

PC	Object	I	Line	Source
A			1	TITLE "***** TRSDOS *****"
A			2	
A			3	define startup,space=RAM,ORG=%0000
A			4	.ASSUME ADL=0
A			5	
B			0	include "copyright_messages.s"
B			1	
B			2	; * Copyright 2021-25, Daniel Paul Martin.
B			3	
B			4	; * All Rights Reserved
B			5	;
B			6	; * This is UNPUBLISHED PROPRIETARY SOURCE CODE of Daniel Paul Martin and may
B			7	; * contain proprietary, confidential and trade secret information of
B			8	; * Daniel Paul Martin and/or his partners.
B			9	;
B			10	; * The contents of this file may not be disclosed to third parties, copied or
B			11	; * duplicated in any form, in whole or in part, without the prior written
B			12	; * permission of Daniel Paul Martin.
B			13	
A			7	
A			8	; After reset, control is turned to 1st step, CPU initialization.
A			9	; This code begins in system overlay region, @ 1E00h.
A			10	
A			11	SUBTITLE "< SYSRES - operating system resident code. >"
A			12	
A			13	
B			0	include "D_library.s"
C			0	include "D_language_macros.s"
C			1	BREAK.disable MACRO
C			2	scope
C			3	LD hl,SFLAG\$
C			4	set 4,(hl) ; Disable BREAK.
C			5	ENDMAC BREAK.disable
C			6	
C			7	BREAK.enable macro
C			8	scope
C			9	LD hl,SFLAG\$
C			10	res 4,(hl) ; Turn BREAK on.
C			11	endmac BREAK.enable
C			12	
C			13	
C			14	DELAY MACRO count ; Delay macro, count is delay duration.
C			15	scope
C			16	IFMA count
C			17	call delay ; Call delay using default value.
C			18	ELSE
C			19	ld b,count ; Load delay value.
C			20	call delay+2 ; Skip into delay routine, not to load default value.
C			21	ENDIF
C			22	ENDMAC DELAY ; End of.
C			0	include "D_language_MATH.s"
C			1	DEC_32 MACRO pointer
C			2	scope
C			3	ld hl,pointer ; 32 bit interrupt counter.
C			4	dec (hl) ; INC LSB.
C			5	jr nz,end_dec_32 ; Did not roll over to zero?
C			6	inc hl ; Bump to next digit.
C			7	dec (hl) ; Increase by one since we had carry from last digit.
C			8	jr nz,end_dec_32 ; Did this digit carry?
C			9	inc hl ; Bump pointer to next digit.
C			10	dec (hl) ; Increase counter by 1.
C			11	jr nz,end_dec_32 ; Test if that increase caused carry to next column.

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYSRES - operating system resident code. &gt;

```

PC      Object      I  Line      Source ..\..\TRSDOS_GLOBAL\macros\D_language_macros\D_language_MATH.s
C      12          inc    hl          ; Bump pointer to next column.
C      13          dec    (hl)        ; INC MSB of 32 bit 4 byte counter if needed.
C      14      end_dec_32    EQU    $
C      15          ENDMAC    DEC_32          ; End of macro.
C      16
C      17      DECHEX          MACRO mystring
C      18          ld    hl,mystring
C      19          SVC    svc_dechex
C      20          ENDMAC    DECHEX
C      21
C      22      DIV16:          MACRO arg1,arg2
C      23          ld    hl,arg1
C      24          ld    c,arg2
C      25          SVC    svc_div16
C      26          ENDMAC
C      27
C      28      DIV8:          MACRO arg1,arg2
C      29          ld    e,arg1
C      30          ld    c,arg2
C      31          SVC    svc_div8
C      32          ENDMAC
C      33
C      34      HEXDEC:          MACRO valu, result
C      35          ld    hl,valu
C      36          ld    de,result
C      37          SVC    svc_hexdec
C      38          ENDMAC    HEXDEC
C      39
C      40      INC_32          MACRO pointer
C      41      scope
C      42          ld    hl,pointer          ; 32 bit interrupt counter.
C      43          inc    (hl)          ; INC LSB.
C      44          jr    nz,end_inc_32          ; Did not roll over to zero?
C      45          inc    hl          ; Bump to next digit.
C      46          inc    (hl)          ; Increase by one since we had carry from last digit.
C      47          jr    nz,end_inc_32          ; Did this digit carry?
C      48          inc    hl          ; Bump pointer to next digit.
C      49          inc    (hl)          ; Increase counter by 1.
C      50          jr    nz,end_inc_32          ; Test if that increase caused carry to next column.
C      51          inc    hl          ; Bump pointer to next column.
C      52          inc    (hl)          ; INC MSB of 32 bit 4 byte counter if needed.
C      53      end_inc_32    EQU    $
C      54          ENDMAC    INC_32          ; End of macro.
C      55
C      56      MUL16:          MACRO multiplicand, multiplier
C      57          ld    hl,multiplicand
C      58          ld    c,multiplier
C      59          SVC    svc_mul16
C      60          ENDMAC
C      61
C      62      MUL8:          MACRO multiplicand, multiplier
C      63          ld    c,multiplicand
C      64          ld    e,multiplier
C      65          SVC    svc_mul8
C      66          ENDMAC
C      0          include "D_language_misc.s"
C      1      DIE          macro
C      2          scope
C      3          jp    $          ; Endless, no return.
C      4          endmac

```

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYSRES - operating system resident code. &gt;

```

PC      Object      I  Line      Source ..\..\TRSDOS_GLOBAL\macros\D_language_macros\D_language_misc.s
C        5
C        6
C        7      dump_buff MACRO  BUFF, LENGTH
C        8
C        9          PUMP  0, dmp_cr$
C       10          ld    de, BUFF
C       11          ld    b, LENGTH
C       12      $more_dmp ld    a, (de)
C       13          inc   de
C       14          push  bc
C       15          push  de
C       16          cvt_ascii  dmp_string$
C       17          PUMP  0, dmp_string$
C       18          pop   de
C       19          pop   bc
C       20          LOOP  $more_dmp
C       21          PUMP  0, dmp_cr$
C       22          endmac      dump_buff
C       23
C       24
C       25
C       26      PUMP      MACRO msgdev, msgbuf      ; Pump a screen built in memory to x serial port.
C       27      scope      ; A tribute to Dave Shriner - Display Data Corp.
C       28
C       29      LD      HL, msgbuf      ; Pump from this starting point until ETX.asc  is found.
C       30
C       31      if      msgdev=0
C       32      $0:      LD      A, (HL)      ; Get byte to send.
C       33      INC      HL      ; pointer=pointer+1
C       34      ;      CP      EOT.asc      ; End Of Transmission (block)?
C       35      ;      jr      Z, $0a      ; Return if finished.
C       36      ;      CP      ETX.asc      ; End Of Text?
C       37      ;      jr      Z, $0a      ; Return if finished.
C       38      ;      CP      CR.asc      ; Was it a CR?
C       39      call    z, $1      ; 1st add a LF to stream.
C       40      LD      C, A      ; Output byte in C.
C       41      XOR      A      ; Set Z flag for put.
C       42      CALL    U0DVR      ; Out the UART we go.
C       43      JR      $0
C       44      $1:      ld      a, LF.asc      ; Get a LF.
C       45      LD      C, A      ; Output byte in C.
C       46      XOR      A      ; Set Z flag for put
C       47      CALL    U0DVR      ; Out the UART we go.
C       48      ld      a, CR.asc      ; Now put a CR back in stream.
C       49      ret      ; Return and print CR.
C       50      $0a      endif
C       51
C       52      if      msgdev=1
C       53      $1:      LD      A, (HL)      ; Get byte to send.
C       54      INC      HL      ; pointer=pointer+1
C       55      ;      CP      EOT.asc      ; End Of Transmission (block)?
C       56      ;      jr      Z, $1a      ; Return if finished.
C       57      ;      CP      ETX.asc      ; End Of Text?
C       58      ;      jr      Z, $1a      ; Return if finished.
C       59      LD      C, A      ; Output byte to C.
C       60      XOR      A      ; Set Z flag for put.
C       61      CALL    U1DVR      ; Out the UART we go.
C       62      JR      $1
C       63      $1a      equ      $
C       64      endif

```

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYSRES - operating system resident code. &gt;

PC	Object	I	Line	Source	...
		C	65	ENDMAC	PUMP
		C	0		include "D_language_CONDITIONALS.s"
		C	1	IF_LT_BRANCH	macro where
		C	2		scope
		C	3		jr nc,where
		C	4		endmac IF_LT_BRANCH
		C	5		
		C	6	IF_GT_BRANCH	macro where
		C	7		scope
		C	8		jr c,where
		C	9		endmac IF_GT_BRANCH
		C	10		
		C	11	IF_EQ_BRANCH	macro where
		C	12		scope
		C	13		jr z,where
		C	14		endmac IF_EQ_BRANCH
		C	15		
		C	16	IF_NE_BRANCH	macro where
		C	17		scope
		C	18		jr nz,where
		C	19		endmac IF_NE_BRANCH
		C	20		
		C	21	IF_EQ_GOSUB	macro where
		C	22		scope
		C	23		call z,where
		C	24		endmac IF_EQ_GOSUB
		C	25		
		C	26	IF_GE_GOSUB	macro where
		C	27		scope
		C	28		call z,where ; If EQUAL =.
		C	29		call c,where ; If GREATER THAN > then carry is set.
		C	30		endmac IF_GE_GOSUB
		C	31		
		C	32	IF_GT_GOSUB	macro where
		C	33		scope
		C	34		call c,where ; GREATER THAN >.
		C	35		endmac IF_GT_GOSUB
		C	36		
		C	37	IF_LE_GOSUB	macro where
		C	38		scope
		C	39		call z,where ; If EQUAL =.
		C	40		call nc,where ; If LESS THAN <.
		C	41		endmac IF_LE_GOSUB
		C	42		
		C	43	IF_LT_GOSUB	macro where ; If LESS THAN <.
		C	44		scope
		C	45		call nc,where
		C	46		endmac IF_LT_GOSUB
		C	47		
		C	48	IF_NE_GOSUB	macro where ; If NOT EQUAL <>.
		C	49		scope
		C	50		call nz,where
		C	51		endmac IF_NE_GOSUB
		C	52		
		C	53	IF_PE_GOSUB	macro where ; If parity even then gosub.
		C	54		scope
		C	55		call pe,label
		C	56		endmac IF_PE_GOSUB
		C	57		
		C	58	IF_PO_GOSUB	macro where ; If parity off gosub.

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYSRES - operating system resident code. &gt;

PC	Object	I	Line	Source	...
		C	59	scope	Source ..\..\TRSDOS_GLOBAL\macros\D_language_macros\D_language_CONDITIONALS.s
		C	60	call po,where	
		C	61	endmac IF_PO_GOSUB	
		C	62		
		C	63	IF_EQ_GOTO macro where	
		C	64	scope	
		C	65	jp z,where	
		C	66	endmac IF_EQ_GOTO	
		C	67		
		C	68	IF_GE_GOTO macro where	
		C	69	scope	
		C	70	jp z,where	
		C	71	jp c,where	
		C	72	endmac IF_GE_GOTO	
		C	73		
		C	74	IF_GT_GOTO macro where	
		C	75	scope	
		C	76	jp c,where	
		C	77	endmac IF_GT_GOTO	
		C	78		
		C	79	IF_LE_GOTO macro where	
		C	80	scope	
		C	81	jp z,where	
		C	82	jp nc,where	
		C	83	endmac IF_LE_GOTO	
		C	84		
		C	85	IF_LT_GOTO macro where	
		C	86	scope	
		C	87	jp nc,where	
		C	88	endmac IF_LT_GOTO	
		C	89		
		C	90	IF_NE_GOTO macro where	
		C	91	scope	
		C	92	jp nz,where	
		C	93	endmac IF_NE_GOTO	
		C	94		
		C	95	IF_EQ_RETURN macro ; If equal, z is set, return.	
		C	96	scope	
		C	97	ret z	
		C	98	endmac IF_EQ_RETURN	
		C	99		
		C	100	IF_NE_RETURN macro ; If Z flag NOT set, return.	
		C	101	scope	
		C	102	ret nz	
		C	103	endmac IF_NE_RETURN	
		C	0	include "D_language_FLOW.s"	
		C	1	BRANCH macro where	
		C	2	scope	
		C	3	jr where	
		C	4	endmac BRANCH	
		C	5		
		C	6	CMD_I MACRO cmd ; COMMAND O/S WITH NO RETURN THE #0 STRING.	
		C	7	LD HL,cmd ; GET ADDRESS OF STRING.	
		C	8	SVC svc_cmdi	
		C	9	ENDMAC CMD_I	
		C	10		
		C	11	CMD_R MACRO cmd ; EXECUTE O/S COMMAND WITH RETURN.	
		C	12	LD HL,cmd ; GET ADDRESS OF COMMAND.	
		C	13	SVC svc_cmdr	
		C	14	ENDMAC CMD_R	

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYSRES - operating system resident code. &gt;

PC	Object	I	Line	Source	...	TRSDOS_GLOBAL\macros\D_language_macros\D_language_FLOW.s
		C	15			
		C	16	DEBUG_START	macro	
		C	17	SVC	svc_debug	
		C	18	endmac	DEBUG_START	
		C	19			
		C	20	GOSUB	macro where	
		C	21	scope		
		C	22	CALL	where	
		C	23	endmac	GOSUB	
		C	24			
		C	25	GOTO	macro where	
		C	26	scope		
		C	27	jp	where	
		C	28	endmac	GOTO	
		C	29			
		C	30	RETURN	macro	
		C	31	scope		
		C	32	ret		; Return from call.
		C	33	endmac	RETURN	
		C	34			
		C	35	SYS_ABORT	MACRO	
		C	36	scope		
		C	37	SVC	svc_abort	
		C	38	ENDMAC	SYS_ABORT	
		C	39			
		C	40	SYS_EXIT	MACRO exitcode	
		C	41	scope		
		C	42	ld	hl,exitcode	
		C	43	SVC	svc_zexit	
		C	44	ENDMAC	SYS_EXIT	
		C	45			
		C	46	LOOP	macro location	; B=B-1. If B<>0 then goto location.
		C	47	scope		
		C	48	djnz	location	
		C	49	endmac	LOOP	
		C	50			
		C	51	FOR	MACRO START,STOP,STEP	
		C	52	SCOPE		
		C	53	ld	bc,(\$1)	
		C	54	ld	(\$4),bc	
		C	55	JR	\$5	
		C	56	\$4	DW 0	; I hold count when running (+8).
		C	57	\$1	DW START	; Start count from xx (+10).
		C	58	\$2	DW STOP	; Stop when equal or greater than STOP (+12).
		C	59	\$3	DW STEP	; Step by (+14).
		C	60	\$5	equ \$	
		C	61	ENDMAC	FOR	
		C	62			
		C	63	NEXT	MACRO pointer	
		C	64	scope		
		C	65	ld	hl,(pointer+10)	; Counter.
		C	66	ld	bc,(pointer+16)	; STEP
		C	67	add	hl,bc	
		C	68	JR	C,\$0	; Overflow, exceeds limit of counter, were done.
		C	69	PUSH	HL	
		C	70	LD	BC,(pointer+14)	; Get limit.
		C	71	and	A	; Clear carry flag.
		C	72	SBC	HL,BC	; count-limit.
		C	73	POP	bc	
		C	74	JP	z,pointer+4	; limit=counter? Go around again.

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYSRES - operating system resident code. &gt;

```

PC      Object      I  Line      Source ..\..\TRSDOS_GLOBAL\macros\D_language_macros\D_language_FLOW.s
C       75          JP      m,pointer+4      ; limit>counter? Yes, go around again.
C       76          $0:      equ      $
C       77          endmac      NEXT
C       78
C       79          WHERE:      MACRO                      ; Return out location in memory.
C       80          SVC          svc_where
C       81          ENDMAC
C       0           include "D_language_DATA.s"
C       1          CMP.bytes MACRO length, buff1, buff2, nomatch ; Compare buffers. If <> goto nomatch.
C       2          scope
C       3          ld      b, length      ; Setup to compare length bytes.
C       4          ld      hl, buff1      ; Get address of first buffer.
C       5          ld      de, buff2      ; Get address of second buffer.
C       6          $1      ld      a, (de)      ; Get a byte from rbuffer.
C       7          cp      (hl)      ; Compare to other buffer.
C       8          IF_NE_GOTO nomatch      ; If compare fails match.
C       9          inc     hl      ; Advance pointers for
C      10          inc     de      ; both buffers.
C      11          LOOP    $1      ; Loop until B=0
C      12          endmac      CMP.bytes
C      13
C      14          cvt_asciimacro cvt_buff
C      15          ld      hl, cvt_buff
C      16          ld      c, a
C      17          and     0f0h      ; Strip out upper 4 bits.
C      18          rr      a
C      19          rr      a
C      20          rr      a
C      21          rr      a
C      22          and     0fh
C      23          add     '0'
C      24          cp      ':'      ; Is it >9
C      25          IF_GT_GOTO $no_more2
C      26          add     7      ; Adjust for A-F
C      27          $no_more2 ld      (hl), a
C      28          inc     hl
C      29
C      30          ld      a, c
C      31          and     0fh
C      32          add     '0'
C      33          cp      ':'      ; Is it >9
C      34          IF_GT_GOTO $no_more1
C      35          add     7      ; Adjust for A-F
C      36          $no_more1 ld      (hl), a
C      37          inc     hl
C      38          ld      (hl), SP.asc
C      39          endmac      cvt_asciimacro
C      40
C      41          PUT.char macro PUT_dcb, char, on_error
C      42          ld      de, PUT_dcb
C      43          ld      c, char
C      44          SVC      svc_put
C      45          IFMA      3
C      46          JP      nz, on_error      ; Returned with error.
C      47          else
C      48          ENDIF
C      49          endmac      PUT.char
C      50
C      51          STRING.kbd.get MACRO buffer, max
C      52          scope

```

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYSRES - operating system resident code. &gt;

```

PC      Object      I  Line      Source ..\..\TRSDOS_GLOBAL\macros\D_language_macros\D_language_DATA.s
C      53          ld      hl,buffer
C      54          ld      b,max          ; Max number bytes to input.
C      55          ld      c,0
C      56          SVC     svc_keyin
C      57          endmac      STRING.kbd.get
C      58
C      59          STRING.table.get MACRO      index,table,output
C      60          scope
C      61          ld      a,index          ; Index value of get (1-127) to accumulator.
C      62          dec     a
C      63          add     a,a          ; index=index*2.
C      64          ld      e,a
C      65          ld      d,0          ; d=0 e=index*2.
C      66          ld      hl,table      ; Get base address of table.
C      67          add     hl,de          ; Point hl to index table.
C      68          ld      bc,(hl)      ; Pickup indexed pointer to entry.
C      69          ld      hl,bc
C      70          ld      bc,(hl)      ; 1st word of entry which is length of entry.
C      71          inc     hl          ; Bump past length.
C      72          inc     hl          ; Bump past length and leave hl pointing to entry.
C      73          ld      de,output    ; Point DE to receiver place for data.
C      74          LDIR          ; Copy BC bytes from (HL) to (DE).
C      75          ENDMAC      STRING.table.get
C      76
C      77          ;STRING.table.put MACRO      index,table,input
C      78          ; scope
C      79          ; ld      a,index          ; Index value of put (1-127).
C      80          ; dec     a          ; index=index-1.
C      81          ; add     a,a          ; index=index*2.
C      82          ; ld      e,a
C      83          ; ld      d,0
C      84          ; ld      hl,table      ; Get base address of table.
C      85          ; add     hl,de          ; Add offset & point hl to table entry.
C      86          ; ld      bc,(hl)      ; Pickup pointer to entry.
C      87          ; ld      hl,bc
C      88          ; ld      bc,(hl)      ; 1st word of entry which is length of entry.
C      89          ; inc     hl          ; Bump past length.
C      90          ; inc     hl          ; Bump past length and leave hl pointing to entry.
C      91          ; ld      de,hl        ; Point DE to receiver place for data.
C      92          ; ld      hl,input    ; Get pointer to input.
C      93          ; LDIR          ; Copy BC bytes from (HL) to (DE).
C      94          ; ENDMAC      STRING.table.put
C      95
C      96          ZEROS_RPL_SPACEmacro buffer
C      97          ld      hl,buffer      ; Get address of buffer to scan & convert.
C      98          ld      d,'0'        ; 0 to replace space.
C      99          ld      a,' '        ; Test for space.
C      100         ld      b,5          ; Test 5 char string.
C      101         $1      cp      (hl)
C      102         jr      nz,$2          ; Replace all spaces with 0's.
C      103         ld      (hl),d
C      104         $2      inc     hl
C      105         LOOP    $1
C      106         endmac      ZEROS_RPL_SPACE
C      0          include "D_language_DATETIME.s"
C      1          DATE.get MACRO buffer
C      2          scope
C      3          ld      hl,buffer
C      4          SVC     svc_date
C      5          ENDMAC      DATE.get

```



\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYSRES - operating system resident code. &gt;

```

PC      Object      I  Line      Source ..\..\TRSDOS_GLOBAL\macros\D_language_macros\D_language_DATETIME.s
C        6
C        7      DATE.put MACRO buffer
C        8          scope
C        9          ld    hl,buffer
C       10          SVC   svc_date
C       11          ENDMAC      DATE.put
C       12
C       13      TIME.get MACRO buffer
C       14          scope
C       15          ld    hl,buffer
C       16          SVC   svc_time
C       17          ENDMAC      TIME.get
C       18
C       19      TIME.put MACRO buffer
C       20          scope
C       21          ld    hl,buffer
C       22          SVC   svc_time
C       23          ENDMAC      TIME.put
C        0          include "D_language_MOVES.s"
C        1      MVC      MACRO source,destination,count ; Move Character, from my IBM MVS days.
C        2          scope
C        3          ld    hl,source
C        4          ld    de,destination
C        5          ld    bc,count
C        6          LDIR
C        7          ENDMAC      MVC
C        0          include "D_language_UART0.s"
C        1      ; UART 0 macros.
C        2
C        3
C        4
C        5      UART0.dtr.resetmacro
C        6          scope
C        7          in0    a,(UART0_MCTL)          ; Get current register status.
C        8          and    11111110b          ; Set bit 0 to 1.
C        9          OUT0   (UART0_MCTL),A          ; UART1 tell other end we are ready.
C       10          endmac      UART0.dtr.reset
C       11      UART0.dtr.set macro
C       12          scope
C       13          in0    a,(UART0_MCTL)          ; Get current register status.
C       14          or     1          ; Set bit 0 to 1.
C       15          OUT0   (UART0_MCTL),A          ; UART1 tell other end we are ready.
C       16          endmac      UART0.dtr.set
C       17
C       18      UART0.tst.cts macro          ; Test CTS signal on UART 0.
C       19          scope
C       20          IN0     a,(UART0_MSR)          ; Get CTS status & test if CTS is ready..
C       21          BIT    4,A          ; Test if other end sets CTS true, they are ready to receive.
C       22          endmac      UART0.tst.cts
C       23
C       24      UART0.tst.dcd macro
C       25          scope
C       26          IN0     a,(UART0_MSR)          ; Get DCD status & test if DCD is ready..
C       27          BIT    7,A          ; Test if other end sets DCD true.
C       28          endmac      UART0.tst.dcd
C       29
C       30      UART0.tst.dsr macro
C       31          scope
C       32          IN0     a,(UART0_MSR)          ; Get CTS status & test if CTS is ready..
C       33          BIT    5,A          ; Test if other end sets CTS true, they are ready to receive.

```

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYSRES - operating system resident code. &gt;

```

PC      Object      I  Line      Source ..\..\TRSDOS_GLOBAL\macros\D_language_macros\D_language_UART0.s
C       34          endmac      UART0.tst.dsr
C       35
C       36      UART0.tst.ri  macro
C       37          scope
C       38          IN0      a,(UART0_MSR)          ; Get Ring Indicator & test.
C       39          BIT      6,A          ; Test RI.
C       40          endmac      UART0.tst.ri
C       41
C       42      UART0.tst.rx  macro
C       43          scope
C       44          IN0          A,(UART0_LSR)          ; Char waiting?
C       45          AND      UART_DR          ; 1 or TRUE if yes.
C       46          endmac      UART0.tst.rx
C       47
C       48      UART0.tst.tx  macro
C       49          scope          ; Test UART 1 if xmitter buffer empty and ready.
C       50          IN0      A,(UART0_LSR)
C       51          AND      UART_THRE          ; Returning a NE or <> 0 means ready, 1 means ready.
C       52          endmac      UART0.tst.tx
C       0          include "D_language_UART1.s"
C       1      ; UART 1 macros.
C       2
C       3
C       4
C       5      UART1.dtr.set  macro
C       6          scope
C       7          in0      a,(UART1_MCTL)          ; Get current register status.
C       8          or      1          ; Set bit 0 to 1.
C       9          OUT0     (UART1_MCTL),A          ; UART1 tell other end we are ready.
C      10          endmac      UART1.dtr.set
C      11
C      12      UART1.dtr.resetmacro
C      13          scope
C      14          in0      a,(UART1_MCTL)
C      15          and      0FEh          ; Strip out bit 0 making reset.
C      16          OUT0     (UART1_MCTL),A          ; UART1 tell other end we are ready.
C      17          endmac      UART1.dtr.reset
C      18
C      19      UART1.get.byte macro vector          ; Get char from UART1 via driver.
C      20          scope
C      21          ld      a,1
C      22          or      a          ; On timeout or error goto vector.
C      23          scf
C      24          GOSUB     U1DVR          ; C flag set & NZ set means PUT.
C      25          IF_NE_GOSUB vector          ; If nz we had a timeout or other error.
C      26          endmac      UART1.get.byte
C      27
C      28      UART1.get.bytesmacro count,buffer,vector ; # bytes to get, to buffer, vector on error or timeout
C      29          scope
C      30          ld      b,count          ; Input 5 bytes.
C      31          ld      hl,buffer          ; Get address of buffer.
C      32      $1      UART1.get.byte  vector          ; Get byte from UART1. If timeout goto no_pong.
C      33          ld      (hl),c          ; Store char.
C      34          inc      hl
C      35          LOOP     $1          ; Loop for count bytes.
C      36          endmac      UART1.get.bytes
C      37
C      38      UART1.mctl.put macro payload
C      39          scope
C      40          LD      A,payload          ; Get payload to write in MCTL register.

```

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYSRES - operating system resident code. &gt;

```

PC      Object      I  Line      Source ..\..\TRSDOS_GLOBAL\macros\D_language_macros\D_language_UART1.s
C      41          OUT0 (UART1_MCTL),A          ; UART1 tell other end we are ready.
C      42          endmac      UART1.mctl.put
C      43
C      44      UART1.put.byte macro vector          ; Get char from UART1 via driver.
C      45          scope
C      46          xor    a          ; Set Z flag set means PUT.
C      47          GOSUB U1DVR          ; Call UART driver.
C      48          IF_NE_GOSUB vector          ; If error goto error routine.
C      49          endmac      UART1.put.byte
C      50
C      51      UART1.put.bytesmacro count,buffer,vector
C      52          scope
C      53          ld    b,count          ; Were sending "count" bytes.
C      54          ld    hl,buffer
C      55      $1      ld    c,(hl)
C      56          inc    hl
C      57          UART1.put.byte    vector          ; If nz we had a timeout or other error.
C      58          LOOP    $1
C      59          endmac      UART1.put.bytes
C      60      UART1.rts.set macro
C      61          scope
C      62          in0    a,(UART1_MCTL)          ; Get current register status.
C      63          or    2          ; Set bit 0 to 1.
C      64          OUT0 (UART1_MCTL),A          ; UART1 tell other end we are ready.
C      65          endmac      UART1.rts.set
C      66
C      67      UART1.rts.resetmacro
C      68          scope
C      69          in0    a,(UART1_MCTL)
C      70          and    0FDh          ; Strip out bit 0 making reset.
C      71          OUT0 (UART1_MCTL),A          ; UART1 tell other end we are ready.
C      72          endmac      UART1.rts.reset
C      73
C      74      UART1.tst.cts macro wait_loop          ; Test CTS signal on UART 1.
C      75          scope
C      76          IN0    a,(UART1_MSR)          ; Get CTS status & test if CTS is ready..
C      77          BIT    4,A          ; If other end sets CTS true, they are ready to receive.
C      78          jr    z,wait_loop          ; Loop until ready.
C      79          endmac      UART1.tst.cts
C      80
C      81      UART1.tst.dcd macro
C      82          scope
C      83          IN0    a,(UART1_MSR)          ; Get DCD status & test if DCD is ready..
C      84          BIT    7,A          ; Test if other end sets DCD true.
C      85          endmac      UART1.tst.dcd
C      86
C      87      UART1.tst.dsr macro
C      88          scope
C      89          IN0    a,(UART1_MSR)          ; Get CTS status & test if CTS is ready..
C      90          BIT    5,A          ; Test if other end sets CTS true, they are ready to receive.
C      91          endmac      UART1.tst.dsr
C      92
C      93      UART1.tst.ri macro
C      94          scope
C      95          IN0    a,(UART1_MSR)          ; Get Ring Indicator & test.
C      96          BIT    6,A          ; Test RI.
C      97          endmac      UART1.tst.ri
C      98
C      99      UART1.tst.rx macro wait_loop
C      100         scope

```

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYSRES - operating system resident code. &gt;

```

PC      Object      I  Line      Source ..\..\TRSDOS_GLOBAL\macros\D_language_macros\D_language_UART1.s
C      101          IN0      A,(UART1_LSR)          ; Char waiting?
C      102          AND      UART_DR          ; 1 or TRUE if yes.
C      103          jr      z,wait_loop      ; Loop if not ready.
C      104          endmac      UART1.tst.rx
C      105
C      106          UART1.tst.tx      macro wait_loop
C      107          scope          ; Test UART 1 if xmitter buffer empty and ready.
C      108          IN0      A,(UART1_LSR)
C      109          AND      UART_THRE          ; Returning a NE or <> 0 means ready, 0 means not ready.
C      110          jr      z,wait_loop
C      111          endmac      UART1.tst.tx
B       0          include "DD_library.s"
C       0          include "DD_language_CONDITIONALS.s"
C       1          if_LT_BRANCH      macro where
C       2          scope
C       3          jr      nc,where
C       4          endmac      if_LT_BRANCH
C       5
C       6          if_GT_BRANCH      macro where
C       7          scope
C       8          jr      c,where
C       9          endmac      if_GT_BRANCH
C      10
C      11          if_EQ_BRANCH      macro where
C      12          scope
C      13          jr      z,where
C      14          endmac      if_EQ_BRANCH
C      15
C      16          if_NE_BRANCH      macro where
C      17          scope
C      18          jr      nz,where
C      19          endmac      if_NE_BRANCH
C      20
C      21          if_EQ_GOSUB      macro where
C      22          scope
C      23          call      z,where
C      24          endmac      if_EQ_GOSUB
C      25
C      26          if_GE_GOSUB      macro where
C      27          scope
C      28          call      z,where          ; If EQUAL =.
C      29          call      c,where          ; If GREATER THAN > then carry is set.
C      30          endmac      if_GE_GOSUB
C      31
C      32          if_GT_GOSUB      macro where
C      33          scope
C      34          call      c,where          ; GREATER THAN >.
C      35          endmac      if_GT_GOSUB
C      36
C      37          if_LE_GOSUB      macro where
C      38          scope
C      39          call      z,where          ; If EQUAL =.
C      40          call      nc,where          ; If LESS THAN <.
C      41          endmac      if_LE_GOSUB
C      42
C      43          if_LT_GOSUB      macro where          ; If LESS THAN <.
C      44          scope
C      45          call      nc,where
C      46          endmac      if_LT_GOSUB
C      47

```

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYSRES - operating system resident code. &gt;

PC	Object	I	Line	Source	...
		C	48	if_NE_GOSUB	macro where ; If NOT EQUAL <>.
		C	49	scope	
		C	50	call nz,where	
		C	51	endmac	if_NE_GOSUB
		C	52		
		C	53	if_PE_GOSUB	macro where ; If parity even then gosub.
		C	54	scope	
		C	55	call pe,label	
		C	56	endmac	if_PE_GOSUB
		C	57		
		C	58	if_PO_GOSUB	macro where ; If parity off gosub.
		C	59	scope	
		C	60	call po,where	
		C	61	endmac	if_PO_GOSUB
		C	62		
		C	63	if_EQ_GOTO	macro where
		C	64	scope	
		C	65	jp z,where	
		C	66	endmac	if_EQ_GOTO
		C	67		
		C	68	if_GE_GOTO	macro where
		C	69	scope	
		C	70	jp z,where	
		C	71	jp c,where	
		C	72	endmac	if_GE_GOTO
		C	73		
		C	74	if_GT_GOTO	macro where
		C	75	scope	
		C	76	jp c,where	
		C	77	endmac	if_GT_GOTO
		C	78		
		C	79	if_LE_GOTO	macro where
		C	80	scope	
		C	81	jp z,where	
		C	82	jp nc,where	
		C	83	endmac	if_LE_GOTO
		C	84		
		C	85	if_LT_GOTO	macro where
		C	86	scope	
		C	87	jp nc,where	
		C	88	endmac	if_LT_GOTO
		C	89		
		C	90	if_NE_GOTO	macro where
		C	91	scope	
		C	92	jp nz,where	
		C	93	endmac	if_NE_GOTO
		C	94		
		C	95	if_EQ_RETURN	macro ; If equal, z is set, return.
		C	96	scope	
		C	97	ret z	
		C	98	endmac	if_EQ_RETURN
		C	99		
		C	100	if_NE_RETURN	macro ; If Z flag NOT set, return.
		C	101	scope	
		C	102	ret nz	
		C	103	endmac	if_NE_RETURN
		C	0		include "DD_language_DATETIME.s"
		C	1	date.GET	MACRO buffer
		C	2	scope	
		C	3	ld	hl,buffer

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYSRES - operating system resident code. &gt;

PC	Object	I	Line	Source	...
		C	4	SVC	svc_date
		C	5	ENDMAC	date.GET
		C	6		
		C	7	date.PUT	MACRO buffer
		C	8		scope
		C	9	ld	hl,buffer
		C	10	SVC	svc_date
		C	11	ENDMAC	date.PUT
		C	12		
		C	13	time.GET	MACRO buffer
		C	14		scope
		C	15	ld	hl,buffer
		C	16	SVC	svc_time
		C	17	ENDMAC	time.GET
		C	18		
		C	19	time.PUT	MACRO buffer
		C	20		scope
		C	21	ld	hl,buffer
		C	22	SVC	svc_time
		C	23	ENDMAC	time.PUT
		C	0		include "DD_language_DEVICE.s"
		C	1	device.KI.DCB	macro storage
		C	2	dw	0 ; Container to hold vector to
		C	3	db	"KI"
		C	4	if	storage<>0
		C	5	ds	storage
		C	6	endif	
		C	7	endmac	device.KI.DCB
		C	8		
		C	9	device.D0.DCB	macro storage
		C	10	dw	0 ; Container to hold vector to
		C	11	db	"D0"
		C	12	if	storage<>0
		C	13	ds	storage
		C	14	endif	
		C	15	endmac	device.D0.DCB
		C	16		
		C	17	device.PR.DCB	macro storage
		C	18	dw	0 ; Container to hold vector to
		C	19	db	"PR"
		C	20	if	storage<>0
		C	21	ds	storage
		C	22	endif	
		C	23	endmac	device.PR.DCB
		C	24		
		C	25	device.CL.DCB	macro storage
		C	26	dw	0 ; Container to hold vector to
		C	27	db	"CL"
		C	28	if	storage<>0
		C	29	ds	storage
		C	30	endif	
		C	31	endmac	device.CL.DCB
		C	32		
		C	33	device.U1.DCB	macro storage
		C	34	dw	0 ; Container to hold vector to
		C	35	db	"U1"
		C	36	if	storage<>0
		C	37	ds	storage
		C	38	endif	
		C	39	endmac	device.U1.DCB

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYSRES - operating system resident code. &gt;

PC	Object	I	Line	Source	...
		C	0	include	"DD_language_flow.s"
		C	1	branch	macro where
		C	2	scope	
		C	3	jr	where
		C	4	endmac	branch
		C	5		
		C	6	cmd_i	MACRO cmd ; COMMAND O/S WITH NO RETURN THE #0 STRING.
		C	7	LD	HL,cmd ; GET ADDRESS OF STRING.
		C	8	SVC	svc_cmdi
		C	9	ENDMAC	cmd_i
		C	10		
		C	11	cmd_r	MACRO cmd ; EXECUTE O/S COMMAND WITH RETURN.
		C	12	LD	HL,cmd ; GET ADDRESS OF COMMAND.
		C	13	SVC	svc_cmdr
		C	14	ENDMAC	cmd_r
		C	15		
		C	16	debug.START	macro
		C	17	SVC	svc_debug
		C	18	endmac	debug.START
		C	19		
		C	20	error.ABORT	macro errnum
		C	21	scope	
		C	22	ld	c,errnum
		C	23	SVC	svc_error
		C	24	endmac	error.ABORT
		C	25		
		C	26	gosub	macro where
		C	27	scope	
		C	28	CALL	where
		C	29	endmac	gosub
		C	30		
		C	31	goto	macro where
		C	32	scope	
		C	33	jp	where
		C	34	endmac	goto
		C	35		
		C	36	return	macro
		C	37	scope	
		C	38	ret	; Return from call.
		C	39	endmac	return
		C	40		
		C	41	sys.ABORT	MACRO
		C	42	scope	
		C	43	SVC	svc_abort
		C	44	ENDMAC	sys.ABORT
		C	45		
		C	46	sys_exit	MACRO exitcode
		C	47	scope	
		C	48	ld	hl,exitcode
		C	49	SVC	svc_zexit
		C	50	ENDMAC	sys_exit
		C	51		
		C	52	loop	macro location ; B=B-1. If B<>0 then goto location.
		C	53	scope	
		C	54	djnz	location
		C	55	endmac	loop
		C	56		
		C	57	for	MACRO START,STOP,STEP
		C	58	SCOPE	
		C	59	ld	bc,(\$1)

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYSRES - operating system resident code. &gt;

```

PC      Object      I  Line      Source  ..\..\TRSDOS_GLOBAL\macros\DD_language_macros\DD_language_flow.s
C      60          ld      ($4),bc
C      61          JR      $5
C      62      $4      DW      0          ; I hold count when running (+8).
C      63      $1      DW      START      ; Start count from xx (+10).
C      64      $2      DW      STOP      ; Stop when equal or greater than STOP (+12).
C      65      $3      DW      STEP      ; Step by (+14).
C      66      $5      equ      $
C      67          ENDMAC      for
C      68
C      69      next      MACRO pointer
C      70          scope
C      71          ld      hl,(pointer+10)      ; Counter.
C      72          ld      bc,(pointer+16)      ; STEP
C      73          add     hl,bc
C      74          JR      C,$0          ; Overflow, exceeds limit of counter, were done.
C      75          PUSH   HL
C      76          LD      BC,(pointer+14)      ; Get limit.
C      77          and     A          ; Clear carry flag.
C      78          SBC     HL,BC          ; count-limit.
C      79          POP    bc
C      80          JP      z,pointer+4      ; limit=counter? Go around again.
C      81          JP      m,pointer+4      ; limit>counter? Yes, go around again.
C      82      $0:      equ      $
C      83          endmac      next
C      84
C      85      where      MACRO          ; Return out location in memory.
C      86          SVC     svc_where
C      87          ENDMAC      where
C      88
C      89      include "DD_language_IO.s"
C      90
C      91      get.IO      macro module_dcb,count,buffer,on_error
C      92          scope
C      93          ld      b,count          ; Pickup number of bytes to get.
C      94          ld      hl,buffer      ; Address of buffer to store chars.
C      95          ld      de,module_dcb      ; DCB of device performing GET.
C      96      $1      SVC     svc_get      ; Perform GET.
C      97          IF_NE_GOSUB on_error      ; If error CALL error routine.
C      98          ld      (hl),a          ; Store char.
C      99          inc     hl          ; Bump pointer + 1.
C      100         LOOP   $1          ; Loop until all bytes read.
C      101         endmac      get.IO
C      102
C      103      put.CHAR      macro PUT_dcb,char,on_error
C      104          ld      de,PUT_dcb
C      105          ld      c,char
C      106          SVC     svc_put
C      107          IFMA     3
C      108          JP      nz,on_error      ; Returned with error.
C      109          else
C      110          ENDIF
C      111         endmac      put.CHAR
C      112
C      113
C      114      put.IO      macro device,count,fm__pointer,on_error
C      115          scope
C      116          ld      b,count          ; Were sending "count" bytes.
C      117          ld      hl,fm__pointer
C      118      $1      ld      c,(hl)
C      119          inc     hl
C      120          ld      de,device      ; Get pointer to DCB in DE.
C      121          SVC     svc_put      ; Send byte using put.

```



\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYSRES - operating system resident code. &gt;

```

PC      Object      I  Line      Source ..\..\TRSDOS_GLOBAL\macros\DD_language_macros\DD_language_IO.s
C      32          IF_NE_GOSUB on_error          ; If error goto error routine.
C      33          LOOP $1
C      34          endmac      put.IO
C      35
C      36      put.IO.MSG      macro dev_dcb,buffer,on_error      ; Output a message line to device.
C      37          scope
C      38          ld      de,dev_dcb      ; Get pointer to output device control block.
C      39          ld      hl,buffer      ; Buffer with line to output.
C      40          SVC      svc_msg      ; OS sends message to device.
C      41          IF_NE_GOTO on_error      ; Vector if error.
C      42          endmac      put.IO.MSG
C      43
C      44      put.LOG.MSG      macro buffer,on_error      ; Log message in buffer to system log.
C      45          scope
C      46          ld      hl,buffer
C      47          SVC      svc_logger      ; Log message.
C      48          IF_NE_GOTO on_error
C      49          endmac      put.LOG.MSG
C      50
C      51      put.LOG.MSG.DSP      macro buffer,on_error      ; Log message to system log & display.
C      52          scope
C      53          ld      hl,buffer
C      54          SVC      svc_logot
C      55          IF_NE_GOTO on_error      ; GOTO if error on logging.
C      56          endmac      put.LOG.MSG.DSP
C      57
C      58      get.KBD.STRING      MACRO buffer,max
C      59          scope
C      60          ld      hl,buffer
C      61          ld      b,max      ; Max number bytes to input.
C      62          ld      c,0
C      63          SVC      svc_keyin
C      64          endmac      get.KBD.STRING
C      65
C      66      get.TABLE.STRING MACRO      index,table,output
C      67          scope
C      68          ld      a,index      ; Index value of get (1-127) to accumulator.
C      69          dec      a
C      70          add      a,a      ; index=index*2.
C      71          ld      e,a
C      72          ld      d,0      ; d=0 e=index*2.
C      73          ld      hl,table      ; Get base address of table.
C      74          add      hl,de      ; Point hl to index table.
C      75          ld      bc,(hl)      ; Pickup indexed pointer to entry.
C      76          ld      hl,bc
C      77          ld      bc,(hl)      ; 1st word of entry which is length of entry.
C      78          inc      hl      ; Bump past length.
C      79          inc      hl      ; Bump past length and leave hl pointing to entry.
C      80          ld      de,output      ; Point DE to receiver place for data.
C      81          LDIR      ; Copy BC bytes from (HL) to (DE).
C      82          ENDMAC      get.TABLE.STRING
C      0          include      "DD_language_MACROS.s"
C      1      break.DISABLE MACRO
C      2          scope
C      3          LD      hl,SFLAG$
C      4          set      4,(hl)      ; Disable BREAK.
C      5          ENDMAC      break.DISABLE
C      6
C      7      break.ENABLE macro
C      8          scope

```

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYSRES - operating system resident code. &gt;

```

PC      Object      I  Line      Source ..\..\TRSDOS_GLOBAL\macros\DD_language_macros\DD_language_MACROS.s
C        9          C        LD      hl,SFLAG$
C       10          C        res     4,(hl)          ; Turn BREAK on.
C       11          C        endmac    break.enable
C       12
C       13
C       14          C        time.DELAY  MACRO count          ; Delay macro, count is delay duration.
C       15          C        scope
C       16          C        IFMA  count
C       17          C        call  delay          ; Call delay using default value.
C       18          C        ELSE
C       19          C        ld     b,count          ; Load delay value.
C       20          C        call  delay+2        ; Skip into delay routine, not to load default value.
C       21          C        ENDIF
C       22          C        ENDMAC    time.DELAY      ; End of.
C        0          C        include  "DD_language_MATH.s"
C        1          C        dec.32      MACRO pointer
C        2          C        scope
C        3          C        ld     hl,pointer      ; 32 bit interrupt counter.
C        4          C        dec     (hl)          ; INC LSB.
C        5          C        jr     nz,end_dec_32      ; Did not roll over to zero?
C        6          C        inc     hl          ; Bump to next digit.
C        7          C        dec     (hl)          ; Increase by one since we had carry from last digit.
C        8          C        jr     nz,end_dec_32      ; Did this digit carry?
C        9          C        inc     hl          ; Bump pointer to next digit.
C       10          C        dec     (hl)          ; Increase counter by 1.
C       11          C        jr     nz,end_dec_32      ; Test if that increase caused carry to next column.
C       12          C        inc     hl          ; Bump pointer to next column.
C       13          C        dec     (hl)          ; INC MSB of 32 bit 4 byte counter if needed.
C       14          C        end_dec_32  EQU     $
C       15          C        ENDMAC    dec.32          ; End of macro.
C       16
C       17          C        cvt.DEC.HEX      MACRO mystring
C       18          C        ld     hl,mystring
C       19          C        SVC     svc_dechex
C       20          C        ENDMAC    cvt.DEC.HEX
C       21
C       22          C        div.16:    MACRO arg1,arg2
C       23          C        ld     hl,arg1
C       24          C        ld     c,arg2
C       25          C        SVC     svc_div16
C       26          C        ENDMAC    div.16
C       27
C       28          C        div.8:     MACRO arg1,arg2
C       29          C        ld     e,arg1
C       30          C        ld     c,arg2
C       31          C        SVC     svc_div8
C       32          C        ENDMAC    div.8
C       33
C       34          C        cvt.HEX.DEC:  MACRO valu, result
C       35          C        ld     hl,valu
C       36          C        ld     de,result
C       37          C        SVC     svc_hexdec
C       38          C        ENDMAC    cvt.HEX.DEC
C       39
C       40          C        inc.32      MACRO pointer
C       41          C        scope
C       42          C        ld     hl,pointer      ; 32 bit interrupt counter.
C       43          C        inc     (hl)          ; INC LSB.
C       44          C        jr     nz,end_inc_32      ; Did not roll over to zero?
C       45          C        inc     hl          ; Bump to next digit.

```

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYSRES - operating system resident code. &gt;

```

PC      Object      I  Line      Source ..\..\TRSDOS_GLOBAL\macros\DD_language_macros\DD_language_MATH.s
C      46          inc    (hl)          ; Increase by one since we had carry from last digit.
C      47          jr     nz,end_inc_32    ; Did this digit carry?
C      48          inc    hl              ; Bump pointer to next digit.
C      49          inc    (hl)            ; Increase counter by 1.
C      50          jr     nz,end_inc_32    ; Test if that increase caused carry to next column.
C      51          inc    hl              ; Bump pointer to next column.
C      52          inc    (hl)            ; INC MSB of 32 bit 4 byte counter if needed.
C      53      end_inc_32    EQU    $
C      54          ENDMAC    inc.32          ; End of macro.
C      55
C      56      mul.16          MACRO multiplicand, multiplier
C      57          ld      hl,multiplicand
C      58          ld      c,multiplier
C      59          SVC     svc_mul16
C      60          ENDMAC    mul.16
C      61
C      62      mul.8          MACRO multiplicand, multiplier
C      63          ld      c,multiplicand
C      64          ld      e,multiplier
C      65          SVC     svc_mul8
C      66          ENDMAC    mul.8
C      0          include "DD_language_MOVES.s"
C      1      mov.BYT          MACRO source,destination,count ; Move Character, from my IBM MVS days.
C      2          scope
C      3
C      4          ; On entry, HL points to location containing bytes to move.
C      5          ; DE contains pointer to destination.
C      6          ; Length is maximum number of bytes to move (16 bits).
C      7
C      8          ; On exit.
C      9          ; BC=0
C     10          ; HL = Pointer next char to move.
C     11          ; DE = Points to next place in destination buffer to store a char.
C     12
C     13          ld      hl,source
C     14          ld      de,destination
C     15          ld      bc,count
C     16          LDIR
C     17
C     18          ENDMAC    mov.BYT
C     19
C     20
C     21      mov.CHR          MACRO source,destination,length,stop ; Move string from source to dest for length
C     22          scope          ; bytes or until stop byte is encountered.
C     23
C     24          ; On entry, HL points to location containing bytes to move.
C     25          ; Destination contains pointer of destination.
C     26          ; Length is maximum number of bytes to move (16 bits).
C     27          ; Move terminates before max number bytes if byte=stop value.
C     28
C     29          ; On exit.
C     30          ; BC=0 or BC= # chars not moved bc of termination character.
C     31          ; HL = Pointer next char to move.
C     32          ; DE = Points to next place in destination buffer to store a char.
C     33          ; A = Termination char.
C     34
C     35
C     36          ld      hl,source          ; Where we are moving from.
C     37          ld      de,destination      ; Get destination in memory.
C     38          ld      bc,length          ; Get how many characters to move.

```

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYSRES - operating system resident code. &gt;

```

PC      Object      I  Line      Source  ..\..\TRSDOS_GLOBAL\macros\DD_language_macros\DD_language_MOVES.s
C      39          dec  de          ; Start back one byte.
C      40      $0:    ld    a,stop    ; What character stops moving operation.
C      41          cpi              ; Compare (HL) with A register. BC=BC-1
C      42          IF_EQ_BRANCH      $1      ; Did not get termination character? Keep looping until done.
C      43          dec  hl
C      44          inc  de
C      45          ld    a,(hl)        ; Get character.
C      46          ld    (de),a       ; Store in new destination.
C      47          inc  hl
C      48          jp    v,$0          ; Keep looping until done (BC=0).
C      49      $1:    inc  de
C      50      $2:    endmac          mov.CHR
C      0          include "DD_language_STRINGS.s"
C      1      cvt.HEX.ASCII          macro cvt_buff
C      2          ld    hl,cvt_buff
C      3          ld    c,a
C      4          and  0f0h          ; Strip out upper 4 bits.
C      5          rr    a
C      6          rr    a
C      7          rr    a
C      8          rr    a
C      9          and  0fh
C     10          add  '0'
C     11          cp    ':'          ; Is it >9
C     12          IF_GT_GOTO $no_more2
C     13          add  7              ; Adjust for A-F
C     14      $no_more2 ld    (hl),a
C     15          inc  hl
C     16
C     17          ld    a,c
C     18          and  0fh
C     19          add  '0'
C     20          cp    ':'          ; Is it >9
C     21          IF_GT_GOTO $no_more1
C     22          add  7              ; Adjust for A-F
C     23      $no_more1 ld    (hl),a
C     24          inc  hl
C     25          ld    (hl),SP.asc
C     26          endmac          cvt.HEX.ASCII
C     27      ; On entry.
C     28      ; Location is a pointer to string in memory.
C     29      ; Terminator is value search is terminated on when finding in string.
C     30
C     31      ; On exit.
C     32      ; A = terminator byte.
C     33      ; HL = Pointer to terminator byte.
C     34      ; BC = Count of bytes from beginning or length in 2s complement.
C     35
C     36      find.BYT          macro location,terminator
C     37          scope
C     38
C     39          ld    a,terminator    ; Stop counter if hitting this.
C     40          ld    hl,location    ; Pointer to data.1
C     41          ld    bc,0          ; Start counter @ 00.
C     42      $1:    cpi              ; Compare (HL) to A...HL=HL+1, BC=BC-1.
C     43          IF_NE_BRANCH $1      ; If now byte were looking for loop.
C     44          dec  hl
C     45          endmac          find.BYT
C     46
C     47      get.STR.TBL          MACRO index,table,output

```

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYSRES - operating system resident code. &gt;

PC	Object	I	Line	Source	...	TRSDOS_GLOBAL\macros\DD_language_macros\DD_language_STRINGS.s
		C	48	scope		
		C	49	ld a,index		; Index value of get (1-127) to accumulator.
		C	50	dec a		
		C	51	add a,a		; index=index*2.
		C	52	ld e,a		
		C	53	ld d,0		; d=0 e=index*2.
		C	54	ld hl,table		; Get base address of table.
		C	55	add hl,de		; Point hl to index table.
		C	56	ld bc,(hl)		; Pickup indexed pointer to entry.
		C	57	ld hl,bc		
		C	58	ld bc,(hl)		; 1st word of entry which is length of entry.
		C	59	inc hl		; Bump past length.
		C	60	inc hl		; Bump past length and leave hl pointing to entry.
		C	61	ld de,output		; Point DE to receiver place for data.
		C	62	LDIR		; Copy BC bytes from (HL) to (DE).
		C	63	ENDMAC	get.STR.TBL	
		C	64			
		C	65	mov.STR	MACRO source,destination	; Move string... LENGTH.STRING.
		C	66	scope		
		C	67	ld hl,source+2		; P/U source address.
		C	68	ld bc,(source)		; Get address of string\$
		C	69	ld (destination),bc		; Store length in destination string.
		C	70	ld de,destination+2		; Get destination to DE.
		C	71	LDIR		; Move a block of bytes from (HL) to (DE) until BC=0
		C	72	ENDMAC	mov.STR	
		C	73			
		C	74	put.STR.TBL	MACRO index,table,input	
		C	75	scope		
		C	76	ld a,index		; Index value of put (1-127).
		C	77	dec a		; index=index-1.
		C	78	add a,a		; index=index*2.
		C	79	ld e,a		
		C	80	ld d,0		
		C	81	ld hl,table		; Get base address of table.
		C	82	add hl,de		; Add offset & point hl to table entry.
		C	83	ld bc,(hl)		; Pickup pointer to entry.
		C	84	ld hl,bc		
		C	85	ld bc,(hl)		; 1st word of entry which is length of entry.
		C	86	inc hl		; Bump past length.
		C	87	inc hl		; Bump past length and leave hl pointing to entry.
		C	88	ld de,hl		; Point DE to receiver place for data.
		C	89	ld hl,input		; Get pointer to input.
		C	90	LDIR		; Copy BC bytes from (HL) to (DE).
		C	91	ENDMAC	put.STR.TBL	
		C	92			
		C	93	cvt_ZEROS2SPACE	macro buffer	
		C	94	ld hl,buffer		; Get address of buffer to scan & convert.
		C	95	ld d,'0'		; 0 to replace space.
		C	96	ld a,' '		; Test for space.
		C	97	ld b,5		; Test 5 char string.
		C	98	\$1	cp (hl)	
		C	99	jr nz,\$2		; Replace all spaces with 0's.
		C	100	ld (hl),d		
		C	101	\$2	inc hl	
		C	102	LOOP	\$1	
		C	103	endmac	cvt_ZEROS2SPACE	
		C	0	include	"DD_language_SYS_DATA.s"	
		C	1	get.avail.DCB	MACRO	; Get 1st empty device control block (DCB).
		C	2	ld de,0		; Tell SVC to recover 1st empty DCB.
		C	3	SVC	svc_gtdcb	

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYSRES - operating system resident code. &gt;

PC	Object	I	Line	Source	...	Source
		C	4	endmac		get.avail.DCB
		C	5			
		C	6	get.DCB	MACRO	DCB,not_found
		C	7		ld hl,DCB	; Get device control block (DCB).
		C	8		inc hl	
		C	9		inc hl	
		C	10		ld de,(hl)	
		C	11		SVC svc_gtdcb	
		C	12		IF_NE_GOSUB not_found	
		C	13		ld (DCB),hl	; Store DCB pointer to UART1 in DCB container.
		C	14		endmac	get.DCB
		C	15			
		C	16	get.DCT	MACRO	num ; Get DCT for num (0-7).
		C	17		ld c,num	
		C	18		SVC svc_gtdct	
		C	19		endmac	get.DCT ; Return address of DCT in IY register.
		C	20			
		C	21	get.ECI	macro	
		C	22		FLAGS.get	
		C	23		ld a,(IY+zEFLAG\$)	; Get ECI value.
		C	24		endmac	get.ECI
		C	25			
		C	26	put.ECI	MACRO	value ; SET ECI FLAG TO value.
		C	27		get.FLAGS	; GET FLAGS POINTER IN IY.
		C	28		LD A,value	; GET VALUE.
		C	29		LD (IY+zEFLAG\$),A	; STORE VALUE.
		C	30		ENDMAC	put.ECI
		C	31			
		C	32	get.FLAGS	MACRO	
		C	33		scope	
		C	34		SVC svc_flags	
		C	35		ENDMAC	get.FLAGS
		C	36			
		C	37	get.MODULE	macro	name,not_found
		C	38		scope	
		C	39		ld de,name	
		C	40		inc de	
		C	41		inc de	
		C	42		SVC svc_gtmod	
		C	43		IF_NE_GOSUB not_found	
		C	44		ld (name),hl	; Store pointer to module.
		C	45		endmac	get.MODULE
		C	0		include	"DD_language_CONSOLE.s"
		C	1	cls	MACRO	
		C	2		scope	
		C	3		SVC svc_cls	
		C	4		ENDMAC	cls
		C	5			
		C	6	get.KBD.KEY	macro	on_error ; Get a key from keyboard.
		C	7		scope	
		C	8		SVC svc_key	; Go get key, wait until one available.
		C	9		IF_NZ_GOTO on_error	; KI possibly routed?
		C	10		endmac	get.KBD.KEY
		C	11			
		C	12	get.KBD.LINE	macro	num_char,buffer,on_BREAK ; Input a line from keyboard.
		C	13		scope	
		C	14		ld b,num_char	; max number of chars to input.
		C	15		ld c,0	; Per programmers ref manual c=0.
		C	16		ld hl,buffer	; Buffer size = num_char+1.
		C	17		SVC svc_keyin	; Input line.

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYSRES - operating system resident code. &gt;

```

PC      Object      I  Line      Source ..\..\TRSDOS_GLOBAL\macros\DD_language_macros\DD_language_CONSOLE.s
C      18          C      18          IF_NZ_GOTO on_BREAK          ; Vector to BREAK handler.
C      19          C      19          endmac          get.KBD.LINE
C      20
C      21          get.KBD.SCAN          macro          ; Scan keyboard & return in A. NZ=ERROR in A.
C      22          C      22          scope
C      23          C      23          SVC          svc_kbd          ; Invoke supervisor call.
C      24          C      24          endmac          get.KBD.SCAN
C      25
C      26          put.DSP.BUFFER          macro buffer          on_error
C      27          C      27          SVC
C      28          C      28          ld          hl,buffer          ; Get the buffer address. Any valid ld hl,
C      29          $more_blk          C      29          ld          a,(hl)          ; p/u a char.
C      30          C      30          cp          ETX.asc          ; Is it end of block?
C      31          C      31          IF_EQ_BRANCH $blk_done          ; No more chars ETX received.
C      32          C      32          dsp.BYTE a
C      33          C      33          inc          hl          ; Bump to next char.
C      34          C      34          jr          $more_blk          ; Display more chars.
C      35          $blk_done          C      35          ENDMAC          put.DSP.BUFFER          ; 0Dh or 04h (CR or ETX).
C      36
C      37          put.DSP.BYTE          macro char          ; Display char.
C      38          C      38          ld          c,char          ; char can be any valid ld c,
C      39          C      39          SVC          svc_dsp          ; Ask OS to display char.
C      40          C      40          ENDMAC          put.DSP.BYTE
C      41
C      42          put.DSP.LINE          macro buffer
C      43          C      43          ld          hl,buffer          ; Get the buffer address. Any valid ld hl,
C      44          C      44          SVC          svc_dsply          ; Ask OS to display up to 1st
C      45          C      45          ENDMAC          put.DSP.LINE          ; 0Dh or 04h (CR or ETX).
C      46
C      47          locate.CURSOR          macro x,y
C      48          C      48          scope
C      49          C      49          ld          b,3          ; Function 3, relocate cursor.
C      50          C      50          ld          h,x
C      51          C      51          ld          l,y          ; Get column.
C      52          C      52          SVC          svc_vdctl
C      53          C      53          ENDMAC          locate.CURSOR
C      54
C      55          ring.BELL          macro terminal,times          ; Ring bell on terminal & get operator attention.
C      56          C      56          scope
C      57          C      57          ld          b,times
C      58          $9          C      58          push          bc
C      59          C      59          PUT.char          terminal,BELL.asc,$BELL.done
C      60          $BELL.done          C      60          pop          bc
C      61          C      61          LOOP          $9
C      62          C      62          endmac          ring.BELL
C      63
C      64          scrollaway          macro num
C      65          C      65          ld          b,num
C      66          $loop          C      66          ld          a,CR.asc
C      67          C      67          put.DSP.BYTE A
C      68          C      68          LOOP          $loop
C      69          C      69          endmac          scrollaway
C      0          C      0          include "DD_language_UART0.s"
C      1          C      1          ; UART 0 macros.
C      2
C      3
C      4
C      5          reset.UART0.DTRmacro
C      6          C      6          scope
C      7          C      7          in0          a,(UART0_MCTL)          ; Get current register status.

```

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYSRES - operating system resident code. &gt;

```

PC      Object      I  Line      Source ..\..\TRSDOS_GLOBAL\macros\DD_language_macros\DD_language_UART0.s
C        8          C        and 11111110b          ; Set bit 0 to 1.
C        9          C        OUT0 (UART0_MCTL),A          ; UART1 tell other end we are ready.
C       10          C        endmac      reset.UART0.DTR
C       11          C        set.UART0.DTR macro
C       12          C        scope
C       13          C        in0 a, (UART0_MCTL)          ; Get current register status.
C       14          C        or 1          ; Set bit 0 to 1.
C       15          C        OUT0 (UART0_MCTL),A          ; UART1 tell other end we are ready.
C       16          C        endmac      set.UART0.DTR
C       17
C       18          C        test.UART0.CTS macro          ; Test CTS signal on UART 0.
C       19          C        scope
C       20          C        IN0 a, (UART0_MSR)          ; Get CTS status & test if CTS is ready..
C       21          C        BIT 4,A          ; Test if other end sets CTS true, they are ready to receive.
C       22          C        endmac      test.UART0.CTS
C       23
C       24          C        test.UART0.DCD macro
C       25          C        scope
C       26          C        IN0 a, (UART0_MSR)          ; Get DCD status & test if DCD is ready..
C       27          C        BIT 7,A          ; Test if other end sets DCD true.
C       28          C        endmac      test.UART0.DCD
C       29
C       30          C        test.UART0.DSR macro
C       31          C        scope
C       32          C        IN0 a, (UART0_MSR)          ; Get CTS status & test if CTS is ready..
C       33          C        BIT 5,A          ; Test if other end sets CTS true, they are ready to receive.
C       34          C        endmac      test.UART0.DSR
C       35
C       36          C        test.UART0.RI macro
C       37          C        scope
C       38          C        IN0 a, (UART0_MSR)          ; Get Ring Indicator & test.
C       39          C        BIT 6,A          ; Test RI.
C       40          C        endmac      test.UART0.RI
C       41
C       42          C        test.UART0.RX macro
C       43          C        scope
C       44          C        IN0 A, (UART0_LSR)          ; Char waiting?
C       45          C        AND UART_DR          ; 1 or TRUE if yes.
C       46          C        endmac      test.UART0.RX
C       47
C       48          C        test.UART0.TX macro
C       49          C        scope          ; Test UART 1 if xmitter buffer empty and ready.
C       50          C        IN0 A, (UART0_LSR)
C       51          C        AND UART_THRE          ; Returning a NE or <> 0 means ready, 1 means ready.
C       52          C        endmac      test.UART0.TX
C        0          C        include "DD_language_UART1.s"
C        1          C        ; UART 1 macros.
C        2
C        3
C        4          C        reset.UART1.DTRmacro
C        5          C        scope
C        6          C        in0 a, (UART1_MCTL)
C        7          C        and 0FEh          ; Strip out bit 0 making reset.
C        8          C        OUT0 (UART1_MCTL),A          ; UART1 tell other end we are ready.
C        9          C        endmac      reset.UART1.DTR
C       10
C       11          C        reset.UART1.RTSmacro
C       12          C        scope
C       13          C        in0 a, (UART1_MCTL)
C       14          C        and 0FDh          ; Strip out bit 0 making reset.

```



\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYSRES - operating system resident code. &gt;

```

PC      Object      I  Line      Source ..\..\TRSDOS_GLOBAL\macros\DD_language_macros\DD_language_UART1.s
C      15          OUT0 (UART1_MCTL),A          ; UART1 tell other end we are ready.
C      16          endmac      reset.UART1.RTS
C      17      set.UART1.DTR macro
C      18          scope
C      19          in0  a,(UART1_MCTL)          ; Get current register status.
C      20          or   1          ; Set bit 0 to 1.
C      21          OUT0 (UART1_MCTL),A          ; UART1 tell other end we are ready.
C      22          endmac      set.UART1.DTR
C      23
C      24      set.UART1.MCTL macro payload
C      25          scope
C      26          LD    A,payload          ; Get payload to write in MCTL register.
C      27          OUT0 (UART1_MCTL),A          ; UART1 tell other end we are ready.
C      28          endmac      set.UART1.MCTL
C      29
C      30      set.UART1.RTS macro
C      31          scope
C      32          in0  a,(UART1_MCTL)          ; Get current register status.
C      33          or   2          ; Set bit 0 to 1.
C      34          OUT0 (UART1_MCTL),A          ; UART1 tell other end we are ready.
C      35          endmac      set.UART1.RTS
C      36
C      37      test.UART1.CTS macro wait_loop          ; Test CTS signal on UART 1.
C      38          scope
C      39          IN0  a,(UART1_MSR)          ; Get CTS status & test if CTS is ready..
C      40          BIT  4,A          ; If other end sets CTS true, they are ready to receive.
C      41          jr   z,wait_loop          ; Loop until ready.
C      42          endmac      test.UART1.CTS
C      43
C      44      test.UART1.DCD macro
C      45          scope
C      46          IN0  a,(UART1_MSR)          ; Get DCD status & test if DCD is ready..
C      47          BIT  7,A          ; Test if other end sets DCD true.
C      48          endmac      test.UART1.DCD
C      49
C      50      test.UART1.DSR macro
C      51          scope
C      52          IN0  a,(UART1_MSR)          ; Get CTS status & test if CTS is ready..
C      53          BIT  5,A          ; Test if other end sets CTS true, they are ready to receive.
C      54          endmac      test.UART1.DSR
C      55
C      56      test.UART1.RI macro
C      57          scope
C      58          IN0  a,(UART1_MSR)          ; Get Ring Indicator & test.
C      59          BIT  6,A          ; Test RI.
C      60          endmac      test.UART1.RI
C      61
C      62      test.UART1.RX macro wait_loop
C      63          scope
C      64          IN0  A,(UART1_LSR)          ; Char waiting?
C      65          AND  UART_DR          ; 1 or TRUE if yes.
C      66          jr   z,wait_loop          ; Loop if not ready.
C      67          endmac      test.UART1.RX
C      68
C      69      test.UART1.TX macro wait_loop
C      70          scope          ; Test UART 1 if xmitter buffer empty and ready.
C      71          IN0  A,(UART1_LSR)
C      72          AND  UART_THRE          ; Returning a NE or <> 0 means ready, 0 means not ready.
C      73          jr   z,wait_loop
C      74          endmac      test.UART1.TX

```

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYSRES - operating system resident code. &gt;

PC	Object	I	Line	Source
		C	0	include "DD_language_VOLUMES.s"
		C	1	check.VOLUME           MACRO           drv,onerror
		C	2	ld           c,drv
		C	3	SVC           svc_ckdrv
		C	4	call          nz,onerror
		C	5	endmac        check.VOLUME
		C	0	include       "DD_language_files.s"
		C	1	LOAD       macro filespecs
		C	2	scope
		C	3	
		C	4	ld     de,filespecs
		C	5	SVC    svc_load
		C	6	endmac    LOAD
		C	7	
		C	8	RUNmacro filespecs
		C	9	scope
		C	10	
		C	11	ld     de,filespecs
		C	12	SVC    svc_run
		C	13	endmac    RUN
		A	16	
		B	0	include "trs80_macros.s"
		B	1	BANK           MACRO       num
		B	2	ld       c,num
		B	3	ld       b,0
		B	4	SVC       svc_zbank
		B	5	ENDMAC       BANK
		B	6	
		B	7	CKDRIVE       MACRO       drv,onerror
		B	8	ld       c,drv
		B	9	SVC       svc_ckdrv
		B	10	call      nz,onerror
		B	11	endmac     CKDRIVE
		B	12	
		B	13	DCINIT:       MACRO       drive,inierr
		B	14	ld       c,drive
		B	15	SVC       svc_dcinit
		B	16	jp       nz,inierr
		B	17	ENDMAC     DCINIT
		B	18	
		B	19	DCRESET       MACRO       drive
		B	20	ld       c,drive
		B	21	SVC       svc_dcres
		B	22	ENDMAC     DCRESET
		B	23	
		B	24	DCSTATUS   MACRO       drive,onerror
		B	25	ld       c,drive
		B	26	SVC       svc_dcstat
		B	27	jp       nz,onerror
		B	28	ENDMAC     DCSTATUS
		B	29	
		B	30	DISABLEVIDEO   MACRO
		B	31	CALL       DIS_DO_RAM
		B	32	ENDMAC
		B	33	
		B	34	ENABLEVIDEO    MACRO
		B	35	CALL       ENA_DO_RAM
		B	36	ENDMAC
		B	37	
		B	38	IPL       MACRO

\*\*\*\*\* TRSDOS \*\*\*\*\*  
 < SYSRES - operating system resident code. >

PC	Object	I	Line	Source	...	...	...	...
		B	39	SVC	svc_ipl			
		B	40	ENDMAC	IPL			
		B	41					
		B	42					
		B	43	MSG	MACRO	dat,device		
		B	44		LD	DE,device		
		B	45		LD	HL,dat		
		B	46		SVC	svc_msg		
		B	47		ENDMAC			
		B	48					
		B	49	PAUSE:	MACRO	wait		
		B	50		ld	bc,wait		
		B	51		SVC	svc_pause		
		B	52		ENDMAC			
		B	53					
		B	54	PRT.msg:	MACRO	mess		
		B	55		ld	hl,mess		
		B	56		SVC	svc_printz		
		B	57		ENDMAC	PRT.msg		
		B	58					
		B	59	PRT.char:	MACRO	txt		
		B	60		ld	c,txt		
		B	61		SVC	svc_prt		
		B	62		ENDMAC	PRT.char		
		B	63					
		B	64	READ.dir	macro	fdec,drv,onerror		; Read directory entry code, on drive, if error got
		B	65		LD	a,fdec		
		B	66		ld	b,a		
		B	67		ld	c,drv		
		B	68		SVC	svc_dirrd		
		B	69		jp	nz,onerror		
		B	70		endmac	READ.dir		
		B	71					
		B	72	READ.hdr	MACRO	buffer,drive		
		B	73		LD	C,drive		
		B	74		LD	HL,buffer		
		B	75		SVC	svc_rdhdr		
		B	76		ENDMAC			
		B	77					
		B	78	READ.sec	MACRO	buff,drive,cylinder,sector		
		B	79		LD	HL,buff		
		B	80		LD	D,cylinder		
		B	81		LD	E,sector		
		B	82		LD	C,drive		
		B	83		SVC	svc_rdsec		;Read sector SVC049.
		B	84		ENDMAC			
		B	85					
		B	86	READ.ssc	MACRO	buffer,drive,cylinder,sector		
		B	87					
		B	88					
		B	89					
		B	90					
		B	91					
		B	92					
		B	93					
		B	94					
		B	95					
		B	96					
		B	97					
		B	98					

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYSRES - operating system resident code. &gt;

PC	Object	I	Line	Source	..\\..\\TRSDOS_GLOBAL\\macros\\trs80_macros.s
		B	99		
		B	100	; Exit: A Passes the error return code if an error is encountered.	
		B	101	; Z Set if no error is encountered.	
		B	102		
		B	103		
		B	104	ld	hl,buffer
		B	105	ld	d,cylinder
		B	106	ld	e,sector
		B	107	ld	c,drive
		B	108	SVC	svc_rdssc ; Read a system sector, SVC085.
		B	109	ENDMAC	
		B	110		
		B	111	RESELECT	MACRO drive
		B	112	ld	c,drive
		B	113	SVC	svc_rslct
		B	114	ENDMAC	
		B	115		
		B	116	RESTORE0	MACRO drive
		B	117	ld	c,drive
		B	118	SVC	svc_rstor
		B	119	ENDMAC	
		B	120		
		B	121	SVC	MACRO num
		B	122	ld	a,num
		B	123	restart28	
		B	124	ENDMAC	SVC
		B	125		
		B	126	SCANKI	MACRO ;Scan KI device.
		B	127	SVC	svc_kbd
		B	128	ENDMAC	
		B	129		
		B	130	SEEKCYL	MACRO drive,cylinder ; Seek to cylinder on drive.
		B	131	ld	c,drive
		B	132	ld	d,cylinder
		B	133	SVC	svc_seekcyl
		B	134	ENDMAC	
		B	135		
		B	136	SELECT	MACRO drive,onerror ; Select drive.
		B	137	ld	c,drive
		B	138	SVC	svc_slct
		B	139	jp	nz,onerror
		B	140	ENDMAC	
		B	141		
		B	142		
		B	143	SOUND:	MACRO tone,duration
		B	144	ld	b,duration tone
		B	145	SVC	svc_sound
		B	146	ENDMAC	
		B	147		
		B	148	STEPIN	MACRO drive
		B	149	ld	c,drive
		B	150	SVC	svc_stepi
		B	151	ENDMAC	
		B	152		
		B	153	VIEWDIR	MACRO drive,buffer ; Send directory of drive to *do.
		B	154	ld	hl,buffer
		B	155	ld	b,0 ; Display dir on *do.
		B	156	ld	c,drive
		B	157	SVC	svc_dodir
		B	158	ENDMAC	VIEWDIR

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYSRES - operating system resident code. &gt;

```

PC      Object      I  Line      Source ..\..\TRSDOS_GLOBAL\macros\trs80_macros.s
B      159
B      160      WRITESEC  MACRO buffer,drive,cylinder,sector      ; Write disc sector.
B      161              ld      hl,buffer
B      162              ld      c,drive
B      163              ld      d,cylinder
B      164              ld      e,sector
B      165              SVC      svc_wrsec
B      166              ENDMAC      WRITESEC
B      167
B      168      WRITESSC  MACRO buffer,drive,cylinder,sector
B      169
B      170      ; WRSSC svc_54 will pass a function code 14 to a disk driver.
B      171      ; It is used to write a system sector (directory cylinder).
B      172      ; Disk controller must support IBM Data Address Mark convention.
B      173      ; Controller command should denote "deleted data mark", or X'F8' in lieu of standard data mark (X'F
B      174      ; This distinct mark is used in @RDSEC command to detect presence of a system (directory) sector.
B      175      ; Other than this Data Address Mark variation, @WRSSC is the same as @WRSEC.
B      176      ; DOS will use @WRSSC for all writes to a directory cylinder.
B      177
B      178      ; Registers Affected: AF [Note: DOS saves BC, IY; drivers should save any other registers they use]
B      179
B      180      ; Entry:
B      181
B      182      ; C Contains the logical drive number.
B      183      ; D Contains the number of the cylinder to write.
B      184      ; E Contains the number of the sector to write.
B      185      ; HL A pointer to the buffer containing the sector of data.
B      186
B      187      ; Exit.
B      188
B      189      ; A Will contain the error code if an error was encountered.
B      190      ; Z Set if the operation was successful.
B      191
B      192              ld      hl,buffer
B      193              ld      c,drive
B      194              ld      d,cylinder
B      195              ld      e,sector
B      196              SVC      svc_wrssc
B      197              ENDMAC      WRITESSC
B      0      include "ez80_macros.s"
B      1      RTC.lock  macro
B      2              ld      a,00100000b      ; Lock clock & disable interrupts.
B      3              OUT0    (RTC_CTRL),a      ; Execute.
B      4              ENDMAC      RTC.lock
B      5
B      6      RTC.interrupt.true  macro
B      7              ld      a,10100000b      ; Enable interrupts & lock clock.
B      8              OUT0    (RTC_CTRL),a      ; Execute.
B      9              ENDMAC      RTC.interrupt.true
B      10
B      11      RTC.unlock  macro
B      12              ld      a,00100001b      ; Unlock & disable interrupts.
B      13              OUT0    (RTC_CTRL),a
B      14              endmac      RTC.unlock
B      15
B      16      RTC.HOUR.get  macro destination
B      17              in0      a,(RTC_HRS)      ;
B      18              ld      destination,a
B      19              ENDMAC
B      20

```

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYSRES - operating system resident code. &gt;

PC	Object	I	Line	Source	...
		B	21	RTC.HOUR.put	macro source
		B	22	ld	a,source
		B	23	out0	(RTC_HRS),a ;
		B	24	ENDMAC	
		B	25		
		B	26	RTC.MIN.get	macro destination
		B	27	in0	a,(RTC_MIN) ;
		B	28	ld	destination,a
		B	29	ENDMAC	
		B	30		
		B	31	RTC.MIN.put	macro source
		B	32	ld	a,source
		B	33	out0	(RTC_MIN),a ;
		B	34	ENDMAC	
		B	35		
		B	36	RTC.SEC.get	macro destination
		B	37	in0	a,(RTC_SEC) ;
		B	38	ld	destination,a
		B	39	ENDMAC	
		B	40		
		B	41	RTC.SEC.put	macro source
		B	42	ld	a,source
		B	43	out0	(RTC_SEC),a ;
		B	44	ENDMAC	
		B	45		
		B	46	RTC.STAT.get	macro destination
		B	47	in0	a,(RTC_CTRL)
		B	48	ld	destination,a
		B	49	ENDMAC	
		B	50		
		B	51	RTC.STAT.put	macro source
		B	52	ld	a,source
		B	53	out0	(RTC_CTRL),a
		B	54	endmac	
		B	55		
		B	56	RTC.CEN.get	macro destination
		B	57	in0	a,(RTC_CEN)
		B	58	or	a
		B	59	jr	nz,\$1
		B	60	ld	a,20h
		B	61	\$1	ld destination,a
		B	62	ENDMAC	
		B	63		
		B	64	RTC.CEN.put	macro source
		B	65	ld	a,source
		B	66	out0	(RTC_CEN),A
		B	67	ENDMAC	
		B	68		
		B	69		
		B	70	RTC.YEAR.get	macro destination
		B	71	in0	a,(RTC_YR)
		B	72	or	a
		B	73	jr	nz,\$1
		B	74	ld	a,23h
		B	75	\$1	ld destination,a
		B	76	ENDMAC	
		B	77		
		B	78	RTC.YEAR.put	macro source
		B	79	ld	a,source
		B	80	out0	(RTC_YR),A ;

\*\*\*\*\* TRSDOS \*\*\*\*\*  
 < SYSRES - operating system resident code. >

PC	Object	I	Line	Source	...
		B	81	ENDMAC	
		B	82		
		B	83	RTC.DOM.get	macro destination
		B	84	in0	a, (RTC_DOM)
		B	85	or	a
		B	86	jr	nz, \$1
		B	87	ld	a, 20h
		B	88	\$1	ld destination, a
		B	89	ENDMAC	
		B	90		
		B	91	RTC.DOM.put	macro source
		B	92	ld	a, source
		B	93	out0	(RTC_DOM), A
		B	94	endmac	
		B	95		
		B	96	RTC.DOW.get	macro destination
		B	97	in0	a, (RTC_DOW)
		B	98	or	a
		B	99	jr	nz, \$1
		B	100	ld	a, 02h
		B	101	\$1	ld destination, a
		B	102	ENDMAC	
		B	103		
		B	104	RTC.DOW.put	macro source
		B	105	ld	a, source
		B	106	out0	(RTC_DOW), A
		B	107	endmac	
		B	108		
		B	109		
		B	110	RTC.MON.get	macro destination
		B	111	in0	a, (RTC_MON)
		B	112	or	a
		B	113	jr	nz, \$1
		B	114	ld	a, 07h
		B	115	\$1	ld destination, a
		B	116	ENDMAC	
		B	117		
		B	118	RTC.MON.put	macro source
		B	119	ld	a, source
		B	120	out0	(RTC_MON), A
		B	121	ENDMAC	
		B	0	include	"trsos_macros.s"
		B	1	INPUT_U.str	MACRO buffer, max ; Buffer and MAX chars to input.
		B	2	scope	
		B	3	ld	hl, buffer
		B	4	ld	b, max ; Max number bytes to input.
		B	5	jr	\$1 ; Branch to beginning.
		B	6		
		B	7	\$2:	ld c, a ; Output BS just received.
		B	8	CALL	TXUART0 ; Send it.
		B	9	ld	c, SP.asc ; Now space out char we are deleting.
		B	10	CALL	TXUART0
		B	11	ld	c, BS.asc ; Backspace again to new location.
		B	12	CALL	TXUART0
		B	13	dec	hl ; Back pointer in buffer up by -1.
		B	14	pop	bc ; Recover B counter.
		B	15	INC	B
		B	16	cp	max
		B	17	jr	nc, \$1
		B	18	ld	b, max

\*\*\*\*\* TRSDOS \*\*\*\*\*  
 < SYSRES - operating system resident code. >

PC	Object	I	Line	Source	...
		B	19		
		B	20	\$1: push bc	; Save counter as other routines may clobber it.
		B	21	CALL getchar	; Using UART NOT TRSDOS get one byte from UART0.
		B	22	CP BS.asc	; Is is backspace?
		B	23	jr z,\$2	; If yes gp to backspace routine above.
		B	24	ld (hl),a	; Put char in buff.
		B	25	inc hl	; Bump pointer to next place in buffer.
		B	26	cp CR.asc	; Was it a CR?
		B	27	jr z,\$4	; Exit input as terminalted by CR.
		B	28	ld c,a	
		B	29	CALL TXUART0	; Output character.
		B	30	\$3 pop bc	; Recover counter.
		B	31	djnz \$1	; Counter = counter - 1. If counter <> 0 branch \$1.
		B	32	xor a	; All is well set Z flag.
		B	33	jr \$5	; Finished, exit.
		B	34	\$4: or a	; Make it non zero.
		B	35	\$5: endmac	INPUT_U.str ; MACRO is done.
		B	36		
		B	37	KEY_U.get: MACRO dest	
		B	38	call getchar	
		B	39	IFMA 1	
		B	40	ld dest,a	
		B	41	else	
		B	42	ENDIF	
		B	43	ENDMAC	KEY_U.get
		A	20		
		B	0	include "equates-global-lowcore.s"	; Global equates defined by ZDS.
		B	1	; Hardware vectors.	
		B	2		
		B	3	; Memory Routines.	
		B	4		
		B	5	xdef BAR\$	; 'BAR\$ Symbol is declared global to external modules..'
		B	6	xdef BUR\$	; 'BUR\$ Symbol is declared global to external modules..'
		B	7	xdef ZHIGH\$	; 'ZHIGH\$ Symbol is declared global to external modules..'
		B	8	xdef LBANK\$	; 'LBANK\$ Symbol is declared global to external modules..'
		B	9	xdef PHIGH\$	; 'PHIGH\$ Symbol is declared global to external modules..'
		B	10		
		B	11	; Video modules.	
		B	12		
		B	13	xdef ADDR_2_ROWCOL	; 'ADDR_2_ROWCOL Symbol is declared global to external modules..'
		B	14	xdef GET_z_ROWCOL	; 'GET_z_ROWCOL Symbol is declared global to external modules..'
		B	15	xdef DIS_DO_RAM	; 'DIS_DO_RAM Symbol is declared global to external modules..'
		B	16	xdef DODATA\$	; 'DODATA\$ Symbol is declared global to external modules..'
		B	17	xdef DO_CONTROL	; 'DO_CONTROL Symbol is declared global to external modules..'
		B	18	xdef DO_DSPCHAR	; 'DO_DSPCHAR Symbol is declared global to external modules..'
		B	19	xdef DO_INVERT_DIS	; 'DO_INVERT_DIS Symbol is declared global to external modules..'
		B	20	xdef DO_INVERT_ENA	; 'DO_INVERT_ENA Symbol is declared global to external modules..'
		B	21	xdef DO_INVERT_OFF	; 'DO_INVERT_OFF Symbol is declared global to external modules..'
		B	22	xdef DO_MASK	; 'DO_MASK Symbol is declared global to external modules..'
		B	23	xdef DO_RET	; 'DO_RET Symbol is declared global to external modules..'
		B	24	xdef DO_RET1	; 'DO_RET1 Symbol is declared global to external modules..'
		B	25	xdef DO_SCROLL	; 'DO_SCROLL Symbol is declared global to external modules..'
		B	26	xdef DO_TABS	; 'DO_TABS Symbol is declared global to external modules..'
		B	27	xdef ENADIS_DO_RAM	; 'ENADIS_DO_RAM Symbol is declared global to external modules..'
		B	28	xdef PUT_z_ROWCOL	; 'PUT_z_ROWCOL Symbol is declared global to external modules..'
		B	29	xdef ROWCOL_2_ADDR	; 'ROWCOL_2_ADDR Symbol is declared global to external modules..'
		B	30	xdef z_VDCTL	; 'z_VDCTL Symbol is declared global to external modules..'
		B	31	xdef zVDCTL	; 'zVDCTL Symbol is declared global to external modules..'
		B	32	xdef zVDCTL3	; 'zVDCTL3 Symbol is declared global to external modules..'
		B	33		



```
< SYSRES - operating system resident code. >
```

PC	Object	I	Line	Source	...	TRSDOS_GLOBAL\equates-global-lowcore.s
		B	34	; Keyboard module/routines.		
		B	35			
		B	36			
		B	37	xdef	TYPHK\$	; 'TYPHK\$ Symbol is declared global to external modules..'
		B	38	xdef	TYPTSK\$	; 'TYPTSK\$ Symbol is declared global to external modules..'
		B	39	xdef	ZKBD	; 'ZKBD Symbol is declared global to external modules..'
		B	40	xdef	ZKEY	; 'ZKEY Symbol is declared global to external modules..'
		B	41	xdef	ZKEYIN	; 'ZKEYIN Symbol is declared global to external modules..'
		B	42	xdef	ZKITSK	; 'ZKITSK Symbol is declared global to external modules..'
		B	43	xdef	ZKLTSK	; 'ZKLTSK Symbol is declared global to external modules..'
		B	44	xdef	KIDATA\$	; 'KIDATA\$ Symbol is declared global to external modules..'
		B	45	xdef	BREAKz	; 'BREAK? Symbol is declared global to external modules..'
		B	46			
		B	47			
		B	48	; Test system flag table.		
		B	49			
		B	50	XDEF	PFLAG\$	
		B	51			
		B	52			
		B	53	; Function calls.		
		B	54			
		B	55			
		B	56	xdef	ZBANK	; 'ZBANK Symbol is declared global to external modules..'
		B	57	xdef	zspace.BYTEIO\$	; 'BYTEIO\$ Symbol is declared global to external modules..'
		B	58			
		B	59	xdef	ZCHNIO	; 'ZCHNIO Symbol is declared global to external modules..'
		B	60	xdef	ZCKBRKC	; 'ZCKBRKC Symbol is declared global to external modules..'
		B	61			
		B	62	xdef	DATEz	; 'ZDATE Symbol is declared global to external modules..'
		B	63	xdef	ZDIV16	; 'ZDIV16 Symbol is declared global to external modules..'
		B	64	xdef	ZDSP	; 'ZDSP Symbol is declared global to external modules..'
		B	65	xdef	ZDSPLY	; 'ZDSPLY Symbol is declared global to external modules..'
		B	66			
		B	67			
		B	68	xdef	ZGET	; 'ZGET Symbol is declared global to external modules..'
		B	69			
		B	70	xdef	ZHEX16	; 'ZHEX16 Symbol is declared global to external modules..'
		B	71	xdef	ZHEX8	; 'ZHEX8 Symbol is declared global to external modules..'
		B	72	xdef	ZHEXDEC	; 'ZHEXDEC Symbol is declared global to external modules..'
		B	73			
		B	74	xdef	ZJCL	; 'ZJCL Symbol is declared global to external modules..'
		B	75			
		B	76			
		B	77	xdef	ZLOGOT	; 'ZLOGOT Symbol is declared global to external modules..'
		B	78			
		B	79	xdef	ZMSG	; 'ZMSG Symbol is declared global to external modules..'
		B	80	xdef	ZMUL16	; 'ZMUL16 Symbol is declared global to external modules..'
		B	81			
		B	82	xdef	ZPRINT	; 'ZPRINT Symbol is declared global to external modules..'
		B	83	xdef	ZPRT	; 'ZPRT Symbol is declared global to external modules..'
		B	84	xdef	ZPUT	; 'ZPUT Symbol is declared global to external modules..'
		B	85	xdef	PUT_z	; 'PUT_@ Symbol is declared global to external modules..'
		B	86			
		B	87			
		B	88	xdef	ZWEOF	; 'ZWEOF Symbol is declared global to external modules..'
		B	89	xdef	ZWHERE	; 'ZWHERE Symbol is declared global to external modules..'
		B	90	xdef	ZWRITE	; 'ZWRITE Symbol is declared global to external modules..'
		B	91	xdef	ZWRSEC	; 'ZWRSEC Symbol is declared global to external modules..'
		B	92	xdef	ZWRSSC	; 'ZWRSSC Symbol is declared global to external modules..'
		B	93	xdef	ZWRTRK	; 'ZWRTRK Symbol is declared global to external modules..'

```
< SYSRES - operating system resident code. >
```

PC	Object	I	Line	Source	...	...	TRSDOS_GLOBAL\equates\equates-global-lowcore.s		
		B	94						
		B	95						
		B	96				; File control blocks.		
		B	97						
		B	98		xdef	CFGFCB\$		; 'CFGFCB\$	Symbol is declared global to external modules..'
		B	99		xdef	CFCB\$		; 'CFCB\$	Symbol is declared global to external modules..'
		B	100		xdef	JFCB\$		; 'JFCB\$	Symbol is declared global to external modules..'
		B	101		xdef	SFCB\$		; 'SFCB\$	Symbol is declared global to external modules..'
		B	102						
		B	103				; Device control blocks.		
		B	104						
		B	105		xdef	DODCB\$		; 'DODCB\$	Symbol is declared global to external modules..'
		B	106		xdef	JCLCB\$		; 'JCLCB\$	Symbol is declared global to external modules..'
		B	107		xdef	JDCB\$		; 'JDCB\$	Symbol is declared global to external modules..'
		B	108		xdef	JLDCB\$		; 'JLDCB\$	Symbol is declared global to external modules..'
		B	109		xdef	KIDCB\$		; 'KIDCB\$	Symbol is declared global to external modules..'
		B	110		xdef	PRDCB\$		; 'PRDCB\$	Symbol is declared global to external modules..'
		B	111		xdef	SIDCB\$		; 'SIDCB\$	Symbol is declared global to external modules..'
		B	112		xdef	SODCB\$		; 'SODCB\$	Symbol is declared global to external modules..'
		B	113		xdef	U0DCB\$		; 'S1DCB\$	Symbol is declared global to external modules..'
		B	114						
		B	115				; Drive Control Tables.		
		B	116						
		B	117						
		B	118		xdef	DCT\$		; 'DCT\$	Symbol is declared global to external modules..'
		B	119		XDEF	DCT0\$			
		B	120		XDEF	DCT1\$			
		B	121		XDEF	DCT6\$			
		B	122		XDEF	FDCDVR			
		B	123						
		B	124				; Machine code blocks.		
		B	125						
		B	126		xdef	BRKVEC\$		; 'BRKVEC\$	Symbol is declared global to external modules..'
		B	127		XDEF	zNMI			
		B	128		XDEF	zRST00			
		B	129		XDEF	zRST38			
		B	130		xdef	FDDINT\$		; 'FDDINT\$	Symbol is declared global to external modules..'
		B	131		xdef	FEMSK\$		; 'FEMSK\$	Symbol is declared global to external modules..'
		B	132		xdef	WRINT\$		; 'WRINT\$	Symbol is declared global to external modules..'
		B	133		xdef	INTIM\$		; 'INTIM\$	Symbol is declared global to external modules..'
		B	134		xdef	INTMSK\$		; 'INTMSK\$	Symbol is declared global to external modules..'
		B	135		xdef	INTVC\$		; 'INTVC\$	Symbol is declared global to external modules..'
		B	136						
		B	137				; OS data blocks.		
		B	138						
		B	139		xdef	DAYTBL\$		; 'DAYTBL\$	Symbol is declared global to external modules..'
		B	140		xdef	INBUF\$		; 'INBUF\$	Symbol is declared global to external modules..'
		B	141		xdef	MAXDAY\$		; 'MAXDAY\$	Symbol is declared global to external modules..'
		B	142		xdef	OSRLS\$		; 'OSRLS\$	Symbol is declared global to external modules..'
		B	143		xdef	OSVER\$		; 'OSVER\$	Symbol is declared global to external modules..'
		B	144		xdef	OVRLY\$		; 'OVRLY\$	Symbol is declared global to external modules..'
		B	145		xdef	PAKNAM\$		; 'PAKNAM\$	Symbol is declared global to external modules..'
		B	146		xdef	SVCRET\$		; 'SVCRET\$	Symbol is declared global to external modules..'
		B	147		xdef	SVCTAB\$		; 'SVCTAB\$	Symbol is declared global to exter
		B	148		XDEF	FLGTAB\$			
		B	149		XDEF	SFLAG\$			
		B	150		xdef	USTOR\$		; 'USTOR\$	Symbol is declared global to external modules..'
		B	151		xdef	zUSA		; 'zUSA	Symbol is declared global to external modules..'
		B	152		xdef	PCSAVE\$		; 'PCSAVE\$	Symbol is declared global to external modules..'
		B	153		xdef	PDRV\$		; 'PDRV\$	Symbol is declared global to external modules..'

```
< SYSRES - operating system resident code. >
```

PC	Object	Line	Source	Symbol	Symbol is declared global to external modules..'
		B 154			
		B 155			
		B 156			
		B 157	XDEF	ZLOAD	
		B 158	XDEF	ZICNFG	
		B 159	XDEF	ZCMNDI	
		B 160	xdef	CASHK\$	; 'CASHK\$ Symbol is declared global to external modules..'
		B 161	xdef	CKOPENZ	; 'CKOPEN@ Symbol is declared global to external modules..'
		B 162	xdef	CYL_GRN	; 'CYL_GRN Symbol is declared global to external modules..'
		B 163	XDEF	zCTL	
		B 164	xdef	DATE\$	; 'DATE\$ Symbol is declared global to external modules..'
		B 165	xdef	DBGSV\$	; 'DBGSV\$ Symbol is declared global to external modules..'
		B 166	xdef	DCBKL\$	; 'DCBKL\$ Symbol is declared global to external modules..'
		B 167	xdef	DSKTYP\$	; 'DSKTYP\$ Symbol is declared global to external modules..'
		B 168	xdef	DTPMT\$	; 'DTPMT\$ Symbol is declared global to external modules..'
		B 169	xdef	DVREND\$	; 'DVREND\$ Symbol is declared global to external modules..'
		B 170	xdef	DVRHI\$	; 'DVRHI\$ Symbol is declared global to external modules..'
		B 171	Xdef	EXTDBG\$	; 'EXTDBG\$ Symbol is declared global to external modules..'
		B 172	xdef	HKRES\$	; 'HKRES\$ Symbol is declared global to external modules..'
		B 173	xdef	JRET\$	; 'JRET\$ Symbol is declared global to external modules..'
		B 174	xdef	LDRV\$	; 'LBANK\$ Symbol is declared global to external modules..'
		B 175	xdef	LSVC\$	; 'LSVC\$ Symbol is declared global to external modules..'
		B 176	xdef	COREMIN\$	; 'MINCOR\$\$ Symbol is declared global to external modules..'
		B 177	xdef	MONTBL\$	; 'MONTBL\$ Symbol is declared global to external modules..'
		B 178	xdef	OPREG_SV_AREA	; 'OPREG_SV_AREA Symbol is declared global to external modules..'
		B 179	xdef	OPREG_SV_PTR	; 'OPREG_SV_PTR\$ Symbol is declared global to external modules..'
		B 180	xdef	ORARETZ	; 'ORARET@ Symbol is declared global to external modules..'
		B 181	xdef	zPAUSE	; 'PAUSE@ Symbol is declared global to external modules..'
		B 182	xdef	PUTAZDE	; 'PUTA@DE Symbol is declared global to external modules..'
		B 183	xdef	RSTOR\$	; 'RSTOR\$ Symbol is declared global to external modules..'
		B 184	xdef	RWRITZ	; 'RWRIT@ Symbol is declared global to external modules..'
		B 185	xdef	SBUFF\$	; 'SBUFF\$ Symbol is declared global to external modules..'
		B 186	xdef	SETzEXEC	; 'SET@EXEC Symbol is declared global to external modules..'
		B 187	xdef	SET_SCROLL	; 'SET_SCROLL Symbol is declared global to external modules..'
		B 188	xdef	STACK\$	; 'STACK\$ Symbol is declared global to external modules..'
		B 189	xdef	SYSERR\$	; 'SYSERR\$ Symbol is declared global to external modules..'
		B 190	xdef	TCB\$	; 'TCB\$ Symbol is declared global to external modules..'
		B 191	xdef	TIME\$	; 'TIME\$ Symbol is declared global to external modules..'
		B 192	xdef	TIMER\$	; 'TIMER\$ Symbol is declared global to external modules..'
		B 193	xdef	TIMSL\$	; 'TIMSL\$ Symbol is declared global to external modules..'
		B 194	xdef	TMPMT\$	; 'TMPMT\$ Symbol is declared global to external modules..'
		B 195	xdef	TRACE_INT	; 'TRACE_INT Symbol is declared global to external modules..'
		B 196	xdef	ZERO\$	; 'ZERO\$ Symbol is declared global to external modules..'
		B 197	xdef	ZEROAZ	; 'ZEROA@ Symbol is declared global to external modules..'
		B 198	xdef	DzFBYT8	; 'D@FBYT8 Symbol is declared global to external modules..'
		B 199	xdef	KCKz	; 'KCK@ Symbol is declared global to external modules..'
		B 200	xdef	DCTBYT8z	; 'DCTBYT8@ Symbol is declared global to external modules..'
		B 201	xdef	DCTFLDz	; 'DCTFLD@ Symbol is declared global to external modules..'
		B 202			
		B 203	xdef	U1DVR	
		B 204	xdef	U0DVR	
		B 205	XDEF	TXUART0	
		B 206			
		B 207	xdef	net_volume	
		B 208			
		B 209	; TRS_OS associated xdefs.		
		B 210			
		B 211	XDEF	int_tmr1_cnt	
		B 212	XDEF	pump_switch_value	
		B 213	XDEF	pump_switch	

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYSRES - operating system resident code. &gt;

PC	Object	I	Line	Source
		B	214	XDEF GETDATE
		B	215	XDEF GETTIME
		B	216	XDEF ramdrv\$
		B	217	XDEF reboot
		B	218	XDEF bin2bcd
		B	219	
		B	220	; eZ80 associated xdef's.
		B	221	
		B	222	
		B	223	XDEF EMAC.RX.handler
		B	224	XDEF EMAC.TX..handler
		B	225	XDEF EMAC.SYS.handler
		B	226	XDEF PLL.handler
		B	227	XDEF FLASH.handler
		B	228	XDEF TIMER0.handler
		B	229	XDEF f91_TIMER1.handler
		B	230	XDEF TIMER2.handler
		B	231	XDEF TIMER3.handler
		B	232	XDEF STRAY.handler
		B	233	XDEF RTC.handler
		B	234	XDEF UART0.handler
		B	235	XDEF UART1.handler
		B	236	XDEF I2C.handler
		B	237	XDEF SPI.handler
		B	238	XDEF PORT.A.BIT0.handler
		B	239	XDEF PORT.A.BIT1.handler
		B	240	XDEF PORT.A.BIT2.handler
		B	241	XDEF PORT.A.BIT3.handler
		B	242	XDEF PORT.A.BIT4.handler
		B	243	XDEF PORT.A.BIT5.handler
		B	244	XDEF PORT.A.BIT6.handler
		B	245	XDEF PORT.A.BIT7.handler
		B	246	XDEF PORT.B.BIT0.handler
		B	247	XDEF PORT.B.BIT1.handler
		B	248	XDEF PORT.B.BIT2.handler
		B	249	XDEF PORT.B.BIT3.handler
		B	250	XDEF PORT.B.BIT4.handler
		B	251	XDEF PORT.B.BIT5.handler
		B	252	XDEF PORT.B.BIT6.handler
		B	253	XDEF PORT.B.BIT7.handler
		B	254	XDEF PORT.C.BIT0.handler
		B	255	XDEF PORT.C.BIT1.handler
		B	256	XDEF PORT.C.BIT2.handler
		B	257	XDEF PORT.C.BIT3.handler
		B	258	XDEF PORT.C.BIT4.handler
		B	259	XDEF PORT.C.BIT5.handler
		B	260	XDEF PORT.C.BIT6.handler
		B	261	XDEF PORT.C.BIT7.handler
		B	262	XDEF PORT.D.BIT0.handler
		B	263	XDEF PORT.D.BIT1.handler
		B	264	XDEF PORT.D.BIT2.handler
		B	265	XDEF PORT.D.BIT3.handler
		B	266	XDEF PORT.D.BIT4.handler
		B	267	XDEF PORT.D.BIT5.handler
		B	268	XDEF PORT.D.BIT6.handler
		B	269	XDEF PORT.D.BIT7.handler
		B	270	XDEF f92_TIMER1.handler
		B	271	XDEF TIMER4.handler
		B	272	XDEF TIMER5.handler
		B	0	include "equates-global-xref.s"

; Global equates defined by ZDS.

\*\*\*\*\* TRSDOS \*\*\*\*\*  
< Equates. >

PC	Object	I	Line	Source	..\..\TRSDOS_GLOBAL\equates\equates-global-xref.s
		B	1	SUBTITLE	"< Equates. >"
		B	2	newpage	

\*\*\*\*\* TRSDOS \*\*\*\*\*  
 < Equates. >

PC	Object	I	Line	Source	...
		B	3	scope	
		B	4		
		B	5	; Global equates needed to assemble o/s, ;for assembly of system.	
		B	6		
		B	7	XREF	_SYS_CLK_SRC
		B	8	XREF	_SYS_CLK_FREQ
		B	9	XREF	_OSC_FREQ
		B	10	XREF	_OSC_FREQ_MULT
		B	11	XREF	__PLL_CTL0_INIT_PARAM
		B	12	XREF	__stack
		B	13	XREF	__CS0_LBR_INIT_PARAM
		B	14	XREF	__CS0_UBR_INIT_PARAM
		B	15	XREF	__CS0_CTL_INIT_PARAM
		B	16	XREF	__CS1_LBR_INIT_PARAM
		B	17	XREF	__CS1_UBR_INIT_PARAM
		B	18	XREF	__CS1_CTL_INIT_PARAM
		B	19	XREF	__CS2_LBR_INIT_PARAM
		B	20	XREF	__CS2_UBR_INIT_PARAM
		B	21	XREF	__CS2_CTL_INIT_PARAM
		B	22	XREF	__CS3_LBR_INIT_PARAM
		B	23	XREF	__CS3_UBR_INIT_PARAM
		B	24	XREF	__CS3_CTL_INIT_PARAM
		B	25	XREF	__CS0_BMC_INIT_PARAM
		B	26	XREF	__CS1_BMC_INIT_PARAM
		B	27	XREF	__CS2_BMC_INIT_PARAM
		B	28	XREF	__CS3_BMC_INIT_PARAM
		B	29	XREF	__FLASH_CTL_INIT_PARAM
		B	30	XREF	__FLASH_ADDR_U_INIT_PARAM
		B	31	XREF	__RAM_CTL_INIT_PARAM
		B	32	XREF	__RAM_ADDR_U_INIT_PARAM
		A	23		
		B	0	include "equires-eZ80.inc"	; General eZ80 equates needed in eZ80 family
		B	1	NEWPAGE	

PC	Object	I	Line	Source	..\\..\\TRSDOS_GLOBAL\\equates\\equates-eZ80.inc
		B	2	SUBTITLE	"< Common equates for eZ80 chip. >"
		B	3	scope	
		B	4		
		B	5		
		B	6	; UART MASK equates.	
		B	7		
	00000020	B	8	UART_THRE EQU 20h	; bit mask for okay to Tx.
	00000001	B	9	UART_DR EQU 01h	; bit mask for Rx'd char ready.
		B	10		
	000000C0	B	11	BRG0_DLR_L EQU 0C0h	; BRG 0 Divisor Latch Register, Low Byte.
	000000C1	B	12	BRG0_DLR_H EQU 0C1h	; BRG 0 Divisor Latch Register, High Byte.
		B	13		
	00000020	B	14	LCK_STATUS EQU %20	
		B	15		
		B	16	; * WDT	
		B	17		
	00000093	B	18	WDT_CTL: .equ %93	
	00000094	B	19	WDT_RR: .equ %94	
		B	20		
		B	21	; * PORT	
		B	22		
	0000009A	B	23	PB_DR: .equ %9A	
	0000009B	B	24	PB_DDR: .equ %9B	
	0000009C	B	25	PB_ALT1: .equ %9C	
	0000009D	B	26	PB_ALT2: .equ %9D	
	0000009E	B	27	PC_DR: .equ %9E	
	0000009F	B	28	PC_DDR: .equ %9F	
	000000A0	B	29	PC_ALT1: .equ %A0	
	000000A1	B	30	PC_ALT2: .equ %A1	
	000000A2	B	31	PD_DR: .equ %A2	
	000000A3	B	32	PD_DDR: .equ %A3	
	000000A4	B	33	PD_ALT1: .equ %A4	
	000000A5	B	34	PD_ALT2: .equ %A5	
		B	35		
		B	36	; * CS	
		B	37		
	000000A8	B	38	CS0_LBR: .equ %A8	
	000000A9	B	39	CS0_UBR: .equ %A9	
	000000AA	B	40	CS0_CTL: .equ %AA	
	000000AB	B	41	CS1_LBR: .equ %AB	
	000000AC	B	42	CS1_UBR: .equ %AC	
	000000AD	B	43	CS1_CTL: .equ %AD	
	000000AE	B	44	CS2_LBR: .equ %AE	
	000000AF	B	45	CS2_UBR: .equ %AF	
	000000B0	B	46	CS2_CTL: .equ %B0	
	000000B1	B	47	CS3_LBR: .equ %B1	
	000000B2	B	48	CS3_UBR: .equ %B2	
	000000B3	B	49	CS3_CTL: .equ %B3	
		B	50		
		B	51	; * RAMCTL	
		B	52		
	000000B4	B	53	RAM_CTL: .equ %B4	
	000000B4	B	54	RAM_CTL0: .equ %B4	
	000000B5	B	55	RAM_ADDR_U: .equ %B5	
		B	56		
		B	57	; * SPI	
		B	58		
	000000B8	B	59	SPI_BRG_L: .equ %B8	
	000000B9	B	60	SPI_BRG_H: .equ %B9	
	000000BA	B	61	SPI_CTL: .equ %BA	

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; Common equates for eZ80 chip. &gt;

PC	Object	I	Line	Source	...	...	...	...	...
	000000BB	B	62	SPI_SR:	.equ		%BB		
	000000BC	B	63	SPI_RBR:	.equ		%BC		
	000000BC	B	64	SPI_TSR:	.equ		%BC		
		B	65						
		B	66	;*	IR				
		B	67						
	000000BF	B	68	IR_CTL:	.equ		%BF		
		B	69						
		B	70	;*	UART0				
		B	71						
	000000C0	B	72	UART0_RBR:	.equ		%C0		
	000000C0	B	73	UART0_THR:	.equ		%C0		
	000000C0	B	74	UART0_BRG_L:	.equ		%C0		
	000000C1	B	75	UART0_IER:	.equ		%C1		
	000000C1	B	76	UART0_BRG_H:	.equ		%C1		
	000000C2	B	77	UART0_IIR:	.equ		%C2		
	000000C2	B	78	UART0_FCTL:	.equ		%C2		
	000000C3	B	79	UART0_LCTL:	.equ		%C3		
	000000C4	B	80	UART0_MCTL:	.equ		%C4		
	000000C5	B	81	UART0_LSR:	.equ		%C5		
	000000C6	B	82	UART0_MSR:	.equ		%C6		
	000000C7	B	83	UART0_SPR:	.equ		%C7		
		B	84						
		B	85	;*	I2C				
		B	86						
	000000C8	B	87	I2C_SAR:	.equ		%C8		
	000000C9	B	88	I2C_XSAR:	.equ		%C9		
	000000CA	B	89	I2C_DR:	.equ		%CA		
	000000CB	B	90	I2C_CTL:	.equ		%CB		
	000000CC	B	91	I2C_SR:	.equ		%CC		
	000000CC	B	92	I2C_CCR:	.equ		%CC		
	000000CD	B	93	I2C_SRR:	.equ		%CD		
		B	94						
		B	95						
		B	96						
		B	97	;*	UART1				
		B	98						
	000000D0	B	99	UART1_RBR:	.equ		%D0		
	000000D0	B	100	UART1_THR:	.equ		%D0		
	000000D0	B	101	UART1_BRG_L:	.equ		%D0		
	000000D1	B	102	UART1_IER:	.equ		%D1		
	000000D1	B	103	UART1_BRG_H:	.equ		%D1		
	000000D2	B	104	UART1_IIR:	.equ		%D2		
	000000D2	B	105	UART1_FCTL:	.equ		%D2		
	000000D3	B	106	UART1_LCTL:	.equ		%D3		
	000000D4	B	107	UART1_MCTL:	.equ		%D4		
	000000D5	B	108	UART1_LSR:	.equ		%D5		
	000000D6	B	109	UART1_MSR:	.equ		%D6		
	000000D7	B	110	UART1_SPR:	.equ		%D7		
		B	111						
		B	112						
		B	113	;*	CLK				
		B	114						
	000000DB	B	115	CLK_PPD1:	.equ		%DB		
	000000DC	B	116	CLK_PPD2:	.equ		%DC		
		B	117						
		B	118						
		B	119	;*	RTC Registers				
		B	120						
	000000E0	B	121	RTC_SEC:	.equ		%E0		



\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; Common equates for eZ80 chip. &gt;

```

PC      Object      I  Line      Source ..\..\TRSDOS_GLOBAL\equates\equates-eZ80.inc
000000E1      B    122      RTC_MIN:      .equ      %E1
000000E2      B    123      RTC_HRS:      .equ      %E2
000000E3      B    124      RTC_DOW:      .equ      %E3
000000E4      B    125      RTC_DOM:      .equ      %E4
000000E5      B    126      RTC_MON:      .equ      %E5
000000E6      B    127      RTC_YR:      .equ      %E6
000000E7      B    128      RTC_CEN:      .equ      %E7
000000E8      B    129      RTC_ASEC:      .equ      %E8
000000E9      B    130      RTC_AMIN:      .equ      %E9
000000EA      B    131      RTC_AHRS:      .equ      %EA
000000EB      B    132      RTC_ADOW:      .equ      %EB
000000EC      B    133      RTC_ACTRL:      .equ      %EC
000000ED      B    134      RTC_CTRL:      .equ      %ED
          B    135
          B    136      ;* CSBMC Registers
          B    137
000000F0      B    138      CS0_BMC:      .equ      %F0
000000F1      B    139      CS1_BMC:      .equ      %F1
000000F2      B    140      CS2_BMC:      .equ      %F2
000000F3      B    141      CS3_BMC:      .equ      %F3
          B    142
          B    143      ;* FLASH Registers
          B    144
000000F5      B    145      FLASH_KEY:      .equ      %F5
000000F6      B    146      FLASH_DATA:      .equ      %F6
000000F7      B    147      FLASH_ADDR_U:      .equ      %F7
000000F8      B    148      FLASH_CTRL:      .equ      %F8
000000F9      B    149      FLASH_FDIV:      .equ      %F9
000000FA      B    150      FLASH_PROT:      .equ      %FA
000000FB      B    151      FLASH_IRQ:      .equ      %FB
000000FC      B    152      FLASH_PAGE:      .equ      %FC
000000FD      B    153      FLASH_ROW:      .equ      %FD
000000FE      B    154      FLASH_COL:      .equ      %FE
000000FF      B    155      FLASH_PGCTL:      .equ      %FF
          B    156
          B    157      ; Programmable Reload Timers.
          B    158
00000092      B    159      TMR_ISS      equ      %92
          A    25
          B     0      include "equates-eZ80F91.inc"      ; eZ80f91 specific equates.
          B     1
          B     2      SUBTITLE      "< Common equates for eZ80F91. >"
          B     3      NEWPAGE

```

\*\*\*\*\* TRSDOS \*\*\*\*\*  
 < Common equates for eZ80F91. >

PC	Object	I	Line	Source	..\\..\\TRSDOS_GLOBAL\\equate\\equate-eZ80F91.inc
		B	4	scope	
		B	5		
		B	6	; eZ80F91.inc	
		B	7	; eZ80F91 Registers	
		B	8		
		B	9		
		B	10		
		B	11	; PRODUCT_ID	
		B	12		
00000000		B	13	ZDI_ID_L: .equ	%00
00000001		B	14	ZDI_ID_H: .equ	%01
00000002		B	15	ZDI_ID_REV: .equ	%02
		B	16		
		B	17	; INTERRUPT	
		B	18		
00000010		B	19	INT_P0: .equ	%10
00000011		B	20	INT_P1: .equ	%11
00000012		B	21	INT_P2: .equ	%12
00000013		B	22	INT_P3: .equ	%13
00000014		B	23	INT_P4: .equ	%14
00000015		B	24	INT_P5: .equ	%15
		B	25		
		B	26	; EMACC	
		B	27		
00000020		B	28	EMAC_TEST: .equ	%20
00000021		B	29	EMAC_CFG1: .equ	%21
00000022		B	30	EMAC_CFG2: .equ	%22
00000023		B	31	EMAC_CFG3: .equ	%23
00000024		B	32	EMAC_CFG4: .equ	%24
00000025		B	33	EMAC_STAD_0: .equ	%25
00000026		B	34	EMAC_STAD_1: .equ	%26
00000027		B	35	EMAC_STAD_2: .equ	%27
00000028		B	36	EMAC_STAD_3: .equ	%28
00000029		B	37	EMAC_STAD_4: .equ	%29
0000002A		B	38	EMAC_STAD_5: .equ	%2A
0000002B		B	39	EMAC_TPTV_L: .equ	%2B
0000002C		B	40	EMAC_TPTV_H: .equ	%2C
0000002D		B	41	EMAC_IPGT: .equ	%2D
0000002E		B	42	EMAC_IPGR1: .equ	%2E
0000002F		B	43	EMAC_IPGR2: .equ	%2F
00000030		B	44	EMAC_MAXF_L: .equ	%30
00000031		B	45	EMAC_MAXF_H: .equ	%31
00000032		B	46	EMAC_AFR: .equ	%32
00000033		B	47	EMAC_HTBL_0: .equ	%33
00000034		B	48	EMAC_HTBL_1: .equ	%34
00000035		B	49	EMAC_HTBL_2: .equ	%35
00000036		B	50	EMAC_HTBL_3: .equ	%36
00000037		B	51	EMAC_HTBL_4: .equ	%37
00000038		B	52	EMAC_HTBL_5: .equ	%38
00000039		B	53	EMAC_HTBL_6: .equ	%39
0000003A		B	54	EMAC_HTBL_7: .equ	%3A
0000003B		B	55	EMAC_MIIMGT: .equ	%3B
0000003C		B	56	EMAC_CTLD_L: .equ	%3C
0000003D		B	57	EMAC_CTLD_H: .equ	%3D
0000003E		B	58	EMAC_RGAD: .equ	%3E
0000003F		B	59	EMAC_FIAD: .equ	%3F
00000040		B	60	EMAC_PTMR: .equ	%40
00000041		B	61	EMAC_RST: .equ	%41
00000042		B	62	EMAC_TLBP_L: .equ	%42
00000043		B	63	EMAC_TLBP_H: .equ	%43

\*\*\*\*\* TRSDOS \*\*\*\*\*  
 < Common equates for eZ80F91. >

PC	Object	I	Line	Source	...	...	...
	00000044	B	64	EMAC_BP_L:	.equ	%44	
	00000045	B	65	EMAC_BP_H:	.equ	%45	
	00000046	B	66	EMAC_BP_U:	.equ	%46	
	00000047	B	67	EMAC_RHBP_L:	.equ	%47	
	00000048	B	68	EMAC_RHBP_H:	.equ	%48	
	00000049	B	69	EMAC_RRP_L:	.equ	%49	
	0000004A	B	70	EMAC_RRP_H:	.equ	%4A	
	0000004B	B	71	EMAC_BUFSZ:	.equ	%4B	
	0000004C	B	72	EMAC_IEN:	.equ	%4C	
	0000004D	B	73	EMAC_ISTAT:	.equ	%4D	
	0000004E	B	74	EMAC_PRSD_L:	.equ	%4E	
	0000004F	B	75	EMAC_PRSD_H:	.equ	%4F	
	00000050	B	76	EMAC_MIISTAT:	.equ	%50	
	00000051	B	77	EMAC_RWP_L:	.equ	%51	
	00000052	B	78	EMAC_RWP_H:	.equ	%52	
	00000053	B	79	EMAC_TRP_L:	.equ	%53	
	00000054	B	80	EMAC_TRP_H:	.equ	%54	
	00000055	B	81	EMAC_BLKSLFT_L:	.equ	%55	
	00000056	B	82	EMAC_BLKSLFT_H:	.equ	%56	
	00000057	B	83	EMAC_FDATA_L:	.equ	%57	
	00000058	B	84	EMAC_FDATA_H:	.equ	%58	
	00000059	B	85	EMAC_FFLAGS:	.equ	%59	
	00000059	B	86	EMAC_FLAGS:	.equ	%59	
		B	87				
		B	88	; * PLL			
		B	89				
	0000005C	B	90	PLL_DIV_L:	.equ	%5C	
	0000005D	B	91	PLL_DIV_H:	.equ	%5D	
	0000005E	B	92	PLL_CTL0:	.equ	%5E	
	0000005F	B	93	PLL_CTL1:	.equ	%5F	
		B	94				
		B	95	; * TIMER			
		B	96				
	00000060	B	97	TMR0_CTL:	.equ	%60	
	00000061	B	98	TMR0_IER:	.equ	%61	
	00000062	B	99	TMR0_IIR:	.equ	%62	
	00000063	B	100	TMR0_DR_L:	.equ	%63	
	00000063	B	101	TMR0_RR_L:	.equ	%63	
	00000064	B	102	TMR0_DR_H:	.equ	%64	
	00000064	B	103	TMR0_RR_H:	.equ	%64	
	00000065	B	104	TMR1_CTL:	.equ	%65	
	00000066	B	105	TMR1_IER:	.equ	%66	
	00000067	B	106	TMR1_IIR:	.equ	%67	
	00000068	B	107	TMR1_DR_L:	.equ	%68	
	00000068	B	108	TMR1_RR_L:	.equ	%68	
	00000069	B	109	TMR1_DR_H:	.equ	%69	
	00000069	B	110	TMR1_RR_H:	.equ	%69	
	0000006A	B	111	TMR1_CAP_CTL:	.equ	%6A	
	0000006B	B	112	TMR1_CAPA_L:	.equ	%6B	
	0000006C	B	113	TMR1_CAPA_H:	.equ	%6C	
	0000006D	B	114	TMR1_CAPB_L:	.equ	%6D	
	0000006E	B	115	TMR1_CAPB_H:	.equ	%6E	
	0000006F	B	116	TMR2_CTL:	.equ	%6F	
	00000070	B	117	TMR2_IER:	.equ	%70	
	00000071	B	118	TMR2_IIR:	.equ	%71	
	00000072	B	119	TMR2_DR_L:	.equ	%72	
	00000072	B	120	TMR2_RR_L:	.equ	%72	
	00000073	B	121	TMR2_DR_H:	.equ	%73	
	00000073	B	122	TMR2_RR_H:	.equ	%73	
	00000074	B	123	TMR3_CTL:	.equ	%74	

\*\*\*\*\* TRSDOS \*\*\*\*\*  
 < Common equates for eZ80F91. >

PC	Object	I	Line	Source	...	...	Source
	00000075	B	124	TMR3_IER:	.equ	%75	
	00000076	B	125	TMR3_IIR:	.equ	%76	
	00000077	B	126	TMR3_DR_L:	.equ	%77	
	00000077	B	127	TMR3_RR_L:	.equ	%77	
	00000078	B	128	TMR3_DR_H:	.equ	%78	
	00000078	B	129	TMR3_RR_H:	.equ	%78	
	0000007B	B	130	TMR3_CAP_CTL:	.equ	%7B	
	0000007C	B	131	TMR3_CAPA_L:	.equ	%7C	
	0000007D	B	132	TMR3_CAPA_H:	.equ	%7D	
	0000007E	B	133	TMR3_CAPB_L:	.equ	%7E	
	0000007F	B	134	TMR3_CAPB_H:	.equ	%7F	
	00000080	B	135	TMR3_OC_CTL1:	.equ	%80	
	00000081	B	136	TMR3_OC_CTL2:	.equ	%81	
	00000082	B	137	TMR3_OC0_L:	.equ	%82	
	00000083	B	138	TMR3_OC0_H:	.equ	%83	
	00000084	B	139	TMR3_OC1_L:	.equ	%84	
	00000085	B	140	TMR3_OC1_H:	.equ	%85	
	00000086	B	141	TMR3_OC2_L:	.equ	%86	
	00000087	B	142	TMR3_OC2_H:	.equ	%87	
	00000088	B	143	TMR3_OC3_L:	.equ	%88	
	00000089	B	144	TMR3_OC3_H:	.equ	%89	
		B	145				
		B	146	; * PWM			
		B	147				
	00000079	B	148	PWM_CTL1:	.equ	%79	
	0000007A	B	149	PWM_CTL2:	.equ	%7A	
	0000007B	B	150	PWM_CTL3:	.equ	%7B	
	0000007C	B	151	PWM0R_L:	.equ	%7C	
	0000007D	B	152	PWM0R_H:	.equ	%7D	
	0000007E	B	153	PWM1R_L:	.equ	%7E	
	0000007F	B	154	PWM1R_H:	.equ	%7F	
	00000080	B	155	PWM2R_L:	.equ	%80	
	00000081	B	156	PWM2R_H:	.equ	%81	
	00000082	B	157	PWM3R_L:	.equ	%82	
	00000083	B	158	PWM3R_H:	.equ	%83	
	00000084	B	159	PWM0F_L:	.equ	%84	
	00000085	B	160	PWM0F_H:	.equ	%85	
	00000086	B	161	PWM1F_L:	.equ	%86	
	00000087	B	162	PWM1F_H:	.equ	%87	
	00000088	B	163	PWM2F_L:	.equ	%88	
	00000089	B	164	PWM2F_H:	.equ	%89	
	0000008A	B	165	PWM3F_L:	.equ	%8A	
	0000008B	B	166	PWM3F_H:	.equ	%8B	
		B	167				
		B	168	; * PORT			
		B	169				
	00000096	B	170	PA_DR:	.equ	%96	
	00000097	B	171	PA_DDR:	.equ	%97	
	000000A6	B	172	PA_ALT0:	.equ	%A6	
	00000098	B	173	PA_ALT1:	.equ	%98	
	00000099	B	174	PA_ALT2:	.equ	%99	
	000000CF	B	175	PD_ALT0:	.equ	%CF	
	000000CE	B	176	PC_ALT0:	.equ	%CE	
	000000A7	B	177	PB_ALT0:	.equ	%A7	
		B	178				
		B	179				
	000000B6	B	180	MBIST_GPR:	.equ	%B6	
	000000B7	B	181	MBIST_EMUR:	.equ	%B7	
		B	0	INCLUDE "equates-ez80f92.inc"			; ez80f92 specific equates.
		B	1	;*****			

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; Common equates for eZ80F91. &gt;

PC	Object	I	Line	Source	..\\..\\TRSDOS_GLOBAL\\equate\\equate-ez80f92.inc
		B	2	;* eZ80F92.inc	
		B	3	;*	
		B	4	;* eZ80F92 Registers	
		B	5	;*	
		B	6	*****	
		B	7	;* Start eZ80F92 Include file	
		B	8		
		B	9	;* TIMER registers	
		B	10		
00000080		B	11	f92_TMR0_CTL: .equ %80	
00000081		B	12	f92_TMR0_DR_L: .equ %81	
00000081		B	13	f92_TMR0_RR_L: .equ %81	
00000082		B	14	f92_TMR0_DR_H: .equ %82	
00000082		B	15	f92_TMR0_RR_H: .equ %82	
00000083		B	16	f92_TMR1_CTL: .equ %83	
00000084		B	17	f92_TMR1_DR_L: .equ %84	
00000084		B	18	f92_TMR1_RR_L: .equ %84	
00000085		B	19	f92_TMR1_DR_H: .equ %85	
00000085		B	20	f92_TMR1_RR_H: .equ %85	
00000086		B	21	f92_TMR2_CTL: .equ %86	
00000087		B	22	f92_TMR2_DR_L: .equ %87	
00000087		B	23	f92_TMR2_RR_L: .equ %87	
00000088		B	24	f92_TMR2_DR_H: .equ %88	
00000088		B	25	f92_TMR2_RR_H: .equ %88	
00000089		B	26	f92_TMR3_CTL: .equ %89	
0000008A		B	27	f92_TMR3_DR_L: .equ %8a	
0000008A		B	28	f92_TMR3_RR_L: .equ %8a	
0000008B		B	29	f92_TMR3_DR_H: .equ %8b	
0000008B		B	30	f92_TMR3_RR_H: .equ %8b	
0000008C		B	31	f92_TMR4_CTL: .equ %8c	
0000008D		B	32	f92_TMR4_DR_L: .equ %8d	
0000008D		B	33	f92_TMR4_RR_L: .equ %8d	
0000008E		B	34	f92_TMR4_DR_H: .equ %8e	
0000008E		B	35	f92_TMR4_RR_H: .equ %8e	
0000008F		B	36	f92_TMR5_CTL: .equ %8f	
00000090		B	37	f92_TMR5_DR_L: .equ %90	
00000090		B	38	f92_TMR5_RR_L: .equ %90	
00000091		B	39	f92_TMR5_DR_H: .equ %91	
00000091		B	40	f92_TMR5_RR_H: .equ %91	
		B	41		
		B	42		
		B	43	;* End eZ80F92 inc file	
		B	44		
		B	45		
		A	28	; INCLUDE "equate-ez80f93.inc"	; eZ80f93 specific equates.
		A	29		
		B	0	include "equate-ASCII.s"	; Needed ASCII equates.
		B	1	SUBTITLE "< ASCII equates. >"	
		B	2	newpage	

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; ASCII equates. &gt;

PC	Object	I	Line	Source	...	...	Source	...	...	Source
		B	3				scope			
		B	4							
		B	5				< ASCII equates >			
		B	6							
	00000000	B	7	NULL.asc	EQU	0				; Null Filler.
	00000001	B	8	SOH.asc	EQU	1				; SOH -Start of Heading
	00000002	B	9	STX.asc	EQU	2				; Start of Text
	00000003	B	10	ETX.asc	EQU	3				; END OF TEXT.
	00000004	B	11	EOT.asc	EQU	4				; End of transmission.
	00000005	B	12	ENQ.asc	EQU	5				; Inquiry.
	00000006	B	13	ACK.asc	EQU	6				; Acknowledge.
	00000007	B	14	BELL.asc	EQU	7				; Ring terminal bell.
	00000008	B	15	BS.asc	EQU	8				; ASCII Backspace.
	00000009	B	16	HT.asc	EQU	9				; Horizontal Tebulation.
	0000000A	B	17	LF.asc	EQU	10				; Line Feed.
	0000000B	B	18	VT.asc	EQU	11				; Vertical Tabulation.
	0000000C	B	19	FF.asc	EQU	12				; Form Feed.
	0000000D	B	20	CR.asc	EQU	13				; Carriage Return.
	0000000E	B	21	SO.asc	EQU	14				; Shift Out.
	0000000F	B	22	SI.asc	EQU	15				; Shift In.
	00000010	B	23	DLE.asc	EQU	16				; Data link escape
	00000011	B	24	DC1.asc	EQU	17				; Device control 1
	00000012	B	25	DC2.asc	EQU	18				; Device control 2
	00000013	B	26	DC3.asc	EQU	19				; Device control 3
	00000014	B	27	DC4.asc	EQU	20				; Device control 4
	00000015	B	28	NAK.asc	EQU	21				; Negative acknowledge
	00000016	B	29	SYN.asc	EQU	22				; Synchronous idle
	00000017	B	30	ETB.asc	EQU	23				; End of transmission block
	00000018	B	31	CAN.asc	EQU	24				; Cancel
	00000019	B	32	EM.asc	EQU	25				; End of medium
	0000001A	B	33	SUB.asc	EQU	26				; Substitute
	0000001B	B	34	ESC.asc	EQU	27				; Escape.
	0000001C	B	35	FS.asc	EQU	28				; File separator
	0000001D	B	36	GS.asc	equ	29				; Group seperator (TRS-80 CLEAR key).
	0000001E	B	37	RS.asc	EQU	30				; Record separator
	0000001F	B	38	US.asc	EQU	31				; Unit separator.
	00000020	B	39	SP.asc	EQU	32				; Space.
	0000007F	B	40	DEL.asc	EQU	127				; Delete.
	0000007F	B	41	RUBOUT.asc	EQU	127				; RUBOUT.
		A	31							
		B	0				include "equates-trs-flags.s"			; Pointers to TRSDOS internal flags.
		B	1							
		B	2				; Flag equate constants (added as offsets to flag base address )			
		B	3							
		B	4							
	0000006A	B	5	ZAFLAG\$	EQU	0+FLGTAB\$				
	0000006B	B	6	ZBFLAG\$	EQU	1+FLGTAB\$				
	0000006C	B	7	ZCFLAG\$	EQU	2+FLGTAB\$				
	0000006D	B	8	ZDFLAG\$	EQU	3+FLGTAB\$				
	0000006E	B	9	ZEFLAG\$	EQU	4+FLGTAB\$				
	0000006F	B	10	ZFFLAG\$	EQU	5+FLGTAB\$				
	00000070	B	11	ZGFLAG\$	EQU	6+FLGTAB\$				
	00000071	B	12	ZHFLAG\$	EQU	7+FLGTAB\$				
	00000072	B	13	ZIFLAG\$	EQU	8+FLGTAB\$				
	00000073	B	14	ZJFLAG\$	EQU	9+FLGTAB\$				
	00000074	B	15	ZKFLAG\$	EQU	10+FLGTAB\$				
	00000075	B	16	ZLFLAG\$	EQU	11+FLGTAB\$				
	00000076	B	17	ZMFLAG\$	EQU	12+FLGTAB\$				
	00000077	B	18	ZNFLAG\$	EQU	13+FLGTAB\$				
	00000078	B	19	ZOFLAG\$	EQU	14+FLGTAB\$				

\*\*\*\*\* TRSDOS \*\*\*\*\*  
< ASCII equates. >

PC	Object	I	Line	Source ..\..\TRSDOS_GLOBAL\equates\equates-trs-flags.s	
	00000079	B	20	ZPFLAG\$ EQU 15+FLGTAB\$	
	0000007A	B	21	ZQFLAG\$ EQU 16+FLGTAB\$	
	0000007B	B	22	ZRFLAG\$ EQU 17+FLGTAB\$	
	0000007C	B	23	ZSFLAG\$ EQU 18+FLGTAB\$	
	0000007D	B	24	ZTFLAG\$ EQU 19+FLGTAB\$	
	0000007E	B	25	ZUFLAG\$ EQU 20+FLGTAB\$	
	0000007F	B	26	ZVFLAG\$ EQU 21+FLGTAB\$	
	00000080	B	27	ZWFLAG\$ EQU 22+FLGTAB\$	
	00000081	B	28	ZXFLAG\$ EQU 23+FLGTAB\$	
	00000082	B	29	ZYFLAG\$ EQU 24+FLGTAB\$	
	00000083	B	30	ZZFLAG\$ EQU 25+FLGTAB\$	
		B	0	include "equates-trs-api.s"	; Constants needed for TRSDOS API interface.
		B	1		
		B	2	SUBTITLE "< Equates for TRS80 API. >"	
		B	3	NEWPAGE	

\*\*\*\*\* TRSDOS \*\*\*\*\*  
 < Equates for TRS80 API. >

PC	Object	I	Line	Source	..\\..\\TRSDOS_GLOBAL\\equate\\equate-trs-api.s
		B	4	scope	
		B	5		
		B	6	; Supervisor call equates for TRSDOS/LSDOS	
		B	7		
	00000015	B	8	svc_abort	equ 21
	0000001D	B	9	svc_adtsk	equ 29
	00000066	B	10	svc_zbank	equ 102
	0000003D	B	11	svc_bksp	equ 61
	00000067	B	12	svc_break	equ 103
	00000014	B	13	svc_chnio	equ 20
	0000006A	B	14	svc_ckbrkc	equ 106
	00000021	B	15	svc_ckdrv	equ 33
	0000003E	B	16	svc_ckeof	equ 62
	0000001C	B	17	svc_cktsk	equ 28
	0000003C	B	18	svc_close	equ 60
	00000069	B	19	svc_cls	equ 105
	00000018	B	20	svc_cmndi	equ 24
	00000019	B	21	svc_cmndr	equ 25
	00000005	B	22	svc_ctl	equ 5
	00000012	B	23	svc_date	equ 18
	0000002A	B	24	svc_dcinit	equ 42
	0000002B	B	25	svc_dcres	equ 43 ; Disk controller reset.
	00000031	B	26	svc_dcstat	equ 49
	0000001B	B	27	svc_debug	equ 27 ; Enter system debugger.
	00000060	B	28	svc_dechex	equ 96
	00000057	B	29	svc_dirrd	equ 87
	0000005D	B	30	svc_div8	equ 93
	0000005E	B	31	svc_div16	equ 94
	00000022	B	32	svc_dodir	equ 34 ; Display directory.
	00000002	B	33	svc_dsp	equ 2
	0000000A	B	34	svc_dsply	equ 10
	0000001A	B	35	svc_error	equ 26 ; Error SVC.
	00000016	B	36	svc_zexit	equ 22 ; Exit back to O/S.
	00000065	B	37	svc_flags	equ 101
	00000003	B	38	svc_get	equ 3
	00000051	B	39	svc_gtdct	equ 81
	00000053	B	40	svc_gtmod	equ 83 ; Get address of module in memory.
	00000052	B	41	svc_gtdcb	equ 82 ; Get device control block address.
	00000061	B	42	svc_hexdec	equ 97
	00000000	B	43	svc_ipl	equ 0
	00000008	B	44	svc_kbd	equ 8
	00000001	B	45	svc_key	equ 1
	00000009	B	46	svc_keyin	equ 9
	0000004C	B	47	svc_load	equ 76 ; Load a file.
	0000000B	B	48	svc_logger	equ 11 ; Send message to system log.
	0000000C	B	49	svc_logot	equ 12 ; Send message to system log & console.
	0000000D	B	50	svc_msg	equ 13
	0000005B	B	51	svc_mul16	equ 91
	0000005A	B	52	svc_mul8	equ 90
	00000010	B	53	svc_pause	equ 16
	0000000E	B	54	svc_printz	equ 14
	00000006	B	55	svc_prt	equ 6
	00000004	B	56	svc_put	equ 4
	0000004D	B	57	svc_run	equ 77
	00000068	B	58	svc_sound	equ 104
	00000030	B	59	svc_rdhdr	equ 48
	00000031	B	60	svc_rdsec	equ 49
	00000055	B	61	svc_rdssc	equ 85 ; Read a system sector.
	0000002F	B	62	svc_rslct	equ 47 ; Reselect busy drive until ready.
	0000002C	B	63	svc_rstor	equ 44 ; Restore drive to CYL 0.



\*\*\*\*\* TRSDOS \*\*\*\*\*

< Equates for TRS80 API. >

PC	Object	I	Line	Source	...	...	TRSDOS_GLOBAL\equates\equates-trs-api.s
	0000002E	B	64	svc_seekcyl:	equ	46	
	00000029	B	65	svc_slct	equ	41	
	0000002D	B	66	svc_stepi	equ	45	; Issue step in to drive controller.
	00000013	B	67	svc_time	equ	19	; System time.
	0000000F	B	68	svc_vdctl	equ	15	; Video control SVC.
	00000007	B	69	svc_where	equ	7	; Resolve runtime address.
	00000035	B	70	svc_wrsec	equ	53	; Write a disk sector.
	00000036	B	71	svc_wrssc	equ	54	; Write a systems sector.
		B	72				
		B	73				
		B	74	; Logic equates.			
		B	75				
	00000000	B	76	FALSE	EQU	0	; Common usage in many modules.
	FFFFFFFF	B	77	TRUE	EQU	-1	; Common usage in many modules.
		B	78				
		B	79				
		B	80	; <Instructions.>			
		B	81				
	000000C9	B	82	RETOPCODE:	EQU	0C9h	
		B	83	restart28	EQU	"	RST 28h"
		B	84				
		A	34				
		B	0	include "equates-versions.s"			
		B	1	NEWPAGE			

```
***** TRSDOS *****
```

PC	Object	I	Line	Source	..\\..\\TRSDOS_GLOBAL\\equates\\equates-versions.s
		B	2	SUBTITLE	"< Equates supporting multiple serial ports, memory, video, languages - OS
		B	3	scope	
		B	4		
		B	5	; < Assignments for OS assembly/build. >	
		B	6		
		B	7	; < Serial port assignments. >	
		B	8		
00000000		B	9	DEVICE.console EQU 0	; Serial port to use as console.
00000001		B	10	DEVICE.serialnet EQU 1	; Serial port to use for serial networking.
		B	11		
		B	12		
00038400		B	13	DEVICE.console.baud EQU 230400	; Starting console baudrate.
00038400		B	14	DEVICE.serialnet.baud EQU 230400	; Starting UART1 baudrate.
		B	15		
0000000D		B	16	f91_console_baud EQU ((_SYS_CLK_FREQ / DEVICE.console.baud) /16)	; Default baud rate for f91
00000005		B	17	f92_console_baud EQU ((20000000 / DEVICE.console.baud) /16)	; Default baud rate for f92
		B	18		
0000000D		B	19	f91_serialnet_baud EQU ((_SYS_CLK_FREQ / DEVICE.serialnet.baud) /16)	; Default baud rate for
00000005		B	20	f92_serialnet_baud EQU ((20000000 / DEVICE.serialnet.baud) /16)	; Default baud rate for f92
		B	21		
		B	22	; < Systems terminal refresh updates. >	
		B	23		
00000003		B	24	video_refresh EQU 3	; Refresh counter rate. 60hz is divided by video_refresh & determin
00000014		B	25	pump_retard EQU 20	; Initial amount to delay when pump 1st started. 0 makes max ticks of c
		B	26		
		B	27		
		B	28	; < Memory map/assignments for OS build. >	
		B	29		
00000000		B	30	START\$ EQU 0000h	; Base offset of conditional assembly.
		B	31		
0000F800		B	32	CRTBGN\$ EQU zspace.VIDEO\$	; 1920 bytes video / keyboard RAM.
0000FF7F		B	33	CRTEND\$ EQU zspace.VIDEO\$+CRTSIZE-1	
		B	34		
00002300		B	35	DIRBUF\$ EQU COREMAX\$-100h	; File buffer.
		B	36		
00002400		B	37	COREMAX\$ EQU START\$+2400h	; Start of application area when no library zone used.
00003000		B	38	COREMIN\$ EQU START\$+3000h	; Start of application area with active library area.
0000F5F4		B	39	COREEND\$ EQU INTERRUPT.handler-0ch	; End+1 of usable memory for high memory routines.
		B	40		
0000FFFF		B	41	ENDRAM\$ EQU 10000H-1	
		B	42		
		B	43	; < Version flags ultralite, lite, standard and advanced. >	
		B	44		
FFFFFFFF		B	45	zULITE EQU TRUE	; HIGH memory F400h-FFFFh contains video, no bank switching.
00000000		B	46	zLITE EQU FALSE	; No bank RAM, full video RAM switching
00000000		B	47	zSTANDRD EQU FALSE	; Supports video and bank RAM as standard M4.
00000000		B	48	zADVANCd EQU FALSE	; STANDRD features plus bank memory to 256*32K banks.
		B	49		
		B	50	; < Country codes. >	
		B	51		
00000000		B	52	zINTL EQU FALSE	
FFFFFFFF		B	53	zUSA EQU TRUE	
		B	54		
		B	55	; < Language flags. >	
		B	56		
00000000		B	57	zCZECH EQU FALSE	
00000000		B	58	zDEUTSCH EQU FALSE	
FFFFFFFF		B	59	zENGLISH EQU TRUE	
00000000		B	60	zFRENCH EQU FALSE	
00000000		B	61	zGERMAN EQU zDEUTSCH	

\*\*\*\*\* TRSDOS \*\*\*\*\*  
< Equates supporting multiple serial ports, memory, video, languages - OS BUILDS. >

PC	Object	I	Line	Source	..\\..\\TRSDOS_GLOBAL\\equate\\equate-versions.s
		B	62		
		B	63	; < Hardware platform specific equates. >	
		B	64		
		B	65	;_EZ80ACCLAIM! EQU 1	
		B	66		
	0000003C	B	67	HEART_RATE EQU 60	
		B	68		
	00000000	B	69	zHZ50 EQU FALSE	
	FFFFFFFF	B	70	zHZ60 EQU TRUE	
		B	71		
	00000000	B	72	zMOD2 EQU FALSE	
	00000000	B	73	zMOD4 EQU FALSE	
	FFFFFFFF	B	74	zMOD5 EQU TRUE	
		B	75		
		B	76	; <TRSDOS versions flags.>	
		B	77		
	FFFFFFFF	B	78	zBLD631 EQU TRUE	
	FFFFFFFF	B	79	zBLD631C EQU TRUE	
	FFFFFFFF	B	80	zBLD631D EQU TRUE	
	FFFFFFFF	B	81	zBLD631E EQU TRUE	
	FFFFFFFF	B	82	zBLD631F EQU TRUE	
	FFFFFFFF	B	83	zBLD631G EQU TRUE	
	FFFFFFFF	B	84	zBLD631H EQU TRUE	
		A	36		
		B	0	include "equate-memory.s" ; Memory map & configuration for build.	
		B	1	SUBTITLE <* Parameters defining memory model available to OS. *>	
		B	2	NEWPAGE	

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt;\* Parameters defining memory model available to OS. \*&gt;

PC	Object	I	Line	Source	...	...	...	...	...
		B	3	scope					
		B	4						
		B	5	; <* Drivers. In order KI DO PR FD S0 S1 *>					
		B	6						
	000008F0	B	7	driver.KI EQU START\$+08F0h				; Keyboard driver.	
	00000B88	B	8	driver.DO EQU START\$+0B88h				; Begin video driver.	
	00000E01	B	9	driver.PR EQU START\$+0E01h				; Start printer driver.	
	00000E3D	B	10	driver.FD EQU START\$+0E3Dh				; Floppy Disk Driver entry point.	
	00000000	B	11	driver.S0 EQU START\$+0000h				; UART 0 driver.	
	00000000	B	12	driver.S1 EQU START\$+0000h				; UART 1 driver.	
		B	13						
	00000000	B	14	handler.zRST00 EQU 0000h				; IPL SVC.	
	00000000	B	15	handler.FDDINT\$ EQU 0000h				; NOP or DI (F3h) for	
	00000010	B	16	handler.zRST10 EQU 0010h					
	00000018	B	17	handler.zRST18 EQU 0018h					
	00000020	B	18	handler.zRST20 EQU 0020h					
		B	19	; handler.zRST28 EQU RST28				; System SVC processor	
		B	20	; handler.zRST30 EQU zDEBUG				; DEBUG call address	
	00000038	B	21	handler.zRST38 EQU 038h				; RST38z Interrupt RST	
	0000F600	B	22	INTERRUPT.handler EQU 0F600h					
		B	23						
		B	24	; These equates define storage space to position individual modules.					
		B	25						
	00000000	B	26	memory.PAGE0\$ EQU START\$+0000h					
	00000100	B	27	memory.PAGE1\$ EQU START\$+0100h					
	00000200	B	28	memory.PAGE2\$ EQU START\$+0200h					
	00000300	B	29	memory.PAGE3\$ EQU START\$+0300h					
	00000400	B	30	memory.PAGE4\$ EQU START\$+0400h					
	00000500	B	31	memory.PAGE5\$ EQU START\$+0500h					
		B	32						
	00000380	B	33	zspace.STACK\$ EQU START\$+0380h				; Start of stace.	
		B	34						
	000008F0	B	35	zspace.DRIVERS\$ EQU START\$+08F0h				; Beginning of driver zone. KIDVR is 1st driver.	
		B	36						
	00001300	B	37	zspace.BYTEIO\$ EQU START\$+1300h				; Beginning of byte I/O routines.	
	00001948	B	38	zspace.MEMORY\$ EQU START\$+1948h				; HIGH/LOW memory functions.	
	0000196A	B	39	zspace.SVCHANDLER\$ EQU START\$+196Ah				; SVC handler.	
	000019F6	B	40	zspace.IOFUNCTION\$ EQU START\$+19F6h					
	00001A43	B	41	zspace.RST28\$ EQU START\$+1A43h				; RST28 Routine.	
	00001AA0	B	42	zspace.EXOVR\$ EQU START\$+1AA0h				; Execute overlay.	
	00001AF4	B	43	zspace.SYSERROR\$ EQU START\$+1AF4h				; Error executing SVC.	
	00001B38	B	44	zspace.LOAD\$ EQU START\$+1B38h				; Beginning of LOAD routine.	
	00001B56	B	45	zspace.LOADER\$ EQU START\$+1B56h				; Beginning of LOADER routine	
	00001BFF	B	46	zspace.RST38\$ EQU START\$+1BFFh				; RST38 handler.	
	00001C94	B	47	zspace.TASKER\$ EQU START\$+1c94h					
		B	48						
	00001D00	B	49	zspace.SYSBUF\$ EQU START\$+1D00h				; System buffer, 256 byte.	
	00001E00	B	50	zspace.OVERLAY\$ EQU START\$+1E00h				; Systems overlay area, 1.5K bytes.	
	00002400	B	51	zspace.LIBRARY\$ EQU START\$+2400h				; Library modules execution area 3072 bytes.	
	00003000	B	52	zspace.APPLICATION\$ EQU START\$+3000h				; Application execution area, aprox 51.2K ending at COR	
	0000F600	B	53	zspace.HIGHMEMORY\$ EQU START\$+0F600h				; Begin of protected HIGH MEMORY AREA & old keyboard sc	
	0000F800	B	54	zspace.VIDEO\$ EQU START\$+0F800h				; 1920 bytes video & type ahead buffer.	
		B	55						
	00000003	B	56	quiet_default_flg EQU START\$+3				; Holds flag for quiet & terminal type on default startup.	
		B	0	include "equates-keyboard.s"				; Driver for keyboard.	
		B	1	NEWPAGE					

```
<* Parameters defining memory model available to OS. *>
```

PC	Object	I	Line	Source	...	...	TRSDOS_GLOBAL\equates\equates-keyboard.s
		B	2	SUBTITLE		"< Common keyboard equates for OS BUILD. >"	
		B	3	scope			
		B	4				
	0000F401	B	5	KB0	EQU	0F401h	;Row 0 RAM address
	0000F401	B	6	KB1	EQU	0F401h	
	0000F440	B	7	KB7	EQU	0F440h	
		B	8	;KB67	EQU	0F460h	;Keyboard rows 6&7.
	0000F480	B	9	SHIFT	EQU	0F480h	;Row 7 RAM address
	0000FF80	B	10	TYPBUF	EQU	0FF80h	
		B	11				
		B	0	include		"equates-video.s"	; Driver for video memory.
		B	1	SUBTITLE		"< Common equates for OS VIDEO parameters. >"	
		B	2	NEWPAGE			

***** TRSDOS *****				
< Common equates for OS VIDEO parameters. >				
PC	Object	I	Line	Source ..\..\TRSDOS_GLOBAL\equates\equates-video.s
		B	3	scope
		B	4	
		B	5	; < Screen and control equates. >
		B	6	
	FFFFFFB0	B	7	NEGLINE EQU -LINESIZ
	00000007	B	8	SCRPROT EQU 7 ; Bits 0-2: scroll protect.
	0000001F	B	9	CLRFRM EQU EQU 1Fh ; Clear form.
	00000004	B	10	CTL EQU 4 ; Bit 4, display controls
	0000001C	B	11	HOME EQU EQU 1Ch
	00000003	B	12	TABS EQU 3 ; Bit 3: 0=tabs, 1=chars
		B	13	
		B	14	
		B	15	; <Video display definitions>
		B	16	
		B	17	
	00000050	B	18	LINESIZ EQU 80
	00000018	B	19	NUMROWS EQU 24
	00000780	B	20	CRTSIZE EQU LINESIZ*NUMROWS ; 24x80=1920 bytes.
		A	40	
	000004A8	A	41	freelow equ zspace.BYTEIO\$-DVREND\$
	000009F4	A	42	usedhi equ FFFFh-INTERRUPT.handler-0bh
		A	43	
		A	44	ORG memory.PAGE0\$ ; And we begin.
000000		A	45	trs_os
		B	0	include "page0.s" ; Critical machine vectors, hardware data &
		B	1	SUBTITLE"<* SYSRES Page 0 - RST's, data, and buffers *> "
		B	2	newpage

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt;\* SYSRES Page 0 - RST's, data, and buffers \*&gt;

PC	Object	I	Line	Source	.. \SYSRES\lowcore\page0.s
		B	3	scope	
		B	4	DEFINE	PAGE0,ORG=%0000
		B	5		
		B	6		
000000	C3001E	B	7	zRST00	JP zspace.OVERLAY\$ ; Let the party begin.
000003	04	B	8	db	00000100b ; XY...ZZZ X=1=quiet X=0=normal. Y=1 network, y=0 no net. ZZZ defau
000004	00	B	9	BANKRES	db 0
000005	C9	B	10	CPMEMULATOR	ret ;CP/M emulator SVC.
000006	0000	B	11	dw	0
000008	C9	B	12	zRST08	RET
000009	0000	B	13	DW	0
00000B	0000	B	14	SVCRET\$	DW 0 ;Return address from SVC
00000D	00	B	15	LSVC\$	DB 0 ;Last SVC executed
00000E	F3	B	16	FDDINT\$	DI ;NOP or DI (F3h) for
00000F	C9	B	17	RET	;System (Smooth)
000010	C9	B	18	zRST10	RET
000011	0000	B	19	DW	0
		B	20		
000013	00 00 00	B	21	USTOR\$	BLKB 3,0 ;User storage area
000016	0000	B	22	DW	0 ;IV.DMPVID
000018	C9	B	23	zRST18	RET
000019	0000	B	24	DW	0
00001B	01	B	25	PDRV\$	DB 1 ; Current drive, physical.
00001C	0000	B	26	PHIGH\$	DW 0 ; Physical HIGH\$. This filled in at initialization.
00001E	0030	B	27	zLOW\$	DW COREMIN\$ ; Lowest usable memory
000020	C9	B	28	zRST20	RET
000021	0000	B	29	DW	0
000023	00	B	30	LDRV\$	DB 0 ;Current drive, logical
000024	0000	B	31	JDCB\$	DW 0 ;Saved FCB pointer
000026	0000	B	32	JRET\$	DW 0 ;Saved I/O return address
000028	C3 5B 1A	B	33	zRST28	jp RST28 ;System SVC processor
00002B	55	B	34	TIMSL\$	DB 55h ;Fast=55, slow=FF
00002C	00	B	35	TIMER\$	DB 0 ;RTC heartbeat counter.
00002D	000000	B	36	TIME\$	db 0,0,0 ;SS:MM:HH storage area
000030	C3 A0 19	B	37	zRST30	JP zDEBUG ;DEBUG call address
000033	17	B	38	DATE\$	DB 23
000034	11	B	39		DB 17
000035	03	B	40		DB 03
000036	4C	B	41		DB 76 ; Day of year.
000037	03	B	42		DB 3 ; YY/DD/MM/packed
		B	43		
000038	C3 38 00	B	44	zRST38	jp zRST38 ; Interrupt RST
		B	45		
00003B	01	B	46	OSRLS\$	DB 01h ; <631>OS Release #
		B	47		
		B	48		; INTIM\$ stores the image read from RDINTSTATUS*
		B	49		
00003C	00	B	50	INTIM\$	DB 0 ;Interrupt latch image
		B	51		
		B	52		; INTMSK\$ masks the image read from RDINTSTATUS*
		B	53		; LDOS 6.x permits only RS-232 RCV INT, IOBUS INT,
		B	54		; and RTC INT to be used by the TASKER off of RST38
		B	55		
00003D	2C	B	56	INTMSK\$	DB 2Ch ;Mask for INTIM\$
		B	57		
		B	58		; INTVC\$ stores the eight vectors associated
		B	59		; with the INTIM\$ bit assignments
		B	60		
		B	61		
00003E	481C 481C 481C	B	62	INTVC\$	DW RETINST,RETINST,RETINST,RETINST ;Primary interrupts

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt;\* SYSRES Page 0 - RST's, data, and buffers \*&gt;

```

PC      Object      I  Line      Source  ..\SYSRES\lowcore\page0.s
000044 481C
000046 481C 941C 481C      B    63              DW      RETINST,TASKER,RETINST,RETINST
00004C 481C
                                B    64
                                B    65      ; TCB$ stores the TCB vectors for task slots 0-11
                                B    66      ;
00004E E91C E91C E91C      B    67      TCB$      DW      NOTASK,NOTASK,NOTASK,NOTASK      ;Interrupt task table, IM 1.
000054 E91C
000056 E91C E91C E91C      B    68              DW      NOTASK,NOTASK,NOTASK,NOTASK      ;Interrupt task table, IM 1.
00005C E91C
00005E E91C E91C E91C      B    69              DW      NOTASK,NOTASK,NOTASK,NOTASK      ;Interrupt task vectors.
000064 E91C
                                B    70
                                B    71      ; NMI vector used in disk I/O
                                B    72
000066 C9      B    73      zNMI      RET
000067 0000      B    74              DW      0
                                B    75
                                B    76      ; OVRLY$ stores the system"s overlay request #
                                B    77
000069 00      B    78      OVRLY$      DB      0      ;Current overlay resident
                                B    79
                                B    80      ;{> FLGTAB$ stores 26 system flags and images (A-Z).
                                B    81      ;{> A pointer to this table is obtained from SVC-zFLAGS.
                                B    82
                                B    83      FLGTAB$  EQU      $
                                B    84
00006A 01      B    85      AFLAG$      DB      1      ; A FLAG - Start CYL for Allocation search
00006B 00      B    86      BFLAG$      DB      0      ; B FLAG
                                B    87
                                B    88      ; CFLAG$ assignments.
                                B    89      ;      0 - Cannot change HIGH$ via SVC-100
                                B    90      ;      1 - zCMNDR in execution
                                B    91      ;      2 - zKEYIN request from SYS1
                                B    92      ;      3 - System request for drivers, filters, DCTs
                                B    93      ;      4 - zCMNDR to only execute LIB commands
                                B    94      ;      5 - Sysgen inhibit bit
                                B    95      ;      6 - zERROR inhibit display
                                B    96      ;      7 - zERROR to use user (DE) buffer
                                B    97
00006C 00      B    98      CFLAG$      DB      0      ; Condition flag
                                B    99
                                B   100      ; DFLAG$ assignments:
                                B   101      ;      0 - SPOOL is active
                                B   102      ;      1 - TYPE ahead is active
                                B   103      ;      2 - VERIFY is on
                                B   104      ;      3 - SMOOTH active
                                B   105      ;      4 - MemDISK active
                                B   106      ;      5 - FORMS active
                                B   107      ;      6 - KSM active
                                B   108      ;      7 - accept GRAPHICS in screen print
                                B   109
00006D 0A      B   110      DFLAG$      DB      00001010B      ; DEV Flag (SMOOTH,TYPE)
                                B   111
                                B   112      ; EFLAG$ - Assignments: (sys13 usage)
                                B   113      ; use only bits 4, 5 and 6 to indicate user
                                B   114      ; entry code to be passed to SYS13. SYS13
                                B   115      ; will be executed from SYS1 if this byte
                                B   116      ; is NON/0, bit 4, 5 and 6 will be merged into
                                B   117      ; the SYS13 (1000,1111b) overlay request

```



\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt;\* SYSRES Page 0 - RST's, data, and buffers \*&gt;

PC	Object	I	Line	Source	.. \SYSRES \lowcore \page0.s
		B	118		
00006E	00	B	119	EFLAG\$ DB 0	; Flag E
00006F	00	B	120	FEMSK\$ DB 0	; Port FE mask
000070	00	B	121	GFLAG\$ DB 0	; G Flag.
000071	00	B	122	HFLAG\$ DB 0	; Flag H.
		B	123		
		B	124	; IFLAG\$ - Assignments: (INTERNATIONAL)	
		B	125	; 0 - FRENCH	
		B	126	; 1 - GERMAN	
		B	127	; 2 - SWISS	
		B	128	; 3 -	
		B	129	; 4 -	
		B	130	; 5 -	
		B	131	; 6 - Special DMP mode ON/OFF	
		B	132	; 7 - '7' bit mode ON/OFF	
		B	133		
000072	00	B	134	IFLAG\$ DB 0	; Flag I
000073	00	B	135	JFLAG\$ DB 0	; Flag J
		B	136		
		B	137	; KFLAG\$ assignments:	
		B	138	; 0 - BREAK latch	
		B	139	; 1 - PAUSE latch	
		B	140	; 2 - ENTER latch	
		B	141	; 3 - reserved	
		B	142	; 4 - reserved	
		B	143	; 5 - CAPs lock	
		B	144	; 6 - reserved	
		B	145	; 7 - character in TYPE ahead	
		B	146		
000074	00	B	147	KFLAG\$ DB 0	; Keyboard flag
		B	148		
		B	149	; LFLAG\$ assignments:	
		B	150	; 0 - inhibit step rate question in FORMAT	
		B	151	; 4 - inhibit 8" query in FLOPPY/DCT	
		B	152	; 5 - inhibit # sides question in FORMAT	
		B	153	; 6,7 - Reserved for IM 2 hardware	
		B	154		
		B	155		
000075	01	B	156	LFLAG\$ DB 00000001B	; LDOS feature inhibit
		B	157		
		B	158	; MODOUT\$ mask assignments:	
		B	159	; 0 -	
		B	160	; 1 - cassette motor on/off	
		B	161	; 2 - mode select (0 = 80/64, 1 = 40/32)	
		B	162	; 3 - enable alternate character set	
		B	163	; 4 - enable external I/O	
		B	164	; 5 - video wait states (0 = disable, 1 = enable)	
		B	165	; 6 - clock speed ( 1 = 4 Mhz, 0 = 2 MHz)	
		B	166	; 7 -	
		B	167		
000076	78	B	168	MODOUT\$ DB 78h	; MODOUT port image (FAST)
		B	169		
		B	170	; NFLAG\$ - Network flag\$	
		B	171	; 0 - Allow setting of file open bit in DIR	
		B	172	; 1 / 5 - Reserved	
		B	173	; 6 - Set if in Task Processor	
		B	174	; 7 - Reserved	
		B	175		
000077	01	B	176	NFLAG\$ DB 1	; Inhibit open bit in DIR
		B	177		

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt;\* SYSRES Page 0 - RST's, data, and buffers \*&gt;

PC	Object	I	Line	Source
		B	178	; OPREG\$ - memory management image port
		B	179	; 0 - SEL0 - Select map overlay bit 0
		B	180	; 1 - SEL1 - Select map overlay bit 1
		B	181	; 2 - 80/64 - 1 = 80 x 24
		B	182	; 3 - Inverse video
		B	183	; 4 - MBIT0 - memory map bit 0
		B	184	; 5 - MBIT1 - memory map bit 1
		B	185	; 6 - FXUPMEM - fix upper memory
		B	186	; 7 - PAGE - page 1K video RAM (set for 80x24)
		B	187	;
		B	188	
000078	87	B	189	OPREG\$ DB 87h ; Memory management image
		B	190	
		B	191	; PFLAG\$ - Printer flag
		B	192	; 7 = Printer spooler is paused
		B	193	; 0 - 6 = Reserved
		B	194	
000079	00	B	195	PFLAG\$ DB 0
00007A	00	B	196	QFLAG\$ DB 0
		B	197	
		B	198	; RFLAG\$ - Retry init for FDC driver
		B	199	
00007B	08	B	200	RFLAG\$ DB 8 ; FDC retry count >=2
		B	201	
		B	202	; SFLAG\$ assignments:
		B	203	; 0 - inhibit file open bit
		B	204	; 1 - set to 1 if bit-2 set & EXEC file opened
		B	205	; 2 - set by zRUN to permit load of EXEC file
		B	206	; 3 - SYSTEM (FAST)
		B	207	; 4 - BREAK key disabled
		B	208	; 5 - JCL active
		B	209	; 6 - force extended error messages
		B	210	; 7 - DEBUG to be turned on after load
		B	211	
00007C	08	B	212	SFLAG\$ DB 00001000b ; System flag (FAST)
		B	213	
		B	214	; Machine TYPE assignment:
		B	215	; All values are in decimal
		B	216	;
		B	217	; 2 = TRS-80 Model 2
		B	218	; 4 = TRS-80 Model 4
		B	219	; 5 = TRS-80 MODEL 4P
		B	220	; 12 = TRS-80 Model 12
		B	221	; 16 = TRS-80 Model 16
		B	222	
00007D	05	B	223	TFLAG\$ DB 5 ; Model 4 assignment
00007E	00	B	224	UFLAG\$ DB 0 ; Flag U
		B	225	
		B	226	; Video FLAG\$ assignments:
		B	227	; 0-3 - Set blink rate (1=fastest,7=slowest)
		B	228	; 4 - display CLOCK
		B	229	; 5 - cursor blink toggle bit
		B	230	; 6 - Inhibit blinking cursor (user)
		B	231	; 7 - Inhibit blinking cursor (system)
		B	232	
00007F	40	B	233	VFLAG\$ DB 40h ; Blink,Slow,No clock
		B	234	
		B	235	; WRINT\$ - interrupt mask register
		B	236	; 0 - enable 1500 baud rising edge
		B	237	; 1 - enable 1500 baud falling edge

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt;\* SYSRES Page 0 - RST's, data, and buffers \*&gt;

PC	Object	I	Line	Source	..\\SYSRES\\lowcore\\page0.s
		B	238	; 2 - enable real time clock	
		B	239	; 3 - enable I/O bus interrupts	
		B	240	; 4 - enable RS-232 transmit interrupts	
		B	241	; 5 - enable RS-232 receive data interrupts	
		B	242	; 6 - enable RS-232 error interrupt	
		B	243		
000080	00	B	244	WRINT\$ DB 0	; WRINTMASK port image
000081	00	B	245	XFLAG\$ DB 0	; Flag x
		B	246		
		B	247	; Bits 0-7 indicate new style dating on drives 0-7	
		B	248		
000082	FF	B	249	YFLAG\$ DB 0FFh	
000083	00	B	250	ZFLAG\$ DB 0	; Z flag
		B	251		
		B	252	; Contents are high-order byte of SVC table	
		B	253		
000084	01	B	254	MSBSVC DB SVCTAB\$>>8	; MSB of SVC table
		B	255		
		B	256	; OSVER\$ stores the operating system version	
		B	257		
000085	63	B	258	OSVER\$ DB 63h	; OS version #
		B	259		
		B	260	; Vector for config initialization	
		B	261		
000086	C9	B	262	zICNFG RET	; Initialization config
000087	0000	B	263	DW 0	
		B	264		
		B	265	; Chain vector for KI task processor	
		B	266		
000089	C9	B	267	zKITSK RET	; Keyboard task routine
00008A	0000	B	268	DW 0	
		B	269		
		B	270	; System File Control Block for overlays	
		B	271		
00008C	800000	B	272	SFCB\$ DB 80h,0,0	; System /SYS FCB
00008F	001D	B	273	DW SBUFF\$	
000091	00	B	274	DB 0	
000092	00000000 0000FFFF	B	275	DW 0,0,0,-1,0,-1,-1	
00009A	0000FFFF FFFF				
		B	276		
		B	277	; 32-byte DEBUG save area	
		B	278		
0000A0	00 00 00 00 00 00 00	B	279	DBGSV\$ BLKB 32,0	
0000A6	00 00 00 00 00 00 00				
0000AC	00 00 00 00 00 00 00				
0000B2	00 00 00 00 00 00 00				
0000B8	00 00 00 00 00 00 00				
0000BE	00 00				
		B	280		
		B	281	; Job Control Language File Control Block	
		B	282		
0000C0	00 00 00	B	283	JFCB\$ BLKB 3,0	
0000C3	001D	B	284	DW SBUFF\$	
0000C5	00 00 00 00 00 00 00	B	285	BLKB 27,0	
0000CB	00 00 00 00 00 00 00				
0000D1	00 00 00 00 00 00 00				
0000D7	00 00 00 00 00 00 00				
0000DD	00 00 00				
		B	286		
		B	287	; System Command Line file control block	

\*\*\*\*\* TRSDOS \*\*\*\*\*  
<\* SYSRES Page 0 - RST's, data, and buffers \*>

PC	Object	I	Line	Source	..\\SYSRES\\lowcore\\page0.s
		B	288		
	000000E0	B	289	CFCB\$ EQU \$	; Command Interpreter FCB.
0000E0	434F4E46 49472F53	B	290	CFGFCB\$ DB	"CONFIG/SYS.CCC:0",ETX.asc
0000E8	59532E43 43433A30				
0000F0	03				
0000F1	00 00 00 00 00 00	B	291	BLKB	15,0
0000F7	00 00 00 00 00 00				
0000FD	00 00 00				
		B	0	include "page1.s"	; SVC vector table.
		B	1	SUBTITLE	"< SYSRES Page 1 >"
		B	2	newpage	

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYSRES Page 1 &gt;

PC	Object	I	Line	Source	.. \SYSRES\lowcore\page1.s
		B	3	scope	
		B	4	DEFINE	PAGE1,ORG=%0100
		B	5		
		B	6	; Update to LSDOS 6.3.1 2022-09-16 dpm	
		B	7		
		B	8	; Page 1 - System Supervisor Call Table.	
		B	9		
		B	10	ORG	memory.PAGE1\$
		B	11		
	00000100	B	12	SVCTAB\$ EQU	\$
		B	13		
000100	0000	B	14	DW	zRST00 ;SVC 0, IPL.
000102	2806	B	15	DW	zKEY ;SVC 1, Get byte from console.
000104	4206	B	16	DW	zDSP ;SVC 2, Send byte to console, DSP.
000106	3806	B	17	DW	zGET ;SVC 3, Get a byte from I/O stream/file, GET.
000108	4506	B	18	DW	zPUT ;SVC 4. PUT byte in I/O stream.
00010A	2306	B	19	DW	zCTL ;SVC 5. Control a device chain.
00010C	3D06	B	20	DW	zPRT ;SVC 6. Send a character to *PR device.
00010E	7919	B	21	DW	zWHERE ;SVC 7. Resolve run-time address.
000110	3506	B	22	DW	zKBD ;SVC 8. Scan the *KI device.
000112	8505	B	23	DW	zKEYIN ;SVC 9. Obtain a line of characters from *KI (or JCL).
000114	2D05	B	24	DW	zDSPLY ;SVC 10. DSPLY.
000116	0305	B	25	DW	zLOGGER ;SVC 11. Send a message to the Job Log (*JL).
000118	0005	B	26	DW	zLOGOT ;SVC 12. Display and log a message (*DO and *JL).
00011A	3005	B	27	DW	zMSG ;SVC 13. Send a message line to a device.
00011C	2805	B	28	DW	zPRINT ;SVC 14. Send a message line to *PR device.
00011E	990B	B	29	DW	zVDCTL ;15
000120	8203	B	30	DW	zPAUSE ;16
000122	8719	B	31	DW	zPARAM ;17
000124	2014	B	32	DW	zDATE ;18
000126	8D07	B	33	DW	zTIME ;19
000128	8906	B	34	DW	zCHNIO ;20
00012A	081B	B	35	DW	zABORT ;21
00012C	0B1B	B	36	DW	zEXIT ;22
00012E	F41A	B	37	DW	SVCERR ;23
000130	7E19	B	38	DW	zCMNDI ;24
000132	7B19	B	39	DW	zCMNDR ;25
000134	0F1B	B	40	DW	zERROR ;26
000136	A019	B	41	DW	zDEBUG ;27
000138	F51C	B	42	DW	zCKTSK ;28
00013A	DA1C	B	43	DW	zADTSK ;29
00013C	D71C	B	44	DW	zRMTSK ;30
00013E	EB1C	B	45	DW	zRPTSK ;31
000140	D01C	B	46	DW	zKLTSK ;32
000142	9319	B	47	DW	zCKDRV ;33
000144	AF19	B	48	DW	zDODIR ;34
000146	AC19	B	49	DW	zRAMDIR ;35
000148	F41A	B	50	DW	SVCERR ;36
00014A	F41A	B	51	DW	SVCERR ;37
00014C	F41A	B	52	DW	SVCERR ;38
00014E	F41A	B	53	DW	SVCERR ;39
000150	B519	B	54	DW	zDCSTAT ;40
000152	BC19	B	55	DW	zSLCT ;41
000154	C019	B	56	DW	zDCINIT ;42
000156	C419	B	57	DW	zDCRES ;43
000158	C819	B	58	DW	zRSTOR ;44
00015A	CC19	B	59	DW	zSTEPI ;45
00015C	D019	B	60	DW	zSEEK ;46
00015E	D419	B	61	DW	zRSLCT ;47
000160	D819	B	62	DW	zRDHDR ;48

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYSRES Page 1 &gt;

PC	Object	I	Line	Source	.. \SYSRES\lowcore\page1.s
000162	F419	B	63	DW	zRDSEC ;49
000164	DC19	B	64	DW	zVRSEC ;50
000166	E019	B	65	DW	zRDTRK ;51
000168	E419	B	66	DW	zHDFMT ;52
00016A	E819	B	67	DW	zWRSEC ;53
00016C	EC19	B	68	DW	zWRSSC ;54
00016E	F019	B	69	DW	zWRTRK ;55
000170	9619	B	70	DW	zRENAME ;56
000172	A619	B	71	DW	zREMOVE ;57
000174	8D19	B	72	DW	zINIT ;58
000176	8A19	B	73	DW	zOPEN ;59
000178	9919	B	74	DW	zCLOSE ;60
00017A	AD14	B	75	DW	zBKSP ;61
00017C	8F15	B	76	DW	zCKEOF ;62
00017E	DA14	B	77	DW	zLOC ;63
000180	0515	B	78	DW	zLOF ;64
000182	C914	B	79	DW	zPEOF ;65
000184	5B14	B	80	DW	zPOSN ;66
000186	1315	B	81	DW	zREAD ;67
000188	C214	B	82	DW	zREW ;68
00018A	9A14	B	83	DW	zRREAD ;69
00018C	AD13	B	84	DW	zRWRTIT ;70
00018E	A013	B	85	DW	zSEEKSC ;71
000190	5714	B	86	DW	zSKIP ;72
000192	6015	B	87	DW	zVER ;73
000194	3014	B	88	DW	zWEOF ;74
000196	3115	B	89	DW	zWRITE ;75
000198	381B	B	90	DW	zLOAD ;76
00019A	1D1B	B	91	DW	zRUN ;77
00019C	8119	B	92	DW	zFSPEC ;78
00019E	8419	B	93	DW	zFEXT ;79
0001A0	9C19	B	94	DW	zFNAME ;80
0001A2	1E1A	B	95	DW	zGTDCT ;81
0001A4	9019	B	96	DW	zGTDCCB ;82
0001A6	B219	B	97	DW	zGTMOD ;83
0001A8	F41A	B	98	DW	SVCERR ;84
0001AA	D818	B	99	DW	zRDSSC ;85
0001AC	7418	B	100	DW	zGATRD ;86
0001AE	BB18	B	101	DW	zDIRRD ;87
0001B0	0318	B	102	DW	zDIRWR
0001B2	7518	B	103	DW	zGATWR
0001B4	0A19	B	104	DW	zMUL8
0001B6	C906	B	105	DW	zMUL16 ;88-91
0001B8	F41A	B	106	DW	SVCERR
0001BA	2719	B	107	DW	zDIV8
0001BC	E306	B	108	DW	zDIV16
0001BE	F806	B	109	DW	zHEXDEC+2 ;92-95
0001C0	E103	B	110	DW	zDECHEX
0001C2	F606	B	111	DW	zHEXDEC
0001C4	C207	B	112	DW	zHEX8
0001C6	BD07	B	113	DW	zHEX16 ;96-99
0001C8	4819	B	114	DW	zHIGH
0001CA	6A19	B	115	DW	zFLAGS
0001CC	7708	B	116	DW	zBANK
0001CE	6F19	B	117	DW	zBREAK ;100-103
0001D0	9203	B	118	DW	zSOUND
0001D2	4505	B	119	DW	zCLS
0001D4	5305	B	120	DW	zCKBRKC
0001D6	3509	B	121	DW	zVDPRT ;104-107
0001D8	F41A	B	122	DW	SVCERR

\*\*\*\*\* TRSDOS \*\*\*\*\*  
< SYSRES Page 1 >

PC	Object	I	Line	Source	.. \SYSRES\lowcore\page1.s
0001DA	F41A	B	123	DW	SVCERR
0001DC	F41A	B	124	DW	SVCERR
0001DE	F41A	B	125	DW	SVCERR ;108-111
0001E0	F41A	B	126	DW	SVCERR
0001E2	F41A	B	127	DW	SVCERR
0001E4	F41A	B	128	DW	SVCERR
0001E6	F41A	B	129	DW	SVCERR ;112-115
0001E8	F41A	B	130	DW	SVCERR
0001EA	F41A	B	131	DW	SVCERR
0001EC	F41A	B	132	DW	SVCERR
0001EE	F41A	B	133	DW	SVCERR ;116-119
0001F0	F41A	B	134	DW	SVCERR
0001F2	F41A	B	135	DW	SVCERR
0001F4	F41A	B	136	DW	SVCERR
0001F6	F41A	B	137	DW	SVCERR ;120-123
0001F8	F41A	B	138	DW	SVCERR
0001FA	F41A	B	139	DW	SVCERR
0001FC	F41A	B	140	DW	SVCERR
0001FE	F41A	B	141	DW	SVCERR ;124-127
		B	0	include	"page2.s" ; Device Control Blocks & spares before stac
		B	1	SUBTITLE	"< SYSRES Page 2 >"
		B	2	newpage	

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYSRES Page 2 &gt;

PC	Object	I	Line	Source	.. \SYSRES\lowcore\page2.s
		B	3		scope
		B	4		
		B	5	ORG	memory.PAGE2\$
		B	6		
		B	7	;	Page 2 - Device Control Blocks.
		B	8		
000200	00	B	9	BUR\$ DB 00h	; Bank use RAM.
000201	FE	B	10	BAR\$ DB 0feh	; Bank available RAM.
000202	14	B	11	LBANK\$ DB 20	; Dir cyl & logical bank.
000203	01	B	12	JCLCB\$ DB 1	; Mini-DCB for JCL gets.
000204	0000	B	13	dw 0	; Pointer to input JCL FCB.
000206	580E	B	14	DVRHI\$ DW DVREND\$	; Start of low I/O zone.
		B	15		
		B	16	;	Keyboard Device Control Block.
		B	17		
000208	05	B	18	KIDCB\$ DB 5	;Permit CTL, GET
000209	F008	B	19	DW KIDVR	
00020B	0000004B 49	B	20	DB 0,0,0,"KI"	
		B	21		
		B	22	;	Display output Device Control Block (Video).
		B	23		
000210	07	B	24	DODCB\$ DB 7	;Permit CTL, PUT, GET
000211	880B	B	25	DW DODVR	
000213	00000044 4F	B	26	DB 0,0,0,"D0"	
		B	27		
		B	28	;	Printer Device Control Block.
		B	29		
000218	06	B	30	PRDCB\$ DB 6	;Permit CTL, PUT
000219	010E	B	31	DW PRDVR	
00021B	00000050 52	B	32	DB 0,0,0,"PR"	
		B	33		
		B	34	;	Standard input, Device Control Block.
		B	35		
000220	15	B	36	SIDCB\$ DB 15h	;Routed to *KI
000221	0802	B	37	DW KIDCB\$	
000223	0D000053 49	B	38	DB 0Dh,0,0,"SI"	
		B	39		
		B	40	;	Standard output, Device Control Block.
		B	41		
000228	17	B	42	SODCB\$ DB 17h	;Routed to *D0
000229	1002	B	43	DW DODCB\$	;DODCB\$
00022B	0F000053 4F	B	44	DB 0Fh,0,0,"S0"	
		B	45		
		B	46	;	Job control Device Control Block.
		B	47		
000230	0A	B	48	JLDCB\$ DB 0Ah	
000231	0000	B	49	dw 0	
000233	0A00004A 4C	B	50	db 0Ah,0,0,"JL"	
		B	51		
		B	52		
000238	07	B	53	U0DCB\$ DB 7	;Permit CTL, PUT, GET
000239	D910	B	54	DW U0DVR	
00023B	00000043 4C	B	55	DB 00h,0,0,"CL"	
		B	56		
000240	07	B	57	U1DCB\$ DB 7	;Permit CTL, PUT, GET
000241	F511	B	58	DW U1DVR	
000243	00000055 31	B	59	DB 00h,0,0,"U1"	
		B	60		
		B	61		
		B	62	;	Spare Device Control Block.



\*\*\*\*\* TRSDOS \*\*\*\*\*  
< SYSRES Page 2 >

PC	Object	I	Line	Source	..\SYSRES\lowcore\page2.s
		B	63		
	00000248	B	64	S1DCB\$ EQU \$	;1st spare DCB
		B	65		
	00000031	B	66	DCBKL\$ EQU JLD CB\$&0FFh+1	;Non-killable DCB"s
		B	0	include "page3.s"	; Page 3-System stack/Sysinfo, stack 128 byt
		B	1	SUBTITLE	"< SYSRES Page 3 Stack @end of this page.>"
		B	2	newpage	

\*\*\*\*\* TRSDOS \*\*\*\*\*

< SYSRES Page 3 Stack @end of this page.>

PC	Object	I	Line	Source ..\SYSRES\lowcore\page3.s	
		B	3		
		B	4	; Page 3 - System stack and Sysinfo section	
		B	5		
		B	6	ORG	memory.PAGE3\$ ; Page 3 - System stack/Sysinfo section, start stack 128 bytes
		00000380	B	7	STACK\$ EQU \$+80h
		B	8		
		B	9		
		A	50		
		A	51	ORG	zspace.STACK\$ ; Stack begins here.
		A	52		
		B	0	include "svc16-pause.s"	; Where pause will be.
		B	1	SUBTITLE	"< Pause and Sound routines, >"
		B	2	newpage	

\*\*\*\*\* TRSDOS \*\*\*\*\*  
 < Pause and Sound routines, >

PC	Object	I	Line	Source	.. \SYSRES \SVC \svc16-pause.s
		B	3	scope	
		B	4	; Update to LSDOS 6.3.1 2022-09-16 dpm	
		B	5		
000380	0000	B	6	DW 00	; Stack guard.
		B	7		
	00000382	B	8	PAUSEz EQU STACK\$+2	;Where pause will be
	00000382	B	9	zPAUSE EQU \$	;Pause routine
		B	10		
000382	C5	B	11	PUSH BC	;Save the count
000383	3A 7C 00	B	12	LD A,(SFLAG\$)	;If system (FAST)
000386	CB5F	B	13	BIT 3,A	;then double it
000388	C4 8C 03	B	14	CALL NZ,CDLOOP	;Call if fast
00038B	C1	B	15	POP BC	;Restore the count
00038C	0B	B	16	CDLOOP DEC BC	;Count down routine
00038D	78	B	17	LD A,B	
00038E	B1	B	18	OR C	
00038F	20 FB	B	19	JR NZ,CDLOOP	
000391	C9	B	20	RET	
		B	0	include "driver-sound.s"	; Driver for sound.
		B	1		
		B	2	ORG 0392h	
		B	3	SUBTITLE "< SOUND/ASM - Contains SOUND routine >"	
		B	4	NEWPAGE	

\*\*\*\*\* TRSDOS \*\*\*\*\*  
 < SOUND/ASM - Contains SOUND routine >

PC	Object	I	Line	Source	.. \SYSRES\drivers\driver-sound.s
		B	5	SCOPE	
		B	6		
		B	7	; zSOUND SVC-104 - Operates sound generator	
		B	8	; B => sound function	
		B	9	; Bits 0-2 <0-7> = note # (0 highest)	
		B	10	; Bits 3-7 <0-31> = relative sound duration	
		B	11	; All regs except A unchanged	
		B	12	; Z-flag set on exit	
		B	13	; Note that interrupts disabled during duration	
		B	14		
		B	15		
000392	C5	B	16	zSOUND	PUSH BC ;Save registers
000393	D5	B	17		PUSH DE
000394	E5	B	18		PUSH HL
		B	19		
		B	20	ring.BELL	U0DCB\$,0
		B+	20	scope	
000395	0600	B+	20		ld b,0
000397	C5	B+	20	__19	push bc
		B+	20		PUT.char U0DCB\$,BELL.asc,__1BELL.done
000398	113802	B+	20		ld de,U0DCB\$
00039B	0E07	B+	20		ld c,BELL.asc
		B+	20		SVC svc_put
00039D	3E04	B+	20		ld a,svc_put
00039F	EF	B+	20		restart28
		B+	20		ENDMAC SVC
		B+	20		IFMA 3
0003A0	C2 A3 03	B+	20		JP nz,__1BELL.done ; Returned with error.
		B+	20		else
		B+	20		ENDIF
		B+	20		endmac PUT.char
0003A3	C1	B+	20	__1BELL.done	pop bc
		B+	20		LOOP __19
		B+	20		scope
0003A4	10 F1	B+	20		djnz __19
		B+	20		endmac LOOP
		B+	20		endmac ring.BELL
0003A6	E1	B	21		POP HL
0003A7	D1	B	22		POP DE
0003A8	C1	B	23		POP BC
0003A9	AF	B	24		xor a
0003AA	C9	B	25		RET
0003AB	0D0A03	B	26	newline	db CR.asc,LF.asc,ETX.asc
		B	27		
		B	0		include "svc96-dechex.s" ; DECHEX.
		B	1		; Process decimal assignment.
		B	2		
		B	3		ORG 03E1h
		B	4		
0003E1	010000	B	5	zDECHEX	LD BC,0 ;Init value to zero
0003E4	7E	B	6	DEC1	LD A,(HL) ;P/u a char
0003E5	D630	B	7		SUB 30h ;Cvrt to binary
0003E7	D8	B	8		RET C ;Return if < "0"
0003E8	FE0A	B	9		CP 10 ;Ck for bad decimal
0003EA	D0	B	10		RET NC ;Ret if not 0-9
0003EB	C5	B	11		PUSH BC ;Exchange BC & HL
0003EC	E3	B	12		EX (SP),HL ;& save HL on stack
0003ED	29	B	13		ADD HL,HL ;Multiply by 10
0003EE	29	B	14		ADD HL,HL
0003EF	09	B	15		ADD HL,BC

\*\*\*\*\* TRSDOS \*\*\*\*\*  
< SOUND/ASM - Contains SOUND routine >

PC	Object	I	Line	Source	..\\SYSRES\\SVC\\svc96-dechex.s
0003F0	29	B	16	ADD	HL,HL
0003F1	0600	B	17	LD	B,0 ;Merge in new digit
0003F3	4F	B	18	LD	C,A ;New digit to C
0003F4	09	B	19	ADD	HL,BC ;& add it in
0003F5	44	B	20	LD	B,H ;Current value to BC
0003F6	4D	B	21	LD	C,L
0003F7	E1	B	22	POP	HL ;Recover HL pointer
0003F8	23	B	23	INC	HL
0003F9	18 E9	B	24	JR	DEC1 ;Loop
		B	0	include	"page4.s"
		B	1	SUBTITLE	"< Page 4 >"
		B	2	newpage	

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; Page 4 &gt;

PC	Object	I	Line	Source	.. \SYSRES\lowcore\page4.s
		B	3		scope
		B	4		
		B	5	ORG	memory.PAGE4\$ ; Page 4 - Miscellaneous stuff
		B	6		
000400	63	B	7	DB	63h ; Operating system version
000401	C9	B	8	ZERO\$ DB	C9h ; Config on BOOT, yes = 0.
	00000401	B	9	MAXDAY\$ EQU	\$-1 ; Max days per month.
		B	10		
000402	1F1C1F1E 1F1E1F1F	B	11	DB	31,28,31,30,31,30,31,31,30,31,30,31
00040A	1E1F1E1F				
		B	12		
00040E	F4F5	B	13	ZHIGH\$ DW	COREEND\$ ; Highest available memory
		B	14		
000410	5452532D 4F53	B	15	PAKNAM\$ db	"TRS-OS"
000416	20576869 736B6572	B	16	PAKVER\$ db	" Whiskers" ; <7.0.0>
00041E	73				
		B	17	;	DB 0,0 ; <631H>Level-1H
00041F	03	B	18		DB ETX.asc
		B	19		
		B	20	;	Command line input buffer & AUTO buffer area. Input buffer - 80 bytes.
		B	21		
000420	0D	B	22	INBUF\$ db	CR.asc ; Total input buffer size of 80 char.
000421	20 20 20 20 20 20	B	23	blk	78,020h
000427	20 20 20 20 20 20				
00042D	20 20 20 20 20 20				
000433	20 20 20 20 20 20				
000439	20 20 20 20 20 20				
00043F	20 20 20 20 20 20				
000445	20 20 20 20 20 20				
00044B	20 20 20 20 20 20				
000451	20 20 20 20 20 20				
000457	20 20 20 20 20 20				
00045D	20 20 20 20 20 20				
000463	20 20 20 20 20 20				
000469	20 20 20 20 20 20				
00046F	0D	B	24		db CR.asc
		B	25		
		B	26	;	CR.asc, ' ,CR.asc
		B	27	;	*backup :0 :1 (new,q=n) ',CR.asc
		B	28	;	*backup :0 :1 (s,i,q=n) ',CR.asc
		B	29	;	*COPY SYSTEM/JCL *D0 ',CR.asc
		B	30	;	copy system/jcl:0 dan:0 ',CR.asc
		B	31	;	*do dan/jcl ',CR.asc
		B	32	;	*dir (s,i,sort=no,n) ',CR.asc
		B	33	;	*dircheck :1 ',CR.asc
		B	34	;	*dct :1 ',CR.asc
		B	35	;	*device (b) ',CR.asc
		B	36	;	*dir \$/cmd (s,i,sort=no,n) ',CR.asc
		B	37	;	*format :1 (cyl=40,sides=2,q=n,abs) ',CR.asc
		B	38	;	*format :7 (abs,q=n,cyl=80,dden,sides=2) ',CR.asc
		B	39	;	*FORMAT :1 (SYSTEM) ',CR.asc
		B	40	;	*free :0 ',CR.asc
		B	41	;	*HOST25 ',CR.asc
		B	42	;	*LIST SYSTEM/JCL ',CR.asc
		B	43	;	*LIST SYSTEM/JCL (HEX,NS=ON) ',CR.asc
		B	44	;	*log :0 ',CR.asc
		B	45	;	*mapper :1 ',CR.asc
		B	46	;	*memdir ',CR.asc
		B	47	;	*memory ',CR.asc
		B	48	;	*memory (clear) ',CR.asc

\*\*\*\*\* TRSDOS \*\*\*\*\*  
< Page 4 >

PC	Object	I	Line	Source ..\SYSRES\lowcore\page4.s	
		B	49	; '*memory (clear)	',CR.asc
		B	50	; '*purge basic/\$:1 (q=n)	',CR.asc
		B	51	; '*purge :1 (s,i,q=n)	',CR.asc
		B	52	; '*remove sys0/sys.system6:0;	',CR.asc
		B	53	; '*SYSTEM (DRIVE=0,WP)	',CR.asc
		B	0	include "dct.s"	
		B	1	SUBTITLE "< Drive Code Table >"	
		B	2	newpage	

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; Drive Code Table &gt;

PC	Object	I	Line	Source	.. \SYSRES\lowcore\dct.s
		B	3		scope
		B	4		
	00000470	B	5	DCT\$	EQU \$ ; System drive code table (DCT).
		B	6		
	00000470	B	7	DCT0\$	EQU \$
000470	C3 58 0E	B	8		JP FDCDVR ; Logical volume 0. DCT+0,1,2 jumps to driver for this volume.
000473	40	B	9	DB	40h ; DCT+3 WP & double density. 44H.
000474	00	B	10	db	00h ; DCT+4 cxxxxxxb c=1 inhibit check drive, x unused.
000475	00	B	11	db	00h ; DCT+5 UNUSED <Current Cylinder>.
000476	27	B	12	db	27h ; DCT+6 Max Cyl=40.
000477	11	B	13	db	11h ; DCT+7 hhhsssssb hhh=heads assigned to volume. sssss=sectors per track..
000478	45	B	14	db	45h ; DCT+8 gggsssssb ggg=granule's per track -- sssss=sectors per granule.
000479	01	B	15	db	01h ; DCT+9 Directory location cylinder.
		B	16		
	0000047A	B	17	DCT1\$	EQU \$
00047A	C3 58 0E	B	18	jp	FDCDVR ; Logical volume 1.
00047D	40	B	19	DB	40h ; DCT+3 No WP & single density. 00H.
00047E	00	B	20	db	00h ; DCT+4 cxxxxxxb c=1 inhibit check drive, x unused.
00047F	00	B	21	db	00h ; DCT+5 UNUSED <Current Cylinder>.
000480	27	B	22	db	27h ; DCT+6 Max Cyl=80.
000481	11	B	23	db	11h ; DCT+7 hhhsssssb hhh=heads assigned to volume. sssss=sectors per track..
000482	45	B	24	db	45h ; DCT+8 gggsssssb ggg=granule's per track -- sssss=sectors per granule.
000483	01	B	25	db	01h ; DCT+9 Directory location cylinder.
		B	26		
	00000484	B	27	DCT2\$	EQU \$ ; Logical volume 2
000484	C9	B	28	RET	
000485	580E	B	29	dw	FDCDVR
000487	00	B	30	DB	00h ; DCT+3 No WP & single density. 00H.
000488	00	B	31	db	00h ; DCT+4 cxxxxxxb c=1 inhibit check drive, x unused.
000489	27	B	32	db	27h ; DCT+5 UNUSED <Current Cylinder>.
00048A	11	B	33	db	11h ; DCT+6 Max Cyl=35.
00048B	45	B	34	db	45h ; DCT+7 hhhsssssb hhh=heads assigned to volume. sssss=sectors per track..
00048C	01	B	35	db	01h ; DCT+8 gggsssssb ggg=granule's per track -- sssss=sectors per granule.
00048D	01	B	36	db	01h ; DCT+9 Directory location cylinder.
		B	37		
	0000048E	B	38	DCT3\$	EQU \$
00048E	C9	B	39	RET	
00048F	580E	B	40	dw	FDCDVR
000491	00	B	41	DB	00h ; DCT+3 No WP & single density. 00H.
000492	00	B	42	db	00h ; DCT+4 cxxxxxxb c=1 inhibit check drive, x unused.
000493	27	B	43	db	27h ; DCT+5 UNUSED <Current Cylinder>.
000494	11	B	44	db	11h ; DCT+6 Max Cyl=35.
000495	45	B	45	db	45h ; DCT+7 hhhsssssb hhh=heads assigned to volume. sssss=sectors per track..
000496	01	B	46	db	01h ; DCT+8 gggsssssb ggg=granule's per track -- sssss=sectors per granule.
000497	01	B	47	db	01h ; DCT+9 Directory location cylinder.
		B	48		
	00000498	B	49	DCT4\$	EQU \$ ; Logical volume 4.
000498	C9	B	50	RET	
000499	580E	B	51	dw	FDCDVR
00049B	00	B	52	DB	00h ; DCT+3 No WP & single density. 00H.
00049C	00	B	53	db	00h ; DCT+4 cxxxxxxb c=1 inhibit check drive, x unused.
00049D	27	B	54	db	27h ; DCT+5 UNUSED <Current Cylinder>.
00049E	11	B	55	db	11h ; DCT+6 Max Cyl=35.
00049F	45	B	56	db	45h ; DCT+7 hhhsssssb hhh=heads assigned to volume. sssss=sectors per track..
0004A0	01	B	57	db	01h ; DCT+8 gggsssssb ggg=granule's per track -- sssss=sectors per granule.
0004A1	01	B	58	db	01h ; DCT+9 Directory location cylinder.
		B	59		
	000004A2	B	60	DCT5\$	EQU \$ ; Logical volume 5.
0004A2	C9	B	61	RET	
0004A3	580E	B	62	dw	FDCDVR



\*\*\*\*\* TRSDOS \*\*\*\*\*  
 < Drive Code Table >

```

PC      Object      I   Line      Source ..\SYSRES\lowcore\dct.s
0004A5 00           B    63          DB    00h          ; DCT+3 No WP & single density. 00H.
0004A6 00           B    64          db    00h          ; DCT+4 cxxxxxxb c=1 inhibit check drive, x unused.
0004A7 27           B    65          db    27h          ; DCT+5 UNUSED <Current Cylinder>.
0004A8 11           B    66          db    11h          ; DCT+6 Max Cyl=35.
0004A9 45           B    67          db    45h          ; DCT+7 hhhsssssb hhh=heads assigned to volume. sssss=sectors per track..
0004AA 01           B    68          db    01h          ; DCT+8 gggsssssb ggg=granule's per track -- sssss=sectors per granule.
0004AB 01           B    69          db    01h          ; DCT+9 Directory location cylinder.
                                B    70
                                B    71      DCT6$      EQU          $          ; Logical volume 6.
0004AC C9           B    72          RET
0004AD 580E         B    73          dw    FDCDVR
0004AF 00           B    74          DB    00h          ; DCT+3 No WP & single density. 00H.
0004B0 00           B    75          db    00h          ; DCT+4 cxxxxxxb c=1 inhibit check drive, x unused.
0004B1 27           B    76          db    27h          ; DCT+5 UNUSED <Current Cylinder>.
0004B2 11           B    77          db    11h          ; DCT+6 Max Cyl=35.
0004B3 45           B    78          db    45h          ; DCT+7 hhhsssssb hhh=heads assigned to volume. sssss=sectors per track..
0004B4 01           B    79          db    01h          ; DCT+8 gggsssssb ggg=granule's per track -- sssss=sectors per granule.
0004B5 01           B    80          db    01h          ; DCT+9 Directory location cylinder.
                                B    81
                                B    82      DCT7$      EQU          $          ; Logical volume 7.
0004B6 C9           B    83          RET
0004B7 580E         B    84          dw    FDCDVR
0004B9 00           B    85          DB    00h          ; DCT+3 No WP & single density. 00H.
0004BA 00           B    86          db    00h          ; DCT+4 cxxxxxxb c=1 inhibit check drive, x unused.
0004BB 27           B    87          db    27h          ; DCT+5 UNUSED <Current Cylinder>.
0004BC 11           B    88          db    11h          ; DCT+6 Max Cyl=35.
0004BD 45           B    89          db    45h          ; DCT+7 hhhsssssb hhh=heads assigned to volume. sssss=sectors per track..
0004BE 01           B    90          db    01h          ; DCT+8 gggsssssb ggg=granule's per track -- sssss=sectors per granule.
0004BF 01           B    91          db    01h          ; DCT+9 Directory location cylinder.
                                B     0      include "sysinfo.s"
                                B     1      SUBTITLE    "< SYSINFO Section >"
                                B     2      ; Update to LSDOS 6.3.1 2022-09-16 dpm
                                B     3
                                B     4      ; SYSINFO - miscellaneous information
                                B     5
0004C0 FF           B     6      DSKTYP$  DB     -1      ;0 = DATA, <> 0 = SYS
0004C1 00           B     7          DB     0        ;Reserved
0004C2 00           B     8      DTPMT$   DB     0        ;Date prompt at boot
0004C3 00           B     9      TMPMT$   DB     0        ;Time prompt at boot
0004C4 00           B    10      RSTOR$   DB     0        ;Suppress restores on BOOT
0004C5 00           B    11          DB     0        ;Reserved
                                B    12
                                B    13      ; Note from Pete Cervasio about this byte:
                                B    14      ;
                                B    15      ; Byte X'C6' of the sysinfo sector holds the backup
                                B    16      ; limit count for backup-limited diskettes. If this
                                B    17      ; value is 0FFH, then BACKUP will refuse to back up
                                B    18      ; backup-limit protected files (BIT 4 of DIR+1), which
                                B    19      ; is documented as "file modified when date not set".
                                B    20      ; When this byte is 1-0FEH, then that's the number of
                                B    21      ; backups remaining. Normal disks just have a 0 here.
                                B    22      ; Bit 4 of GAT+X'CD' is set when a disk is a protected
                                B    23      ; disk. This was determined by tracing through the
                                B    24      ; BACKUP/CMD source code.
                                B    25
0004C6 00           B    26          db     0        ;X'C6' - Backup limit count.
                                B    27
0004C7 53756E4D 6F6E5475 B    28      DAYTBL$  DB          "SunMonTueWedThuFriSat"
0004CF 65576564 54687546
0004D7 72695361 74

```

\*\*\*\*\* TRSDOS \*\*\*\*\*  
< SYSINFO Section >

PC	Object	I	Line	Source
0004DC	4A616E46 65624D61	B	29	Source ..\SYSRES\lowcore\sysinfo.s
0004E4	72417072 4D61794A			MONTBL\$ DB "JanFebMarAprMayJunJulAugSepOctNovDec"
0004EC	756E4A75 6C417567			
0004F4	5365704F 63744E6F			
0004FC	76446563			
		B	30	
		B	31	
		B	32	; End of low core assignments
		A	59	
		A	60	ORG memory.PAGE5\$ ; ODVR/ASM - LS-DOS 6.2
		A	61	
		A	62	SUBTITLE "<SYSRES - Device I/O handling ~ Log out routine - display & log>"
		A	63	newpage

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt;SYSRES - Device I/O handling ~ Log out routine - display &amp; log&gt;

PC	Object	I	Line	Source	D:\TRS-OS\Jan2026\TRSDOS_7\TRSDOS.s
		A	64	scope	
		B	0	include "svc12-logot.s"	
		B	1		
		B	2	; Update to LSDOS 6.3.1 2022-09-16 dpm	
		B	3		
000500	CD 2D 05	B	4	zLOGOT	CALL zDSPLY
		B	5		
		B	0	include "svc11-logger.s"	
000503	3A3002	B	1	zLOGGER	LD A,(JLDCB\$) ;If NIL, don't do
000506	EE08	B	2	XOR 8	;anything.
000508	E608	B	3	AND 8	
00050A	C8	B	4	RET Z	
00050B	E5	B	5	PUSH HL	;Save pointer to command.
00050C	21 1D 05	B	6	LD HL,LOGBUF	;Get time string into buf.
00050F	E5	B	7	PUSH HL	
000510	CD 8D 07	B	8	CALL zTIME	
000513	E1	B	9	POP HL	
000514	113002	B	10	LD DE,JLDCB\$	;Log the time
000517	CD 30 05	B	11	CALL zMSG	
00051A	E1	B	12	POP HL	;Log the command
00051B	18 13	B	13	JR zMSG	
00051D	68683A6D 6D3A7373	B	14	LOGBUF	DB "hh:mm:ss ",ETX.asc
000525	202003	B	15		
		B	0	include "svc14-print.s"	
		B	1	; Line print routine.	
		B	2	; Update to LSDOS 6.3.1 2022-09-16 dpm	
		B	3		
000528	111802	B	4	zPRINT	LD DE,PRDCB\$ ;Printer DCB
00052B	18 03	B	5	JR zMSG	
		B	6		
00052D	111002	B	7	zDSPLY	LD DE,DODCB\$ ; Line display routine., get Video DCB.
		B	8		
		B	0	include "SVC13-msg.s"	
		B	1	; Device message routine.	
		B	2	; Update to LSDOS 6.3.1 2022-09-16 dpm	
		B	3		
000530	E5	B	4	zMSG	PUSH HL ;Save pointer to message
000531	7E	B	5	?\$1	LD A,(HL) ;P/u a message character
000532	FE03	B	6	CP ETX.asc	;Exit on ETX
000534	28 0D	B	7	JR Z,\$?3	
000536	FE0D	B	8	CP CR.asc	;Exit & put on ENTER
000538	28 06	B	9	JR Z,\$?2	
00053A	C4 45 06	B	10	CALL NZ,zPUT	;Else put the char
00053D	23	B	11	INC HL	& loop on no error
00053E	28 F1	B	12	JR Z,\$?1	;else fall thru & exit
000540	CC 45 06	B	13	?\$2	CALL Z,zPUT
000543	E1	B	14	?\$3	POP HL
000544	C9	B	15	RET	
		B	0	include "SVC105-cls.s"	
		B	1		
		B	2	; Update to LSDOS 6.3.1 2022-09-16 dpm	
		B	3	; Clear screen routine.	
		B	4		
000545	3E1C	B	5	zCLS	LD A,HOME ;Cursor home to 0,0
000547	CD 4D 05	B	6		CALL DSPBYT
00054A	C0	B	7	RET NZ	;Return on error
00054B	3E1F	B	8	LD A,CLRFRM	;Clear to end of frame
00054D	D5	B	9	DSPBYT:	PUSH DE
00054E	CD 42 06	B	10		CALL zDSP

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt;SYSRES - Device I/O handling ~ Log out routine - display &amp; log&gt;

PC	Object	I	Line	Source	.. \SYSRES \SVC \SVC105-cl.s
000551	D1	B	11	POP	DE
000552	C9	B	12	RET	
		B	0	include	"SVC106-ckbrkc.s"
		B	1		
		B	2	; Update to LSDOS 6.3.1 2022-09-16 dpm	
		B	3		
000553		B	4	zCKBRKC	; Check and Clear <BREAK> bit SVC.
		B	5		
000553	E5	B	6	PUSH	HL ;Save registers
000554	21 74 00	B	7	LD	HL,KFLAG\$ ;Point to KFLAG\$
000557	CB46	B	8	BIT	0,(HL) ;Check break bit
000559	28 1A	B	9	JR	Z,NOBRK ;and ret if none
00055B	F5	B	10	PUSH	AF ;Save flags
00055C	C5	B	11	PUSH	BC
00055D	D5	B	12	PUSH	DE
00055E	CB86	B	13	BRKTEST	RES 0,(HL) ;Reset the break bit
000560	01000B	B	14	LD	BC,0B00h ;Wait more than 1/30
000563	CD 82 03	B	15	CALL	PAUSEz ;of a second
000566	CB46	B	16	BIT	0,(HL) ;Test the bit again
000568	20 F4	B	17	JR	NZ,BRKTEST ;Loop until gone
00056A	110802	B	18	LD	DE,KIDCB\$ ;Point at keyboard &
00056D	3E03	B	19	LD	A,03 ;clear buffer with
00056F	CD 23 06	B	20	CALL	zCTL ;control 3 call
000572	D1	B	21	POP	DE
000573	C1	B	22	POP	BC ;Recover registers
000574	F1	B	23	POP	AF ;Recover FLAGS
000575	E1	B	24	NOBRK	POP HL
000576	C9	B	25	RET	
		B	0	include	"SVC09-keyin.s"
		B	1	SCOPE	
		B	2	; Update to LSDOS 6.3.1 2022-09-16 dpm	
		B	3	; Keyboard line input routine.	
		B	4	; Backspace to beginning of line.	
		B	5		
000577	CD DB 05	B	6	\$?4	CALL \$?6 ;Backspace
00057A	2B	B	7	DEC	HL ;Get the char prior.
00057B	7E	B	8	LD	A,(HL) ;to the current
00057C	23	B	9	INC	HL
00057D	FE0A	B	10	CP	0Ah ;Return if line feed
00057F	C8	B	11	RET	Z
000580	78	B	12	\$?5	LD A,B ;Check for empty buffer
000581	B9	B	13	CP	C
000582	20 F3	B	14	JR	NZ,\$?4 ;Loop if not
000584	C9	B	15	RET	;else return
		B	16		
000585	E5	B	17	zKEYIN	PUSH HL ;Save buffer pointer
000586	48	B	18	LD	C,B ;Set C = buffer size
000587	11 28 06	B	19	\$?1	LD DE,zKEY ; Init for standard input
00058A	3A 7C 00	B	20	LD	A,(SFLAG\$) ;If JCL is active,
00058D	E620	B	21	AND	20h ;then use the JCL input
00058F	28 02	B	22	JR	Z,\$?0 ;Must loop here in case
000591	1E 30	B	23	LD	E,zJCL&0FFh ;JCL exits with //STOP
000593	ED53 98 05	B	24	\$?0	LD (\$?1A+1),DE
000597	CD 00 00	B	25	\$?1A	call \$-\$ ;Get a key.
00059A	20 3D	B	26	JR	NZ,\$?3B ;Back on error
00059C	FE80	B	27	CP	80h ;Break?
00059E	28 75	B	28	JR	Z,\$?10
0005A0	FE20	B	29	CP	20h ;Go if not a control.
0005A2	30 20	B	30	JR	NC,\$?2
0005A4	FE0D	B	31	CP	CR.asc ;Carriage return?

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt;SYSRES - Device I/O handling ~ Log out routine - display &amp; log&gt;

PC	Object	I	Line	Source	.. \SYSRES \SVC \SVC09-keyin.s
0005A6	28 6E	B	32	JR	Z,\$?11
0005A8	FE1F	B	33	CP	1Fh ;Clear?
0005AA	28 25	B	34	JR	Z,\$?3
0005AC	11 87 05	B	35	LD	DE,\$?1 ;Set return address.
0005AF	D5	B	36	PUSH	DE
0005B0	FE08	B	37	CP	08h ;Backspace?
0005B2	28 27	B	38	JR	Z,\$?6
0005B4	FE18	B	39	CP	18h ;Backspace to BOL?
0005B6	28 C8	B	40	JR	Z,\$?5
0005B8	FE09	B	41	CP	09h ;Tab?
0005BA	28 3C	B	42	JR	Z,\$?8
0005BC	FE12	B	43	CP	'R'&1Fh ;CTL-R?
0005BE	28 2A	B	44	JR	Z,\$?7
0005C0	FE0A	B	45	CP	LF.asc ;Line feed?
0005C2	C0	B	46	RET	NZ ;Ret if none above
0005C3	D1	B	47	POP	DE ;Pop the return
0005C4	77	B	48	LD	(HL),A ;Stuff the char
0005C5	78	B	49	LD	A,B ;Check on buffer full
0005C6	B7	B	50	OR	A
0005C7	28 BE	B	51	JR	Z,\$?1 ;Loop if so
0005C9	7E	B	52	LD	A,(HL) ;else get char
0005CA	23	B	53	INC	HL ;& bump pointer
0005CB	05	B	54	DEC	B ;Count down
0005CC	CD 42 06	B	55	CALL	zDSP ;Display entry
0005CF	18 06	B	56	JR	\$?3A ;then loop
		B	57		
		B	58		
		B	59		
0005D1	CD 45 05	B	60	CALL	zCLS
0005D4	41	B	61	LD	B,C ;Reset to start of
0005D5	E1	B	62	POP	HL ;line & start of
0005D6	E5	B	63	PUSH	HL ;buffer
0005D7	28 AE	B	64	LD	A,\$?3A
0005D9	18 3B	B	65	JR	\$?3B
		B	66		
		B	67		
		B	68		
0005DB	78	B	69	LD	A,B ;If buffer is empty,
0005DC	B9	B	70	CP	C ;return
0005DD	C8	B	71	RET	Z
0005DE	2B	B	72	DEC	HL ;else do the backspace
0005DF	7E	B	73	LD	A,(HL)
0005E0	FE0A	B	74	CP	LF.asc ;Last char a linefeed?
0005E2	23	B	75	INC	HL
0005E3	C8	B	76	RET	Z ;Return if so
0005E4	2B	B	77	DEC	HL
0005E5	04	B	78	INC	B ;Add back one char
0005E6	3E08	B	79	LD	A,BS.asc ;Backspace the cursor
0005E8	18 58	B	80	jr	zDSP
		B	81		
		B	82		
0005EA	3A 6C 00	B	83	LD	A,(CFLAG\$) ;Test if SYS1 KEYIN bit
0005ED	E604	B	84	AND	4 ;is set (bit 2)
0005EF	C8	B	85	RET	Z ;Ignore CTL if not
0005F0	78	B	86	LD	A,B ;If not at 1st position,
0005F1	B9	B	87	CP	C ;don't permit it
0005F2	C0	B	88	RET	NZ
0005F3	E1	B	89	POP	HL ;Pop return to KEY
0005F4	E1	B	90	POP	HL ;Point to command buffer
0005F5	C3 2D 05	B	91	JP	zDSPLY ;Display the old command

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt;SYSRES - Device I/O handling ~ Log out routine - display &amp; log&gt;

PC	Object	I	Line	Source	.. \SYSRES \SVC \SVC09-keyin.s
		B	92		
		B	93	; Tab entered	
		B	94		
0005F8	E5	B	95	\$?8	PUSH HL ;Get pos on line
0005F9	CD F1 0D	B	96		CALL ADDR_2_ROWCOL ;Get row,col in HL
0005FC	7D	B	97		LD A,L ;Xfer column to A
0005FD	E1	B	98		POP HL
0005FE	E607	B	99		AND 7
000600	ED44	B	100		NEG ;Negate and add tab
000602	C608	B	101		ADD A,8
000604	5F	B	102		LD E,A ;Reg E has tab length
000605	78	B	103	\$?9	LD A,B ;Check on buffer full
000606	B7	B	104		OR A
000607	C8	B	105		RET Z
000608	3E20	B	106		LD A,' ' ;Put spaces until
00060A	77	B	107		LD (HL),A ;tab expanded
00060B	23	B	108		INC HL
00060C	CD 4D 05	B	109		CALL DSPBYT
00060F	C0	B	110		RET NZ
000610	05	B	111		DEC B ;Dec buffer remaining
000611	1D	B	112		DEC E ;Dec tab count
000612	C8	B	113		RET Z
000613	18 F0	B	114		JR \$?9
		B	115		
		B	116	; Exit KEYIN routine	
		B	117		
000615	37	B	118	\$?10	SCF ;BREAK exit with CF
000616	F5	B	119	\$?11	PUSH AF ;Save flag
000617	3E0D	B	120		LD A,CR.asc ;Stuff CR at end
000619	77	B	121		LD (HL),A
00061A	CD 42 06	B	122		CALL zDSP ;& display it
00061D	79	B	123		LD A,C ;Calculate # of chars
00061E	90	B	124		SUB B ;entered.
00061F	47	B	125		LD B,A
000620	F1	B	126		POP AF ;Rcvr flag
000621	E1	B	127		POP HL ;Restore buffer ptr
000622	C9	B	128		RET
		B	0		include "SVC05-ctl.s"
		B	1	; Update to LSDOS 6.3.1 2022-09-16 dpm	
		B	2		
		B	3	SCOPE	
		B	4		
		B	5	; Device byte I/O handler.	
		B	6	; C => Byte/character if PUT or CTL.	
		B	7	; DE => Device control block.	
		B	8		
000623	C5	B	9	zCTL	PUSH BC
000624	0604	B	10		LD B,4 ;Bit 2, CTL
000626	18 20	B	11		JR IOBGN
000628	CD 35 06	B	12	zKEY	CALL zKBD ;Scan the keyboard
00062B	C8	B	13		RET Z ;Ret if key available
00062C	B7	B	14		OR A ;Return if error
00062D	28 F9	B	15		JR Z,zKEY
00062F	C9	B	16		RET
		B	17		
		B	0		include "IOBGN.s"
		B	1		
		B	2	; Update to LSDOS 6.3.1 2022-09-16 dpm	
		B	3		
000630	110302	B	4	zJCL	LD DE,JCLCB\$ ;JCL file FCB

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt;SYSRES - Device I/O handling ~ Log out routine - display &amp; log&gt;

PC	Object	I	Line	Source	.. \SYSRES\lowcore\IOBGN.s
000633	18 03	B	5	JR	zGET
000635	110802	B	6	zKBD	LD DE,KIDCB\$ ;KIDCB\$, keyboard DCB
000638	C5	B	7	zGET	PUSH BC
000639	0601	B	8	LD	B,1 ;Bit 0, GET
00063B	18 0B	B	9	JR	IOBGN
00063D	111802	B	10	zPRT	LD DE,PRDCB\$ ;Printer DCB
000640	18 03	B	11	JR	zPUT
000642	111002	B	12	zDSP	LD DE,DODCB\$ ;DODCB\$ - Video DCB
000645	C5	B	13	zPUT	PUSH BC
000646	0602	B	14	LD	B,2 ;Bit 1, PUT
		B	15		
		B	16		; I/O begin on entry - DE DCB.
		B	17		
000648	DDE5	B	18	IOBGN	PUSH IX ;Save the registers
00064A	E5	B	19		PUSH HL
00064B	D5	B	20		PUSH DE ;Xfer DCB to IX
00064C	DDE1	B	21		POP IX
00064E	D5	B	22		PUSH DE
00064F	4F	B	23	LD	C,A ;Xfer the I/O char
000650	21 80 06	B	24	LD	HL,zRSTREG ;Restore register routine
000653	3A0202	B	25	LD	A,(LBANK\$) ;If bank 0 is not
000656	B7	B	26	OR	A ;resident, need to
000657	28 0E	B	27	JR	Z,\$?0 ; Skip <get> resident!
		B	28		
		B	29		; Some other bank is resident - invoke bank 0
		B	30		
000659	C5	B	31		PUSH BC ;Save reg again
00065A	AF	B	32	XOR	A ;Prepare for bank-0
00065B	47	B	33	LD	B,A
00065C	4F	B	34	LD	C,A
00065D	CD 77 08	B	35	CALL	zBANK ;Invoke bank-0
000660	60	B	36	LD	H,B ;Get old bank data
000661	69	B	37	LD	L,C ;into reg HL
000662	C1	B	38	POP	BC ;Rcvr BC
000663	E5	B	39	PUSH	HL ;Bank data to stack
000664	21 79 06	B	40	LD	HL,RSTBNK ;Set return address
000667	E5	B	41	\$_?0	PUSH HL ;to restore registers
000668	1A	B	42	LD	A,(DE) ;p/u DCB type byte
000669	B7	B	43	OR	A
00066A	C8	B	44	RET	Z ;Back if nothing
00066B	FE08	B	45	CP	8 ;Ck on GET/PUT/CTL
00066D	30 1A	B	46	JR	NC,zCHNIO ;Branch if special
00066F	DD6E01	B	47	LD	L,(IX+1) ;else p/u the vector
000672	DD6602	B	48	LD	H,(IX+2)
000675	78	B	49	\$_?1	LD A,B ;Xfer I/O code
000676	FE02	B	50	CP	2 ;Set flags state
000678	E9	B	51	JP	(HL)
000679	C1	B	52	RSTBNK	POP BC ;Get old bank data
00067A	F5	B	53		PUSH AF ;Can"t affect AF
00067B	79	B	54		LD A,C ;Request to A
00067C	CD 77 08	B	55		CALL zBANK ;Bring back original bank
00067F	F1	B	56		POP AF
000680	D1	B	57	zRSTREG	POP DE ;Restore regs
000681	E1	B	58		POP HL
000682	DDE1	B	59		POP IX
000684	C1	B	60		POP BC
000685	C9	B	61		RET
		B	0		include "chainio.s" ; IO chaining,
		B	1		SUBTITLE "< Chain IO. >"
		B	2		newpage

\*\*\*\*\* TRSDOS \*\*\*\*\*  
< Chain IO. >

PC	Object	I	Line	Source	.. \SYSRES\lowcore\chainio.s
		B	3		scope
		B	4		
		B	5		; Update to LSDOS 6.3.1 2022-09-16 dpm
		B	6		
000686	E5	B	7	\$?2	PUSH HL
000687	DDE1	B	8		POP IX
000689	DD6E01	B	9	zCHNIO	LD L,(IX+1) ;P/u vector address
00068C	DD6602	B	10		LD H,(IX+2)
00068F	DD7E00	B	11	\$?3	LD A,(IX+0) ;P/u the DCB type
000692	B7	B	12		OR A ;File Control Block?
000693	FA0013	B	13		JP M,zspace.BYTEIO\$
000696	CB5F	B	14		BIT 3,A ;Test NIL bit 2nd
000698	20 24	B	15		JR NZ,\$?5
00069A	CB67	B	16		BIT 4,A ;Routed?
00069C	20 E8	B	17		JR NZ,\$?2 ;Go get routed DCB
00069E	CB6F	B	18		BIT 5,A ;If not linked, then
0006A0	28 30	B	19		JR Z,\$?1 ;must be filtered
0006A2	E5	B	20		PUSH HL ;Point to the link DCB
0006A3	DDE1	B	21		POP IX
0006A5	DD7003	B	22		LD (IX+3),B ;Save the direction
0006A8	DDE5	B	23		PUSH IX
0006AA	CD 89 06	B	24		CALL zCHNIO ;I/O to 1st device
0006AD	DDE1	B	25		POP IX
0006AF	DD4603	B	26		LD B,(IX+3) ;P/u the direction
0006B2	20 0C	B	27		JR NZ,\$?6 ;Go on NZ flag
		B	28		
		B	29		; Z-flag on return - check input/output
		B	30		
0006B4	CB40	B	31		BIT 0,B ;If input & got char,
0006B6	DD6E04	B	32	\$?4	LD L,(IX+4) ;p/u the linked DCB
0006B9	DD6605	B	33		LD H,(IX+5)
0006BC	28 C8	B	34		JR Z,\$?2
0006BE	BF	B	35	\$?5	CP A
0006BF	C9	B	36		RET
		B	37		
		B	38		; 1st link got NZ condition - if input, get link
		B	39		
0006C0	CB40	B	40	\$?6	BIT 0,B ;Was it input/output?
0006C2	28 03	B	41		JR Z,\$?7 ;Output is error
0006C4	B7	B	42		OR A ;If A=0, then no input
0006C5	28 EF	B	43		JR Z,\$?4
0006C7	B7	B	44	\$?7	OR A
0006C8	C9	B	45		RET
		B	0		include "svc91-multiply16.s" ; MULTIPLY, .
		B	1		SUBTITLE "< MULDIV/ASM - 16 x 8 multiplication & division. >"
		B	2		SCOPE
		B	3		; 2022-09-16 updated V 6.3.1 dpm
		B	4		
		B	5		; Multiply HL by A - SVC 91
		B	6		; HL => multiplicand
		B	7		; A => multiplier
		B	8		; HLA <= 24-bit result
		B	9		; DE destroyed
		B	10		
0006C9	C5	B	11	zMUL16	PUSH BC ;Save reg BC
0006CA	EB	B	12		EX DE,HL ;Multiplicand to DE
0006CB	4F	B	13		LD C,A ;& multiplier to C
0006CC	210000	B	14		LD HL,0 ;Init value to zero
0006CF	7D	B	15		LD A,L ;in regs HLA
0006D0	0608	B	16		LD B,8 ;Init for 8-bit mult



\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; MULDIV/ASM - 16 x 8 multiplication &amp; division. &gt;

```

PC      Object      I  Line      Source  ..\SYSRES\SVC\svc91-multiply16.s
0006D2 29           B    17      $?1ADD   HL,HL      ;Shift to next place
0006D3 17           B    18          RLA          ;Use A for bits 16-23
0006D4 CB01         B    19          RLC      C      ;Multiply this bit?
0006D6 30 03        B    20          JR      NC,$?2      ;Go if not
0006D8 19           B    21          ADD   HL,DE      ;Else add multiplicand
0006D9 CE00         B    22          ADC   A,0        ;& any overflow to 16
0006DB 10 F5        B    23      $?2DJNZ  $?1        ;Loop for 8 bits
0006DD 4F           B    24          LD     C,A        ;Tempty save
0006DE 7D           B    25          LD     A,L        ;Xfer low-order to A
0006DF 6C           B    26          LD     L,H        ;Xfer mid-order to L
0006E0 61           B    27          LD     H,C        ;Xfer hi-order to H
0006E1 C1           B    28          POP    BC
0006E2 C9           B    29          RET
B    30
B    0              include "svc94-divide16.s"          ; DIVIDE
B    1              SUBTITLE  "< DIVIDE >"
B    2              scope
B    3
B    4              ; 2022-09-16 Update V 6.3.1 dpm
B    5
B    6              ;      Divide HL by A - SVC 94
B    7              ;      HL => dividend
B    8              ;      A  => divisor
B    9              ;      HL <= resulting quotient
B   10              ;      A  <= remainder
B   11
B   12
0006E3 D5           B   13      zDIV16          PUSH    DE      ;Save this reg pair
0006E4 57           B   14          LD      D,A        ;Xfer divisor to D
0006E5 1E10         B   15          LD      E,16      ;Init for 16-bits
0006E7 AF           B   16          XOR      A
0006E8 29           B   17      $?1          ADD   HL,HL      ;Rotate dividend
0006E9 17           B   18          RLA
0006EA 38 03        B   19          JR      C,$?2      ;carry into bit-16
0006EC BA           B   20          CP      D        ;Compare divisor
0006ED 38 02        B   21          JR      C,$?3      ;Go if no subtract
0006EF 92           B   22      $?2          SUB   D        ;else subtract divisor
0006F0 2C           B   23          INC   L        ;Set lo-order
0006F1 1D           B   24      $?3          DEC   E        ;Count down one bit
0006F2 20 F4        B   25          JR      NZ,$?1      ;Loop for 16-bits
0006F4 D1           B   26          POP    DE
0006F5 C9           B   27          RET
A   77
A   78
A   79
A   80              ;      ORG      06f6h
B    0              include "svc97-hex16ascii.s"          ; HEXDEC.
B    1              SCOPE
B    2              ; Update to LSDOS 6.3.1 2022-09-16 dpm
B    3
B    4              ; 2022-09-16 Update V 6.3.1 dpm
B    5
B    6              ;  zHEXDEC - SVC 97
B    7              ;  Routine to convert 16-bit hexadecimal to decimal
B    8              ;  HL => value
B    9              ;  DE => buffer pointer of 5-character buffer
B   10              ;  HL <= destroyed (always set to zero)
B   11              ;  DE <= Buffer+5
B   12              ;  BC <= destroyed
B   13              ;  Z  <= set

```

\*\*\*\*\* TRSDOS \*\*\*\*\*  
< DIVIDE >

PC	Object	I	Line	Source	.. \SYSRES \SVC \svc97-hex16ascii.s
		B	14		;
0006F6	0605	B	15	ZHEXDEC	LD B,5 ;Length max
0006F8	3E20	B	16	ZHEXD	LD A,' ' ;Load blank
0006FA	12	B	17	HEXDEC1	LD (DE),A ;To string
0006FB	13	B	18		INC DE ;Bump pointer
0006FC	10 FC	B	19	DJNZ	HEXDEC1 ;Go for length
0006FE	D5	B	20	PUSH	DE ;Save end +1
0006FF	1B	B	21	DEC	DE ;Adjust back
000700	3E0A	B	22	HEXDEC2	LD A,10 ;Base to convert to
000702	CD E3 06	B	23	CALL	ZDIV16 ;HL+A = HL/A
000705	C630	B	24	ADD	A,'0' ;Add ASCII to result
000707	12	B	25	LD	(DE),A ;to user string
000708	1B	B	26	DEC	DE ;Move back
		B	27		
		B	28		; Check if done
		B	29		
000709	7C	B	30	LD	A,H ;Get subtotal remainder
00070A	B5	B	31	OR	L ;Done?
00070B	20 F3	B	32	JR	NZ,HEXDEC2 ;Go till completed
00070D	D1	B	33	POP	DE ;Restore end+1
00070E	C9	B	34	RET	;Return Z
		B	0		include "driver-CLOCK.s" ; Hardware task stuff.
		B	1		SUBTITLE "< Real Time Clock interrupt processor - maintaining date/time via software >"
		B	2		newpage

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; Real Time Clock interrupt processor - maintaining date/time via software &gt;

PC	Object	I	Line	Source	.. \SYSRES\drivers\driver-CLOCK.s
		B	3		SCOPE
		B	4		
		B	5		; Time clock - Real Time Clock interrupt processor.
		B	6		
00070F	3A 97 0B	B	7	TIMTSK\$	LD A,(CRSAVE) ;If cursor not on,
000712	B7	B	8		OR A ;then don't blink
000713	21 7F 00	B	9		LD HL,VFLAG\$ ;Point to video flag
000716	28 29	B	10		jr z,\$?2 ;z,
		B	11		
		B	12		; Check if cursor needs blinking blink if so.
		B	13		
000718	CB7E	B	14		BIT 7,(HL) ; Check system INHIBIT blinking.
00071A	CBBE	B	15		RES 7,(HL) ; Allow blink next time
00071C	20 23	B	16		JR NZ,\$?2 ; Branch around cursor blink.
00071E	34	B	17		INC (HL) ; Increment the counter
00071F	CB5E	B	18		BIT 3,(HL) ; Test if = 8
000721	28 1E	B	19		JR Z,\$?2 ; IF <> 8 branch
000723	CB9E	B	20		RES 3,(HL) ; IF counter=8 then counter=0.
000725	CB76	B	21		BIT 6,(HL) ; BLINK/SOLID cursor test.
		B	22		
000727	20 18	B	23		JR nZ,\$?2 ;NOSOLID ; Branch for blink.
000729	00	B	24		nop ; nop to fill memory locations & retain length.
00072A	00	B	25		nop ;SET 5,(HL) ; Force SOLID mode.
		B	26		
00072B	CD 17 08	B	27	NOSOLID	CALL ENADIS_DO_RAM ; Bring up the video RAM
00072E	7E	B	28		LD A,(HL) ; Grab the toggle bit
00072F	EE20	B	29		XOR 20h ; and flip it
000731	77	B	30		LD (HL),A
000732	E620	B	31		AND 20h ; Was it on?
000734	ED5B 95 0B	B	32		LD DE,(CURSOR) ; Get the cursor pos
000738	3A 97 0B	B	33		LD A,(CRSAVE) ; and char under cursor
00073B	20 03	B	34		JR NZ,\$?1 ; Put character if flip on
00073D	3A 98 0B	B	35		LD A,(CRSCHAR) ; else put the cursor
000740	12	B	36	\$?1	LD (DE),A ; Put the char
000741	CB66	B	37	\$?2	BIT 4,(HL) ; Is clock on (VFLAG\$)?
000743	28 04	B	38		JR Z,\$?3 ; OFF? Go if OFF..
000745	11 4E 07	B	39		LD DE,CLOCK ; Set to display clock
000748	D5	B	40		PUSH DE
000749	C9	B	41	\$?3	ret
		B	42		
00074A	DD36031E	B	43		LD (IX+3),30 ; Reset for one second
	0000074D	B	44	HERTZ\$	EQU \$-1
		B	0		include "display_clock.s"
		B	1		; Clock display processor
		B	2		
00074E	CD 17 08	B	3	CLOCK	CALL ENADIS_DO_RAM ; Bring up the video
000751	2145F8	B	4		LD HL,CRTBGN\$+69 ; Point to display CRT
000754	18 37	B	5		jr zTIME
		B	6		
		A	84		
		A	85		; org 078dh
		B	0		include "SVC19-time.s"
		B	1		org 078DH
00078D	CD B8 12	B	2	zTIME	CALL GETTIME ; Original--> ld de,2fh now embeded in GETTIME.
000790	0E3A	B	3		LD C,':' ;Set the separator
000792	0603	B	4	TIME1	LD B,3 ;Init for three fields
000794	1A	B	5	TIME2	LD A,(DE) ;Get a field item
000795	362F	B	6		LD (HL),2Fh ;Init display
000797	34	B	7	TIME3	INC (HL) ;Bump until proper digit
000798	D60A	B	8		SUB 10

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; Real Time Clock interrupt processor - maintaining date/time via software &gt;

```

PC      Object      I  Line      Source  ..\SYSRES\SVC\SVC19-time.s
00079A 30 FB        B    9          JR      NC,TIME3
00079C C63A        B   10          ADD     A,3Ah          ;Correct the remainder
00079E 23          B   11          INC     HL              ;Bump to next display
00079F 77          B   12          LD      (HL),A          ;& stuff the digit
0007A0 23          B   13          INC     HL
0007A1 05          B   14          DEC     B
0007A2 C8          B   15          RET     Z              ;Back when done
0007A3 71          B   16          LD      (HL),C          ;else stuff separator
0007A4 23          B   17          INC     HL
0007A5 1B          B   18          DEC     DE              ;next field
0007A6 18 EC        B   19          JR      TIME2          ;& loop
                                A   87
                                B    0          include "svc18-date.s"          ; Return system DATE.
                                B    1          ; <631> Return formatted date, HL => user buffer
                                B    2
                                B    3          xdef    DATEz
                                B    4
                                B    5          ORG     07a8h
                                B    6
0007A8 CD 85 12     B    7          DATEz      call   GETDATE
                                B    8          ;          LD      DE,DATE$
0007AB 0E2F        B    9          LD          C,'/'
0007AD 18 E3        B   10          JR      TIME1
                                B    0          include "traceroutine.s"
                                B    1          ;   Dynamic Trace routine
                                B    2
                                B    3
0007AF 0000        B    4          PCSAVE$      DW      00          ;PC at entry to RST 38
                                B    5          TRACE_INT  EQU    $
                                B    6
0007B1 B307        B    7          DW      $+2          ;This TCB + 2
0007B3 2A AF 07     B    8          LD      HL,(PCSAVE$)      ;Get interrupt PC value
0007B6 EB          B    9          EX      DE,HL          ;Program counter to DE
0007B7 CD 17 08     B   10          CALL   ENADIS_DO_RAM      ;Bring up the video
0007BA 213EF8       B   11          LD      HL,CRTBGN$+62      ;CRT locn row 0, col 63
                                A   90
                                A   91          ;          ORG     07BDH
                                B    0          include "svc99-hex16decascii.s"          ; Hexadecimal display routine
                                B    1          ;   Hexadecimal display routine
                                B    2
0007BD 7A          B    3          ZHEX16      LD      A,D          ;Convert reg D to
0007BE CD C2 07     B    4          CALL   ZHEX8          ;two hex digits
0007C1 7B          B    5          LD      A,E          ;Convert reg E to
                                B    0          include "svc98-hex8ascii.s"
                                B    1          ;   Hexadecimal display routine
                                B    2
0007C2 F5          B    3          ZHEX8          PUSH   AF          ;two hex digits
0007C3 1F          B    4          RRA              ;Do left nybble first
0007C4 1F          B    5          RRA
0007C5 1F          B    6          RRA
0007C6 1F          B    7          RRA
0007C7 CD CB 07     B    8          CALL   HXD1          ;Bits 0-3 stuffed in hex
0007CA F1          B    9          POP     AF          ;Reget the byte
0007CB E60F        B   10          HXD1          AND     0Fh          ;& use right nybble
0007CD C690        B   11          ADD     A,90h          ;Convert nybble to hex
0007CF 27          B   12          DAA
0007D0 CE40        B   13          ADC     A,40h
0007D2 27          B   14          DAA
0007D3 77          B   15          LD      (HL),A          ;Stuff in (HL)
0007D4 23          B   16          INC     HL

```

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; Real Time Clock interrupt processor - maintaining date/time via software &gt;

PC	Object	I	Line	Source	.. \SYSRES \SVC \svc98-hex8ascii.s
0007D5	C9	B	17	RET	;Return Z
		B	0	include "scan-kflag.s"	
		B	1	; Scan for PAUSE or BREAK & set KFLAG\$.	
		B	2		
0007D6	00	B	3	KCKz	nop
0007D7	00	B	4		nop
0007D8	00	B	5		; CALL ENADIS_DO_RAM ; Bring up the keyboard
0007D9	21 74 00	B	6	LD	HL,KFLAG\$ ; Hang onto flag
0007DC	3A80F4	B	7	LD	A,(SHIFT) ; P/u SHIFT row & ignore
0007DF	E607	B	8	AND	7 ; CTRL key pressed
0007E1	2F	B	9	CPL	
0007E2	CB57	B	10	BIT	2,A
0007E4	C8	B	11	RET	Z ; Back if CTRL
		B	12		
		B	13	; Set carry flag if a SHIFT key is down	
		B	14		
0007E5	C601	B	15	ADD	A,1 ; Set CF if no SHIFT
0007E7	3F	B	16	CCF	; Set CF if SHIFT
0007E8	30 0B	B	17	JR	NC,KCK1 ; No pause if no SHIFT
0007EA	3A01F4	B	18	LD	A,(KB1) ; Test for "z"
		B	19	; IF	zUSA
0007ED	CB47	B	20	BIT	0,A
0007EF	28 0B	B	21	JR	Z,KCK1A ; Bypass if no "z"
0007F1	CBCE	B	22	SET	1,(HL) ; Turn on pause bit
0007F3	18 07	B	23	JR	KCK1A
		B	24		
		B	25	; Inhibit test of unshifted BREAK if nested ENA_DO	
		B	26		
0007F5	3A 32 08	B	27	KCK1	LD A,(OPREG_SV_PTR) ; If not at highest level
0007F8	D6 6F	B	28	SUB	OPREG_SV_AREA+1&0FFh ; then don't allow
0007FA	20 09	B	29	JR	NZ,KCK1B ; tasker BREAK handler
0007FC	3A40F4	B	30	KCK1A	LD A,(KB7) ; Check on BREAK & ENTER
0007FF	CB47	B	31	BIT	0,A ; Check on ENTER
000801	28 02	B	32	JR	Z,KCK1B ; Go if not
000803	CBD6	B	33	SET	2,(HL) ; else note set
000805	CB57	B	34	KCK1B	BIT 2,A ; Is <BREAK> depressed?
000807	F5	B	35	PUSH	AF
000808	28 0B	B	36	JR	Z,KCK2 ; Go if not
00080A	38 09	B	37	JR	C,KCK2 ; Ignore if shifted
00080C	3A 7C 00	B	38	LD	A,(SFLAG\$) ; Permit break bit only
00080F	CB67	B	39	BIT	4,A ; if BREAK enabled?
000811	20 02	B	40	JR	NZ,KCK2
000813	CBC6	B	41	SET	0,(HL) ; Turn on BREAK bit
000815	F1	B	42	KCK2	POP AF ; C=shift, NZ=break
000816	C9	B	43	RET	
		A	95		
		A	96	org	0817h
		B	0	include "videoRAM.s"	; Video RAM control code.
		B	1	SUBTITLE	"< SYSRES Video RAM bank handling. >"
		B	2	newpage	

\*\*\*\*\* TRSDOS \*\*\*\*\*  
 < SYSRES Video RAM bank handling. >

PC	Object	I	Line	Source	.. \SYSRES\lowcore\videoRAM.s
		B	3	SCOPE	
		B	4		
		B	5	; Routine to enable / disable video RAM, changing stack if necessary.	
		B	6		
000817	F5	B	7	ENADIS_DO_RAM push af	
000818	3A 85 F7	B	8	ld a,(pump_switch_value)	; Get value to place in switch instruction.
00081B	32 40 F7	B	9	ld (pump_switch),a	; Enable pump counter and pumping.
00081E	F1	B	10	pop af	
00081F	C9	B	11	ret	
		B	12		
000820	F3	B	13	di	
		B	14	;LD (HLSAV),HL	;Save HL but not on stack
		B	15	;PUSH AF	;Save AF
		B	16	;POP HL	
		B	17	; LD (AFSAV),HL	
		B	18	; LD HL,0C03h	;Can"t exceed X'F3FC"
		B	19	; ADD HL,SP	
000821	30 09	B	20	JR NC,\$?1	
		B	21		
		B	22	; <* Switch to the system stack *>	
		B	23		
000823	E1	B	24	POP HL	;Transfer RET address
000824	ED73 63 08	B	25	LD (SPSAV),SP	;Save stack pointer
000828	31 40 03	B	26	LD SP,STACK\$-40h	;Keep room at top
00082B	E5	B	27	PUSH hl	;Put RET back
00082C	21 46 08	B	28	\$?1 LD HL,DIS_DO_RAM	;Stack return to disable
00082F	E3	B	29	EX (SP),HL	;video RAM below RET
000830	E5	B	30	PUSH HL	
000831	21 6E 08	B	31	LD HL,OPREG_SV_AREA	
	00000832	B	32	OPREG_SV_PTR EQU \$-2	
000834	23	B	33	INC HL	;Get next save location
000835	3A 78 00	B	34	LD A,(OPREG\$)	;P/u port mask
000838	30 02	B	35	JR NC,\$?2	;Bypass if NC (no stack switch)
00083A	E67F	B	36	AND 7Fh	;Strip bit 7 to use as flag
00083C	77	B	37	\$?2 LD (HL),A	;Save current state
00083D	E6FC	B	38	AND 0FCh	;Strip SEL1 & SEL0
00083F	F682	B	39	OR 82h	;Set SEL1,0 = (1,0) & NZ condx
000841	18 15	B	40	JR D00PREG	;Set new assignment.
		B	41		
		B	42	org 0846h	;maintain lowcore
000846	C9	B	43	DIS_DO_RAM ret	;DI ;Routine to disable video RAM
000847	22 6B 08	B	44	LD (HLSAV),HL	;Can"t while we test stack
00084A	F5	B	45	PUSH AF	
00084B	E1	B	46	POP HL	;Save AF
00084C	22 66 08	B	47	LD (AFSAV),HL	
00084F	2A 32 08	B	48	LD HL,(OPREG_SV_PTR)	
000852	7E	B	49	LD A,(HL)	;P/u previous state
000853	CB7F	B	50	BIT 7,A	;Test if we switch stack
000855	CBFF	B	51	SET 7,A	;Make sure PAGE is set
000857	2B	B	52	DEC HL	
000858	22 32 08	B	53	D00PREG LD (OPREG_SV_PTR),HL	
00085B	32 78 00	B	54	LD (OPREG\$),A	;Restore port image.
		B	55	;OUT (ZOPREG),a	
00085E	00	B	56	nop	; Memory holder for out.
00085F	00	B	57	nop	; Memory holder for out.
000860	20 03	B	58	JR NZ,\$?3	
		B	59		
		B	60	; Switch back to the old stack	
		B	61		
000862	31 00 00	B	62	LD SP,\$-\$	;Get the old stack

\*\*\*\*\* TRSDOS \*\*\*\*\*  
< SYSRES Video RAM bank handling. >

PC	Object	I	Line	Source	..\\SYSRES\\lowcore\\videoRAM.s
	00000863	B	63	SPSAV	EQU \$-2
000865	21 00 00	B	64	\$?3	LD HL,\$-\$
	00000866	B	65	AFSAV	EQU \$-2
000868	E5	B	66		PUSH HL ;Restore AF
000869	F1	B	67		POP AF
00086A	21 00 00	B	68		LD HL,\$-\$ ;Restore HL
	0000086B	B	69	HLSAV	EQU \$-2
00086D	FB	B	70		EI
00086E	C9	B	71		RET
	0000086E	B	72	OPREG_SV_AREA	EQU \$-1
00086F	00000000 00000000	B	73		DB 0,0,0,0,0,0,0,0
		A	98		
		A	99	;	ORG 0877H
		B	0		include "svc102-banks.S"
		B	1		SUBTITLE "< Bank handling >"
		B	2		newpage

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; Bank handling &gt;

PC	Object	I	Line	Source	.. \SYSRES \SVC \svc102-banks.S
		B	3		SCOPE
		B	4		
		B	5		; Bank selection SVC handler
		B	6		; HL=> transfer address for function B=0
		B	7		; C => Bank request <0-2>; Set bit 7 to transfer.
		B	8		;
		B	9		; B => Request function
		B	10		; 0 => Select bank C
		B	11		; 1 => Reset in-use bit of bank C
		B	12		; 2 => Test in-use bit of bank C
		B	13		; 3 => Set in-use bit of bank C
		B	14		
		B	15		
000877	E67F	B	16	zBANK	AND 7Fh ;Strip possible bit 7
000879	FE03	B	17	CP	2+1 ;Bank out of range?
00087B	D2 ED 0D	B	18	JP	NC,PERR
		B	19		
00087E	05	B	20	DEC	B
00087F	FA B3 08	B	21	JP	M,SELECTBANK ;Go if function 0, bank select C
		B	22		
000882	0E86	B	23	LD	C,86h ;Set for reset BUR\$
000884	28 29	B	24	JR	Z,rst_bur ;Go if function 1, reset in use bit of bank C..
		B	25		
000886	0E46	B	26	LD	C,46h ;Set for test BUR\$
000888	05	B	27	DEC	B
000889	28 14	B	28	JR	Z,tst_but ;Go if function 2, test in-use bit of bank C.
		B	29		
00088B	05	B	30	DEC	B
00088C	28 09	B	31	JR	Z,set_bur ;Go if function 3, set in-use BUR\$ bit of bank C.
00088E	05	B	32	DEC	B
00088F	C2 ED 0D	B	33	PERRX	JP NZ,PERR ;SVC parameter error
000892	3A0202	B	34	LD	A,(LBANK\$) ;P/u current bank
000895	BF	B	35	CP	A
000896	C9	B	36	RET	
000897	47	B	37	set_bur	LD B,A ;Save the bank requested
000898	CD 9F 08	B	38	CALL	tst_but ;Test if in use already
00089B	C0	B	39	RET	NZ ;Back if error
00089C	78	B	40	LD	A,B ;Reget the request #
00089D	0EC6	B	41	LD	C,0C6h ;Set for set BUR\$
00089F	E607	B	42	tst_but	AND 7 ;Strip to bank 0-7
0008A1	07	B	43	RLCA	;Shift <0-2> to <3-5>
0008A2	07	B	44	RLCA	
0008A3	07	B	45	RLCA	
0008A4	B1	B	46	OR	C ;Merge the code type
0008A5	32 B0 08	B	47	LD	(rst_bur+1),A ;Change the OP code
0008A8	AF	B	48	XOR	A ;Init Z-flag
0008A9	3E08	B	49	LD	A,8 ;Init "Device not avail
0008AB	E5	B	50	PUSH	HL ;Don't alter HL
0008AC	210002	B	51	LD	HL,BUR\$ ;Point to bank-used-RAM
0008AF	CB46	B	52	rst_bur	BIT 0,(HL) ;*** Modified instruction
0008B1	E1	B	53	POP	HL
0008B2	C9	B	54	RET	
		B	55		
0008B3	E5	B	56	SELECTBANK	PUSH HL ;Ck if stack is in upper
0008B4	210580	B	57	LD	HL,8005h ;bank area
0008B7	39	B	58	ADD	HL,SP
0008B8	E1	B	59	POP	HL
		B	60		
0008B9	DA ED 0D	B	61	JP	C,PERR ;Error if > X"7FFE"
		B	62		



\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; Bank handling &gt;

PC	Object	I	Line	Source	.. \SYSRES \SVC \svc102-banks.S
0008BC	FE01	B	63	CP	1 ;Change <0, 1, 2, 3>
0008BE	17	B	64	RLA	;to <1, 2, 4, 6>
0008BF	47	B	65	LD	B,A ;& save for later
0008C0	3A0102	B	66	LD	A,(BAR\$) ;P/u Bank Avail Ram
0008C3	A0	B	67	AND	B ;Is the bank installed?
0008C4	20 C9	B	68	JR	NZ,PERRX ;Error if not in machine
0008C6	78	B	69	LD	A,B ;Get the requested bank
0008C7	1F	B	70	RRA	;Change <1, 2, 4> to
0008C8	3F	B	71	CCF	;<0, 2, 3> {CF on 0
0008C9	CE00	B	72	ADC	A,0 ;switched to 2 & 4}
0008CB	07	B	73	RLCA	;Shift bits 0-1
0008CC	07	B	74	RLCA	;to 4-5 (MBIT0,1)
0008CD	07	B	75	RLCA	
0008CE	07	B	76	RLCA	
0008CF	47	B	77	LD	B,A ;Save bit mask
0008D0	3A 78 00	B	78	LD	A,(OPREG\$) ;P/u current memory
0008D3	E68F	B	79	AND	08Fh ;configuration &
0008D5	B0	B	80	OR	B ;mask off old &
0008D6	32 78 00	B	81	LD	(OPREG\$),A ;merge the new
0008D9	00	B	82	nop	
0008DA	00	B	83	nop	; OUT (084h),a ; Switch hardware.....
0008DB	3A0202	B	84	LD	A,(LBANK\$) ;Get old bank #
0008DE	47	B	85	LD	B,A ;& save it
0008DF	79	B	86	LD	A,C ;P/u new bank #
0008E0	E67F	B	87	AND	7Fh ;Strip any bit-7
0008E2	320202	B	88	LD	(LBANK\$),A ;& save new bank #
0008E5	A9	B	89	XOR	C ;Keep bit-7
0008E6	B0	B	90	OR	B ;Merge in new bank #
0008E7	4F	B	91	LD	C,A ;& replace into C
0008E8	CB79	B	92	BIT	7,C ;Transfer to new bank?
0008EA	0600	B	93	LD	B,0 ;Init for invoke later
0008EC	C8	B	94	RET	Z ;No if bit-7 = 0
0008ED	E3	B	95	EX	(SP),HL ;Exchange RET with new
0008EE	BF	B	96	CP	A ;transfer & go to it.
0008EF	C9	B	97	RET	
		A	101		
		A	102		SUBTITLE "<Device I/O zspace.DRIVERS\$ >"
		A	103		
		A	104		org zspace.DRIVERS\$
		A	105		
000008F0		A	106	z\$SYS	EQU \$ ; Pointer for zGTMOD.
		A	107		
		B	0		INCLUDE "driver-KIDVR.s" ; Keyboard driver.
		B	1		SUBTITLE "< Console Keyboard Driver, KIDVR/ASM - LS-DOS 6.2 >"
		B	2		newpage

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; Console Keyboard Driver, KIDVR/ASM - LS-DOS 6.2 &gt;

PC	Object	I	Line	Source	.. \SYSRES\drivers\driver-KIDVR.s
		B	3		SCOPE
		B	4		
		B	5	; TYPBUF+0 = On/Off Flag	
		B	6	; TYPBUF+1 = Storage pointer	
		B	7	; TYPBUF+2 = Retrieve pointer	
		B	8	; TYPBUF+3 = Start of actual buffer	
		B	9		
		B	10	ORG	driver.KI
0008F0	18 31	B	11	KIDVR	JR KIBGN ;Branch around linkage
0008F2	870B	B	12	DW	driver.DO-1 ;KILAST ;Last byte used
0008F4	03244B49	B	13	DB	3, "\$KI"
0008F8	0802	B	14	DW	KIDCB\$ ;Pointer to DCB
0008FA	0000	B	15	DW	0 ;Spare
0008FC	00	B	16	KIDATA\$	DB 0 ;Last key entered
0008FD	00	B	17	DB	0 ;Repeat time check
	00000002	B	18	RPTINIT	EQU \$-KIDATA\$
0008FE	16	B	19	DB	22 ;22 * 33.3ms = .733 sec
	00000003	B	20	RPTRATE	EQU \$-KIDATA\$
0008FF	02	B	21	DB	2 ;2 x RTC rate
	00000004	B	22	KBROW0	EQU \$-KIDATA\$
000900	FFFFFFFF	B	23	DB	-1, -1, -1, -1 ;Image of rows 0-3
	00000008	B	24	KBROW4	EQU \$-KIDATA\$
000904	FFFF	B	25	DB	-1, -1 ;Image of rows 4-5
	0000000A	B	26	KBROW6	EQU \$-KIDATA\$
000906	FFFF	B	27	DB	-1, -1 ;Image of rows 6-7
		B	28		
		B	29	;	Conversion table for keyboard row 7/8
		B	30		
000908	0D1D1F1F	B	31	KBTBL	DB CR.asc, 1Dh, 1Fh, 1Fh ;<ENTER> <CLEAR>
00090C	80000B1B	B	32	DB	80h, 0, 0Bh, 1Bh ;<BREAK> <UPARW>
000910	0A1A0818	B	33	DB	LF.asc, 1Ah, 8, 18h ;<DNARW> <LTARW>
000914	09192020	B	34	DB	9, 19h, 20h, 20h ;<RTARW> <SPACE>
000918	81918292	B	35	DB	81h, 91h, 82h, 92h ;<F1> <F2>
00091C	8393	B	36	DB	83h, 93h ;<F3>
		B	37		
		B	38	;	Table to generate 5B-5F, 7B-7F
		B	39		
00091E	2C2F2E3B 0D	B	40	SPCLTB	DB ", / . ; ", CR.asc
		B	41		
		B	42	;	Entry to keyboard driver
		B	43		
000923	79	B	44	KIBGN	LD A, C ;Get the character
000924	F5	B	45	PUSH	AF ;Save flags
000925	CD 89 00	B	46	CALL	ZKITSK ;Hook for KI task
000928	F1	B	47	POP	AF
		B	48		
		B	49	;	Screen print (Control-*) processing
		B	50		
000929	CD D5 0A	B	51	CALL	TYPABD ;Chain downstream.
		B	52		
00092C	D0	B	53	RET	NC ;Ret if not <CONTROL>
00092D	F5	B	54	PUSH	AF ;Save flag state
00092E	FE3A	B	55	CP	' : '
000930	28 02	B	56	JR	Z, \$?1 ;Go if screen print
000932	F1	B	57	POP	AF
000933	C9	B	58	RET	
		B	59		
		B	60	;	Perform a screen print
		B	61		
000934	F1	B	62	\$?1	POP AF ;Clean the stack

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; Console Keyboard Driver, KIDVR/ASM - LS-DOS 6.2 &gt;

PC	Object	I	Line	Source	.. \SYSRES\drivers\driver-KIDVR.s
000935	3A 6D 00	B	63	zVDPRT	LD A,(DFLAG\$) ;Check on Graphic bit
000938	07	B	64		RLCA
000939	3E3E	B	65		LD A,3Eh ;Init for LD a, '.'
00093B	30 02	B	66		JR NC,\$+4 ;Go if not Graphic
00093D	3EFE	B	67		LD A,0FEh ;Change to CPR n
00093F	32 5C 09	B	68		LD (\$?4),A ;Stuff cpr or ld
000942	21 74 00	B	69		LD HL,KFLAG\$ ;Reset the BREAK bit
000945	CB86	B	70		RES 0,(HL)
000947	E5	B	71		PUSH HL ;Save on stack
000948	210000	B	72		LD HL,0 ;Init for row,col
00094B	0601	B	73	\$?2	LD B,1 ;Get a character at the
00094D	CD 99 0B	B	74		CALL zVDCTL ;row-H, col-L
000950	20 27	B	75		JR NZ,\$?6 ;Go on error
000952	FE20	B	76		CP 20h
000954	30 02	B	77		JR NC,\$+4 ;Convert control codes
000956	C640	B	78		ADD A,40h ;to cap A-Z, +
000958	FE80	B	79		CP 80h ;Cvrt anything from X"80"
00095A	38 02	B	80		JR C,\$?5 ;thru X'FF' to a '.'
00095C	3E2E	B	81	\$?4	LD A, '.' ;unless graphic bit set
00095E	CD 3D 06	B	82	\$?5	CALL zPRT ;Print the char & loop
000961	20 16	B	83		JR NZ,\$?6
000963	2C	B	84		INC L ;Bump column counter
000964	7D	B	85		LD A,L ;Check for end-of-line
000965	D650	B	86		SUB 80
000967	20 E2	B	87		JR NZ,\$?2 ;Loop if not EOL
000969	6F	B	88		LD L,A ;Reset to column 0
00096A	2D	B	89		DEC L ;Adj for CR force
00096B	E3	B	90		EX (SP),HL ;Get KFLAG\$
00096C	CB46	B	91		BIT 0,(HL) ;Exit with A=0 on
00096E	E3	B	92		EX (SP),HL ;entrance of BREAK
00096F	20 08	B	93		JR NZ,\$?6
000971	24	B	94		INC H ;Bump row counter
000972	7C	B	95		LD A,H ;Test for end of screen
000973	FE18	B	96		CP 24
000975	3E0D	B	97		LD A,CR.asc
000977	20 E5	B	98		JR NZ,\$?5 ;Put the CR & loop
000979	3E0D	B	99	\$?6	LD A,CR.asc ;Close out with CR if
00097B	CD 3D 06	B	100		CALL zPRT ;BREAK key detected
00097E	E1	B	101		POP HL ;Pop the KFLAG
00097F	CB86	B	102		RES 0,(HL) ;& reset BREAK bit
000981	18 32	B	103		JR NOCHAR
		B	104		
		B	105		; Driver to scan the keyboard
		B	106		
		B	107		SCOPE
000983	DD21 FC 08	B	108	KISCAN	LD IX,KIDATA\$ ;Point to data area
000987	21 00 09	B	109		LD HL,KIDATA\$+KBROW0 ;Load kbd image start
00098A	0101F4	B	110		LD BC,KB0 ;Load start of keyboard
00098D	1600	B	111		LD D,0 ;Zero the key counter
00098F	0A	B	112	\$?1	LD A,(BC) ;Load 1st char from kbd
000990	5F	B	113		LD E,A
000991	AE	B	114		XOR (HL) ;XOR with old value
000992	20 26	B	115		JR NZ,\$?2 ;Go if different
000994	14	B	116		INC D ;Bump key counter
000995	23	B	117		INC HL ;Bump image pointer
000996	CB01	B	118		RLC C ;Go to next row
000998	F2 8F 09	B	119		JP P,\$?1 ;Loop until end of rows
00099B	0A	B	120		LD A,(BC) ;Get row 7
00099C	E678	B	121		AND 078h ;Strip SHIFT, CTL
00099E	5F	B	122		LD E,A

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; Console Keyboard Driver, KIDVR/ASM - LS-DOS 6.2 &gt;

PC	Object	I	Line	Source	..\\SYSRES\\drivers\\driver-KIDVR.s
00099F	AE	B	123	XOR	(HL)
0009A0	20 18	B	124	JR	NZ,\$?2
0009A2	DD7E00	B	125	LD	A,(IX+0) ;Key down? It"s same as
0009A5	B7	B	126	OR	A ;the last if so
0009A6	28 0D	B	127	JR	Z,NOCHAR ;Ret if no key
0009A8	3A 2C 00	B	128	LD	A,(TIMER\$) ;Do we repeat the
0009AB	DD9601	B	129	SUB	(IX+1) ;same key?
0009AE	DD96 02	B	130	SUB	(IX+RPTINIT) ;Beyond 0.75 seconds?
0009B1	0A	B	131	LD	A,(BC) ;Get back row 7
0009B2	5F	B	132	LD	E,A
0009B3	38 69	B	133	JR	C,\$?10 ;Go if yes
0009B5	F601	B	134	NOCHAR OR	1 ;Else don"t repeat
0009B7	3E00	B	135	LD	A,0 ;Show NZ with A=0
0009B9	C9	B	136	RET	
		B	137		
		B	138	; Found change in key matrix	
		B	139		
0009BA	73	B	140	\$?2 LD	(HL),E ;Stuff KB image with new
0009BB	A3	B	141	AND	E ;KB row value
0009BC	CA 88 0A	B	142	JP	Z,NOKEY ;Go if new is none
		B	143	;	
		B	144	; Convert the depressed key	
		B	145	;	
0009BF	5F	B	146	LD	E,A ;Save the active bit
0009C0	7A	B	147	LD	A,D ;Calculate 8 * row
0009C1	07	B	148	RLCA	
0009C2	07	B	149	RLCA	
0009C3	07	B	150	RLCA	
0009C4	57	B	151	LD	D,A ;Save 8 * row
0009C5	0E01	B	152	LD	C,1 ;Add 8 * row + column
0009C7	79	B	153	\$?3 LD	A,C
0009C8	A3	B	154	AND	E ;Check if bits match
0009C9	20 19	B	155	JR	NZ,\$?6 ;Go if match
0009CB	14	B	156	INC	D ;else bump value
0009CC	CB01	B	157	RLC	C ;Shift compare bit
0009CE	18 F7	B	158	JR	\$?3 ;Loop to test next
		B	159		
		B	160	; Key pressed was not an alpha	
		B	161		
0009D0	D690	B	162	\$?4 SUB	90h ;Adjust for non-alpha
0009D2	30 52	B	163	JR	NC,\$?9 ;Go if special key
0009D4	C640	B	164	ADD	A,40h ;Cvrt to numeric/symbol
0009D6	FE3C	B	165	CP	3Ch ;Manipulate to get
0009D8	38 02	B	166	JR	C,\$?5 ;proper code
0009DA	EE10	B	167	XOR	10h
0009DC	CB43	B	168	\$?5 BIT	0,E ;Check SHIFT
0009DE	28 60	B	169	JR	Z,\$?11 ;Go if unshift
0009E0	EE10	B	170	XOR	10h ;else adjust for SHIFT
0009E2	18 5C	B	171	JR	\$?11
		B	172		
		B	173	; Found a key - Set up the function codes	
		B	174		
0009E4	3A80F4	B	175	\$?6 LD	A,(SHIFT) ;P/u the SHIFT key
0009E7	5F	B	176	LD	E,A ;Merge RH & LH Shift keys
0009E8	E602	B	177	AND	2 ;Only merge bit 1
0009EA	0F	B	178	RRCA	;Bit 1 to bit 0
0009EB	B3	B	179	OR	E ;Merge bits 0 & 1
0009EC	5F	B	180	LD	E,A ;Value of (RHorLH) shift
0009ED	7A	B	181	LD	A,D ;Load semi-converted
0009EE	C660	B	182	ADD	A,60h ;If alpha, convert to

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; Console Keyboard Driver, KIDVR/ASM - LS-DOS 6.2 &gt;

PC	Object	I	Line	Source	.. \SYSRES\drivers\driver-KIDVR.s
0009F0	FE80	B	183	CP 80h	;correct value
0009F2	21 74 00	B	184	LD HL,KFLAG\$	
0009F5	30 D9	B	185	JR NC,\$?4	;Go if not alpha
		B	186		
		B	187		; Alpha <z-Z> - If caps lock or <SHIFT>,
		B	188		; Convert to caps unless CLEAR
		B	189		
0009F7	CB53	B	190	BIT 2,E	;CTRL key down?
0009F9	20 18	B	191	JR NZ,CTLA2Z	;CTRL sets <00-1A>
0009FB	FE60	B	192	CP 60h	;Invert z and `
0009FD	20 04	B	193	JR NZ,\$?7	
0009FF	EE20	B	194	XOR 20h	;Invert & bypass test
000A01	18 0A	B	195	JR \$?8	;for CAPs lock
000A03	DDCB 0A 4E	B	196	BIT 1,(IX+KBROW6)	;If CLEAR, don't test
000A07	20 04	B	197	JR NZ,\$?8	;for CAPs lock
000A09	CB6E	B	198	BIT 5,(HL)	;Caps lock?
000A0B	20 31	B	199	JR NZ,TGLCASE	
000A0D	CB43	B	200	BIT 0,E	;Shift key down?
000A0F	28 2F	B	201	JR Z,\$?11	;Bypass if not shifted
000A11	18 2B	B	202	JR TGLCASE	;Convert to upper case
000A13	D660	B	203	CTLA2Z SUB 60h	;Convert CTRL A-Z
000A15	20 29	B	204	JR NZ,\$?11	;Go on A-Z
000A17	CB43	B	205	BIT 0,E	;Shifted?
000A19	37	B	206	SCF	;Set C-flag for CTL-z
000A1A	C8	B	207	RET Z	; & return if unshifted
000A1B	3E1C	B	208	LD A,1Ch	;else set EOF error
000A1D	C9	B	209	RET	
000A1E	3A 2C 00	B	210	LD A,(TIMER\$)	;Advance time check
000A21	DD86 03	B	211	ADD A,(IX+RPTRATE)	;by 0.067 seconds
000A24	18 72	B	212	JR \$?12	;Go output the key
		B	213		
		B	214		; Special keys - rows 6 & 7
		B	215		
000A26	FE0B	B	216	CP 11	;Compress F1-F3 keys
000A28	28 4F	B	217	JR Z,CAPSKEY	;while checking for CAP
000A2A	38 02	B	218	JR C,\$+4	;F1-F3 to 8-10
000A2C	D604	B	219	SUB 4	
000A2E	21 08 09	B	220	LD HL,KBTL	;Pt to special char table
000A31	07	B	221	RLCA	;Index into table,
000A32	CB43	B	222	BIT 0,E	;shifted code is +1
000A34	28 01	B	223	JR Z,\$+3	
000A36	3C	B	224	INC A	
000A37	4F	B	225	LD C,A	;Index the table
000A38	0600	B	226	LD B,0	
000A3A	09	B	227	ADD HL,BC	
000A3B	7E	B	228	LD A,(HL)	;Load char from table
000A3C	18 02	B	229	JR \$?11	;Bypass restore of char
000A3E	EE20	B	230	TGLCASE XOR 20h	;Toggle the case
000A40	FE80	B	231	CP 80h	;BREAK key?
000A42	20 0F	B	232	JR NZ,\$?11A	;Ck on <BREAK> disable
000A44	21 7C 00	B	233	LD HL,SFLAG\$	;Break disabled?
000A47	CB66	B	234	BIT 4,(HL)	
000A49	20 07	B	235	JR NZ,\$?11B	;Don't set bit if disabl
000A4B	21 74 00	B	236	LD HL,KFLAG\$	
000A4E	CBC6	B	237	SET 0,(HL)	;otherwise set it
000A50	18 01	B	238	JR \$?11A	
000A52	17	B	239	RLA	;Rotate bit-7 out
000A53	DDCB 0A 4E	B	240	BIT 1,(IX+KBROW6)	;CLEAR key pressed?
000A57	28 0E	B	241	JR Z,NOTALPH	;Go if not down
000A59	57	B	242	LD D,A	;Save code

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; Console Keyboard Driver, KIDVR/ASM - LS-DOS 6.2 &gt;

PC	Object	I	Line	Source	.. \SYSRES\drivers\driver-KIDVR.s
000A5A	CBAF	B	243	RES	5,A ;Set to upper-case for
000A5C	D641	B	244	SUB	'A' ;test A-Z
000A5E	FE1A	B	245	CP	'Z'-'A'+1
000A60	7A	B	246	LD	A,D ;Get back actual char
000A61	30 02	B	247	JR	NC,\$+4 ;Go if not A-Z
000A63	EE20	B	248	XOR	20h ;Shift keyboard case
000A65	F680	B	249	OR	80h ;Set bit 7 for CLEAR key
000A67	CB43	B	250	NOTALPH	BIT 0,E ;SHIFT key down?
000A69	28 19	B	251	JR	Z,FIXCLR ;Go if not
000A6B	FE9F	B	252	GOTSHFT	CP 9Fh ;Shift-clear?
000A6D	28 13	B	253	JR	Z,FIXSCL ;Go if so
000A6F	FE20	B	254	TSTSPA	CP 20h ;Shift 0 or shift sp?
000A71	20 16	B	255	JR	NZ,KEYOK ;Go if not
000A73	DDCB 08 46	B	256	BIT	0,(IX+KBROW4) ;Ck zero key
000A77	28 10	B	257	JR	Z,KEYOK ;Go if not down
		B	258		
		B	259		; Toggle the caps lock bit in the KFLAG\$
		B	260		
000A79	3E20	B	261	CAPSKEY	LD A,20h ;CAPs wasn't 20h
000A7B	21 74 00	B	262	CASHK\$	LD HL,KFLAG\$ ;Reverse case by
000A7E	AE	B	263	XOR	(HL) ;flipping bit 5
000A7F	77	B	264	LD	(HL),A
000A80	18 06	B	265	JR	NOKEY
000A82	EE80	B	266	FIXSCL	XOR 80h ;Reset bit 7
000A84	FE9F	B	267	FIXCLR	CP 9Fh ;Clear key?
000A86	20 01	B	268	JR	NZ,KEYOK ;Go if not
000A88	AF	B	269	NOKEY	XOR A
000A89	DD7700	B	270	KEYOK	LD (IX+0),A
000A8C	018401	B	271	LD	BC,0184h ;Delay
000A8F	CD 82 03	B	272	TYPHK\$	CALL PAUSEz
000A92	3A 2C 00	B	273	LD	A,(TIMER\$) ;Set initialization
000A95	DD86 02	B	274	DELAY2	ADD A,(IX+RPTINIT) ;repeat key delay
000A98	DD7701	B	275	\$?12	LD (IX+1),A ;Save new repeat time
000A9B	DD7E00	B	276	LD	A,(IX+0) ;Check if any key
000A9E	B7	B	277	OR	A ;code was saved
000A9F	CA B5 09	B	278	JP	Z,NOCHAR ;Ret if none
000AA2	CB53	B	279	BIT	2,E ;Shift key down?
000AA4	37	B	280	SCF	;Init carry
000AA5	20 04	B	281	JR	NZ,SPECL ;Ret if CTRL
000AA7	3F	B	282	CCF	
000AA8	CB7F	B	283	DVREXIT	BIT 7,A ;Z-flag set on non-CLEAR
000AAA	C8	B	284	RET	Z ;Go if not CLEAR+key
000AAB	F5	B	285	SPECL	PUSH AF ;Save code
000AAC	21 1E 09	B	286	\$?13	LD HL,SPCLTB ;Special char table
000AAF	CBBF	B	287	RES	7,A ;Turn off "CLEAR"
000AB1	015B05	B	288	LD	BC,5<<8 5Bh ;5 chars, starting char
000AB4	30 01	B	289	JR	NC,\$+3 ;if not CTRL
000AB6	05	B	290	DEC	B ;else only 4
000AB7	BE	B	291	SPCLLP	CP (HL) ;Is this it?
000AB8	28 12	B	292	JR	Z,HIT ;Go if so
000ABA	EE10	B	293	XOR	10h ;Flip shift state
000ABC	BE	B	294	CP	(HL) ;Is that it?
000ABD	28 0B	B	295	JR	Z,HITWS ;Go if so
000ABF	EE10	B	296	XOR	10h ;Flip back
000AC1	23	B	297	INC	HL ;Bump specl table ptr
000AC2	0C	B	298	INC	C ;Bump "convert to" char
000AC3	10 F2	B	299	DJNZ	SPCLLP ;Loop through table
000AC5	F1	B	300	POP	AF ;Not found in table
000AC6	38 0A	B	301	JR	C,CKCTL2 ;Ck CTL for C-flag
000AC8	BF	B	302	CKCTL1	CP A ;Set Z-flag

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; Console Keyboard Driver, KIDVR/ASM - LS-DOS 6.2 &gt;

```

PC      Object      I  Line  Source  ..\SYSRES\drivers\driver-KIDVR.s
000AC9  C9           B   303      RET
000ACA  CBE9        B   304  HITWS      SET    5,C          ;Move to LC set
000ACC  F1          B   305  HIT        POP    AF          ;Restore orig char
000ACD  79          B   306      LD      A,C          ;Load converted one
000ACE  30 F8        B   307  CKCTL      JR      NC,CKCTL1      ;Go if ctl key not down
000AD0  E61F        B   308      AND    1Fh          ;Force ctl code
000AD2  BF          B   309  CKCTL2     CP      A          ;Set Z-flag
000AD3  37           B   310      SCF          ;Set C-flag for CTRL
000AD4  C9           B   311      RET
B       312
B       313      ;  Check the type ahead buffer for any character.
B       314
B       315      SCOPE
000AD5          B   316  TYPAMD      ;      CALL  ENADIS_D0_RAM      ;Bring up Keyboard ram.
B       317
000AD5  2180FF       B   318      LD      HL,TYPBUF      ;P/u start of type buffer
000AD8  36FF       B   319      LD      (HL),0FFh      ;Turn off type ahead
000ADA  38 1D       B   320      JR      C,GET_kbd_char  ;Go on zGET
000ADC  28 21       B   321      JR      Z,TYPON        ;No PUT to *KI
000ADE  FE03       B   322      CP      3          ;CTL 3 function?
000AE0  CA 41 0B    B   323      JP      Z,CLRTP        ;Clear buffer if so
000AE3  3C         B   324      INC     A
000AE4  28 03       B   325      JR      Z,CTLFF        ;Go if CTL 255 function
000AE6  AF         B   326      XOR     A          ;Nothing done, No error
000AE7  18 16       B   327      JR      TYPON
B       328
B       329      ;  Handle CTL-255 - scan keyboard into user rowbuf
B       330      ;*****
000AE9  2101F4       B   331  CTLFF      LD      HL,KB0          ;Start of keyboard image
000AEC  0608       B   332      LD      B,8          ;Do 8 rows
000AEE  7E         B   333  $?0      LD      A,(HL)          ;P/u the image
000AEF  FD7700       B   334      LD      (IY),A        ;and xfer to user buffer
000AF2  FD23       B   335      INC     IY
000AF4  CB15       B   336      RL      L
000AF6  10 F6       B   337      DJNZ   $?0
000AF8  C9           B   338      RET
B       339
B       340
B       341      ;  No character in type ahead buffer - get from kbd
B       342
000AF9  E5         B   343  GET_kbd_char  PUSH   HL
000AFA  37         B   344      SCF
000AFB  CD D9 10     B   345      CALL   U0DVR ;      CALL  KISCAN          ;Call keyboard driver
000AFE  E1         B   346      POP    HL          ;Rcvr switch
000AFF  3600       B   347  TYPON      LD      (HL),0      ;Type ahead back on
000B01  C9           B   348      RET
B       349
B       350      ;  Type ahead task 10 - scans keyboard & saves key
B       351
000B02  040B       B   352  TYPTSK$     DW      $?5          ;Task entry for processor
000B04  3A 6D 00     B   353  $?5      LD      A,(DFLAG$)      ;If type-ahead suppressed
000B07  E602       B   354      AND    2h          ;then return
000B09  C8         B   355      RET     Z
000B0A  CD 17 08     B   356      CALL   ENADIS_D0_RAM      ;Bring up the keyboard
000B0D  2180FF       B   357      LD      HL,TYPBUF      ;P/u type switch
000B10  7E         B   358      LD      A,(HL)          ;If previous driver is
000B11  B7         B   359      OR     A          ;currently executing,
000B12  C0         B   360      RET     NZ          ;do not stack more keys
000B13  23         B   361      INC     HL          ;Bump to PUTPTR
000B14  E5         B   362      PUSH   HL          ;& save it

```

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; Console Keyboard Driver, KIDVR/ASM - LS-DOS 6.2 &gt;

PC	Object	I	Line	Source	.. \SYSRES\drivers\driver-KIDVR.s
000B15	CD 83 09	B	363	KIH00K	CALL KISCAN ;and scan for a character
000B18	E1	B	364		POP HL
000B19	C0	B	365		RET NZ ;Ret if no char
000B1A	F5	B	366		PUSH AF ;else xfer char
000B1B	C1	B	367		POP BC ;& flag to BC
000B1C	FE80	B	368		CP 80h ;Check for <BREAK>
000B1E	F5	B	369		PUSH AF
000B1F	E5	B	370		PUSH HL
000B20	CC 42 0B	B	371		CALL Z,\$?6 ;If so clear type buf
000B23	E1	B	372		POP HL ;Restore
000B24	F1	B	373		POP AF
000B25	FEC0	B	374		CP 0C0h ;If CLEAR z, reset keybuf
000B27	28 19	B	375		JR Z,\$?6
000B29	5E	B	376		LD E,(HL) ;P/u PUTPTR & compare
000B2A	7B	B	377		LD A,E ;GETPTR
000B2B	23	B	378		INC HL
000B2C	BE	B	379		CP (HL)
000B2D	28 21	B	380		JR Z,\$?8 ;Jump if key buffer empty
000B2F	3A 2C 00	B	381		LD A,(TIMER\$) ;Check if we expired the
000B32	DD86 03	B	382		ADD A,(IX+RPTRATE) ;time interval between
000B35	DDBE01	B	383		CP (IX+1) ;repeating keys
000B38	20 12	B	384		JR NZ,\$?7 ;Go if time not up
000B3A	DD86 03	B	385		ADD A,(IX+RPTRATE) ;Re-adjust time check so
000B3D	DD7701	B	386		LD (IX+1),A ;we don't repeat in
000B40	C9	B	387		RET ;type-ahead task
		B	388		
		B	389		; CLEAR z control key entered, clear the buffer
		B	390		
000B41	23	B	391	CLRTYP	INC HL ;Bump to PUT pointer
000B42	AF	B	392	\$?6	XOR A
000B43	77	B	393		LD (HL),A ;1st PUT is loc"n 0
000B44	23	B	394		INC HL ;Pt to GETPTR
000B45	77	B	395		LD (HL),A ;1st GET is loc"n 0
000B46	21 74 00	B	396	R7KFLG	LD HL,KFLAG\$ ;Show buffer empty
000B49	CBBE	B	397		RES 7,(HL)
000B4B	C9	B	398		RET
		B	399		
		B	400		; Char to stuff - check if buffer will overflow
		B	401		
000B4C	7B	B	402	\$?7	LD A,E ;P/u current PUT pointer
000B4D	3C	B	403		INC A ;If the next loc"n wraps
000B4E	BE	B	404		CP (HL) ;to the GET loc"n,
000B4F	C8	B	405		RET Z ;don't permit overrun
000B50	E5	B	406	\$?8	PUSH HL ;Save ptr to GETPTR
000B51	23	B	407		INC HL ;Pt to start of keybuf
000B52	1600	B	408		LD D,0 ;& calculate PUT loc"n
000B54	19	B	409		ADD HL,DE
000B55	70	B	410		LD (HL),B ;Store the char
000B56	21 74 00	B	411		LD HL,KFLAG\$ ;Show type buffer
000B59	CBFE	B	412		SET 7,(HL) ;is not empty
000B5B	E1	B	413		POP HL ;Rcvr ptr to GETPTR
000B5C	2B	B	414		DEC HL ;Backup to PUTPTR
000B5D	34	B	415		INC (HL) ;Bump past the char
000B5E	3E50	B	416		LD A,80 ;Check for >80
000B60	BE	B	417		CP (HL)
000B61	D0	B	418		RET NC ;Back if not over 80
000B62	72	B	419		LD (HL),D ;else reset to 1st
000B63	C9	B	420		RET ;position in buf (0)
		B	421		
		B	422		; Type ahead buffer area



\*\*\*\*\* TRSDOS \*\*\*\*\*  
< Console Keyboard Driver, KIDVR/ASM - LS-DOS 6.2 >

PC	Object	I	Line	Source	..\SYSRES\drivers\driver-KIDVR.s
		B	423		
		B	424		
	00000B63	B	425	KILAST	EQU \$-1
		B	426		
		B	0	INCLUDE	"driver-DODVR.s" ; Video driver.
		B	1		
		B	2	SUBTITLE	"< Video / Console Display Driver. V 6.3.1>"
		B	3		DODVR/ASM - LS-DOS 6.2 / LSDOS 6.3
		B	4		driver-DODVR.s Update to LSDOS 6.3.1 2022-09-16 dpm
		B	5		newpage

\*\*\*\*\* TRSDOS \*\*\*\*\*  
 < Video / Console Display Driver. V 6.3.1>

PC	Object	I	Line	Source	.. \SYSRES\drivers\driver-D0DVR.s
		B	6		SCOPE
		B	7		
		B	8		ORG driver.D0 ; Driver entry point
		B	9		
000B88	18 12	B	10	D0DVR	jr DOBGN ;Branch around linkage
000B8A	000E	B	11		DW DOEND ;Last memory location used
000B8C	0324444F	B	12		DB 3,"\$D0"
000B90	1002	B	13		DW D0DCB\$ ;DCB used
000B92	0000	B	14		DW 0 ;Reserved
	00000B94	B	15	D0DATA\$	EQU \$
	00000000	B	16	D0_MASK	EQU \$-D0DATA\$
000B94	00	B	17		DB 0 ;Tab/Spec, Scroll protect
000B95	00F8	B	18	CURSOR	DW CRTBGN\$
000B97	20	B	19	CRSAVE	DB 20h ;Character under cursor
000B98	5F	B	20	CRSCHAR	DB '_' ;Cursor character
		B	21		
		B	22		
000B99	C3 42 0D	B	23	zVDCTL	JP z_VDCTL ; Entry from SVC 15, zVDCTL.
		B	24		
		B	25		; Continue regular driver functions
		B	26		
000B9C	DD21 94 0B	B	27	DOBGN	LD IX,D0DATA\$
000BA0	CD 17 08	B	28		CALL ENADIS_D0_RAM ;Bring up the video RAM
000BA3	DA B3 0D	B	29		JP C,\$?0 ;Go on 'GET' request
000BA6	CD B3 0D	B	30		CALL \$?0 ;Handle cursor
000BA9	C5	B	31		PUSH BC ;Need to save C
000BAA	79	B	32		LD A,C ;Get char to display
000BAB	DDCB 00 66	B	33		BIT CTL,(IX+D0_MASK) ;Display controls set?
000BAF	20 09	B	34		JR NZ,\$?1A ;Go if so
000BB1	B7	B	35		OR A ;Char a 0?
000BB2	CA 9F 0C	B	36		JP Z,TGGLCTL ;Switch Bit CTL if so
000BB5	FE20	B	37		CP 20h ;Video control char?
000BB7	DA 44 0C	B	38		JP C,D0_CONTROL ;Go if so
000BBA	FEC0	B	39	\$?1A	CP 0C0h ;Tab or special?
000BBC	38 06	B	40		JR C,DONORM ;Go on normal characters
		B	41		
		B	42		; Character is => 0C0h
		B	43		
000BBE	DDCB 00 5E	B	44		BIT TABS,(IX+D0_MASK) ;Tabs or spec chars
000BC2	28 26	B	45		JR Z,D0_TABS ;Go if video tabs
		B	46		
		B	47		; Character is not tab expansion - do it.
		B	48		
000BC4	CD B8 0C	B	49	DONORM	CALL D0_DSPCHAR ;Display the char
000BC7	DDCB 00 A6	B	50		RES CTL,(IX+D0_MASK) ;Turn off CTL bit
000BCB	C1	B	51	D0_RET	POP BC ;Get orig char
000BCC	F3	B	52	D0_RET1	DI ;Disable intr
000BCD	3A 97 0B	B	53		LD A,(CRSAVE) ;If a cursor is on, then
000BD0	B7	B	54		OR A ;we need to save the
000BD1	28 10	B	55		JR Z,\$?1 ;current char & display
000BD3	1A	B	56		LD A,(DE) ;the cursor character
000BD4	32 97 0B	B	57		LD (CRSAVE),A ;Save current char
000BD7	3A 7F 00	B	58		LD A,(VFLAG\$) ;Allow tasker to blink
000BDA	CBBF	B	59		RES 7,A
000BDC	32 7F 00	B	60		LD (VFLAG\$),A
000BDF	3A 98 0B	B	61		LD A,(CRSCHAR) ;P/u cusor character
000BE2	12	B	62		LD (DE),A ;Put it on the screen
000BE3	ED53 95 0B	B	63	\$?1	LD (CURSOR),DE ;Update cursor position
000BE7	BF	B	64		CP A ;Clear status
000BE8	79	B	65		LD A,C ;Restore the char

\*\*\*\*\* TRSDOS \*\*\*\*\*  
 < Video / Console Display Driver. V 6.3.1>

PC	Object	I	Line	Source	.. \SYSRES\drivers\driver-D0DVR.s
000BE9	C9	B	66		RET
		B	67		
		B	68		; Perform a tab expansion {C0H-FFh}
		B	69		
000BEA	D6C0	B	70	DO_TABS	SUB 0C0h ;Compute spaces
000BEC	28 DD	B	71		JR Z, DO_RET ;Forget it if TAB(0)
000BEE	47	B	72		LD B,A ;Display requested
000BEF	0E20	B	73	\$?2	LD C, ' ' ;number of spaces
000BF1	CD B8 0C	B	74		CALL DO_DSPCHAR
000BF4	10 F9	B	75		DJNZ \$?2
000BF6	18 D3	B	76		JR DO_RET
		B	77		
		B	78		; Routine to move the cursor to begin of line {29}
		B	79		
000BF8	EB	B	80	CRSBOL	EX DE,HL ;Cursor addr to HL
000BF9	CD F4 0D	B	81		CALL ADDR1 ;Find row,col
000BFC	6F	B	82		LD L,A ; Set col to start
000BFD	C3 D0 0D	B	83		JP ROWCOL_2_ADDR ;Calc address of BOL
		B	84		
		B	85		; Routines to turn on/off the cursor {14/15}
		B	86		
000C00	1A	B	87	CRSON	LD A,(DE) ;Get screen character
000C01	32 97 0B	B	88	CRSOFF	LD (CRSAVE),A ;Save zero or CRT char
000C04	C9	B	89		RET
		B	90		
		B	91		; Routine moves cursor to start of video page {28}
		B	92		; set to 80 column, and turns off inverse video.
		B	93		
000C05	1100F8	B	94	CRSHOME	LD DE,CRTBGN\$ ;Home the cursor
000C08	3A 76 00	B	95		LD A,(MODOUT\$) ;P/u the mask &
000C0B	E6FB	B	96		AND 0FBh ;set to 80 cpl
000C0D	CD B2 0C	B	97		CALL SETMOD
000C10	18 7A	B	98		JR DO_INVERT_DIS ;Set to normal video
		B	99		
		B	100		; Routine to backspace & erase cursor {08}
		B	101		
000C12	CD 1B 0C	B	102	BACKSPA	CALL CRSBKSP ;Backspace the cursor
000C15	C8	B	103		RET Z ;If not at start,
000C16	0E20	B	104		LD C, ' ' ;put a space at
000C18	C3 CA 0D	B	105		JP PUT_z ;at the new loc"n
		B	106		
		B	107		; Routine to backspace the cursor {24}
		B	108		
000C1B	3A 76 00	B	109	CRSBKSP	LD A,(MODOUT\$) ;If double width chars,
000C1E	E604	B	110		AND 4 ;need to do twice
000C20	C4 23 0C	B	111		CALL NZ,\$+3
000C23	2100F8	B	112		LD HL,CRTBGN\$ ;See if at home position
000C26	ED52	B	113		SBC HL,DE ;prior to adjusting
000C28	C8	B	114		RET Z
000C29	1B	B	115		DEC DE ;Decrement the cursor pos
000C2A	C9	B	116		RET
		B	117		
		B	118		; Routine to move the cursor up one line {27}.
		B	119		
000C2B		B	120	CRSUP	
000C2B	21B0FF	B	121		LD HL,NEGLINE ;Move up one line
000C2E	18 03	B	122		JR MOVCRS
		B	123		
		B	124		; Routine to move the cursor down one line {26}
		B	125		

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; Video / Console Display Driver. V 6.3.1&gt;

PC	Object	I	Line	Source	Comment
000C30	215000	B	126	CRSDOWN LD HL,LINESIZ	;Add the line length
000C33	19	B	127	MOVCRS ADD HL,DE	;to the current pos
000C34	7C	B	128	LD A,H	;Make sure we did not
000C35	FEF8	B	129	CP CRTBGN\$>>8	;go over the top
000C37	D8	B	130	RET C	
000C38	EB	B	131	EX DE,HL	;& switch back to DE
000C39	1B	B	132	DEC DE	;Adjust for fall thru
000C3A	C3 C3 0C	B	133	JP CRSFRW0	
		B	134		
		B	135	; Set to 40 cpl mode {23}	
		B	136		
000C3D	3A 76 00	B	137	SET40 LD A,(MODOUT\$)	;Get image of the port
000C40	F604	B	138	OR 04h	;Merge in 40 cpl bit
000C42	18 6E	B	139	JR SETMOD	
		B	140		
		B	141	; Routines to parse control functions	
		B	142		
000C44	21 CB 0B	B	143	DO_CONTROL LD HL,DO_RET	;Establish RET
000C47	E5	B	144	PUSH HL	
000C48	FE08	B	145	CP 08h	;Backspace?
000C4A	28 C6	B	146	JR Z,BACKSPA	
000C4C	FE0A	B	147	CP 0Ah	;Line feed?
000C4E	28 02	B	148	JR Z,\$+4	;is same as <ENTER>
000C50	D60D	B	149	SUB 0Dh	;Carriage return?
000C52	CA 02 0D	B	150	JP Z,LINFEEED	
000C55	3D	B	151	DEC A	;Cursor on?
000C56	28 A8	B	152	JR Z,CRSON	
000C58	3D	B	153	DEC A	;Cursor off?
000C59	28 A6	B	154	JR Z,CRSOFF	
000C5B	3D	B	155	DEC A	;Reverse video?
000C5C	28 2B	B	156	JR Z,DO_INVERT_ENA	
000C5E	3D	B	157	DEC A	
000C5F	28 3A	B	158	JR Z,DO_INVERT_OFF	
000C61	D604	B	159	SUB 4	;Swap tab/alternate?
000C63	28 41	B	160	JR Z,TGGLTAB	
000C65	3D	B	161	DEC A	;Special/alternate?
000C66	28 45	B	162	JR Z,TGGLALT	
000C68	3D	B	163	DEC A	;40 cpl?
000C69	28 D2	B	164	JR Z,SET40	
000C6B	3D	B	165	DEC A	;Cursor backspace?
000C6C	28 AD	B	166	JR Z,CRSBKSP	
000C6E	3D	B	167	DEC A	;Cursor forward?
000C6F	28 4A	B	168	JR Z,CRSFRWD	
000C71	3D	B	169	DEC A	;Cursor down?
000C72	28 BC	B	170	JR Z,CRSDOWN	
000C74	3D	B	171	DEC A	;Cursor up?
000C75	28 B4	B	172	JR Z,CRSUP	
000C77	3D	B	173	DEC A	;Cursor home?
000C78	CA 05 0C	B	174	JP Z,CRSHOME	
000C7B	3D	B	175	DEC A	;Cursor BOL?
000C7C	CA F8 0B	B	176	JP Z,CRSBOL	
000C7F	3D	B	177	DEC A	;Clear to EOL?
000C80	CA 12 0D	B	178	JP Z,CLREOL	
000C83	3D	B	179	DEC A	
000C84	CA 1E 0D	B	180	JP Z,CLREOF	;Clear to end-of-frame?
000C87	AF	B	181	XOR A	;Clear A reg.
000C88	C9	B	182	RET	
		B	183		
		B	184	; Routine to enable inverse video	
		B	185		

\*\*\*\*\* TRSDOS \*\*\*\*\*  
 < Video / Console Display Driver. V 6.3.1>

PC	Object	I	Line	Source	.. \SYSRES\drivers\driver-D0DVR.s	
000C89	0608	B	186	DO_INVERT_ENA	LD B,8	;Set for enable
000C8B	21	B	187		DB 21h	;Ignore next load
000C8C	0600	B	188	DO_INVERT_DIS	LD B,0	
000C8E	2A 32 08	B	189		LD HL,(OPREG_SV_PTR)	;Real OPREG\$
000C91	7E	B	190		LD A,(HL)	;P/u OPREG mask
000C92	E6F7	B	191		AND 0F7h	;Strip bit 3
000C94	B0	B	192		OR B	;Set/reset invideo bit
000C95	77	B	193		LD (HL),A	;and restuff
000C96	78	B	194		LD A,B	;Get mode mask byte
000C97	07	B	195		RLCA	;Rotate left 4 times to
000C98	07	B	196		RLCA	;make an 8 into 80h
000C99	07	B	197		RLCA	;for inverse on
000C9A	07	B	198		RLCA	;Inverse off remains 0
000C9B	32 CB 0D	B	199	DO_INVERT_OFF	LD (INVIDEO),A	;Set the mask byte
000C9E	C9	B	200		RET	
		B	201			
		B	202			
		B	203			
000C9F	21 CB 0B	B	204	TGGLCTL	LD HL,DO_RET	;Establish ret addr
000CA2	E5	B	205		PUSH HL	
000CA3	3E10	B	206		LD A,10h	;Toggle bit 4
000CA5	21	B	207		DB 21h	;Ignore next
		B	208			
		B	209			
		B	210			
000CA6	3E08	B	211	TGGLTAB	LD A,8	;Toggle bit 3
000CA8	DDAE 00	B	212		XOR (IX+DO_MASK)	;P/u mask value
000CAB	18 50	B	213		JR SETMASK	
		B	214			
		B	215			
		B	216			
000CAD	3A 76 00	B	217	TGGLALT	LD A,(MODOUT\$)	;P/u port mask
000CB0	EE08	B	218		XOR 8	;Flip the bit
000CB2	32 76 00	B	219	SETMOD	LD (MODOUT\$),A	;Resave port mask.
000CB5	00	B	220		NOP	
000CB6	00	B	221		NOP	
		B	222		;OUT (0ECh),A	;and send the byte
000CB7	C9	B	223		RET	
		B	224			
		B	225			
		B	226			
000CB8	CD CA 0D	B	227	DO_DSPCHAR	CALL PUT_z	;Display the char
		B	228			
		B	229			
		B	230			
000CBB	3A 76 00	B	231	CRSFRWD	LD A,(MODOUT\$)	;If double width chars,
000CBE	E604	B	232		AND 4	;need to do twice
000CC0	28 01	B	233		JR Z,CRSFRW0	
000CC2	13	B	234		INC DE	;Move cursor forward
000CC3	13	B	235	CRSFRW0	INC DE	
000CC4	217FFF	B	236		LD HL,CRTEND\$	;Off the screen?
000CC7	ED52	B	237		SBC HL,DE	
000CC9	D0	B	238		RET NC	;Back if not
000CCA	CD 2B 0C	B	239		CALL CRSUP	;Put cursor back on
000CCD	D5	B	240		PUSH DE	;Save cursor position
		B	241			
000CCE	DD7E 00	B	242	DO_SCROLL	LD A,(IX+DO_MASK)	;Get scroll protect
000CD1	E607	B	243		AND SCRPROT	
000CD3	2100F8	B	244		LD HL,CRTBGN\$	;Point to CRT start
000CD6	118007	B	245		LD DE,CRTSIZE	;P/u CRT size

\*\*\*\*\* TRSDOS \*\*\*\*\*  
< Video / Console Display Driver. V 6.3.1>

PC	Object	I	Line	Source	.. \SYSRES\drivers\driver-D0DVR.s	
000CD9	C5	B	246		PUSH BC	
000CDA	015000	B	247		LD BC,LINESIZ	;Set line size
000CDD	3C	B	248		INC A	;Adjust scroll protect
000CDE	09	B	249	\$?4	ADD HL,BC	;Move logical start
000CDF	EB	B	250		EX DE,HL	;down one line
000CE0	B7	B	251		OR A	;and subtract one line
000CE1	ED42	B	252		SBC HL,BC	;from the CRTSIZE for
000CE3	EB	B	253		EX DE,HL	;each protected line
000CE4	3D	B	254		DEC A	;Dec scroll protect
000CE5	20 F7	B	255		JR NZ,\$?4	;Loop until done
000CE7	D5	B	256		PUSH DE	;Save the move length
000CE8	E5	B	257		PUSH HL	;Save the move-from
000CE9	ED42	B	258		SBC HL,BC	;Move start back one
000CEB	EB	B	259		EX DE,HL	;line, Source =
000CEC	E1	B	260		POP HL	;start + one
000CED	C1	B	261		POP BC	;Get back dest locn
		B	262			
000CEE	EDB0	B	263		LDIR	;Scroll unprotected
000CF0	C1	B	264		POP BC	;Recover line size
000CF1	18 2C	B	265		JR CLREOF1	;Clear to EOF from DE
		B	266			
		B	267			
		B	268			
		B	269			
		B	270			
		B	271			
000CF3	79	B	272	SET_SCROLL	LD A,C	;Get user value
000CF4	E607	B	273		AND 7	;Make modulo 8
000CF6	4F	B	274		LD C,A	
000CF7	3A 94 0B	B	275		LD A,(D0DATA\$)	;P/u current mask
000CFA	E6F8	B	276		AND 0F8h	;Remove current scroll
000CFC	B1	B	277		OR C	;Merge in the new value
000CFD	32 94 0B	B	278	SETMASK	LD (D0DATA\$),A	& reload mask
000D00	AF	B	279		XOR A	;Z-flag return
000D01	C9	B	280		RET	
		B	281			
		B	282			
		B	283			
000D02	CD F8 0B	B	284	LINFEED	CALL CRSBOL	;Move to BOL
000D05	D5	B	285		PUSH DE	;Save cursor position
000D06	CD 30 0C	B	286		CALL CRSDOWN	;Move down one line
000D09	B7	B	287		OR A	;Reset the carry flag
000D0A	2180FF	B	288		LD HL,CRTEND\$+1	& check if off of
000D0D	ED52	B	289		SBC HL,DE	the screen
000D0F	28 BD	B	290		JR Z,D0_SCROLL	;Scroll if so
000D11	E1	B	291		POP HL	;Discard old position
000D12	D5	B	292	CLREOL	PUSH DE	;Save new cursor pos
000D13	CD F8 0B	B	293		CALL CRSBOL	;Get start of line
000D16	214F00	B	294		LD HL,79	;Calculate end of line
000D19	19	B	295		ADD HL,DE	;HL = end of line
000D1A	D1	B	296		POP DE	;DE = current position
000D1B	D5	B	297		PUSH DE	
000D1C	18 04	B	298		JR CLREOF2	;Clear the line
		B	299			
		B	300			
		B	301			
000D1E	D5	B	302	CLREOF	PUSH DE	;Save current cursor pos
000D1F	217FFF	B	303	CLREOF1	LD HL,CRTEND\$	;Point to last RAM byte
000D22	3A CB 0D	B	304	CLREOF2	LD A,(INVIDEO)	;P/u normal/reverse
000D25	CBEF	B	305		SET 5,A	& make it a space

\*\*\*\*\* TRSDOS \*\*\*\*\*  
 < Video / Console Display Driver. V 6.3.1>

PC	Object	I	Line	Source	.. \SYSRES\drivers\driver-D0DVR.s	
000D27	12	B	306	LD	(DE),A	;Stuff the "space"
000D28	B7	B	307	OR	A	;Reset carry for subtract
000D29	ED52	B	308	SBC	HL,DE	;Calculate length
000D2B	28 09	B	309	JR	Z,CLREOF3	;Back if at end already
000D2D	C5	B	310	PUSH	BC	
000D2E	44	B	311	LD	B,H	;Xfer length to BC
000D2F	4D	B	312	LD	C,L	
000D30	62	B	313	LD	H,D	;Xfer start to HL
000D31	6B	B	314	LD	L,E	
000D32	13	B	315	INC	DE	;Bump up by one
000D33	EDB0	B	316	LDIR		;Propagate the space
000D35	C1	B	317	POP	BC	
000D36	D1	B	318	POP	DE	
000D37	C9	B	319	RET		
		B	320			
		B	321			
		B	322		; Routine to stuff the video cursor RAM address	
		B	323			
000D38	CD D0 0D	B	324	zVDCTL3	CALL ROWCOL_2_ADDR	;Calculate video address
000D3B	C0	B	325	RET	NZ	;Back on error
000D3C	F3	B	326	DI		;Disable any video tasks
000D3D	ED53 95 0B	B	327	LD	(CURSOR),DE	;until cursor is updated
000D41	C9	B	328	RET		
		B	329			
000D42	CD 17 08	B	330	z_VDCTL	CALL ENADIS_DO_RAM	;Video control SVC processor, bring up the vide
		B	331			
		B	332		; Test if in Task processor.	
000D45	3A 77 00	B	333	LD	A,(NFLAG\$)	;P/u NFLAG\$
000D48	CB77	B	334	BIT	6,A	;Test for task process
000D4A	20 15	B	335	JR	NZ,VDCTL	;If so skip setup
		B	336			
		B	337			
		B	338		; HANDLES zVDCTL screen set up for normal use	
		B	339			
000D4C	D5	B	340	PUSH	DE	
000D4D	CD B3 0D	B	341	CALL	\$?0	;Normalize character at cursor
000D50	D1	B	342	POP	DE	;Recover value
000D51	D5	B	343	PUSH	DE	
000D52	CD 61 0D	B	344	CALL	VDCTL	;Do function request
000D55	F5	B	345	PUSH	AF	;Save the error status
000D56	F3	B	346	DI		;Stop video tasks tempy
000D57	ED5B 95 0B	B	347	LD	DE,(CURSOR)	
000D5B	CD CC 0B	B	348	CALL	DO_RET1	;Normalize screen and cursor
000D5E	F1	B	349	POP	AF	
000D5F	D1	B	350	POP	DE	
000D60	C9	B	351	RET		
000D61	3E09	B	352	VDCTL	LD A,9	;Check for VIDLINE,
000D63	B8	B	353	CP	B	;function 9
000D64	28 25	B	354	JR	Z,VIDLIN	
000D66	3E2B	B	355	LD	A,43	;Prepare for user ERROR
000D68	05	B	356	DEC	B	
000D69	28 43	B	357	JR	Z,GET_z_ROWCOL	;<C> from row-H, col-L
000D6B	05	B	358	DEC	B	
000D6C	28 58	B	359	JR	Z,PUT_z_ROWCOL	;<C> to row-H, col-L
000D6E	05	B	360	DEC	B	
000D6F	28 C7	B	361	JR	Z,zVDCTL3	;Set cursor to H,L
000D71	05	B	362	DEC	B	
000D72	28 7D	B	363	JR	Z,ADDR_2_ROWCOL	;Cursor row,col to H,L
000D74	1100F8	B	364	LD	DE,CRTBGN\$	;Init to start of video
000D77	05	B	365	DEC	B	

\*\*\*\*\* TRSDOS \*\*\*\*\*  
 < Video / Console Display Driver. V 6.3.1>

PC	Object	I	Line	Source	.. \SYSRES\drivers\driver-D0DVR.s	
000D78	28 2D	B	366	JR	Z,VIDMOV1	;User RAM to video
000D7A	05	B	367	DEC	B	
000D7B	28 22	B	368	JR	Z,VIDMOVE	;Video RAM to user
000D7D	05	B	369	DEC	B	
000D7E	CA F3 0C	B	370	JP	Z,SET_SCROLL	;Set scroll protect
000D81	05	B	371	DEC	B	
000D82	C0	B	372	RET	NZ	;Return if bad request
		B	373		; Establish cursor character	
		B	374			
000D83	E5	B	375	PUSH	HL	
000D84	21 98 0B	B	376	LD	HL,CRSCHAR	;Point to cursor char storage
000D87	7E	B	377	LD	A,(HL)	;P/u current cursor character
000D88	71	B	378	LD	(HL),C	& update with new one
000D89	E1	B	379	POP	HL	
000D8A	C9	B	380	RET		
		B	381			
		B	382		; VIDLIN routine function - 9 in register B	
		B	383			
000D8B	2E00	B	384	VIDLIN	LD L,0	;Always starts at col 0
000D8D	D5	B	385	PUSH	DE	;Save user buffer
000D8E	CD D0 0D	B	386	CALL	ROWCOL_2_ADDR	;Get address to DE
000D91	E1	B	387	POP	HL	;Recover user buffer
000D92	C0	B	388	RET	NZ	;Quit on bad address
000D93	0C	B	389	INC	C	;Check direction
000D94	0D	B	390	DEC	C	;If Z then to screen
000D95	28 01	B	391	JR	Z,MOVLIN	;Set to go
000D97	EB	B	392	EX	DE,HL	;Reverse direction
000D98	015000	B	393	MOVLIN	LD BC,LINESIZ	;Set line size
000D9B	EDB0	B	394	LDIR		;Move it
000D9D	AF	B	395	XOR	A	;Z on RET
000D9E	C9	B	396	RET		
		B	397			
		B	398		; Routine to move video RAM	
		B	399			
000D9F	7C	B	400	VIDMOVE	LD A,H	;Check on user buffer
000DA0	C608	B	401	ADD	A,8	;not above X"F800' &
000DA2	FE2C	B	402	CP	24h+8	;not below X"2400"
000DA4	38 47	B	403	JR	C,PERR	
000DA6	EB	B	404	EX	DE,HL	;Xchng user buffer,screen
000DA7	018007	B	405	VIDMOV1	LD BC,CRTSIZE	;Set for full screen xfer
000DAA	EDB0	B	406	LDIR		
000DAC	BF	B	407	CP	A	;Set Z flag
000DAD	C9	B	408	RET		
		B	409			
		B	410		; Routine to get the character at row,col	
		B	411			
000DAE	CD D0 0D	B	412	GET_z_ROWCOL	CALL ROWCOL_2_ADDR	;Get Address of req
000DB1	1A	B	413	LD	A,(DE)	;P/u the character
000DB2	C9	B	414	RET		;Back on error or no error
		B	415			
		B	416		; Routine to halt blinking cursor & restore char	
		B	417			
000DB3	E5	B	418	\$?0	PUSH HL	
000DB4	21 7F 00	B	419	LD	HL,VFLAG\$	
000DB7	CBFE	B	420	SET	7,(HL)	;Disable blinking cursor
000DB9	E1	B	421	POP	HL	
000DBA	ED5B 95 0B	B	422	LD	DE,(CURSOR)	;Get cursor pos in DE
000DBE	3A 97 0B	B	423	LD	A,(CRSAVE)	;P/u saved character
000DC1	B7	B	424	OR	A	;If one is saved, put
		B	425			;it on screen, else



\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; Video / Console Display Driver. V 6.3.1&gt;

```

PC      Object      I  Line      Source  ..\SYSRES\drivers\driver-D0DVR.s
000DC2  20 09      B   426      JR          NZ,PUTAZDE          ;ignore it.
000DC4  1A      B   427      LD          A,(DE)          ;Cursor not ON but get
000DC5  C9      B   428      RET          ;character anyway
      B   429
      B   430
000DC6      B   431      PUT_z_ROWCOL      ;Routine to put a character at row,col
000DC6  CD D0 0D      B   432      CALL         ROWCOL_2_ADDR      ;Get address of req
000DC9  C0      B   433      RET          NZ          ;Back on error
000DCA  3E00      B   434      PUT_z          LD          A,0          ;Merge in reverse video
      00000DCB      B   435      INVIDEO0      EQU          $-1
000DCC  B1      B   436      OR          C
000DCD  12      B   437      PUTAZDE      LD          (DE),A          ;Put the character
000DCE  BF      B   438      CP          A          ;Set Z-flag for return
000DCF  C9      B   439      RET
      B   440
000DD0      B   441      ROWCOL_2_ADDR      ;Routine to calculate cursor position from row,col
      B   442
000DD0  3E4F      B   443      LD          A,79
000DD2  BD      B   444      CP          L
000DD3  38 18      B   445      JR          C,PERR          ;Error if > 79
000DD5  7C      B   446      LD          A,H          ;P/u row number
000DD6  FE18      B   447      CP          24
000DD8  30 13      B   448      JR          NC,PERR          ;Error if > 23
000DDA  E5      B   449      PUSH        HL
000ddb  C5      B   450      PUSH        BC
000DDC  4D      B   451      LD          C,L          ;Save column
000DDD  06F8      B   452      LD          B,CRTBGN$>>8      ;Set to start of D0 RAM
000DDF  215000      B   453      LD          HL,LINESIZ
000DE2  CD C9 06      B   454      CALL        zMUL16          ;Rows * line size
000DE5  65      B   455      LD          H,L          ;Shift to HL
000DE6  6F      B   456      LD          L,A
000DE7  09      B   457      ADD         HL,BC          ;Add in col & RAM start
000DE8  EB      B   458      EX          DE,HL          ;Address to DE
000DE9  C1      B   459      POP         BC
000DEA  E1      B   460      POP         HL
000DEB  AF      B   461      XOR         A          ;Set Z flag
000DEC  C9      B   462      RET
000DED      B   463      PERR;-->*****
000DED  3E2B      B   464      LD          A,43          ;SVC parameter error
000DEF  B7      B   465      OR          A          ;Set NZ condition
000DF0  C9      B   466      RET
      B   467
000DF1      B   468      ADDR_2_ROWCOL      ;Routine to get row,col of video cursor
      B   469
000DF1  2A 95 0B      B   470      LD          HL,(CURSOR)      ;Get addr in HL
000DF4  7C      B   471      ADDR1      LD          A,H          ;Make address relative
000DF5  E607      B   472      AND         7          ;to origin 0
000DF7  67      B   473      LD          H,A
000DF8  3E50      B   474      LD          A,LINESIZ      ;Set divisor
000DFA  CD E3 06      B   475      CALL        zDIV16
000DFD  65      B   476      LD          H,L          ;Row to register H
000DFE  6F      B   477      LD          L,A          ;Column to register L
000DFF  AF      B   478      XOR         A          ;Set zero return code
000E00  C9      B   479      RET
      00000E00      B   480      DOEND      EQU          $-1
      B   0      INCLUDE "driver-PRDVR.s"          ; Printer driver.
      B   1      SUBTITLE    "< Line Printer Driver. >"
      B   2      newpage

```

\*\*\*\*\* TRSDOS \*\*\*\*\*  
 < Line Printer Driver. >

PC	Object	I	Line	Source
		B	3	scope
		B	4	
		B	5	; Update to LSDOS 6.3.1 2022-09-16 dpm
		B	6	
		B	7	ORG driver.PR
		B	8	
		B	9	; PR driver entry point
		B	10	; It passes X"00"-X"FF"
		B	11	
		B	12	
000E01	18 11	B	13	PRDVR JR PRBGN ; Branch around linkage
000E03	570E	B	14	DW PREND ; Last byte used
000E05	03245052	B	15	DB 3,"\$PR"
000E09	1802	B	16	DW PRDCB\$ ; Pointer to its DCB
000E0B	0000	B	17	DW 0 ; Reserved
000E0D	23303030 30300A	B	18	PR_NET_CMD db "#00000",LF.asc ; Network command...0 is filled with byte to print.
		B	19	
		B	20	; Driver code
		B	21	
000E14	28 09	B	22	PRBGN JR Z,\$?2 ; Go if output
000E16	38 03	B	23	JR C,\$?1 ; Go if input req
		B	24	
		B	25	; Character CTL request
		B	26	
000E18	B7	B	27	OR A ; status else
000E19	28 35	B	28	JR Z,\$?4 ; treat as a Get
		B	29	
		B	30	; Character GET request
		B	31	
000E1B	F6FF	B	32	\$?1 OR 0FFh ;Set nz
000E1D	2F	B	33	CPL ;& A=0 to show
000E1E	C9	B	34	RET ;no char available
		B	35	
		B	36	; Character PUT request
		B	37	
000E1F	2600	B	38	\$?2 ld h,0
000E21	69	B	39	LD l,C ; If CTL 0, return.
		B	40	HEXDEC hl,PR_NET_CMD+1
000E22	E5E1	B+	40	ld hl,hl
000E24	11 0E 0E	B+	40	ld de,PR_NET_CMD+1
		B+	40	SVC svc_hexdec
000E27	3E61	B+	40	ld a,svc_hexdec
000E29	EF	B+	40	restart28
		B+	40	ENDMAC SVC
		B+	40	ENDMAC HEXDEC
		B	41	ZEROS_RPL_SPACE PR_NET_CMD+1 ; Convert leading spaces in buffer to 0's.
000E2A	21 0E 0E	B+	41	ld hl,PR_NET_CMD+1 ; Get address of buffer to scan & convert.
000E2D	1630	B+	41	ld d,'0' ; 0 to replace space.
000E2F	3E20	B+	41	ld a,' ' ; Test for space.
000E31	0605	B+	41	ld b,5 ; Test 5 char string.
000E33	BE	B+	41	__71 cp (hl)
000E34	20 01	B+	41	jr nz,__72 ; Replace all spaces with 0's.
000E36	72	B+	41	ld (hl),d
000E37	23	B+	41	__72 inc hl
		B+	41	LOOP __71
		B+	41	scope
000E38	10 F9	B+	41	djnz __71
		B+	41	endmac LOOP
		B+	41	endmac ZEROS_RPL_SPACE
		B	42	UART1.put.bytes 7,PR_NET_CMD,pr_net_err

\*\*\*\*\* TRSDOS \*\*\*\*\*  
 < Line Printer Driver. >

PC	Object	I	Line	Source	.. \SYSRES\drivers\driver-PRDVR.s
		B+	42		scope
000E3A	0607	B+	42		ld b,7 ; Were sending "count" bytes.
000E3C	21 0D 0E	B+	42		ld hl,PR_NET_CMD
000E3F	4E	B+	42	__91	ld c,(hl)
000E40	23	B+	42		inc hl
		B+	42		UART1.put.byte pr_net_err ; If nz we had a timeout or other error.
		B+	42		scope
000E41	AF	B+	42		xor a ; Set Z flag set means PUT.
		B+	42		GOSUB U1DVR ; Call UART driver.
		B+	42		scope
000E42	CD F5 11	B+	42		CALL U1DVR
		B+	42		endmac GOSUB
		B+	42		IF_NE_GOSUB pr_net_err ; If error goto error routine.
		B+	42		scope
000E45	C4 4B 0E	B+	42		call nz,pr_net_err
		B+	42		endmac IF_NE_GOSUB
		B+	42		endmac UART1.put.byte
		B+	42		LOOP __91
		B+	42		scope
000E48	10 F5	B+	42		djnz __91
		B+	42		endmac LOOP
		B+	42		endmac UART1.put.bytes
000E4A	C9	B	43		ret
		B	44		
000E4B	C1	B	45	pr_net_err	pop bc ; Discard return address of error.
000E4C	3E08	B	46		LD A,8 ; Device not avail...
000E4E	B7	B	47		OR A ; Set NZ condition
000E4F	C9	B	48		RET
		B	49		
000E50	ED38D5	B	50	\$?4	IN0 A,(UART1_LSR) ; Xmit ready?
000E53	E620	B	51		and UART_THRE
000E55	EE20	B	52		xor UART_THRE ; 0 if yes.
000E57	C9	B	53		RET ; Return with answer
		A	111		
	00000E58	A	112	DVREND\$	EQU \$ ; Start of low I/O driver area to 12FFh
		A	113		
		A	114		; SYSGEN saves memory from DVREND\$ to 12FFh to file CONFIG/SYS.CCC
		A	115		
		B	0		INCLUDE "driver-FDCDVR.s" ; Floppy disk driver
		B	1		SUBTITLE "<Floppy Disk Driver>"
		B	2		newpage

\*\*\*\*\* TRSDOS \*\*\*\*\*  
<Floppy Disk Driver>

PC	Object	I	Line	Source	.. \SYSRES\drivers\driver-FDCDVR.s
		B	3		scope
		B	4		
		B	5	; ORG	driver.FD
		B	6		
	00000E57	B	7	PREND EQU	\$-1
		B	8		
		B	9	; FDCDVR/ASM - LS-DOS 6.2	
		B	10		
		B	11	; HL=> HL points to buffer of READ / WRITE.	
		B	12	; D=> track desired.	
		B	13	; E=> sector desired.	
		B	14	; C=> C contains drive #.	
		B	15	; B=> Function in register B.	
		B	16		
000E58	18 2B	B	17	FDCDVR JR	FDCBGN ;Branch to entry code
000E5A	D810	B	18	DW FDCEND	;Last byte used
000E5C	03244644	B	19	DB 3,"\$FD"	;Module name
		B	20		
000E60	000000	B	21	PADDRESS db	0,0,0 ; This driver calculates physical beginning of DISK start.
		B	22		
000E63	00 00 00 00 00 00	B	23	ramdrv\$ blkb	16,0 ; Filled in @IPL with physical pointer to boot volume.
000E69	00 00 00 00 00 00				
000E6F	00 00 00 00				
		B	24		
000E73	00	B	25	secreq\$ db	0
		B	26		
000E74	0000	B	27	fdc_no_sum dw	0 ; Hold sum of physical record w/o offset.
000E76	00	B	28	net_volume db	0 ; Number of network volume, filled in by binding.
000E77	3D303030 30300A03	B	29	net_cmd db	"=000000",LF.asc,ETX.asc
000E7F	00000000	B	30	chk_sum db	0,0,0,0
000E83	00	B	31	rd_retries db	0
000E84	00	B	32	wr_retries db	0
		B	33		
		B	34		
		B	35	; Disk Driver Entry Point	
		B	36	; _____	
		B	37		
000E85	F3	B	38	FDCBGN di	
000E86	FD7205	B	39	ld (IY+5),d	; Update current track.
		B	40		
000E89	79	B	41	ld a,c	; Get drive requested code.
000E8A	32 23 00	B	42	ld (LDRV\$),a	; Store in current logical drive.
		B	43		
000E8D	7B	B	44	ld a,e	; Get sector requested. <<<-----
000E8E	32 73 0E	B	45	ld (secreq\$),a	; Store for later.
		B	46		
000E91	FD7E04	B	47	ld a,(IY+4)	; Get drive select code entry.
000E94	E60F	B	48	and 0Fh	; Strip out drive entry code.
000E96	32 1B 00	B	49	ld (PDRV\$),a	; Store in Physical Drive code.
		B	50		
000E99	78	B	51	LD A,B	; P/u primitive request.
000E9A	E5	B	52	push hl	; Per Soltoff we save any registers we use.
000E9B	C5	B	53	push bc	; Per Soltoff we save any registers we use.
000E9C	E5	B	54	push hl	; Per Soltoff we save any registers we use.
		B	55		
000E9D	0600	B	56	ld b,0	
000E9F	CB21	B	57	sla c	; drive * 2.
		B	58		
000EA1	21 63 0E	B	59	ld hl,ramdrv\$	; Get address of drive vectors.
000EA4	09	B	60	add hl,bc	; drive vector = vector + drive

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt;Floppy Disk Driver&gt;

PC	Object	I	Line	Source	.. \SYSRES\drivers\driver-FDCDVR.s
000EA5	ED27	B	61	ld	hl,(hl) ; Get vector.
000EA7	22 B5 10	B	62	ld	(RAMDRV\$),hl ; Store here so driver can use on this call.
		B	63		
000EAA	21 BF 0E	B	64	LD	HL,funtable\$ ; Base address of function table.
000EAD	CB27	B	65	sla	a ; Function=function*2.
000EAF	0600	B	66	ld	b,0
000EB1	4F	B	67	ld	c,a ; BC contains table offset.
000EB2	09	B	68	add	hl,bc ; Pointer=base+offset.
000EB3	ED07	B	69	ld	bc,(hl) ; Get vector.
000EB5	ED43 BD 0E	B	70	ld	(vecvec\$),bc ; Store in JP instruction below.
		B	71		
000EB9	E1	B	72	POP	HL
000EBA	C1	B	73	pop	bc ; Recovered from save on entry to driver.
000EBB	C5	B	74	push	bc ; Save to recover on exit of driver.
		B	75		
000EBC	C3 BC 0E	B	76	jp	\$
	00000EBD	B	77	vecvec\$	equ \$-2
		B	78		
	00000EBF	B	79	funtable\$	equ \$
000EBF	ED0E	B	80	dw	TSTBSY ; *0 NOP.
000EC1	040F	B	81	dw	SLCT ; *1 Drive select.
000EC3	ED0E	B	82	dw	TSTBSY ; *2 initialize controller.
000EC5	ED0E	B	83	dw	TSTBSY ; *3 Reset controller?
000EC7	FC0E	B	84	dw	RESTOR ; *4 Restore drive to 0
000EC9	F70E	B	85	dw	STPIN ; *5 Step in
000ECB	F20E	B	86	dw	SEEKTRK ; *6 Jump on track seek.
000ECD	ED0E	B	87	dw	TSTBSY ; *7 TEST BUSY.
000ECF	DF0E	B	88	dw	notsupported ; *8 Read header.
000ED1	0F0F	B	89	dw	RDIN ; *9 Read sector.
000ED3	E10E	B	90	dw	VERIFY ; *10 Verify sector readable, verify routine.
000ED5	DF0E	B	91	dw	notsupported ; *11 Read cylinder?
000ED7	ED0E	B	92	dw	TSTBSY ; *12 Format volume
000ED9	C00F	B	93	dw	DOWRIT ; *13 Write a sector.
000EDB	C00F	B	94	dw	DOWRIT ; *14 Write system directory.
000EDD	DF0E	B	95	dw	notsupported ; *15 Write cylinder.
		B	96		
000EDF	18 FE	B	97	notsupported	jr \$
		B	98		
		B	99		
000EE1	7A	B	100	VERIFY	ld a,d ; Get track we just read.
000EE2	FDBE09	B	101	cp	(IY+9) ; Compare directory track to track we just read.
000EE5	20 06	B	102	jr	nz,TSTBSY ; Not a directory read so clean up and exit no error.
000EE7	3E06	B	103	ld	a,6 ; This was directory cylinder request so set error=6
000EE9	B7	B	104	or	a ; NZ for return.
000EEA	C1	B	105	pop	bc ; Recover saved register.
000EEB	E1	B	106	pop	hl ; Recovered from save on entry to driver.
000EEC	C9	B	107	ret	
		B	108		
		B	109		
		B	110		
		B	111		
		B	112		
000EED	C1	B	113	TSTBSY	pop bc ; Recovered from save on entry to driver.
000EEE	E1	B	114	POP	HL ; Per Soltoff we restore on exit any registers we used.
000EEF	AF	B	115	XOR	A ; Clear A & reset Z flag.
000EF0	FB	B	116	ei	
000EF1	C9	B	117	RET	
		B	118		
000EF2	FD7205	B	119	SEEKTRK	LD (IY+5),D ; Update current cylinder
000EF5	18 F6	B	120	JR	TSTBSY

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt;Floppy Disk Driver&gt;

PC	Object	I	Line	Source	.. \SYSRES\drivers\driver-FDCDVR.s
		B	121		
000EF7	FD3505	B	122	STPIN	DEC (iy+5) ; FUNCTION 5 STEP-IN.
000EFA	18 F1	B	123	jr	TSTBSY
		B	124		
000EFC	FD360500	B	125	RESTOR	LD (IY+5),0 ; FUNCTION 4. RESTORE Set track to 0
000F00	18 EB	B	126	jr	TSTBSY
		B	127		
000F02	18 E9	B	128	ICTRL	jr TSTBSY ; Function 2, initialize controller.
		B	129		
000F04	79	B	130	SLCT	ld a,c ; Get drive number.
000F05	32 23 00	B	131	ld	(LDRV\$),a ; Store in logical drive number.
000F08	C1	B	132	pop	bc ; Recovered from save on entry to driver.
000F09	E1	B	133	POP	HL ; FUNCTION 7 TEST BUSY.
000F0A	AF	B	134	XOR	A
000F0B	3E02	B	135	ld	A,00000010b ; Bit one set meaning index pulse.
000F0D	FB	B	136	ei	
000F0E	C9	B	137	RET	; We are solid state drive, were always ready.
		B	138		
		B	139		; I/O request handler read routine.
		B	140		
000F0F	D5	B	141	RDIN	push de ; D > track. E > sector, save for later.
000F10	E5	B	142	PUSH	HL ; Save 16b pointer to buffer.
		B	143		
000F11	CD 85 10	B	144	call	calcsec ; Calculate address of sector to read.
		B	145		
000F14	3A 23 00	B	146	ld	a,(LDRV\$) ; Is this i/o request a network drive?
000F17	47	B	147	ld	b,a
000F18	FE06	B	148	cp	6 ; Drive 6 is reserved for network.
		B	149	IF_NE_GOTO	not_net_rd ; This is not network drive, bypass net_read.
		B+	149	scope	
000F1A	C2 A4 0F	B+	149	jp	nz,not_net_rd
		B+	149	endmac	IF_NE_GOTO
		B	150		
000F1D	3A 76 0E	B	151	ld	a,(net_volume) ; Which volume if any is network?
000F20	B8	B	152	cp	a,b ; Which volume if any is network?
		B	153	IF_NE_GOTO	not_net_rd ; This is not network drive, bypass net read.
		B+	153	scope	
000F21	C2 A4 0F	B+	153	jp	nz,not_net_rd
		B+	153	endmac	IF_NE_GOTO
		B	154		
		B	155		; Read sector from network connection.
		B	156		; READ > NETWORK SECTOR HERE <
		B	157		
000F24	AF	B	158	xor	a
000F25	32 83 0E	B	159	ld	(rd_retries),a ; Start retry counter off at 0.
		B	160		
000F28	3E3C	B	161	LD	a,'<' ; Symbol used for READ command.
000F2A	32 77 0E	B	162	read_retry	ld (net_cmd),a ; Stuff direction in command buffer.
		B	163		
		B	164	HEXDEC	(fdc_no_sum),net_cmd+1 ; Convert 16 bit address to ascii in network command.
000F2D	2A 74 0E	B+	164	ld	hl,(fdc_no_sum)
000F30	11 78 0E	B+	164	ld	de,net_cmd+1
		B+	164	SVC	svc_hexdec
000F33	3E61	B+	164	ld	a,svc_hexdec
000F35	EF	B+	164	restart28	
		B+	164	ENDMAC	SVC
		B+	164	ENDMAC	HEXDEC
		B	165	ZEROS_RPL_SPACE	net_cmd+1 ; Convert leading spaces in buffer to 0's.
000F36	21 78 0E	B+	165	ld	hl,net_cmd+1 ; Get address of buffer to scan & convert.
000F39	1630	B+	165	ld	d,'0' ; 0 to replace space.

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt;Floppy Disk Driver&gt;

PC	Object	I	Line	Source	.. \SYSRES\drivers\driver-FDCDVR.s
000F3B	3E20	B+	165	ld	a, ' ' ; Test for space.
000F3D	0605	B+	165	ld	b,5 ; Test 5 char string.
000F3F	BE	B+	165	__181	cp (hl)
000F40	20 01	B+	165	jr	nz, __182 ; Replace all spaces with 0's.
000F42	72	B+	165	ld	(hl),d
000F43	23	B+	165	__182	inc hl
		B+	165		LOOP __181
		B+	165		scope
000F44	10 F9	B+	165		djnz __181
		B+	165		endmac LOOP
		B+	165		endmac ZEROS_RPL_SPACE
		B	166		
		B	167		PUMP DEVICE.serialnet,net_cmd ; Send <XXXXX LF.asc packet.
		B+	167		scope ; A tribute to Dave Shriner - Display Data Corp.
000F46	21 77 0E	B+	167		LD HL,net_cmd ; Pump from this starting point until ETX.asc is found.
		B+	167		if DEVICE.serialnet=0
		B+	167	__200:	LD A,(HL) ; Get byte to send.
		B+	167		INC HL ; pointer=pointer+1
		B+	167	;	CP EOT.asc ; End Of Transmission (block)?
		B+	167	;	jr Z,\$0a ; Return if finished.
		B+	167		CP ETX.asc ; End Of Text?
		B+	167		jr Z,__200a ; Return if finished.
		B+	167		CP CR.asc ; Was it a CR?
		B+	167		call z,__201 ; 1st add a LF to stream.
		B+	167		LD C,A ; Output byte in C.
		B+	167		XOR A ; Set Z flag for put.
		B+	167		CALL U0DVR ; Out the UART we go.
		B+	167		JR __200
		B+	167	__201:	ld a,LF.asc ; Get a LF.
		B+	167		LD C,A ; Output byte in C.
		B+	167		XOR A ; Set Z flag for put
		B+	167		CALL U0DVR ; Out the UART we go.
		B+	167		ld a,CR.asc ; Now put a CR back in stream.
		B+	167		ret ; Return and print CR.
		B+	167	__200a	endif
		B+	167		if DEVICE.serialnet=1
000F49	7E	B+	167	__201:	LD A,(HL) ; Get byte to send.
000F4A	23	B+	167		INC HL ; pointer=pointer+1
		B+	167	;	CP EOT.asc ; End Of Transmission (block)?
		B+	167	;	jr Z,\$1a ; Return if finished.
		B+	167		CP ETX.asc ; End Of Text?
000F4B	FE03	B+	167		jr Z,__201a ; Return if finished.
000F4D	28 07	B+	167		LD C,A ; Output byte to C.
000F4F	4F	B+	167		XOR A ; Set Z flag for put.
000F50	AF	B+	167		CALL U1DVR ; Out the UART we go.
000F51	CD F5 11	B+	167		JR __201
000F54	18 F3	B+	167		equ \$
00000F56		B+	167	__201a	endif
		B+	167		ENDMAC PUMP
		B	168		
000F56	E1	B	169		pop hl ; Recover destination buffer address.
000F57	E5	B	170		push hl ; Save again for checksum routine.
		B	171		
		B	172		UART1.get.bytes 0,hl,fdc_rd_err ; Get from server 256 byte sector requested.
		B+	172		scope
000F58	0600	B+	172		ld b,0 ; Input 5 bytes.
000F5A	E5E1	B+	172		ld hl,hl ; Get address of buffer.
000F5C		B+	172	__211	UART1.get.byte fdc_rd_err ; Get byte from UART1. If timeout goto no_pong.
		B+	172		scope
000F5C	3E01	B+	172		ld a,1

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt;Floppy Disk Driver&gt;

PC	Object	I	Line	Source	.. \SYSRES\drivers\driver-FDCDVR.s
000F5E	B7	B+	172		or a ; On timeout or error goto vector.
000F5F	37	B+	172		scf
		B+	172		GOSUB U1DVR ; C flag set & NZ set means PUT.
		B+	172		scope
000F60	CD F5 11	B+	172		CALL U1DVR
		B+	172		endmac GOSUB
		B+	172		IF_NE_GOSUB fdc_rd_err ; If nz we had a timeout or other error.
		B+	172		scope
000F63	C4 C0 10	B+	172		call nz, fdc_rd_err
		B+	172		endmac IF_NE_GOSUB
		B+	172		endmac UART1.get.byte
000F66	71	B+	172		ld (hl),c ; Store char.
000F67	23	B+	172		inc hl
		B+	172		LOOP __211 ; Loop for count bytes.
		B+	172		scope
000F68	10 F2	B+	172		djnz __211
		B+	172		endmac LOOP
		B+	172		endmac UART1.get.bytes
		B	173		UART1.get.bytes 4,chk_sum,fdc_rd_err ; Get from server 4 byte checksum.
		B+	173		scope
000F6A	0604	B+	173		ld b,4 ; Input 5 bytes.
000F6C	21 7F 0E	B+	173		ld hl,chk_sum ; Get address of buffer.
000F6F		B+	173	__261	UART1.get.byte fdc_rd_err ; Get byte from UART1. If timeout goto no_pong.
		B+	173		scope
000F6F	3E01	B+	173		ld a,1
000F71	B7	B+	173		or a ; On timeout or error goto vector.
000F72	37	B+	173		scf
		B+	173		GOSUB U1DVR ; C flag set & NZ set means PUT.
		B+	173		scope
000F73	CD F5 11	B+	173		CALL U1DVR
		B+	173		endmac GOSUB
		B+	173		IF_NE_GOSUB fdc_rd_err ; If nz we had a timeout or other error.
		B+	173		scope
000F76	C4 C0 10	B+	173		call nz, fdc_rd_err
		B+	173		endmac IF_NE_GOSUB
		B+	173		endmac UART1.get.byte
000F79	71	B+	173		ld (hl),c ; Store char.
000F7A	23	B+	173		inc hl
		B+	173		LOOP __261 ; Loop for count bytes.
		B+	173		scope
000F7B	10 F2	B+	173		djnz __261
		B+	173		endmac LOOP
		B+	173		endmac UART1.get.bytes
		B	174		
000F7D	E1	B	175		pop hl ; Get pointer to buffer.
		B	176		GOSUB calc_cksum
		B+	176		scope
000F7E	CD D1 10	B+	176		CALL calc_cksum
		B+	176		endmac GOSUB
000F81	47	B	177		ld b,a
000F82	3A 82 0E	B	178		ld a,(chk_sum+3)
000F85	B8	B	179		cp b
		B	180		IF_EQ_GOTO rd_finish
		B+	180		scope
000F86	CA 95 0F	B+	180		jp z,rd_finish
		B+	180		endmac IF_EQ_GOTO
		B	181		
		B	182		; Here we have a error on read.
		B	183		; Bump retry counter.
		B	184		; Set server command to retry.



\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt;Floppy Disk Driver&gt;

PC	Object	I	Line	Source	..\\SYSRES\\drivers\\driver-FDCDVR.s
		B	185		; If retries does not roll over to 0 go around again.
		B	186		
000F89	21 83 0E	B	187	ld hl,rd_retries	
000F8C	34	B	188	inc (hl)	; Bump counter by 1.
		B	189	IF_EQ_GOTO fdc_rd_err	; If counter rolls over to 0 we have 256 retries, abort i/o.
		B+	189	scope	
000F8D	CA C0 10	B+	189	jp z,fdc_rd_err	
		B+	189	endmac IF_EQ_GOTO	
000F90	3E5C	B	190	ld a,'\'	; Retry signal to server.
		B	191	GOTO read_retry	
		B+	191	scope	
000F92	C3 2A 0F	B+	191	jp read_retry	
		B+	191	endmac GOTO	
		B	192		
		B	193		
000F95	D1	B	194	rd_finishPOP de	; Restore DE to track/sector.
000F96	7A	B	195	ld a,d	; Get track we just read.
000F97	FDBE09	B	196	cp (IY+9)	; Compare directory track to track we just read.
000F9A	C2 ED 0E	B	197	jp nz,TSTBSY	; Not system sector? Return no error return & carry on.
000F9D	3E06	B	198	ld a,6	; If we read system sector then set error to 6.
000F9F	B7	B	199	or a	; Flag to non-zero.
000FA0	C1	B	200	pop bc	; Recovered from save on entry to driver.
000FA1	E1	B	201	POP HL	; Per Soltoff we restore on exit any registers we used.
000FA2	FB	B	202	ei	
		B	203	RETURN	; Return with NZ flag set indicating error.
		B+	203	scope	
000FA3	C9	B+	203	ret	; Return from call.
		B+	203	endmac RETURN	
		B	204		
		B	205		; Read sector from RAM drive.
		B	206		; > READ MEMDISK SECTOR HERE <.
		B	207		
000FA4		B	208	not_net_rd	
000FA4	492A 60 0E	B	209	LD.L hl,(PADDRESS)	; HL now contains 24b address pointing to sector in RAM dri
000FA8	AF	B	210	xor a	
000FA9	32 62 0E	B	211	LD (PADDRESS+2),a	; A contains 0 to zero out upper 8 bits of 24b address.
000FAC	D1	B	212	POP de	; Point DE to destination buffer.
000FAD	ED53 60 0E	B	213	LD (PADDRESS),de	; A-=0 from above code.
000FB1	49ED5B 60 0E	B	214	LD.L DE,(PADDRESS)	; DE now contains 24b destination pointer.
000FB6	49010001	B	215	LD.L bc,100h	; Move one SECTOR or 256B.
000FBA	49EDB0	B	216	LDIR.L	; Move sector from RAM DRIVE to BUFFER.
		B	217	GOTO rd_finish	; Received sector, continue to finish.
		B+	217	scope	
000FBD	C3 95 0F	B+	217	jp rd_finish	
		B+	217	endmac GOTO	
		B	218		
000FC0		B	219	DOWRIT	; I/O request handler WRITE routine, write sector to RAM DRIVE.
		B	220		
000FC0	FDCB037E	B	221	bit 7,(iy+3)	; Test WP flag in DCT.
000FC4	28 05	B	222	JR Z,wp_ok	; If disk WP on then we skip following code.
000FC6	C1	B	223	pop bc	; Recovered from save on entry to driver.
000FC7	E1	B	224	POP HL	; Per Soltoff we restore on exit any registers we used.
000FC8	3E0F	B	225	LD A,15	; Error disk WP. Return with NZ & error code 15.
000FCA	C9	B	226	ret	; Return with NZ & disk WP error in A.
		B	227		
000FCB	D5	B	228	wp_ok push de	; D - contains track. E contains sector.
000FCC	E5	B	229	PUSH HL	; Save 16b pointer to buffer.
		B	230		
000FCD	CD 85 10	B	231	call calcsec	; Calculate address of sector to write.
		B	232		

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt;Floppy Disk Driver&gt;

PC	Object	I	Line	Source	.. \SYSRES\drivers\driver-FDCDVR.s
000FD0	3A 23 00	B	233	ld	a, (LDRV\$) ; Is this i/o request a network drive?
000FD3	47	B	234	ld	b, a
000FD4	FE06	B	235	cp	6 ; Drive 6 is reserved for network.
		B	236	IF_NE_GOTO	fdc_not_net_wr ; This is not network drive, bypass net_read.
		B+	236	scope	
000FD6	C2 68 10	B+	236	jp	nz, fdc_not_net_wr
		B+	236	endmac	IF_NE_GOTO
		B	237		
000FD9	3A 76 0E	B	238	ld	a, (net_volume) ; Which volume if any is network?
000FDC	B8	B	239	cp	a, b ; Which volume if any is network?
		B	240	IF_NE_GOTO	fdc_not_net_wr ; This is not network drive, bypass net read.
		B+	240	scope	
000FDD	C2 68 10	B+	240	jp	nz, fdc_not_net_wr
		B+	240	endmac	IF_NE_GOTO
		B	241		
		B	242		; > WRITE sector to NETWORK CONNECTION HERE <
		B	243		
		B	244	HEXDEC	(fdc_no_sum), net_cmd+1 ; Convert 16 bit address to ascii in network command.
000FE0	2A 74 0E	B+	244	ld	hl, (fdc_no_sum)
000FE3	11 78 0E	B+	244	ld	de, net_cmd+1
		B+	244	SVC	svc_hexdec
000FE6	3E61	B+	244	ld	a, svc_hexdec
000FE8	EF	B+	244	restart28	
		B+	244	ENDMAC	SVC
		B+	244	ENDMAC	HEXDEC
		B	245		
		B	246	ZEROS_RPL_SPACE	net_cmd+1 ; Convert leading spaces in buffer to 0's.
000FE9	21 78 0E	B+	246	ld	hl, net_cmd+1 ; Get address of buffer to scan & convert.
000FEC	1630	B+	246	ld	d, '0' ; 0 to replace space.
000FEE	3E20	B+	246	ld	a, ' ' ; Test for space.
000FF0	0605	B+	246	ld	b, 5 ; Test 5 char string.
000FF2	BE	B+	246	cp	(hl)
000FF3	20 01	B+	246	jr	nz, __412 ; Replace all spaces with 0's.
000FF5	72	B+	246	ld	(hl), d
000FF6	23	B+	246	inc	hl
		B+	246	LOOP	__411
		B+	246	scope	
000FF7	10 F9	B+	246	djnz	__411
		B+	246	endmac	LOOP
		B+	246	endmac	ZEROS_RPL_SPACE
		B	247		
000FF9	AF	B	248	xor	a
000FFA	32 84 0E	B	249	ld	(wr_retries), a ; Start retry counter off at 0.
		B	250		
000FFD	3E3E	B	251	LD	a, '>'
000FFF	32 77 0E	B	252	wr_retry ld	(net_cmd), a ; Stuff direction in command buffer.
		B	253		
		B	254	PUMP	DEVICE.serialnet, net_cmd ; Send @<XXXXX LF.asc packet.
		B+	254	scope	; A tribute to Dave Shriner - Display Data Corp.
001002	21 77 0E	B+	254	LD	HL, net_cmd ; Pump from this starting point until ETX.asc is found.
		B+	254	if	DEVICE.serialnet=0
		B+	254	__430: LD	A, (HL) ; Get byte to send.
		B+	254	INC	HL ; pointer=pointer+1
		B+	254	; CP	EOT.asc ; End Of Transmission (block)?
		B+	254	; jr	Z, \$0a ; Return if finished.
		B+	254	CP	ETX.asc ; End Of Text?
		B+	254	jr	Z, __430a ; Return if finished.
		B+	254	CP	CR.asc ; Was it a CR?
		B+	254	call	z, __431 ; 1st add a LF to stream.
		B+	254	LD	C, A ; Output byte in C.

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt;Floppy Disk Driver&gt;

PC	Object	I	Line	Source	.. \SYSRES\drivers\driver-FDCDVR.s
		B+	254	XOR	A ; Set Z flag for put.
		B+	254	CALL	U0DVR ; Out the UART we go.
		B+	254	JR	__430
		B+	254	__431:	ld a,LF.asc ; Get a LF.
		B+	254	LD	C,A ; Output byte in C.
		B+	254	XOR	A ; Set Z flag for put
		B+	254	CALL	U0DVR ; Out the UART we go.
		B+	254	ld	a,CR.asc ; Now put a CR back in stream.
		B+	254	ret	; Return and print CR.
		B+	254	__430a	endif
		B+	254	if	DEVICE.serialnet=1
001005	7E	B+	254	__431:	LD A,(HL) ; Get byte to send.
001006	23	B+	254	INC	HL ; pointer=pointer+1
		B+	254	CP	EOT.asc ; End Of Transmission (block)?
		B+	254	jr	Z,\$1a ; Return if finished.
001007	FE03	B+	254	CP	ETX.asc ; End Of Text?
001009	28 07	B+	254	jr	Z,__431a ; Return if finished.
00100B	4F	B+	254	LD	C,A ; Output byte to C.
00100C	AF	B+	254	XOR	A ; Set Z flag for put.
00100D	CD F5 11	B+	254	CALL	U1DVR ; Out the UART we go.
001010	18 F3	B+	254	JR	__431
	00001012	B+	254	__431a	equ \$
		B+	254	endif	
		B+	254	ENDMAC	PUMP
		B	255		
001012	E1	B	256	pop	hl ; Recover buffer address for read.
001013	E5	B	257	push	hl ; Leaving one copy on stack.
		B	258		
		B	259	UART1.put.bytes	0,hl,fdc_wr_err ; Write to server 256 byte sector.
		B+	259	scope	
001014	0600	B+	259	ld	b,0 ; Were sending "count" bytes.
001016	E5E1	B+	259	ld	hl,hl
001018	4E	B+	259	__441	ld c,(hl)
001019	23	B+	259	inc	hl
		B+	259	UART1.put.byte	fdc_wr_err ; If nz we had a timeout or other error.
		B+	259	scope	
00101A	AF	B+	259	xor	a ; Set Z flag set means PUT.
		B+	259	GOSUB	U1DVR ; Call UART driver.
		B+	259	scope	
00101B	CD F5 11	B+	259	CALL	U1DVR
		B+	259	endmac	GOSUB
		B+	259	IF_NE_GOSUB	fdc_wr_err ; If error goto error routine.
		B+	259	scope	
00101E	C4 C8 10	B+	259	call	nz,fdc_wr_err
		B+	259	endmac	IF_NE_GOSUB
		B+	259	endmac	UART1.put.byte
		B+	259	LOOP	__441
		B+	259	scope	
001021	10 F5	B+	259	djnz	__441
		B+	259	endmac	LOOP
		B+	259	endmac	UART1.put.bytes
		B	260		
001023	E1	B	261	pop	hl ; Get buffer address.
001024	E5	B	262	push	hl ; Leaving one copy on stack.
		B	263	GOSUB	calc_cksum ; Calculate checksum for buffer.
		B+	263	scope	
001025	CD D1 10	B+	263	CALL	calc_cksum
		B+	263	endmac	GOSUB
001028	32 82 0E	B	264	ld	(chk_sum+3),a ; Store these 8 bits in little endian of 32 bit python int.
00102B	F5	B	265	push	af ; Save checksum for now.

\*\*\*\*\* TRSDOS \*\*\*\*\*  
<Floppy Disk Driver>

PC	Object	I	Line	Source	.. \SYSRES\drivers\driver-FDCDVR.s
		B	266		
		B	267		UART1.put.bytes 4,chk_sum,fdc_wr_err1 ; Send server 4 byte int containing checksum.
		B+	267		scope
00102C	0604	B+	267		ld b,4 ; Were sending "count" bytes.
00102E	21 7F 0E	B+	267		ld hl,chk_sum
001031	4E	B+	267	__501	ld c,(hl)
001032	23	B+	267		inc hl
		B+	267		UART1.put.byte fdc_wr_err1 ; If nz we had a timeout or other error.
		B+	267		scope
001033	AF	B+	267		xor a ; Set Z flag set means PUT.
		B+	267		GOSUB U1DVR ; Call UART driver.
		B+	267		scope
001034	CD F5 11	B+	267		CALL U1DVR
		B+	267		endmac GOSUB
		B+	267		IF_NE_GOSUB fdc_wr_err1 ; If error goto error routine.
		B+	267		scope
001037	C4 C7 10	B+	267		call nz,fdc_wr_err1
		B+	267		endmac IF_NE_GOSUB
		B+	267		endmac UART1.put.byte
		B+	267		LOOP __501
		B+	267		scope
00103A	10 F5	B+	267		djnz __501
		B+	267		endmac LOOP
		B+	267		endmac UART1.put.bytes
		B	268		
		B	269		UART1.get.bytes 4,chk_sum,fdc_wr_err1 ; Get 4 byte checksum returning from server.
		B+	269		scope
00103C	0604	B+	269		ld b,4 ; Input 5 bytes.
00103E	21 7F 0E	B+	269		ld hl,chk_sum ; Get address of buffer.
001041		B+	269	__551	UART1.get.byte fdc_wr_err1 ; Get byte from UART1. If timeout goto no_pong.
		B+	269		scope
001041	3E01	B+	269		ld a,1
001043	B7	B+	269		or a ; On timeout or error goto vector.
001044	37	B+	269		scf
		B+	269		GOSUB U1DVR ; C flag set & NZ set means PUT.
		B+	269		scope
001045	CD F5 11	B+	269		CALL U1DVR
		B+	269		endmac GOSUB
		B+	269		IF_NE_GOSUB fdc_wr_err1 ; If nz we had a timeout or other error.
		B+	269		scope
001048	C4 C7 10	B+	269		call nz,fdc_wr_err1
		B+	269		endmac IF_NE_GOSUB
		B+	269		endmac UART1.get.byte
00104B	71	B+	269		ld (hl),c ; Store char.
00104C	23	B+	269		inc hl
		B+	269		LOOP __551 ; Loop for count bytes.
		B+	269		scope
00104D	10 F2	B+	269		djnz __551
		B+	269		endmac LOOP
		B+	269		endmac UART1.get.bytes
		B	270		
00104F	C1	B	271		pop bc ; Recover original checksum, B will have old A.
001050	3A 82 0E	B	272		ld a,(chk_sum+3) ; Compare round trip checksum in buffer to original.
001053	B8	B	273		cp b
		B	274		IF_EQ_GOTO net_wr_done ; Another job well done!
		B+	274		scope
001054	CA 63 10	B+	274		jp z,net_wr_done
		B+	274		endmac IF_EQ_GOTO
		B	275		
001057	21 84 0E	B	276		ld hl,wr_retries

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt;Floppy Disk Driver&gt;

PC	Object	I	Line	Source	.. \SYSRES\drivers\driver-FDCDVR.s
00105A	34	B	277	inc	(hl) ; Ooops bump counter +1.
		B	278	IF_EQ_GOTO	fdc_wr_err ; If rolling over to 0 it means 256 retries, abort this write.
		B+	278	scope	
00105B	CA C8 10	B+	278	jp	z, fdc_wr_err
		B+	278	endmac	IF_EQ_GOTO
00105E	3E2F	B	279	ld	a, '/' ; Signal server this is a write retry using retry char.
		B	280	GOTO	wr_retry
		B+	280	scope	
001060	C3 FF 0F	B+	280	jp	wr_retry
		B+	280	endmac	GOTO
		B	281		
001063	E1	B	282	net_wr_done	pop hl ; Discard buffer address.
001064	D1	B	283	POP	de ; Restore DE to track/sector.
001065	C3 ED 0E	B	284	jp	TSTBSY
		B	285		
001068		B	286	fdc_not_net_wr	; > WRITE sector to MEMDISK HERE <
		B	287		
001068	49ED5B 60 0E	B	288	LD.L	de, (PADDRESS) ; de now contains 24b address pointing to sector in RAM dri
		B	289		
00106D	E1	B	290	POP	hl ; Recover pointer to source.
00106E	22 60 0E	B	291	LD	(PADDRESS), hl ; Place 16 bit destination address in buffer.
001071	AF	B	292	xor	a
001072	32 62 0E	B	293	LD	(PADDRESS+2), a ; A-=0 from above code. Zero out upper 8 bits of 24b address.
		B	294		
001075	492A 60 0E	B	295	LD.L	hl, (PADDRESS) ; HL now contains 24b source pointer.
001079	5B010001 00	B	296	LD.Lil	BC, 100h ; Move one SECTOR or 256B.
00107E	49EDB0	B	297	LDIR.L	; Move sector from BUFFER to RAM DRIVE.
		B	298		
001081	D1	B	299	POP	de ; Restore DE to track/sector.
001082	C3 ED 0E	B	300	jp	TSTBSY
		B	301		
		B	302		; IY points do DCTx\$
		B	303		; D - contains track.
		B	304		; E - contains sector.
		B	305		
		B	306		; Calculate physical address based on track & sector.
		B	307		; Convert TRACK / SECTOR to physical RAM drive address.
		B	308		
		B	309		
		B	310		; 1st determine if sector on side 0 or side 1.
		B	311		
001085	FD7E07	B	312	calcsec	LD A, (IY+7) ; p/u alloc data
001088	E61F	B	313	AND	1FH ; Get highest # sector
00108A	93	B	314	SUB	E ; Form req sector minus
00108B	FDCB03A6	B	315	RES	4, (IY+3) ; init side select to 0
00108F	30 0A	B	316	JR	NC, FRCSID0 ; NC if sector on side 0
001091	FDCB046E	B	317	BIT	5, (IY+4) ; If not 2-sided media
001095	28 04	B	318	JR	Z, FRCSID0 ; don't set side 1
001097	FDCB03E6	B	319	SET	4, (IY+3) ; Set side 1
		B	320		
		B	321		; 2nd calculate sectors per cylinder.
		B	322		; D-contains track, E contains sector.
		B	323		
00109B	D5E1	B	324	FRCSID0	LD hl, DE ; sector in lower 8 bits of HL.
00109D	2600	B	325	ld	h, 0 ; Zero upper 8 bits. HL now contains sector.
00109F	FD7E07	B	326	LD	A, (IY+7) ; P/U sectors per track.
0010A2	E61F	B	327	AND	1Fh ; Mask out all but # SEC per TRK.
0010A4	3C	B	328	INC	A ; Adjust to correct pointer.
0010A5	FDCB046E	B	329	bit	5, (IY+4) ; Test if double sided drive?
0010A9	28 02	B	330	jr	z, stp23 ; NO, skip doubling sectors per track.

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt;Floppy Disk Driver&gt;

PC	Object	I	Line	Source	.. \SYSRES\drivers\driver-FDCDVR.s
0010AB	CB27	B	331	sla a	; A = A * 2, double sided - double sectors per trk.
		B	332		
		B	333		; 3rd perform following math leaving results in HL.
		B	334		; sector requested + (requested cylinder number * sectors per track).
		B	335		
0010AD	5F	B	336	stp23 LD E,A	; Move # of SEC per CYL to E.
0010AE	ED5C	B	337	MLT DE	; we then multiply (# CYL * SEC per CYL) 16b result in DE.
0010B0	19	B	338	add hl,de	; Add in sector number we saved before.
		B	339		
0010B1	22 74 0E	B	340	ld (fdc_no_sum),hl	; Save sum w/o offset for net driver.
		B	341		
		B	342		; 4th, using upper 16 bits of RAM drive base address, add in offset using results of above calculat
		B	343		; Leave results in HL.
		B	344		; RAMDRV\$ below was populated at entry to driver by looking up using drive #.
		B	345		
0010B4	11 B4 10	B	346	ld de,\$	
	000010B5	B	347	RAMDRV\$ equ \$-2	; Upper 16 bits of base RAM drive address.
0010B7	19	B	348	add hl,de	; Sum them together with offset to data.
		B	349		
		B	350		; Lastly, we stuff physical address of sector in PADDRESS.
		B	351		
0010B8	AF	B	352	XOR A	; A = 0 for below code.
0010B9	32 60 0E	B	353	LD (PADDRESS),a	
0010BC	22 61 0E	B	354	LD (PADDRESS+1),hl	
0010BF	C9	B	355	ret	
		B	356		
0010C0	E1	B	357	fdc_rd_err pop hl	; Discard return address.
0010C1	E1	B	358	pop hl	; Discard return address.
0010C2	3E03	B	359	ld a,03h	; Data lost on read.
0010C4	B7	B	360	or a	; Make NZ, error.
0010C5	FB	B	361	ei	
		B	362	RETURN	
		B+	362	scope	
0010C6	C9	B+	362	ret	; Return from call.
		B+	362	endmac RETURN	
		B	363		
0010C7	E1	B	364	fdc_wr_err1 pop hl	; Discard extra data.
0010C8	E1	B	365	fdc_wr_err pop hl	; Buffer address.
0010C9	D1	B	366	pop de	; This vector used for cleanup of 1 extra item on stack.
0010CA	C1	B	367	pop bc	; Recovered from save on entry to driver.
0010CB	E1	B	368	POP HL	; Per Soltoff we restore on exit any registers we used.
0010CC	3E0E	B	369	ld a,14	; Write fault on disk drive.
0010CE	B7	B	370	or a	; Make NZ, error
0010CF	FB	B	371	ei	
		B	372	RETURN	
		B+	372	scope	
0010D0	C9	B+	372	ret	; Return from call.
		B+	372	endmac RETURN	
		B	373		
0010D1	0600	B	374	calc_cksum ld b,0	
0010D3	AF	B	375	xor a	
0010D4	86	B	376	add (hl)	; A= A + (HL)
0010D5	23	B	377	inc hl	
		B	378	LOOP calc_cksum+3	
		B+	378	scope	
0010D6	10 FC	B+	378	djnz calc_cksum+3	
		B+	378	endmac LOOP	
		B	379	RETURN	
		B+	379	scope	
0010D8	C9	B+	379	ret	; Return from call.

\*\*\*\*\* TRSDOS \*\*\*\*\*  
<Floppy Disk Driver>

PC	Object	I	Line	Source ..\SYSRES\drivers\driver-FDCDVR.s	
		B+	379	endmac	RETURN
	000010D8	B	380		
		B	381	FDCEND:	EQU \$-1
		B	0	include	"driver-uart0.s" ; UART 0 driver.
		B	1	SUBTITLE	"< UART 0 - Driver. >"
		B	2	NEWPAGE	

\*\*\*\*\* TRSDOS \*\*\*\*\*  
 < UART 0 - Driver. >

PC	Object	I	Line	Source	.. \SYSRES\drivers\driver-uart0.s
		B	3		SCOPE
		B	4		
		B	5		
0010D9	18 11	B	6	U0DVR	JR U0BGN ;Driver entry point. Branch around linkage
0010DB	F411	B	7	DW U0END	;Last memory location used
0010DD	0324434C	B	8	DB 3,"\$CL"	
0010E1	3802	B	9	DW U0DCB\$	;DCB used.
0010E3	0000	B	10	DW 0	;Reserved.
		B	11		
	000010E5	B	12	CLDATA\$ EQU \$	
	00000000	B	13	MSMASK EQU \$-CLDATA\$	
0010E5	80	B	14	DB 80h	
		B	15		
		B	16		; UART Control Port image
		B	17		; Bit 7: 1 = Even parity, 0 = Odd parity
		B	18		; Bit 6 - 5 = Word length (00=5,10=6,01=7,11=8)
		B	19		; Bit 4: 1 = 2 stop bits, 0 = 1 stop bit
		B	20		; Bit 3: 1 = disable parity, 0 = enable parity
		B	21		; Bit 2: 1 = enable TX data, 0 = break
		B	22		; Bit 1: 0 = Data Terminal Ready
		B	23		; Bit 0: 0 = Request To Send
		B	24		
	00000001	B	25	UCIMAGE EQU \$-CLDATA\$	
0010E6	A4	B	26	DB 0A4H	; 10100101 = 7E1
0010E7	55	B	27	BAUDRT DB 55H	; Init 300 baud
	00000003	B	28	LOGBRK EQU \$-CLDATA\$	
0010E8	03	B	29	DB 03H	; Default is Ctl-C
	00000004	B	30	CLFLG EQU \$-CLDATA\$	
0010E9	00	B	31	DB 00H	; Init no char in buf
	00000005	B	32	CLBUF EQU \$-CLDATA\$	
0010EA	00	B	33	DB 00H	; One-char buffer
0010EB	02	B	34	esccount db 2	; Start counter off at 2.
0010EC	FB	B	35	U0BGN ei	
0010ED	DD21 E5 10	B	36	ld ix,CLDATA\$	; Base address of driver data pool.
0010F1	38 78	B	37	JR C,RXUART0	; Go if input req, GET.
0010F3	CA EB 11	B	38	Jp Z,TXUART0	; Go if output, PUT.
		B	39		
		B	40		; CTL request.
		B	41		
0010F6	79	B	42	CTLUART0 LD A,C	; Get @CTL byte
0010F7	B7	B	43	OR A	; @CTL 00?
0010F8	28 34	B	44	JR Z,CANISND	; Go if so
0010FA	3D	B	45	DEC A	
0010FB	28 7D	B	46	JR Z,CTL1	; Go if CTL 01
0010FD	3D	B	47	DEC A	
0010FE	28 0E	B	48	JR Z,CTL2	; CTL 02 "init uart"
001100	FE02	B	49	CP 4-2	; Wakeup feature?
001102	28 11	B	50	JR Z,CTL4	; Go if wakeup
		B	51	; XOR A	
001104	C9	B	52	RET	
		B	53		
		B	54		; CL initialization routine. Set up DR interrupt
		B	55		; vector and initialize the hardware
		B	56		
001105	3A 00 00	B	57	INIT LD A,(\$-)	; Get WRINT
		B	58	;OUT (@WRINT),A	
001108	CD 0E 11	B	59	CALL CTL2	
00110B	C9	B	60	LINK RET	
00110C	0000	B	61	DB 0,0	
		B	62		



\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; UART 0 - Driver. &gt;

PC	Object	I	Line	Source
		B	63	; Initialize the UART and BRG
		B	64	
00110E	ED4B E6 10	B	65	CTL2            LD    BC,(CLDATA\$+UCIMAGE)    ; Vals from DCB.
		B	66	;OUT (MASRES),A            ; Reprime UART.
001112	79	B	67	LD    A,C
		B	68	;OUT (UARTCTL),A
001113	78	B	69	LD    A,B
		B	70	;OUT (BAUDSET),A
001114	C9	B	71	RET
		B	72	
001115	FDE5	B	73	CTL4            PUSH   IY                   ; Xfer pointer to DE
001117	D1	B	74	POP   DE
001118	7A	B	75	LD    A,D                   ; Test if set or reset
001119	B3	B	76	OR    E
00111A	3EC9	B	77	LD    A,0C9H                ; Init disable reset
00111C	2A 89 11	B	78	LD    HL,(WAKEADR+1)        ; Get old value.
00111F	28 02	B	79	JR    Z,SETWAK              ; Jump if disable
001121	3EC3	B	80	LD    A,0C3H                ; Make enable
001123	32 88 11	B	81	SETWAK           LD    (WAKEADR),A        ; Load the opcode.
001126	ED53 89 11	B	82	LD    (WAKEADR+1),DE        ; Then the address.
00112A	E5	B	83	PUSH   HL                   ; Transfer pointer to IY
00112B	FDE1	B	84	POP   IY
00112D	C9	B	85	RET
		B	86	
		B	87	; Check if ready to send
		B	88	
00112E	ED38C6	B	89	CANISND           IN0    a,(UART0_MSR)        ; Get CTS status & test if CTS is ready..
001131	CB67	B	90	BIT    4,A                ; If other end sets CTS true, they are ready to receive.I just
001133	28 F9	B	91	JR    Z,CANISND            ; If not ready wait. Come on I am waiting in CANISND loop.
		B	92	
001135	ED38C5	B	93	IN0    A,(UART0_LSR)        ; Find out if UART ready to transmit.
001138	2F	B	94	cpl
001139	E620	B	95	AND    UART_THRE           ; Determine if our own UART is ready to send a character.
00113B	F5	B	96	push   AF                ; Save results of this test.
		B	97	
		B	98	; eZ80 modem status register format is as follows.
		B	99	
		B	100	; Bit 7 - CD Carrier detect.
		B	101	; Bit 6 - RI Ring indicator.
		B	102	; Bit 5 - DSR Data Set Ready.
		B	103	; Bit 4 - CTS Clear To Send.
		B	104	
		B	105	; TRS-80 modem status register format is as follows.
		B	106	
		B	107	; Bit 0 - Copy of serial input pin UART (Pin 20 of the DB-25).
		B	108	; Bit 4 - RI Ring Indicator (Pin 22 of the DB-25).
		B	109	; Bit 5 - CD Carrier Detect (Pin 8 of the DB-25).
		B	110	; Bit 6 - DSR Data Set Ready (Pin 6 of the DB-25).
		B	111	; Bit 7 - CTS Clear to Send (Pin 5 of the DB-25).
		B	112	
00113C	ED38C6	B	113	in0    a,(UART0_MSR)        ; Get modem status reg.
00113F	06F0	B	114	ld    b,0F0h               ; Set all upper 4 bits.
		B	115	
		B	116	; Test CD pin on eZ80 & if not set, reset CD on TRS-80 modem status.
		B	117	
001141	CB67	B	118	bit    4,a                ; Test eZ80 CTS pin and make TRS-80 CD pin follow it bc no CD on eZ80.
001143	20 00	B	119	jr    nz,b51               ; Is it ON? If OFF then reset this bit in TRS-80 port image.
		B	120	;                   res    5,b               ; Turn TRS-80 CD OFF.
		B	121	
		B	122	; Test RING on eZ80, if not on turn RING on TRS-80 modem status off.

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; UART 0 - Driver. &gt;

PC	Object	I	Line	Source	..\\SYSRES\\drivers\\driver-uart0.s
		B	123		
001145	CB77	B	124	b51	bit 6,a ; Test eZ80 RING indicator.
001147	20 02	B	125	jr	nz,b41 ; Is it ON? If OFF then reset this bit in TRS-80 port image.
001149	CBA0	B	126	res	4,b ; Turn TRS-80 RI OFF.
		B	127		
		B	128		; Test data set ready and adjust TRS-80 status accordingly.
		B	129		
00114B	CB6F	B	130	b41	bit 5,a ; Test eZ80 Data Set Ready.
00114D	20 02	B	131	jr	nz,b61 ; Is it ON? If OFF then reset this bit in TRS-80 port image.
00114F	CBB0	B	132	res	6,b ; Turn TRS-80 DSR OFF.
		B	133		
		B	134		; Test eZ80 Clear To Send (CTS). Adjust TRS-80 to match.
		B	135		
001151	CB67	B	136	b61	bit 4,a ; Test eZ80 Clear To Send.
001153	20 02	B	137	jr	nz,b71 ; Is it ON? If OFF then reset this bit in TRS-80 port image.
001155	CBB8	B	138	res	7,b ; Turn TRS-80 CTS OFF.
		B	139		
001157	78	B	140	b71	ld a,b
001158	2F	B	141	cpl	
001159	47	B	142	ld	b,a
00115A	F1	B	143	pop	AF ; Recover flag if UART empty. A has image of Modem Status Register.
00115B	78	B	144	ld	a,b ; Get TRS-80 format modem status register in A.
00115C	C0	B	145	RET	NZ ; Return if can't send.
		B	146		
00115D	DDAE 00	B	147	XOR	(IX+MSMASK) ; Mask for which to flip.
001160	1F	B	148	RRA	; Move into bits 0-3
001161	1F	B	149	RRA	
001162	1F	B	150	RRA	
001163	1F	B	151	RRA	
001164	DDA6 00	B	152	AND	(IX+MSMASK) ; Mask for which to check
001167	E60F	B	153	AND	0FH ; Mask off garbage
		B	154		
		B	155	; xor	a ; PLUG IT...say handshake always ready.
001169	78	B	156	LD	A,B ; Get reg back
00116A	C9	B	157	RET	; Ret with Z or NZ.
		B	158		
		B	159		; GET request. UART 0 I/O subroutine to receive byte into reg A.
		B	160		
00116B	CD 90 11	B	161	RXUART0	CALL CKINP ; Chk if avail from port
00116E	C0	B	162	RET	NZ ; Back if none
00116F	DDCB 04 26	B	163	SLA	(IX+CLFLG) ; Check if avail from buf
001173	30 F6	B	164	JR	NC,RXUART0 ; Go if none avail
001175	DD7E 05	B	165	LD	A,(IX+CLBUF) ; Get the char
001178	BF	B	166	CP	A ; Set Z flag & exit.
001179	C9	B	167	RET	
		B	168		
		B	169	; Break request	
		B	170		
00117A	DD7E 01	B	171	CTL1	ld a,(IX+UCIMAGE) ; Get UART CTL image.
00117D	CB97	B	172	res	2,a ; Show BREAK bit.
		B	173	;out	(UARTCTL),a
00117F	C9	B	174	ret	; Return with Z flag.
		B	175		; Data received interrupt handler
		B	176		
001180	DD21 E5 10	B	177	RECVINT	LD IX,CLDATA\$ ; Base of data area.
001184	CD 90 11	B	178	CALL	CKINP ; See if available from port.
001187	78	B	179	LD	A,B
001188	C9	B	180	WAKEADR	RET ; Wakeup if enabled.
001189	0000	B	181	DW	0 ; Space for address.
		B	182		

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; UART 0 - Driver. &gt;

PC	Object	I	Line	Source	.. \SYSRES\drivers\driver-uart0.s
		B	183		; Routine to check on a received character.
		B	184		
00118B	F601	B	185	nochar	or 1 ; Make NZ.
00118D	3E00	B	186	ld	a,0 ; A=0 No error.
00118F	C9	B	187	ret	
		B	188		
001190	ED38C5	B	189	CKINP	IN0 A,(UART0_LSR) ; Get status of data waiting? IN A,(UARTST) ; Check if
001193	E601	B	190	AND	UART_DR ; Char waiting?
001195	28 F4	B	191	JR	Z,nochar ; Make NZ, NO char waiting.
		B	192		
001197	3E80	B	193	ld	a,10000000b
001199	47	B	194	LD	B,A ; Save status.
		B	195		;CPL ; Mask Data Received bit.
		B	196		;AND 80H
		B	197		;LD A,00H ; Set "No error".
		B	198		;RET NZ ; Return if none.
		B	199		;IN A,(DATAREG) ; Get character.
		B	200		
		B	201		;ld b,00000000b ; Set TRS-80 modem status register to char waiting.
		B	202		
00119A	ED38C0	B	203	IN0	A,(UART0_RBR) ; Get byte from UART0.
00119D	4F	B	204	LD	C,A ; Save it in C.
		B	205		
00119E	FE1B	B	206	cp	ESC.asc ; User hit escape? If so inc count by 1.
0011A0	3A EB 10	B	207	ld	a,(escccount)
0011A3	20 0B	B	208	jr	nz,resetesc
0011A5	3D	B	209	dec	a
0011A6	20 0A	B	210	jr	nz,CRCK ; Have not reached 0 yet.
0011A8	3E02	B	211	ld	a,2 ; Reset counter to 2.
0011AA	32 EB 10	B	212	ld	(escccount),a
0011AD	C3 08 1B	B	213	jp	zABORT ; Two counts....were out of here!
		B	214		
		B	215		; Break, Pause and Enter handler routine.
		B	216		
0011B0	3E02	B	217	resetesc	ld a,2 ; Reset counter back to 2.
0011B2	32 EB 10	B	218	CRCK	ld (escccount),a
0011B5	79	B	219		ld a,c
0011B6	21 74 00	B	220	LD	HL,zKFLAG\$ ; KFLAG\$
0011B9	FE0D	B	221	CP	CR.asc ; ENTER char received?
0011BB	20 04	B	222	JR	NZ,PAWSCK ; Go if not
0011BD	CBD6	B	223	SET	2,(hl) ; Set ENTER bit
0011BF	18 20	B	224	JR	RCVEX
0011C1	FE60	B	225	PAWSCK	CP 60H ; Pause char received?
0011C3	20 04	B	226	JR	NZ,BRKCHK ; Go if not
0011C5	CBCE	B	227	SET	1,(HL) ; Set pause bit
0011C7	18 18	B	228	JR	RCVEX
0011C9	DD7E 03	B	229	BRKCHK	LD A,(IX+LOGBRK) ; Break char received?
0011CC	B7	B	230	OR	A ; Check if LOGBRK=0
0011CD	28 12	B	231	JR	Z,RCVEX ; No valid break if =0
0011CF	B9	B	232	CP	C ; Check if a valid break
0011D0	20 0F	B	233	JR	NZ,RCVEX ; Go if so.
		B	234		
		B	235		; A break was received, check system's BREAK disable
		B	236		
0011D2	3A 7C 00	B	237	BRKRECD	LD A,(zSFLAG\$) ; Check if break key.
0011D5	E610	B	238	AND	10H ; is disabled
0011D7	3E00	B	239	LD	A,00H ; Ret NZ & A=0 if
0011D9	C0	B	240	RET	NZ ; the BREAK is disabled
0011DA	21 74 00	B	241	LD	HL,zKFLAG\$ ; Hang onto flag
0011DD	CBC6	B	242	SET	0,(HL) ; Else set break bit

\*\*\*\*\* TRSDOS \*\*\*\*\*  
 < UART 0 - Driver. >

PC	Object	I	Line	Source	.. \SYSRES\drivers\driver-uart0.s
0011DF	0E80	B	243	LD C,80H	; & reset BREAK code
0011E1	DD71 05	B	244	RECVEX LD (IX+CLBUF),C	; Put char in 1 char buf
0011E4	DD36 04 80	B	245	LD (IX+CLFLG),80H	; Set char available
0011E8	79	B	246	ld a,c	
0011E9	BF	B	247	cp A	; Set Z flag.
0011EA	C9	B	248	ret	
		B	249		
		B	250		; Character PUT request. Transmit byte in register C.
		B	251		
0011EB	CD 2E 11	B	252	TXUART0: call CANISND	
0011EE	20 FB	B	253	JR nZ,TXUART0	
0011F0	ED09C0	B	254	OUT0 (UART0_THR),c	
0011F3	79	B	255	ld a,c	
0011F4	C9	B	256	RET	
		B	257		
	000011F4	B	258	U0END: EQU \$-1	
		B	0	include "driver-uart1.s"	; UART 1 driver.
		B	1	SUBTITLE "< UART 1 - Driver. RS-232 TERMINAL I/O subroutines. >"	
		B	2	SCOPE	
		B	3		
		B	4		; Transmit byte in register C.
		B	5		
0011F5	18 0A	B	6	U1DVR jr U1BGN	; Driver entry point, branch around linkage.
		B	7		
0011F7	7512	B	8	DW U1END	; Last memory location used.
0011F9	03245531	B	9	DB 3,"\$U1"	
0011FD	4002	B	10	DW U1DCB\$	; DCB used.
0011FF	0000	B	11	DW 0	; Reserved.
		B	12		
001201	28 40	B	13	U1BGN JR Z,TXUART1	; Go if output, PUT.
001203	38 0C	B	14	JR C,RXUART1	; Go if input, GET.
		B	15		
001205		B	16	CTLUART1	; CTL request - get status of char waiting & return.
		B	17		
001205	79	B	18	LD A,C	; If CTL 0, return
001206	B7	B	19	OR A	; status. Else
001207	28 08	B	20	JR Z,RXUART1	; treat as a Get
		B	21		
		B	22	UART1.tst.rx fdc_ctl_ret	; We return ready/not ready with status.
		B+	22	scope	
001209	ED38D5	B+	22	IN0 A,(UART1_LSR)	; Char waiting?
00120C	E601	B+	22	AND UART_DR	; 1 or TRUE if yes.
00120E	28 00	B+	22	jr z,fdc_ctl_ret	; Loop if not ready.
		B+	22	endmac UART1.tst.rx	
		B	23		
001210	C9	B	24	fdc_ctl_ret RET	
		B	25		
001211		B	26	RXUART1	; GET request, UART 0 I/O subroutines to receive byte into reg A.
		B	27		
001211	3E 19	B	28	ld a,echo_delay_ct	; Setup inits for timeout timer.
001213	32 75 12	B	29	ld (echo_timer),a	
001216	D5	B	30	push de	
001217	110000	B	31	ld de,0	; Inits for 65536 countdown.
		B	32		
		B	33	UART1.dtr.set	; Set DTR active, tell server we are here.
		B+	33	scope	
00121A	ED38D4	B+	33	in0 a,(UART1_MCTL)	; Get current register status.
00121D	F601	B+	33	or 1	; Set bit 0 to 1.
00121F	ED39D4	B+	33	OUT0 (UART1_MCTL),A	; UART1 tell other end we are ready.
		B+	33	endmac UART1.dtr.set	

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; UART 1 - Driver. RS-232 TERMINAL I/O subroutines. &gt;

PC	Object	I	Line	Source	.. \SYSRES\drivers\driver-uart1.s
		B	34	UART1.rts.set	; Set RTS active tell we are ready.
		B+	34	scope	
001222	ED38D4	B+	34	in0 a, (UART1_MCTL)	; Get current register status.
001225	F602	B+	34	or 2	; Set bit 0 to 1.
001227	ED39D4	B+	34	OUT0 (UART1_MCTL),A	; UART1 tell other end we are ready.
		B+	34	endmac	UART1.rts.set
		B	35		
00122A	CD 63 12	B	36	RXUART1WAIT	call uart1_timeout ; Call timeout timer, if counter reaches 0 no return.
		B	37		
		B	38	UART1.tst.rx	RXUART1WAIT ; loop until byte avail.
		B+	38	scope	
00122D	ED38D5	B+	38	IN0 A, (UART1_LSR)	; Char waiting?
001230	E601	B+	38	AND UART_DR	; 1 or TRUE if yes.
001232	28 F6	B+	38	jr z, RXUART1WAIT	; Loop if not ready.
		B+	38	endmac	UART1.tst.rx
		B	39		
		B	40	UART1.rts.reset	; Set RTS inactive telling other end wait.
		B+	40	scope	
001234	ED38D4	B+	40	in0 a, (UART1_MCTL)	
001237	E6FD	B+	40	and 0FDh	; Strip out bit 0 making reset.
001239	ED39D4	B+	40	OUT0 (UART1_MCTL),A	; UART1 tell other end we are ready.
		B+	40	endmac	UART1.rts.reset
		B	41		
00123C	ED08D0	B	42	IN0 c, (UART1_RBR)	; GET char waiting.
		B	43		
00123F	D1	B	44	pop de	; Restore DE to value on entry.
001240	79	B	45	ld a, c	
001241	BF	B	46	cp a	; Set Z flag to show successful completion.
001242	C9	B	47	RET	
		B	48		
001243		B	49	TXUART1	; Character PUT request, transmit byte in register C.
		B	50		
001243	3E 19	B	51	ld a, echo_delay_ct	; Setup inits for timeout timer.
001245	32 75 12	B	52	ld (echo_timer), a	
001248	D5	B	53	push de	
001249	110000	B	54	ld de, 0	; Inits for 65536 countdown.
		B	55		
00124C	CD 63 12	B	56	TXUART1WAIT	call uart1_timeout ; Call timeout timer. If this counter has reached 0 there i
		B	57		
		B	58	UART1.tst.cts	TXUART1WAIT ; Test if UART CTS ready? If <> ready wait in loop, calling tim
		B+	58	scope	
00124F	ED38D6	B+	58	IN0 a, (UART1_MSR)	; Get CTS status & test if CTS is ready..
001252	CB67	B+	58	BIT 4, A	; If other end sets CTS true, they are ready to receive.
001254	28 F6	B+	58	jr z, TXUART1WAIT	; Loop until ready.
		B+	58	endmac	UART1.tst.cts
		B	59		
		B	60	UART1.tst.tx	TXUART1WAIT ; Test if UART1 xmitter empty & ready accept data.
		B+	60	scope	; Test UART 1 if xmitter buffer empty and ready.
001256	ED38D5	B+	60	IN0 A, (UART1_LSR)	
001259	E620	B+	60	AND UART_THRE	; Returning a NE or <> 0 means ready, 0 means not ready.
00125B	28 EF	B+	60	jr z, TXUART1WAIT	
		B+	60	endmac	UART1.tst.tx
		B	61		
00125D	ED09D0	B	62	OUT0 (UART1_THR), C	; Send byte into the ether....so to speak.
		B	63		
001260	D1	B	64	pop de	; Restore DE.
001261	AF	B	65	XOR A	; A=0, no error.
		B	66	RETURN	
		B+	66	scope	
001262	C9	B+	66	ret	; Return from call.

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; UART 1 - Driver. RS-232 TERMINAL I/O subroutines. &gt;

PC	Object	I	Line	Source	.. \SYSRES\drivers\driver-uart1.s
		B+	66	endmac	RETURN
		B	67		
001263	1B	B	68	uart1_timeout	dec de ; count = count - 1.
001264	7A	B	69	ld	a,d
001265	B3	B	70	or	e
		B	71	IF_NE_RETURN	; IF 65536 timer <> 0
		B+	71	scope	
001266	C0	B+	71	ret	nz
		B+	71	endmac	IF_NE_RETURN
001267	3A 75 12	B	72	ld	a,(echo_timer) ; Initialize DE in case we go another round.
00126A	3D	B	73	dec	a ; Timer = timer - 1.
00126B	32 75 12	B	74	ld	(echo_timer),a
		B	75	IF_NE_RETURN	; If timer <> 0, return.
		B+	75	scope	
00126E	C0	B+	75	ret	nz
		B+	75	endmac	IF_NE_RETURN
00126F	D1	B	76	pop	de ; Discard return address in driver.
001270	D1	B	77	pop	de ; Restore register to entry of driver.
001271	3E01	B	78	ld	a,1
001273	B7	B	79	or	a ; Make NZ.
		B	80	RETURN	; Return with error.
		B+	80	scope	
001274	C9	B+	80	ret	; Return from call.
		B+	80	endmac	RETURN
		B	81		
	00000019	B	82	echo_delay_ct	equ 25 ; Single source to change delay for assembly.
001275	19	B	83	echo_timer	db echo_delay_ct ; Holds counter when running timer for timeout detection.
		B	84		
	00001275	B	85	U1END:	EQU \$-1
		B	0	include	"driver-tod.s" ; Time Of Day Clock driver.
		B	1		; TOD - Time of Day Clock.
		B	2		
001276	18 0D	B	3	JR	GETDATE ; Driver entry point. Branch around linkage
001278	FF12	B	4	DW	RTCEND\$ ; Last memory location used
00127A	06525443 445652	B	5	DB	6,"RTCDVR"
001281	0000	B	6	DW	0 ; DCB used.
001283	0000	B	7	DW	0
		B	8		
001285		B	9	GETDATE	RTC.YEAR.get a ; Get year.
001285	ED38E6	B+	9	in0	a,(RTC_YR)
001288	B7	B+	9	or	a
001289	20 02	B+	9	jr	nz,__781
00128B	3E23	B+	9	ld	a,23h
00128D	7F	B+	9	__781	ld a,a
		B+	9	ENDMAC	
00128E	CD ED 12	B	10	call	bcd2bin
001291	C664	B	11	add	100 ; DATE is stored in excess 1900.
001293	32 33 00	B	12	ld	(DATE\$),a ; Store years in excess of 1900.
		B	13		
		B	14	RTC.DOM.get	a ; Get day of month.
001296	ED38E4	B+	14	in0	a,(RTC_DOM)
001299	B7	B+	14	or	a
00129A	20 02	B+	14	jr	nz,__791
00129C	3E20	B+	14	ld	a,20h
00129E	7F	B+	14	__791	ld a,a
		B+	14	ENDMAC	
00129F	CD ED 12	B	15	call	bcd2bin
0012A2	32 34 00	B	16	ld	(DATE\$+1),a ; Get day of month in binary from RTC.
		B	17		
		B	18	RTC.MON.get	a ; Get month in binary from RTC.

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; UART 1 - Driver. RS-232 TERMINAL I/O subroutines. &gt;

PC	Object	I	Line	Source	.. \SYSRES\drivers\driver-tod.s
0012A5	ED38E5	B+	18	in0	a, (RTC_MON)
0012A8	B7	B+	18	or	a
0012A9	20 02	B+	18	jr	nz, __801
0012AB	3E07	B+	18	ld	a, 07h
0012AD	7F	B+	18	__801	ld a, a
		B+	18		ENDMAC
0012AE	CD ED 12	B	19	call	bcd2bin
0012B1	32 35 00	B	20	LD	(DATE\$+2), a ; Point DE to DATE buffer end.
		B	21		
0012B4	11 35 00	B	22	LD	DE, DATE\$+2
0012B7	C9	B	23	ret	
		B	24		
0012B8		B	25	GETTIME	RTC.HOUR.get a ; Get hour in binary from RTC.
0012B8	ED38E2	B+	25	in0	a, (RTC_HRS) ;
0012BB	7F	B+	25	ld	a, a
		B+	25		ENDMAC
0012BC	CD ED 12	B	26	call	bcd2bin
0012BF	32 2F 00	B	27	ld	(TIME\$+2), a
		B	28		
		B	29		RTC.MIN.get a
0012C2	ED38E1	B+	29	in0	a, (RTC_MIN) ;
0012C5	7F	B+	29	ld	a, a
		B+	29		ENDMAC
0012C6	CD ED 12	B	30	call	bcd2bin
0012C9	32 2E 00	B	31	ld	(TIME\$+1), a
		B	32		
		B	33		RTC.SEC.get a
0012CC	ED38E0	B+	33	in0	a, (RTC_SEC) ;
0012CF	7F	B+	33	ld	a, a
		B+	33		ENDMAC
0012D0	CD ED 12	B	34	call	bcd2bin
0012D3	32 2D 00	B	35	ld	(TIME\$), a
		B	36		
0012D6	11 2F 00	B	37	LD	DE, TIME\$+2 ;Point to time\$
0012D9	C9	B	38	RET	
		B	39		
		B	40		;*****
		B	41		; a(BIN) => a(BCD)
		B	42		; [0..99] => [00h..99h]
		B	43		;*****
		B	44		;
0012DA	C5	B	45	bin2bcd	push bc
0012DB	060A	B	46		ld b, 10
0012DD	0EFF	B	47		ld c, -1
0012DF	0C	B	48	div10:	inc c
0012E0	90	B	49		sub b
0012E1	30 FC	B	50		jr nc, div10
0012E3	80	B	51		add a, b
0012E4	47	B	52		ld b, a
0012E5	79	B	53		ld a, c
0012E6	87	B	54		add a, a
0012E7	87	B	55		add a, a
0012E8	87	B	56		add a, a
0012E9	87	B	57		add a, a
0012EA	B0	B	58		or b
0012EB	C1	B	59		pop bc
0012EC	C9	B	60		ret
		B	61		;*****
		B	62		; a(BCD) => a(BIN)
		B	63		; [00h..99h] -> [0..99]

\*\*\*\*\* TRSDOS \*\*\*\*\*

< UART 1 - Driver. RS-232 TERMINAL I/O subroutines. >

PC	Object	I	Line	Source ..\SYSRES\drivers\driver-tod.s
		B	64	;*****
		B	65	
0012ED	C5	B	66	bcd2bin: push bc
0012EE	4F	B	67	ld c,a
0012EF	E6F0	B	68	and 0f0h
0012F1	CB3F	B	69	srl a
0012F3	47	B	70	ld b,a
0012F4	CB3F	B	71	srl a
0012F6	CB3F	B	72	srl a
0012F8	80	B	73	add a,b
0012F9	47	B	74	ld b,a
0012FA	79	B	75	ld a,c
0012FB	E60F	B	76	and 0fh
0012FD	80	B	77	add a,b
0012FE	C1	B	78	pop bc
0012FF	C9	B	79	ret
		B	80	
	000012FF	B	81	RTCEND\$ EQU \$-1
		B	82	
		B	83	
		B	84	
		B	85	
		A	120	
		A	121	ORG zspace.BYTEI0\$ ; Beginning of sys0.
		A	122	
		B	0	INCLUDE "filposn.s" ; File positioning routines. -MUST BE FIRST
		B	1	SUBTITLE "< File positioning subroutines >"
		B	2	NEWPAGE



\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; File positioning subroutines &gt;

PC	Object	I	Line	Source	.. \SYSRES \filposn.s
		B	3		scope
		B	4		
		B	5		; Entry for byte I/O from zGET & zPUT
		B	6		
001300	DDE5	B	7	BYTEIO	PUSH IX
001302	D1	B	8		POP DE ;Transfer DCB to DE
001303	CD 68 15	B	9		CALL CKOPENz ;Ck file open, save regs
001306	DDCB01FE	B	10		SET 7,(IX+1) ;Denote byte or LRec
00130A	78	B	11		LD A,B ;Get type code & test
00130B	FE02	B	12		CP 2 ;For get/put
00130D	79	B	13		LD A,C
00130E	28 1F	B	14		JR Z,WRCHAR ;Go on PUT
001310	30 58	B	15		JR NC,IORETZ ;Ignore if CTL
		B	16		
		B	17		; Get a byte from a file
		B	18		
001312	CD 92 15	B	19	RDCHAR	CALL CKEOF1 ;Ck for end of file
001315	C0	B	20		RET NZ ;Return if at end
001316	DDCB016E	B	21		BIT 5,(IX+1) ;If buffer not current,
00131A	C4 79 13	B	22		CALL NZ,NSEC1 ;read next sector
00131D	C0	B	23		RET NZ
00131E	CD 12 14	B	24		CALL BFRPOS ;Pt to byte posn in bfr
001321	1A	B	25		LD A,(DE) ;P/u the byte
001322	DD3405	B	26		INC (IX+5) ;Inc NEXT ptr
001325	CC 2A 13	B	27		CALL Z,SET5 ;Set bit 5 if zero
001328	BF	B	28		CP A ;Set Z flag--no error
001329	C9	B	29		RET
00132A	DDCB01EE	B	30	SET5	SET 5,(IX+1)
00132E	C9	B	31		RET
		B	32		
		B	33		; Write a byte to a file
		B	34		
00132F	DDCB0076	B	35	WRCHAR	BIT 6,(IX+0) ;Prot level give write acc?
001333	CA C6 13	B	36		JP Z,RWRIT3 ;go if not
001336	F5	B	37		PUSH AF ;Save byte
001337	DDCB016E	B	38		BIT 5,(IX+1) ;Get next sector if
00133B	C4 6C 13	B	39		CALL NZ,WRCH2 ;buffer is not current
00133E	28 03	B	40		JR Z,WRCH1 ;Skip if read was ok
001340	E3	B	41		EX (SP),HL ;Pop stack but keep
001341	E1	B	42		POP HL ;error # in AF
001342	C9	B	43		RET
		B	44		
001343	CD 12 14	B	45	WRCH1	CALL BFRPOS ;Next bfr byte posn
001346	F1	B	46		POP AF
001347	12	B	47		LD (DE),A ;Stuff the byte
001348	DDCB01E6	B	48		SET 4,(IX+1) ;Buffer contains updated data
00134C	DD3405	B	49		INC (IX+5) ;Inc NEXT byte
00134F	F5	B	50		PUSH AF ;Save Z or NZ flag
001350	CC 2A 13	B	51		CALL Z,SET5 ;Set bit 5 if offset 0
001353	CD 92 15	B	52		CALL CKEOF1 ;Check for EOF
001356	20 06	B	53		JR NZ,ATEOFW ;Go if there
001358	DDCB0176	B	54		BIT 6,(IX+1) ;Jump if EOF set to next
00135C	20 09	B	55		JR NZ,DNTSET ;only if at EOF
00135E		B	56	ATEOFW	
00135E	DD7108	B	57		LD (IX+8),C ;Set EOF
001361	DD750C	B	58		LD (IX+12),L
001364	DD740D	B	59		LD (IX+13),H
001367	F1	B	60	DNTSET	POP AF ;Restore offset flag
001368	28 46	B	61		JR Z,RWRIT1 ;Go to write sector if 00
00136A		B	62	IORETZ	

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; File positioning subroutines &gt;

PC	Object	I	Line	Source	.. \SYSRES\filposn.s
00136A	AF	B	63	XOR	A ;Set Z flag--no error
00136B	C9	B	64	RET	
		B	65		
		B	66	; WRCHR needs the next sector - if UPDATE, ck EOF	
		B	67		
00136C	DD7E01	B	68	WRCH2	LD A,(IX+1) ;Ck if UPD bit set
00136F	E607	B	69	AND	7 ;Mask for prot level
001371	FE04	B	70	CP	4 ;Check for UPD
001373	20 04	B	71	JR	NZ,NSEC1 ;Bypass EOF ck on > UPD
001375	CD 92 15	B	72	NXTSECT	CALL CKE0F1 ;Ck for end of file
001378	C0	B	73	RET	NZ ;Can"t extend in update mode.
001379	DD7E01	B	74	NSEC1	LD A,(IX+1) ;Read access?
00137C	E607	B	75	AND	7
00137E	FE06	B	76	CP	6
001380	30 44	B	77	JR	NC,RWRIT3 ;"Illegal access..." if not
001382	CD CB 15	B	78	NSEC2	CALL IOREC ;Calc cylinder/sector
001385	C0	B	79	RET	NZ ; Return on error.
001386	DDCB01AE	B	80	RES	5,(IX+1) ;Show buffer current
00138A	DD6E03	B	81	LD	L,(IX+3) ;P/u buffer address
00138D	DD6604	B	82	LD	H,(IX+4)
001390	CD F4 19	B	83	CALL	zRDSEC ;Read the sector
001393	28 03	B	84	JR	Z,BUMPNRN ;Go if no error
001395	FE06	B	85	CP	6 ;Test for prot sector
001397	C0	B	86	RET	NZ ;Quit if error not 6
001398		B	87	BUMPNRN	
001398	DD340A	B	88	INC	(IX+10) ;Inc the NRN ptr LSB
00139B	20 03	B	89	JR	NZ,ZEROAz
00139D	DD340B	B	90	INC	(IX+11) ;and MSB if necessary
0013A0		B	91	zSEEKSC:	; <631>
0013A0	AF	B	92	ZEROAz	XOR A
0013A1	C9	B	93	RET	
		B	94		
		B	95	; Repositioning needs to write out the buffer	
		B	96		
0013A2	DD7E01	B	97	RWRITz	LD A,(IX+1)
0013A5	E690	B	98	AND	90h ;Test for non-sector i/o and
0013A7	FE90	B	99	CP	90h ;buffer contents changed
0013A9	28 05	B	100	JR	Z,RWRIT1 ;Go if conditions true
0013AB	18 F3	B	101	JR	ZEROAz ;else no need to write
0013AD	CD 68 15	B	102	zRWRIT	CALL CKOPENz ;Ck file open, save regs
0013B0	CD 0B 14	B	103	RWRIT1	CALL GETNRN ;P/u NRN
0013B3	7C	B	104	LD	A,H ;Ignore if rewound
0013B4	B5	B	105	OR	L
0013B5	C8	B	106	RET	Z
0013B6	2B	B	107	DEC	HL ;Dec & reset NRN
0013B7	DD750A	B	108	LD	(IX+10),L
0013BA	DD740B	B	109	LD	(IX+11),H
		B	110		
		B	111	; Check access protection level	
		B	112		
0013BD	DD7E01	B	113	RWRIT2	LD A,(IX+1) ;Get prot
0013C0	E607	B	114	AND	7
0013C2	FE05	B	115	CP	5 ;Update access or better?
0013C4	38 04	B	116	JR	C,RWRIT4
0013C6	3E25	B	117	RWRIT3	LD A,25h ;Illegal access error code
0013C8	B7	B	118	OR	A ;Return NZ
0013C9	C9	B	119	RET	
0013CA		B	120	RWRIT4	
0013CA	E604	B	121	AND	4 ;If UPDATE access, then
0013CC	28 05	B	122	JR	Z,RWRIT5 ;can"t extend if at EOF

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; File positioning subroutines &gt;

PC	Object	I	Line	Source	.. \SYSRES\filposn.s
0013CE	CD 92 15	B	123		CALL CKEOF1
0013D1	20 F3	B	124		JR NZ,RWRIT3 ;so show "Illegal access...
0013D3		B	125	RWRIT5	
0013D3	CD CB 15	B	126		CALL IOREC ;Calculate cylinder & sector
0013D6	C0	B	127		RET NZ
0013D7	DD6E03	B	128		LD L,(IX+3) ;P/u buffer addr
0013DA	DD6604	B	129		LD H,(IX+4)
0013DD	DDCB01A6	B	130		RES 4,(IX+1) ;Altered buffer flag off
0013E1	DDCB00D6	B	131		SET 2,(IX+0) ;Show modification done
0013E5	CD E8 19	B	132		CALL zWRSEC ;for directory mod flag
0013E8	C0	B	133		RET NZ
0013E9	3E00	B	134	VEROP	LD A,0 ;Verify operation if set
0013EB	B7	B	135		OR A
0013EC	C4 DC 19	B	136		CALL NZ,zVRSEC ;Verify if no write error
0013EF	C0	B	137		RET NZ ;Return if wrt/ver error
0013F0	CD 98 13	B	138		CALL BUMPNRN ;Increment NRN
		B	139		
		B	140		; Check if ERN to be set to NRN
		B	141		; Should be done for byte i/o, but not random i/o
		B	142		
0013F3	CD 92 15	B	143		CALL CKEOF1 ;Returns 0 if not at EOF
0013F6	3D	B	144		DEC A ;Set bit 6 if retcod=0
0013F7	DDA601	B	145		AND (IX+1) ;If IX+1, bit 6 set, then
0013FA	E640	B	146		AND 40h ;don't update EOF unless at
0013FC	20 A2	B	147		JR NZ,ZEROAz ;or past the old EOF
0013FE	DD750C	B	148	YESEOF	LD (IX+12),L ;Update ERN
001401	DD740D	B	149		LD (IX+13),H
001404	DDCB015E	B	150		BIT 3,(IX+1) ;Test if ending '!'
001408	20 2C	B	151		jr NZ,WEOF1 ;<631>Upd dir if so
00140A	C9	B	152		RET
00140B		B	153	GETNRN	
00140B	DD6E0A	B	154		LD L,(IX+10) ;Xfer NRN to HL
00140E	DD660B	B	155		LD H,(IX+11)
001411	C9	B	156		RET
001412		B	157	BFRPOS	
001412	DD7E05	B	158		LD A,(IX+5) ;P/u byte offset in buffer
001415	DD8603	B	159		ADD A,(IX+3) ;Add to buffer lsb
001418	5F	B	160		LD E,A
001419	DD7E04	B	161		LD A,(IX+4) ;and adjust buffer MSB
00141C	CE00	B	162		ADC A,0 ;if needed
00141E	57	B	163		LD D,A ;Return DE = posn
00141F	C9	B	164		RET
		B	165		
		B	166		; 6.3.1 >>> changes dpm.
		B	167		
001420	CD A8 07	B	168	zDATE	CALL DATEz ; @DATE handler
001423	E5	B	169		PUSH HL ; Points to end of date buffer
001424	2B	B	170		DEC HL ; back up 2
001425	2B	B	171		DEC HL ; Point to 10s of years
001426	7E	B	172		LD A,(HL) ; Get digit
001427	FE3A	B	173		CP '9'+1 ; greater than 9?
001429	38 03	B	174		JR C,M142E ; jump if not
00142B	D60A	B	175		SUB 0AH ; Convert back to decimal
00142D	77	B	176		LD (HL),A ; Store in buffer
00142E	E1	B	177	M142E	POP HL ; Recover date ptr
00142F	C9	B	178		RET ; Return to caller
		B	179		
		B	180		; Entry to Write an end-of-file mark
		B	181		; This routine relocated here to allow more relative branch references.
		B	182		

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; File positioning subroutines &gt;

PC	Object	I	Line	Source	..\\SYSRES\\filposn.s
		B	183		org 1430H
001430	CD 68 15	B	184	zWE0F	CALL CKOPENz
001433	CD A2 13	B	185		CALL RWRITz ;Write buffer if needed
001436	DD4607	B	186	WE0F1	LD B,(IX+7) ;P/u DEC of FPDE
001439	DD4E06	B	187		LD C,(IX+6) ;P/u drive #
00143C	CD BB 18	B	188		CALL zDIRRD ;Read file"s dir record
00143F	C0	B	189		RET NZ ;Back if read error
001440	2C	B	190		INC L ;Pt to ERN offset
001441	2C	B	191		INC L
001442	2C	B	192		INC L
001443	DD7E08	B	193		LD A,(IX+8) ;P/u EOF offset
001446	77	B	194		LD (HL),A ;Put in direc
001447	111100	B	195		LD DE,17 ;Pt to EOF in dir
00144A	19	B	196		ADD HL,DE
00144B	DD7E0C	B	197		LD A,(IX+12) ;P/u lo EOF
00144E	77	B	198		LD (HL),A ;Put EOF in direc
00144F	23	B	199		INC HL
001450	DD7E0D	B	200		LD A,(IX+13) ;P/u hi EOF
001453	77	B	201		LD (HL),A
001454	C3 03 18	B	202		JP zDIRWR ;Write direc and return
		B	203		
		B	204		
		B	205		; Entry to Skip record routine
		B	206		org 1457H
001457	CD DA 14	B	207	zSKIP	CALL zLOC ;Locate next record
00145A	03	B	208		INC BC ;Step past it
		B	209		
		B	210		; Entry to Position to record routine
		B	211		
00145B	CD 68 15	B	212	zPOSN	CALL CKOPENz
00145E	DDCB01F6	B	213		SET 6,(IX+1) ;Upd eof only if NRN>EOF
001462	DDCB017E	B	214		BIT 7,(IX+1) ;Jump if sector i/o only
001466	28 1D	B	215		JR Z,POSN1
001468	60	B	216		LD H,B ;Record ptr to HL
001469	69	B	217		LD L,C
00146A	DDB609	B	218		OR (IX+9) ;P/u LRL
00146D	28 16	B	219		JR Z,POSN1 ;Skip nxt if LRL=256
00146F	CD C9 06	B	220		CALL zMUL16 ;Calc sector & offset
001472	44	B	221		LD B,H ;Physical sector =>BC
001473	4D	B	222		LD C,L
001474	DD7705	B	223		LD (IX+5),A ;Set byte ptr
001477	DDCB016E	B	224		BIT 5,(IX+1) ;Jump if buffer does not
00147B	20 0B	B	225		JR NZ,POSN2 ;contain current sector
00147D	CD 0B 14	B	226		CALL GETNRN ;P/u the NRN
001480	37	B	227		SCF
001481	ED42	B	228		SBC HL,BC
001483	28 12	B	229		JR Z,\$CKEOF ;Pass on to CKEOF
001485	DD7705	B	230	POSN1	LD (IX+5),A ;Offset in buffer
001488	C5	B	231	POSN2	PUSH BC
001489	CD A2 13	B	232	POSN2A	CALL RWRITz ;Write current if needed
00148C	C1	B	233		POP BC ;before moving
00148D	C0	B	234		RET NZ ;Back on write error
00148E	DD710A	B	235		LD (IX+10),C ;NRN
001491	DD700B	B	236		LD (IX+11),B
001494	CD 2A 13	B	237		CALL SET5 ;Show bfr does not
001497	C3 92 15	B	238	\$CKEOF	JP CKEOF1 ;contain current sector
		B	239		
		B	240		; Entry to force a physical read
		B	241		
00149A	CD 68 15	B	242	zRREAD	CALL CKOPENz

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; File positioning subroutines &gt;

PC	Object	I	Line	Source	..\\SYSRES\\filposn.s
00149D	0E01	B	243		LD C,1 ;Cause ADJUST to bump NRN when called.
00149F	CD 0B 14	B	244	BKSP1	CALL GETNRN ;Get current record #
0014A2	7C	B	245		LD A,H ;If file is rewound,
0014A3	B5	B	246		OR L ;then ignore the req
0014A4	28 15	B	247		JR Z,BKSP0 ;& force OFFSET = 0
0014A6	2B	B	248		DEC HL ;Back up by 1
0014A7	CD BA 15	B	249		CALL ADJ2 ;RET if sector I/O only,
		B	250		;else bump fwd if RREAD
		B	251		;then back up if bit 5=0
0014AA	E5	B	252		PUSH HL ;Will be popped into BC
0014AB	18 DC	B	253		JR POSN2A ;Finish the job
		B	254		
		B	255		; Entry to backspace one logical record
		B	256		
0014AD	CD 68 15	B	257	zBKSP	CALL CKOPENz
0014B0	4F	B	258		LD C,A ;Keep ADJUST from bumping
0014B1	DD4609	B	259		LD B,(IX+9) ;P/u LRL
0014B4	B0	B	260		OR B ;Is it a 0
0014B5	28 E8	B	261		JR Z,BKSP1 ;Go if so
0014B7	DD7E05	B	262		LD A,(IX+5) ;P/u next byte pointer
0014BA	90	B	263		SUB B ;Sub one record length
0014BB	DD7705	B	264	BKSP0	LD (IX+5),A
0014BE	38 DF	B	265		JR C,BKSP1 ;Go if crossed sec bdry
0014C0	AF	B	266		XOR A ;else all done
0014C1	C9	B	267		RET
		B	268		
		B	269		; Entry to Rewind to beginning
		B	270		
0014C2	CD 68 15	B	271	zREW	CALL CKOPENz
0014C5	47	B	272		LD B,A ;Zero NRN
0014C6	4F	B	273		LD C,A
0014C7	18 BC	B	274		JR POSN1 ;Will also zero offset
		B	275		
		B	276		; Entry to Position to end-of-file
		B	277		
0014C9	CD 68 15	B	278	zPEOF	CALL CKOPENz
0014CC	DD4E0C	B	279		LD C,(IX+12) ;ERN to BC
0014CF	DD460D	B	280		LD B,(IX+13)
0014D2	DDB608	B	281		OR (IX+8) ;P/u EOF byte
0014D5	28 AE	B	282		JR Z,POSN1 ;Go if full sector
0014D7	0B	B	283		DEC BC ;Point to last rec
0014D8	18 AB	B	284		JR POSN1 ;Use POSN to get end
		B	285		
		B	286		; Entry to Locate current record number
		B	287		
0014DA	CD 68 15	B	288	zLOC	CALL CKOPENz
0014DD	CD 0B 14	B	289		CALL GETNRN ;P/u NRN
0014E0	CD B7 15	B	290		CALL ADJUST ;Get offset and adj NRN
0014E3	DD5E09	B	291	LOC1	LD E,(IX+9) ;P/u LRL
0014E6	7B	B	292		LD A,E ;Test LRL for zero
0014E7	B7	B	293		OR A ;If zero, then give NRN
0014E8	28 16	B	294		JR Z,LOC3 ;LRL=0, NRN is correct
0014EA	0C	B	295		INC C ;If offset is zero,
0014EB	0D	B	296		DEC C ;then it's at 256,
0014EC	28 01	B	297		JR Z,LOC2 ;and we don't dec NRN
0014EE	2B	B	298		DEC HL
		B	299		
		B	300		; Divide the three byte pointer (HLC) by the LRL
		B	301		
0014EF	CD E3 06	B	302	LOC2	CALL zDIV16 ;Divide (NRN-1)/LRL

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; File positioning subroutines &gt;

PC	Object	I	Line	Source	.. \SYSRES\filposn.s
0014F2	45	B	303	LD	B,L ;Save high order result
0014F3	54	B	304	LD	D,H ;Save possible overflow
0014F4	67	B	305	LD	H,A ;Prepare 2nd dividend
0014F5	69	B	306	LD	L,C ;P/u low order dividend
0014F6	7B	B	307	LD	A,E ;P/u LRL divisor again
0014F7	CD E3 06	B	308	CALL	zDIV16
0014FA	60	B	309	LD	H,B ;Xfer high order result
0014FB	B7	B	310	OR	A ;If remainder, we have a
0014FC	28 01	B	311	JR	Z,\$+3 ;partial record to round
0014FE	23	B	312	INC	HL ;up to next record #
0014FF	7A	B	313	LD	A,D ;Xfer possible overflow
001500	C1	B	314	LOC3 POP	BC ;Pop RESTREG return adr
001501	E3	B	315	EX	(SP),HL ;Exchange value with BC
001502	C5	B	316	PUSH	BC ;Restore RESTREG
	00001503	B	317	ORARETz	EQU \$
001503	B7	B	318	OR	A
001504	C9	B	319	RET	
		B	320		
		B	321		; Entry to Locate the end-of-file record
		B	322		
001505	CD 68 15	B	323	zLOF	CALL CKOPENz
001508	DD6E0C	B	324	LD	L,(IX+12) ;P/u ERN
00150B	DD660D	B	325	LD	H,(IX+13)
00150E	DD4E08	B	326	LD	C,(IX+8) ;EOF byte
001511	18 D0	B	327	JR	LOC1 ;Handle all LRLs
		B	328		
		B	329		; Entry to Read a record
		B	330		
001513	CD 68 15	B	331	zREAD	CALL CKOPENz
001516	E5	B	332	PUSH	HL
001517	CD A2 13	B	333	CALL	RWRITz ;Write buffer if needed
00151A	E1	B	334	POP	HL
00151B	C0	B	335	RET	NZ ;Back on write error
00151C	DD4609	B	336	LD	B,(IX+9) ;P/u LRL
00151F	78	B	337	LD	A,B ;If LRL=256, just
001520	B7	B	338	OR	A
001521	CA 75 13	B	339	JP	Z,NXTSECT ;get the next sector
001524	E5	B	340	RDREC	PUSH HL ;Save buffer posn
001525	C5	B	341	PUSH	BC ;Save LRL
001526	CD 12 13	B	342	CALL	RDCHAR ;Read next byte
001529	C1	B	343	POP	BC
00152A	E1	B	344	POP	HL
00152B	C0	B	345	RET	NZ ;Back on read error
00152C	77	B	346	LD	(HL),A ;Put char into buffer
00152D	23	B	347	INC	HL ;Bump buffer ptr
00152E	10 F4	B	348	DJNZ	RDREC ;Loop for entire record
001530	C9	B	349	RET	
		B	350		
		B	351		; Entry to Write a record
		B	352		
001531	CD 68 15	B	353	zWRITE	CALL CKOPENz
001534	32 EA 13	B	354	WRIT1	LD (VEROP+1),A ;Turn on/off verify
001537	DD4609	B	355	LD	B,(IX+9) ;P/u LRL
00153A	78	B	356	LD	A,B ;Bypass if LRL=256
00153B	B7	B	357	OR	A
00153C	CA BD 13	B	358	JP	Z,RWRIT2
00153F	E5	B	359	PUSH	HL ;Save some FCB values
001540	DD6605	B	360	LD	H,(IX+5) ;P/u buffer offset loc
001543	DD6E08	B	361	LD	L,(IX+8) ;P/U EOF offset byte
001546	E3	B	362	EX	(SP),HL ;Put values on stack

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; File positioning subroutines &gt;

PC	Object	I	Line	Source	..\\SYSRES\\filposn.s
		B	363		;and recover HL
001547	7E	B	364	WRREC	LD A,(HL) ;Pass the logical record
001548	23	B	365		INC HL ;to the writing routine
001549	E5	B	366		PUSH HL ;byte by byte
00154A	C5	B	367		PUSH BC
00154B	CD 2F 13	B	368		CALL WRCHAR
00154E	C1	B	369		POP BC
00154F	E1	B	370		POP HL
001550	20 05	B	371		JR NZ,WRERROR ;Exit and fix FCB
001552	10 F3	B	372		DJNZ WRREC ;Loop for entire record
001554	E3	B	373		EX (SP),HL ;Remove stored FCB info
001555	E1	B	374		POP HL ;Recover HL
001556	C9	B	375		RET
001557	E3	B	376	WRERROR	EX (SP),HL ;Get FCB Values
001558	DD7405	B	377		LD (IX+5),H ;and put them back
00155B	DD7508	B	378		LD (IX+8),L
00155E	E1	B	379		POP HL ;Restore HL
00155F	C9	B	380		RET ;Go back with error
		B	381		
		B	382		; Entry to Verify after write of a record
		B	383		
001560	CD 68 15	B	384	zVER	CALL CKOPENz
001563	3C	B	385		INC A ;Set verify byte
001564	18 CE	B	386		JR WRIT1
001566	37	B	387	LNKFCBz	SCF ;Init to force file open
001567	D2	B	388		DB 0D2h ;test by JP NC,aaaa
001568	1A	B	389	CKOPENz	LD A,(DE) ;Ignore if from LNKFCB
001569	07	B	390		RLCA ;Test hi bit of FCB
00156A	E3	B	391		EX (SP),HL
00156B	22 26 00	B	392		LD (JRET\$),HL ;Save ret
00156E	ED53 24 00	B	393		LD (JDCB\$),DE ;Save DCB
001572	E3	B	394		EX (SP),HL
001573	30 0F	B	395		JR NC,NOTOPEN ;Go if not an open FCB
001575	F1	B	396		POP AF ;Get return
001576	D5	B	397		PUSH DE ;Dcb addr to IX
001577	DDE3	B	398		EX (SP),IX
001579	E5	B	399		PUSH HL ;Save regs
00157A	D5	B	400		PUSH DE
00157B	C5	B	401		PUSH BC
00157C	E5	B	402		PUSH HL ;Etab ret
00157D	21 89 15	B	403		LD HL,RESTREG ;to restore registers
001580	E3	B	404		EX (SP),HL
001581	F5	B	405		PUSH AF ;Put back ret
001582	AF	B	406		XOR A
001583	C9	B	407		RET ;Go back
		B	408		
001584	F1	B	409	NOTOPEN	POP AF
001585	3E26	B	410		LD A,26h ;File not open
001587	B7	B	411		OR A
001588	C9	B	412		RET
		B	413		
001589	C1	B	414	RESTREG	POP BC ;Pop back registers save
00158A	D1	B	415		POP DE ;in CKOPENz
00158B	E1	B	416		POP HL
00158C	DDE1	B	417		POP IX
00158E	C9	B	418		RET
		B	419		
		B	420		; Entry to Check if at end-of-file.
		B	421		
00158F	CD 68 15	B	422	zCKEOF	CALL CKOPENz

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; File positioning subroutines &gt;

PC	Object	I	Line	Source	.. \SYSRES\filposn.s
001592	CD 0B 14	B	423	CKEOF1	CALL GETNRN ;P/U NRN into HL
001595	E5	B	424		PUSH HL ;Save un-adjusted NRN
001596	CD B7 15	B	425		CALL ADJUST ;Adjust for special cases
001599	7C	B	426		LD A,H ;Compare hi byte
00159A	DDBE0D	B	427		CP (IX+13)
00159D	20 0E	B	428		JR NZ,CKEOF2 ;Go if not equal
00159F	7D	B	429		LD A,L ;Compare lo byte
0015A0	DDBE0C	B	430		CP (IX+12)
0015A3	20 08	B	431		JR NZ,CKEOF2 ;Go if not equal
0015A5	0D	B	432		DEC C ;Adjust for 00=256
0015A6	DD7E08	B	433		LD A,(IX+8) ;Compare offset byte
0015A9	3D	B	434		DEC A
0015AA	91	B	435		SUB C
0015AB	3F	B	436		CCF
0015AC	03	B	437		INC BC ;Restore old C value
		B	438		
0015AD	E1	B	439	CKEOF2	POP HL ;Restore unadjusted NRN
0015AE	3E1D	B	440		LD A,1Dh ;Rec # out of range code
0015B0	20 02	B	441		JR NZ,CKEOF3 ;Go if not at EOF
0015B2	3D	B	442		DEC A ;X"1C'=EOF encountered
0015B3	C9	B	443		RET ;Return with NZ flag
0015B4	D0	B	444	CKEOF3	RET NC ;Return with error
0015B5	AF	B	445		XOR A ;No error
0015B6	C9	B	446		RET
		B	447		
		B	448		; File positioning adjustment routines
		B	449		
	000015B7	B	450	ADJUST	EQU \$ ;Entry from zCKEOF & zLOC
0015B7	DD4E05	B	451		LD C,(IX+5) ;Pick up offset
	000015BA	B	452	ADJ2	EQU \$ ;Entry from zBKSP/zRREAD
0015BA	DDCB017E	B	453		BIT 7,(IX+1) ;Sector I/O only?
0015BE	C8	B	454		RET Z ;No adjustment if so
0015BF	79	B	455		LD A,C ;Offset =0? (or "RREAD?")
0015C0	B7	B	456		OR A
0015C1	28 01	B	457		JR Z,\$+3 ;Go if zero
0015C3	23	B	458		INC HL ;Adjust
0015C4	DDCB016E	B	459		BIT 5,(IX+1) ;Check magic bit
0015C8	C0	B	460		RET NZ ;Go if set
0015C9	2B	B	461		DEC HL ;Adjust
0015CA	C9	B	462		RET
		B	463		;
		B	464		; Calculate the cylinder/sector of needed record
		B	465		;
0015CB	CD 0B 14	B	466	IOREC	CALL GETNRN ;P/u record number
0015CE	CD 26 1A	B	467		CALL ZDCTBYT-5 ;Get # of sectors/gran
0015D1	E61F	B	468		AND 1Fh
0015D3	3C	B	469		INC A
0015D4	CD E3 06	B	470		CALL zDIV16 ;By # of sectors/gran
0015D7	32 60 16	B	471		LD (CALS5+1),A ;Sv rmndr (sector offset)
0015DA	DDE5	B	472		PUSH IX ;Xfer fcb to HL
0015DC	E3	B	473		EX (SP),HL
0015DD	010E00	B	474		LD BC,14 ;Pt to 1st extent info
0015E0	09	B	475		ADD HL,BC
0015E1	C1	B	476		POP BC ;Pop gran ptr HL into BC
0015E2	3E05	B	477		LD A,5 ;Init to ck 4 extents
0015E4	110000	B	478		LD DE,0 ;& extended FXDE ptr
0015E7	F5	B	479	GREC1	PUSH AF
0015E8	7E	B	480		LD A,(HL) ;P/u starting cyl byte
0015E9	23	B	481		INC HL ;& bypass if FF
0015EA	3C	B	482		INC A



\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; File positioning subroutines &gt;

PC	Object	I	Line	Source	.. \SYSRES\filposn.s
0015EB	28 0B	B	483	JR	Z,GREC2
0015ED	E5	B	484	PUSH	HL ;Xfer the # of grans up
0015EE	62	B	485	LD	H,D ;to but not including
0015EF	6B	B	486	LD	L,E ;this extent into HL
0015F0	AF	B	487	XOR	A ;Sub gran pointer from
0015F1	ED42	B	488	SBC	HL,BC ;cumulative figure & go
0015F3	38 0E	B	489	JR	C,GREC3 ;if not in previous ext
0015F5	E1	B	490	POP	HL
0015F6	28 29	B	491	JR	Z,CALCSEC
0015F8	23	B	492	GREC2	INC HL
0015F9	F1	B	493	POP	AF
0015FA	3D	B	494	DEC	A
0015FB	28 19	B	495	JR	Z,GREC4 ;Jump when all quads c"kd
0015FD	5E	B	496	LD	E,(HL) ;P/u cumulative # grans
0015FE	23	B	497	INC	HL ;up to but not
0015FF	56	B	498	LD	D,(HL) ;including this extent
001600	23	B	499	INC	HL
001601	18 E4	B	500	JR	GREC1
001603	24	B	501	GREC3	INC H ;Within 256 grans?
001604	7D	B	502	LD	A,L ;Xfer lo-order difference
001605	E1	B	503	POP	HL ;Rcvr # of contig grans
		B	504		;in this extent
001606	20 F0	B	505	JR	NZ,GREC2 ;Go if not within 256
001608	D5	B	506	PUSH	DE ;Save cumulative count
001609	5F	B	507	LD	E,A ;Xfer gran dif (neg)
00160A	7E	B	508	LD	A,(HL) ;P/u # of grans
00160B	E61F	B	509	AND	1Fh ;in this extent
00160D	83	B	510	ADD	A,E ;Add to negative diff
00160E	7B	B	511	LD	A,E ;Put neg diff into A
00160F	D1	B	512	POP	DE
001610	30 E6	B	513	JR	NC,GREC2 ;Go if not in this extent
001612	ED44	B	514	NEG	;Is in this extent, make
001614	18 0B	B	515	JR	CALCSEC ;diff positive & use it
		B	516		
		B	517		; All current quads checked - Need directory info
		B	518		
001616	CD 64 16	B	519	GREC4	CALL ALLOC ;Get # of grans
001619	C0	B	520	RET	NZ ;into the extent
00161A	32 50 16	B	521	LD	(CALS4+1),A ;or error RET
00161D	30 2A	B	522	JR	NC,CALS3 ;Jp if record in 1st ext
00161F	18 1F	B	523	JR	CALS1 ;else jp if in another
		B	524		
		B	525		; Calc sector in gran
		B	526		
001621	32 50 16	B	527	CALCSEC	LD (CALS4+1),A ;Stuff # grans into
001624	46	B	528	LD	B,(HL) ;this extent
001625	2B	B	529	DEC	HL ;P/u # contig grans &
001626	4E	B	530	LD	C,(HL) ;rel start & start cyl
001627	23	B	531	INC	HL
001628	F1	B	532	POP	AF ;Rcvr # of quad
001629	2F	B	533	CPL	
00162A	C604	B	534	ADD	A,4
00162C	30 19	B	535	JR	NC,CALS2 ;Jump if 1st ext or quad
00162E	3C	B	536	INC	A ;If not 1st, set up to move
00162F	07	B	537	RLCA	;matching quad to the
001630	07	B	538	RLCA	;first position by
001631	C5	B	539	PUSH	BC ;shuffling the others up
001632	D5	B	540	PUSH	DE
001633	4F	B	541	LD	C,A ;Get bytes to move
001634	0600	B	542	LD	B,0

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; File positioning subroutines &gt;

PC	Object	I	Line	Source	.. \SYSRES\filposn.s
001636	EB	B	543	EX	DE,HL ;DE = top of last quad
001637	21FCFF	B	544	LD	HL,-4
00163A	19	B	545	ADD	HL,DE ;HL = top of next lower
00163B	EDB8	B	546	LDDR	;Do the shuffle
00163D	EB	B	547	EX	DE,HL
00163E	D1	B	548	POP	DE
00163F	C1	B	549	POP	BC
001640	70	B	550	CALS1	LD (HL),B ;Move info on matching quad
001641	2B	B	551	DEC	HL ;into position
001642	71	B	552	LD	(HL),C
001643	2B	B	553	DEC	HL
001644	72	B	554	LD	(HL),D
001645	2B	B	555	DEC	HL
001646	73	B	556	LD	(HL),E
001647	60	B	557	CALS2	LD H,B ;Xfer start & contig gran
001648	69	B	558	LD	L,C ;Xfer start cylinder
001649	7C	B	559	CALS3	LD A,H
00164A	07	B	560	RLCA	;P/u start gran on track
00164B	07	B	561	RLCA	
00164C	07	B	562	RLCA	
00164D	E607	B	563	AND	7
00164F	C600	B	564	CALS4	ADD A,0 ;P/u # grans into extent
001651	CD 19 19	B	565	CALL	RELCYL ;Calc 1st relative cyl
001654	85	B	566	ADD	A,L ;Add starting cylinder
001655	57	B	567	LD	D,A
001656	78	B	568	LD	A,B ;Rcvr # sectors/gran
001657	E61F	B	569	AND	1Fh
001659	3C	B	570	INC	A
00165A	D5	B	571	PUSH	DE ;Calculate sector offset
00165B	CD 0A 19	B	572	CALL	zMUL8 ;into desired cylinder
00165E	D1	B	573	POP	DE ;for desired granule
00165F	C600	B	574	CALS5	ADD A,0 ;P/u # of excess sectors
001661	5F	B	575	LD	E,A ;over even gran & add
001662	AF	B	576	XOR	A ;to granule sector
001663	C9	B	577	RET	
		B	578		
		B	579		; On entry, gran needed is in BC
		B	580		
001664	CD AE 16	B	581	ALLOC	CALL CYL_GRN ;Find ext cntng gran
001667	C0	B	582	RET	NZ ;Ret on error
001668	E5	B	583	PUSH	HL ;Save starting cyl & gran
001669	60	B	584	LD	H,B ;Xfer granule needed to
00166A	69	B	585	LD	L,C ;HL then calculate how
00166B	AF	B	586	XOR	A ;many grans into this
00166C	ED52	B	587	SBC	HL,DE ;extent is the desired
00166E	7D	B	588	LD	A,L ;granule
00166F	32 A7 16	B	589	LD	(ALL6+1),A ;Stuff rel gran from
001672	E1	B	590	POP	HL ;start of extent
001673	D5	B	591	PUSH	DE ;Save granule count
001674	DDE5	B	592	PUSH	IX ;to extent
001676	E3	B	593	EX	(SP),HL ;FCB pointer to HL
001677	110E00	B	594	LD	DE,14 ;Pt to 1st alloc in FCB
00167A	19	B	595	ADD	HL,DE
00167B	D1	B	596	POP	DE ;Pop starting cylinder
00167C	0605	B	597	LD	B,5 ;to this extent
00167E	7E	B	598	ALL1	LD A,(HL) ;P/u a cyl
00167F	23	B	599	INC	HL ;Does starting cyl of
001680	BB	B	600	CP	E ;needed gran alloc
001681	20 06	B	601	JR	NZ,ALL2 ;appear in this extent?
001683	7E	B	602	LD	A,(HL) ;Now see if needed gran is

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; File positioning subroutines &gt;

PC	Object	I	Line	Source	.. \SYSRES\filposn.s
001684	AA	B	603	XOR D	;in this extent field
001685	E6E0	B	604	AND 0E0h	;by checking its starting gran
001687	28 19	B	605	JR Z,ALL4	
001689	05	B	606	ALL2 DEC B	;Dec the count down loop
00168A	28 05	B	607	JR Z,ALL3	;Done if no match
00168C	23	B	608	INC HL	;Go to next extent
00168D	23	B	609	INC HL	;info in FCB
00168E	23	B	610	INC HL	
00168F	18 ED	B	611	JR ALL1	
001691	D5	B	612	ALL3 PUSH DE	;Save needed extent info
001692	EB	B	613	EX DE,HL	;Set up to shuffle extent
001693	21FCFF	B	614	LD HL,-4	;info
001696	19	B	615	ADD HL,DE	
001697	010C00	B	616	LD BC,12	
00169A	EDB8	B	617	LDDR	
00169C	EB	B	618	EX DE,HL	
00169D	C1	B	619	POP BC	
00169E	AF	B	620	XOR A	;Set Z, no error
00169F	37	B	621	SCF	;Set CF, extent not found
0016A0	18 03	B	622	JR ALL5	
0016A2	72	B	623	ALL4 LD (HL),D	
0016A3	EB	B	624	EX DE,HL	
0016A4	AF	B	625	XOR A	;Set Z no error
0016A5	D1	B	626	ALL5 POP DE	
0016A6	3E00	B	627	ALL6 LD A,0	;# of grans into this ext
0016A8	C9	B	628	RET	;Where desired gran is
		B	629		
		B	630		; Extent is unused - need to allocate more space
		B	631		
0016A9	CD F2 16	B	632	CG06 CALL CG07	;Try to allocate more
0016AC	C1	B	633	POP BC	;Get back desired gran
0016AD	C0	B	634	RET NZ	;Return on error
		B	635		;Look for gran again
		B	636		
		B	637		; Find extent containing desired granule
		B	638		
0016AE	C5	B	639	CYL_GRN PUSH BC	;Save desired gran #
0016AF	110000	B	640	LD DE,0	;Init gran counter
0016B2	DD4607	B	641	LD B,(IX+7)	;P/u DEC of file
0016B5	78	B	642	CG01 LD A,B	
0016B6	32 AC 17	B	643	LD (STUFDEC+1),A	;Stuff
0016B9	DD4E06	B	644	LD C,(IX+6)	;P/u drive for file
0016BC	CD BB 18	B	645	CALL ZDIRRD	;Read its directory
0016BF	011600	B	646	LD BC,22	;Point to 1st extent
0016C2	09	B	647	ADD HL,BC	;of its directory
0016C3	EB	B	648	EX DE,HL	;Gran count to HL
0016C4	C1	B	649	POP BC	;Restore desired gran
0016C5	C0	B	650	RET NZ	;Return on read error
0016C6	1A	B	651	CG02 LD A,(DE)	;Is this extent
0016C7	FEFE	B	652	CP 0FEh	;allocated?
0016C9	30 1F	B	653	JR NC,CG05	;Jump if it is not
0016CB	13	B	654	INC DE	;Point to allocation
0016CC	1A	B	655	LD A,(DE)	;P/u relative gran & #
0016CD	E5	B	656	PUSH HL	;of contiguous grans
0016CE	E61F	B	657	AND 1Fh	;Keep contiguous grans
0016D0	3C	B	658	INC A	;& bump for 0 offset
0016D1	85	B	659	ADD A,L	;Add to count in HL
0016D2	6F	B	660	LD L,A	
0016D3	30 01	B	661	JR NC,CG03	
0016D5	24	B	662	INC H	;Bump hi order

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; File positioning subroutines &gt;

PC	Object	I	Line	Source	.. \SYSRES\filposn.s
0016D6	E5	B	663	CG03	PUSH HL ;Save gran count to
0016D7	2B	B	664	DEC HL ;end of extent	
0016D8	AF	B	665	XOR A ;Test if EOF is in this	
0016D9	ED42	B	666	SBC HL,BC ;allocation	
0016DB	E1	B	667	POP HL	
0016DC	30 04	B	668	JR NC,CG04 ;EOF not > this alloc	
0016DE	13	B	669	INC DE ;Get rid of old	
0016DF	F1	B	670	POP AF ;current quantity	
0016E0	18 E4	B	671	JR CG02 ;Check next extent	
		B	672		
		B	673	; The EOF is within this allocation. Recover	
		B	674	; the allocation data and exit	
		B	675		
0016E2	E1	B	676	CG04	POP HL ;P/u gran count to extent
0016E3	EB	B	677	EX DE,HL ;Gran count to DE	
0016E4	7E	B	678	LD A,(HL) ;P/u granule data	
0016E5	2B	B	679	DEC HL	
0016E6	6E	B	680	LD L,(HL) ;P/u starting cylinder	
0016E7	67	B	681	LD H,A	
0016E8	AF	B	682	XOR A	
0016E9	C9	B	683	RET	
		B	684		
		B	685	; This extent is 1) unused, or 2) FXDE pointer	
		B	686	; and the needed gran has not been found yet	
		B	687		
0016EA	C5	B	688	CG05	PUSH BC ;Gran count to DE &
0016EB	EB	B	689	EX DE,HL ;DIR ptr to HL	
0016EC	20 BB	B	690	JR NZ,CG06 ;Jump if unused	
0016EE	23	B	691	INC HL ;Point to DEC of FXDE	
0016EF	46	B	692	LD B,(HL) ;P/u the DEC	
0016F0	18 C3	B	693	JR CG01 ;& loop	
		B	694		
		B	695	; See if the drive has enough free space left	
		B	696		
0016F2	C5	B	697	CG07	PUSH BC ;Save needed gran
0016F3	DD4E06	B	698	LD C,(IX+6) ;P/u file"s drive	
0016F6	CD 74 18	B	699	CALL zGATRD ;Get GAT	
0016F9	C1	B	700	POP BC ;Rcvr needed gran	
0016FA	C0	B	701	RET NZ ;Return if GAT error	
0016FB	E5	B	702	PUSH HL	
0016FC	60	B	703	LD H,B ;Xfer the requested	
0016FD	69	B	704	LD L,C ;gran to HL &	
0016FE	AF	B	705	XOR A ;subtract current gran	
0016FF	ED52	B	706	SBC HL,DE ;Count to calculate how	
001701	44	B	707	LD B,H ;many excess grans	
001702	4D	B	708	LD C,L ;are needed	
001703	03	B	709	INC BC	
001704	D1	B	710	POP DE ;Rcvr dir byte ptr	
001705	13	B	711	INC DE ;Pt to next DIR byte	
001706	2623	B	712	LD H,DIRBUF\$>>8 ;Start looking at TRK #1	
001708	3A 6A 00	B	713	LD A,(AFLAG\$) ;P/u Search start CYL	
00170B	6F	B	714	LD L,A ;and put it in L	
00170C	C5	B	715	PUSH BC ;Save excess grans needed	
00170D	7B	B	716	LD A,E ;Is this extent the 1st?	
00170E	E61E	B	717	AND 1Eh ;Jump if so, else we can	
001710	FE16	B	718	CP 16h ;use it for allocation	
001712	28 42	B	719	JR Z,CG14	
001714	1D	B	720	DEC E ;Backup to previous	
001715	1D	B	721	DEC E ;extent	
001716	1A	B	722	CG12	LD A,(DE) ;P/u # of contig grans to

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; File positioning subroutines &gt;

PC	Object	I	Line	Source	.. \SYSRES\filposn.s
001717	E61F	B	723	AND	1Fh ;see if the last gran
001719	3C	B	724	INC	A ;used can be extended
00171A	4F	B	725	LD	C,A ;Is current # the max
00171B	FE20	B	726	CP	20h ;an extent can hold?
00171D	28 20	B	727	JR	Z,CG13 ;Jump if a full extent
00171F	1A	B	728	LD	A,(DE) ;(32 grans max) - else
001720	E6E0	B	729	AND	0E0h ;p/u the relative
001722	07	B	730	RLCA	;granule offset
001723	07	B	731	RLCA	
001724	07	B	732	RLCA	
001725	81	B	733	ADD	A,C ;Add the # of contiguous
001726	D5	B	734	PUSH	DE ;granules
001727	CD 19 19	B	735	CALL	RELCYL ;Calc relative cyl needed
00172A	47	B	736	LD	B,A ;Save offset
00172B	4B	B	737	LD	C,E
00172C	D1	B	738	POP	DE
00172D	1B	B	739	DEC	DE ;Backup to starting cyl
00172E	1A	B	740	LD	A,(DE)
00172F	13	B	741	INC	DE ;& repoint to alloc byte
001730	80	B	742	ADD	A,B ;Add cyls used to
001731	6F	B	743	LD	L,A ;starting cyl
001732	2623	B	744	LD	H,DIRBUF\$>>8 ;Is it less than max?
001734	FECB	B	745	CP	0CBh
001736	30 07	B	746	JR	NC,CG13 ;Jump if too big
001738	79	B	747	LD	A,C
001739	46	B	748	LD	B,(HL) ;P/u the cyl"s GAT
00173A	CD 5B 18	B	749	CALL	TSTBIT ;Test if gran is free
00173D	28 4B	B	750	JR	Z,CG21 ;Bypass if free gran
		B	751		
		B	752		; The next gran cannot be used - get another extent
		B	753		
00173F	1C	B	754	CG13	INC E ;Else point to next
001740	1C	B	755		INC E ;extent field
001741	7B	B	756		LD A,E
001742	E61E	B	757		AND 1Eh ;Jump if not on the FXDE
001744	FE1E	B	758		CP 1Eh ;field, else we have to
001746	20 0E	B	759		JR NZ,CG14 ;obtain an FXDE record
		B	760		
		B	761		; Last extent used up, get new dir rec for FXDE
		B	762		
001748	CD A4 17	B	763		CALL CG23 ;Write curent GAT & HIT
00174B	C1	B	764		POP BC
00174C	C0	B	765		RET NZ ;Ret if GAT/HIT error
00174D	C5	B	766		PUSH BC
00174E	CD AF 17	B	767		CALL NEWHIT ;Get new HIT for FXDE
001751	C1	B	768		POP BC
001752	C0	B	769		RET NZ ;Loop to process
001753	C3 AE 16	B	770		JP CYL_GRN ;new extent
		B	771		
		B	772		; Extent is vacant - use it & get new allocation
		B	773		
001756	CD FE 18	B	774	CG14	CALL MAXCYL ;Get highest # cyl
001759	32 60 17	B	775		LD (CG17+1),A ;Stuff highest cyl
00175C	0602	B	776		LD B,2
00175E	7D	B	777	CG16	LD A,L ;Test last cyl used
00175F	FE00	B	778	CG17	CP 0 ;P/u max cyl
001761	30 07	B	779		JR NC,CG18
001763	7E	B	780		LD A,(HL) ;P/u a GAT byte
001764	3C	B	781		INC A
001765	20 10	B	782		JR NZ,CG19 ;Go if space in this cyl

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; File positioning subroutines &gt;

PC	Object	I	Line	Source	.. \SYSRES\filposn.s
001767	2C	B	783	INC	L ;else bump to next one
001768	18 F4	B	784	JR	CG16 ;& loop
00176A	2E00	B	785	CG18	LD L,0 ;Now start from begin
00176C	10 F0	B	786	DJNZ	CG16 ;of disk & recheck
00176E	C1	B	787	POP	BC
00176F	CD A4 17	B	788	CALL	CG23 ;Write out GAT & HIT
001772	C0	B	789	RET	NZ
001773	3E1B	B	790	LD	A,1Bh ;"disk space full"
001775	B7	B	791	OR	A
001776	C9	B	792	RET	
		B	793		
		B	794		; Found available space in cylinder
		B	795		
001777	3EFF	B	796	CG19	LD A,0FFh ;Set DIR extent to FF
001779	12	B	797		LD (DE),A
00177A	0E00	B	798		LD C,0
00177C	46	B	799		LD B,(HL) ;P/u current GAT alloc
00177D	79	B	800	CG20	LD A,C
00177E	CD 5B 18	B	801	CALL	TSTBIT ;Find a free gran
001781	28 07	B	802	JR	Z,CG21 ;& jump when found
001783	1A	B	803	LD	A,(DE) ;else advance starting
001784	C620	B	804	ADD	A,20h ;rel gran value
001786	12	B	805	LD	(DE),A
001787	0C	B	806	INC	C ;Bump pointer to test
001788	18 F3	B	807	JR	CG20 ;next gran
		B	808		
		B	809		; Next gran in line is free - allocate it
		B	810		
00178A	79	B	811	CG21	LD A,C
00178B	CD 68 18	B	812	CALL	SETBIT ;Show it allocated
00178E	B6	B	813	OR	(HL)
00178F	77	B	814	LD	(HL),A
001790	1D	B	815	DEC	E ;Backup to starting cyl
001791	1A	B	816	LD	A,(DE) ;Bump by one to see if
001792	3C	B	817	INC	A ;this alloc is the 1st
001793	20 02	B	818	JR	NZ,CG22 ;one for the extent &
001795	7D	B	819	LD	A,L ;we have to set the
		B	820		;starting cylinder
001796	12	B	821		LD (DE),A ;Stuff starting cyl
001797	1C	B	822	CG22	INC E
001798	1A	B	823	LD	A,(DE) ;Add 1 to # of contiguous
001799	3C	B	824	INC	A ;granules
00179A	12	B	825	LD	(DE),A
00179B	C1	B	826	POP	BC ;Decrement needed gran
00179C	0B	B	827	DEC	BC ;count since we just
00179D	C5	B	828	PUSH	BC ;allocated one
00179E	78	B	829	LD	A,B ;Loop if we need more
00179F	B1	B	830	OR	C ;space allocated
0017A0	C2 16 17	B	831	JP	NZ,CG12
0017A3	C1	B	832	POP	BC
0017A4	DD4E06	B	833	CG23	LD C,(IX+6) ;Else p/u the drive #
0017A7	CD 75 18	B	834	CALL	ZGATWR ;& write out the GAT
0017AA	C0	B	835	RET	NZ
0017AB	0600	B	836	STUFDEC	LD B,0 ;P/u DEC of FPDE
0017AD	18 54	B	837	JR	ZDIRWR
		B	838		
		B	839		; Get new HIT for FXDE
		B	840		
0017AF	DD4E06	B	841	NEWHIT	LD C,(IX+6) ;P/u drive #
0017B2	CD 97 18	B	842	CALL	ZHITRD ;Read the HIT

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; File positioning subroutines &gt;

PC	Object	I	Line	Source	.. \SYSRES\filposn.s
0017B5	C0	B	843	RET	NZ
0017B6	DD7E07	B	844	LD	A,(IX+7) ;P/u FPDE DEC so 1st ck
0017B9	E61F	B	845	AND	1Fh ;will be for next
0017BB	CD 1F 18	B	846	CALL	zSCNHIT ;in line
0017BE	3E1E	B	847	LD	A,1Eh ;Init "full directory...
0017C0	C0	B	848	RET	NZ ;Ret if no space
0017C1	45	B	849	LD	B,L ;Set DEC for
0017C2	7D	B	850	LD	A,L ;directory read
0017C3	32 02 18	B	851	LD	(NHIT3+1),A ; Stuff new DEC from HIT
0017C6	54	B	852	LD	D,H
0017C7	DD5E07	B	853	LD	E,(IX+7) ;P/u current DEC
0017CA	1A	B	854	LD	A,(DE) ;Copy filespec hash code
0017CB	77	B	855	LD	(HL),A ;to new DEC
0017CC	CD 98 18	B	856	CALL	zHITWR
0017CF	CC BB 18	B	857	CALL	Z,zDIRRD
0017D2	C0	B	858	RET	NZ
0017D3	3690	B	859	LD	(HL),90h ;Show dir rec in use as
0017D5	2C	B	860	INC	L ;FXDE record
0017D6	C5	B	861	PUSH	BC ;P/u DEC of FPDE &
0017D7	3A AC 17	B	862	LD	A,(STUFDEC+1) ;stuff it into FXDE"s
0017DA	77	B	863	LD	(HL),A ;DIR+1 to link back
0017DB	2C	B	864	INC	L
0017DC	0614	B	865	LD	B,20 ;Zero out 20 bytes
0017DE	3600	B	866	NHIT1 LD	(HL),0 ;in the FXDE
0017E0	2C	B	867	INC	L
0017E1	10 FB	B	868	DJNZ	NHIT1
0017E3	E5	B	869	PUSH	HL ;Save ptr to 1st extent
0017E4	060A	B	870	LD	B,10 ;Init to X"FF" 10 bytes
0017E6	36FF	B	871	NHIT2 LD	(HL),0FFh ;or 5 extents
0017E8	2C	B	872	INC	L
0017E9	10 FB	B	873	DJNZ	NHIT2
0017EB	D1	B	874	POP	DE ;Rcvr ptr to 1st extent
0017EC	13	B	875	INC	DE ;Pt to allocation byte
0017ED	C1	B	876	POP	BC
0017EE	CD 03 18	B	877	CALL	zDIRWR ;Write FXDE back to disk
0017F1	C0	B	878	RET	NZ ;Return if error
0017F2	3A AC 17	B	879	LD	A,(STUFDEC+1) ;else p/u DEC of FPDE
0017F5	47	B	880	LD	B,A
0017F6	CD BB 18	B	881	CALL	zDIRRD ;Read its directory
0017F9	C0	B	882	RET	NZ ;& return if error
0017FA	7D	B	883	LD	A,L
0017FB	C61E	B	884	ADD	A,1Eh ;Point to FXDE posn
0017FD	6F	B	885	LD	L,A ;in FPDE
0017FE	36FE	B	886	LD	(HL),0FEh ;Show link to FXDE
001800	2C	B	887	INC	L
001801	3600	B	888	NHIT3 LD	(HL),0 ;Show what"s the FXDE DEC
		B	889		
		B	890		; Routine to write a directory sector
		B	891		; B => DEC of FPDE, C => logical drive number
		B	892		; HL <= points to directory record in SBUFF\$
		B	893		
001803	CD 07 18	B	894	zDIRWR	CALL DIRWR ;Permit two attempts
001806	C8	B	895	RET	Z
001807	D5	B	896	DIRWR	PUSH DE ;Save the reg
001808	CD CA 18	B	897	CALL	CALCDIR ;Calc dir cyl
00180B	2E00	B	898	LD	L,0 ;Set buffer to start
00180D	CD EC 19	B	899	CALL	zWRSSC ;Write the sector
001810	CC DC 19	B	900	CALL	Z,zVRSEC ;Verify on no error
001813	D606	B	901	SUB	6
001815	D1	B	902	POP	DE

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; File positioning subroutines &gt;

PC	Object	I	Line	Source	.. \SYSRES\filposn.s
001816	C8	B	903	RET	Z ;Back on system sector
001817	FE09	B	904	CP	0Fh-6 ;WP error?
001819	3E12	B	905	LD	A,18 ;Set dir write error
00181B	C0	B	906	RET	NZ ;if not WP
00181C	D603	B	907	SUB	3
00181E	C9	B	908	RET	
		B	909		
		B	910		; Find a spare HIT entry
		B	911		
00181F	F5	B	912	zSCNHIT	PUSH AF
001820	3E07	B	913	LD	A,7 ;Get highest # sector
001822	CD 2B 1A	B	914	CALL	zDCTBYT ;on a cylinder
001825	D5	B	915	PUSH	DE ;into register E
001826	57	B	916	LD	D,A
001827	E61F	B	917	AND	1Fh
001829	5F	B	918	LD	E,A
00182A	1C	B	919	INC	E ;& get number of heads
00182B	AA	B	920	XOR	D ;into register A
00182C	07	B	921	RLCA	
00182D	07	B	922	RLCA	
00182E	07	B	923	RLCA	
00182F	3C	B	924	INC	A
001830	CD 0A 19	B	925	CALL	zMUL8 ;To calc sectors/cylinder
001833	CD 3B 19	B	926	CALL	CKDBLBIT ;Double if necessary
001836	D1	B	927	POP	DE ;Total sectors per cyl
001837	D602	B	928	SUB	2 ;Reduce for GAT & HIT
001839	32 4D 18	B	929	LD	(NHIT7+1),A ;# of directory sectors
00183C	F1	B	930	POP	AF ;Get DEC init entry
00183D	6F	B	931	LD	L,A
00183E	CD 49 18	B	932	CALL	NHIT6 ;Ck if HIT slot is spare
001841	C8	B	933	RET	Z ;Return if it is spare
001842	2E01	B	934	LD	L,1 ;Start at beginning
001844	2C	B	935	NHIT5	INC L
001845	20 02	B	936	JR	NZ,NHIT6
001847	B4	B	937	OR	H
001848	C9	B	938	RET	
001849	7D	B	939	NHIT6	LD A,L
00184A	E61F	B	940	AND	1Fh
00184C	FE00	B	941	NHIT7	CP 0
00184E	7D	B	942	LD	A,L
00184F	38 05	B	943	JR	C,NHIT8
001851	F61F	B	944	OR	1Fh
001853	6F	B	945	LD	L,A
001854	18 EE	B	946	JR	NHIT5
001856	7E	B	947	NHIT8	LD A,(HL)
001857	B7	B	948	OR	A
001858	C8	B	949	RET	Z
001859	18 E9	B	950	JR	NHIT5
		B	951		
		B	952		; Test if gran is free in GAT
		B	953		
00185B	E607	B	954	TSTBIT	AND 7 ;Get 0 to 7
00185D	07	B	955	RLCA	;Shift to match BIT n,
00185E	07	B	956	RLCA	;opcode
00185F	07	B	957	RLCA	
001860	F640	B	958	OR	40h
001862	32 66 18	B	959	LD	(TBIT1+1),A ;Modify BIT instruction
001865	CB40	B	960	TBIT1	BIT 0,B
001867	C9	B	961	RET	
		B	962		



\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; File positioning subroutines &gt;

PC	Object	I	Line	Source	.. \SYSRES\filposn.s
		B	963	; Set gran to allocated in GAT	
		B	964		
001868	07	B	965	SETBIT	RLCA ;Shift to create opcode
001869	07	B	966		RLCA ;to match current bit
00186A	07	B	967		RLCA
00186B	F6C7	B	968		OR 0C7h
00186D	32 72 18	B	969		LD (SBIT1+1),A ;Create SET n, opcode
001870	AF	B	970		XOR A
001871	CBC7	B	971	SBIT1	SET 0,A
001873	C9	B	972		RET
		B	973		
		B	974	; Routine reads/writes the Granule Allocation Table	
		B	975		
001874	F6	B	976	zGATRD	DB 0F6h ;Set NZ for test
001875	AF	B	977	zGATWR	XOR A ;Set Z for test
001876	D5	B	978		PUSH DE
001877	E5	B	979		PUSH HL
001878	F5	B	980		PUSH AF ;Save flag for test
001879	CD F7 18	B	981		CALL zDIRCYL
00187C	210023	B	982		LD HL,DIRBUF\$
00187F	5D	B	983		LD E,L ;Set E to 0
001880	F1	B	984		POP AF ;Rcvr flag for R/W
001881	28 07	B	985		JR Z,GATRW1 ;Go if zGATWR
001883	CD D8 18	B	986		CALL zRDSSC
001886	3E14	B	987		LD A,14h ;Init "GAT read error"
001888	18 0A	B	988		JR GATRW2
00188A		B	989	GATRW1	
00188A	CD EC 19	B	990		CALL zWRSSC ;Protected sector write
00188D	CC DC 19	B	991		CALL Z,zVRSEC ;Verify if OK
001890	FE06	B	992		CP 6 ;Protected sector?
001892	3E15	B	993		LD A,15h ;Init "GAT write error"
001894	E1	B	994	GATRW2	POP HL
001895	D1	B	995		POP DE
001896	C9	B	996		RET
		B	997		
		B	998	; Read or write the hash index table	
		B	999		
001897	F6	B	1000	zHITRD	DB 0F6h ;Set NZ for test
001898	AF	B	1001	zHITWR	XOR A ;Set Z for test
001899	C5	B	1002		PUSH BC
00189A	D5	B	1003		PUSH DE
00189B	F5	B	1004		PUSH AF ;Save flag for test
00189C	CD F7 18	B	1005		CALL zDIRCYL ;D => directory cylinder
00189F	1E01	B	1006		LD E,1 ;E => HIT sector
0018A1	21 00 1D	B	1007		LD HL,SBUFF\$ ;HL => HIT buffer area
0018A4	F1	B	1008		POP AF ;Rcvr flag for RD/WR
0018A5	28 07	B	1009		JR Z,HITRW1 ;Go if zHITWR
0018A7	CD D8 18	B	1010		CALL zRDSSC ;Read cyl D, sector E
0018AA	3E16	B	1011		LD A,22 ;Init "HIT read error"
0018AC	18 0A	B	1012		JR HITRW2
0018AE	CD EC 19	B	1013	HITRW1	CALL zWRSSC ;Protected sector write
0018B1	CC DC 19	B	1014		CALL Z,zVRSEC ;Verify the write
0018B4	FE06	B	1015		CP 6 ;Protected sector?
0018B6	3E17	B	1016		LD A,23 ;"HIT write error"
0018B8	D1	B	1017	HITRW2	POP DE ;Message for other than
0018B9	C1	B	1018		POP BC ;attempt protected sector
0018BA	C9	B	1019		RET
		B	1020		
		B	1021	; Routine to read a directory sector	
		B	1022	; B => DEC of FPDE, C => logical drive number	

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; File positioning subroutines &gt;

PC	Object	I	Line	Source	.. \SYSRES \filposn.s
		B	1023	; HL <= points to directory record in SBUFF\$	
		B	1024		
0018BB	D5	B	1025	ZDIRRD	PUSH DE
0018BC	CD CA 18	B	1026	CALL	CALCDIR ;Set HL to SBUFF\$
0018BF	E5	B	1027	PUSH	HL
0018C0	2E00	B	1028	LD	L,0 ;Start of bfr
0018C2	CD D8 18	B	1029	CALL	ZRDSSC ;Read it
0018C5	E1	B	1030	POP	HL
0018C6	3E11	B	1031	LD	A,17 ;Init to dir read err
0018C8	D1	B	1032	POP	DE
0018C9	C9	B	1033	RET	
		B	1034		
		B	1035	; Routine to get directory access data	
		B	1036	; B => DEC	
		B	1037	; DE <= cylinder and sector needed	
		B	1038	; HL <= pointer to directory record in SBUFF\$	
		B	1039		
0018CA	CD F7 18	B	1040	CALCDIR	CALL ZDIRCYL ;Get directory cyl in D
0018CD	78	B	1041	LD	A,B ;Calculate record star
0018CE	E6E0	B	1042	AND	0E0h ;from the DEC
0018D0	6F	B	1043	LD	L,A
0018D1	26 1D	B	1044	LD	H,SBUFF\$>>8 ;Point to buffer start
0018D3	A8	B	1045	XOR	B ;Calculate directory
0018D4	C602	B	1046	ADD	A,2 ;sector needed
0018D6	5F	B	1047	LD	E,A
0018D7	C9	B	1048	RET	
		B	1049		
0018D8		B	1050	ZRDSSC	; Read system sector, D=Track, E=Sector, HL=Buffer.
		B	1051		
0018D8	CD F1 18	B	1052	CALL	READIR
0018DB	C8	B	1053	RET	Z
0018DC	D5	B	1054	PUSH	DE
0018DD	110100	B	1055	LD	DE,1 ;Pt to tk 0, sec 1
0018E0	CD F4 19	B	1056	CALL	ZRDSEC ;Read to find dir cyl
0018E3	D1	B	1057	POP	DE
0018E4	C0	B	1058	RET	NZ
0018E5	E5	B	1059	PUSH	HL
0018E6	23	B	1060	INC	HL ;Pt to dir tk #
0018E7	23	B	1061	INC	HL
0018E8	56	B	1062	LD	D,(HL) ;P/u dir tk fm boot
0018E9	2609	B	1063	LD	H,9 ;Update memory table
0018EB	CD 34 1A	B	1064	CALL	DCTFLDz
0018EE	6F	B	1065	LD	L,A
0018EF	72	B	1066	LD	(HL),D
0018F0	E1	B	1067	POP	HL
0018F1		B	1068	READIR	
0018F1	CD F4 19	B	1069	CALL	ZRDSEC ;Retry dir read
0018F4	D606	B	1070	SUB	6 ;Test protected
0018F6	C9	B	1071	RET	
		B	1072	;	
0018F7		B	1073	ZDIRCYL	
0018F7	3E09	B	1074	LD	A,9
0018F9	CD 2B 1A	B	1075	CALL	ZDCTBYT ;Get the dir cylinder
0018FC	57	B	1076	LD	D,A
0018FD	C9	B	1077	RET	
		B	1078		
0018FE		B	1079	MAXCYL	
0018FE	3E06	B	1080	LD	A,6
001900	C5	B	1081	PUSH	BC
001901	DD4E06	B	1082	LD	C,(IX+6)

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; File positioning subroutines &gt;

PC	Object	I	Line	Source	.. \SYSRES\filposn.s
001904	CD 2B 1A	B	1083	CALL	zDCTBYT ;Get highest # cyl
001907	3C	B	1084	INC	A ;Adjust for zero offset
001908	C1	B	1085	POP	BC
001909	C9	B	1086	RET	
		B	1087		
		B	1088		; Multiply register E by register A
		B	1089		
00190A	C5	B	1090	zMUL8	PUSH BC ; Mult A x E
00190B	57	B	1091	LD	D,A
00190C	AF	B	1092	XOR	A
00190D	0608	B	1093	LD	B,8
00190F	87	B	1094	MEA1	ADD A,A
001910	CB23	B	1095	SLA	E
001912	30 01	B	1096	JR	NC,MEA2
001914	82	B	1097	ADD	A,D
001915	10 F8	B	1098	MEA2	DJNZ MEA1
001917	C1	B	1099	POP	BC
001918	C9	B	1100	RET	
		B	1101		; Calculate relative cylinder for granule needed
		B	1102		
		B	1103		
001919	5F	B	1104	RELCYL	LD E,A
00191A	CD 26 1A	B	1105	CALL	zDCTBYT-5 ;Get # of grans/track
00191D	47	B	1106	LD	B,A ;Hang on to this
00191E	07	B	1107	RLCA	
00191F	07	B	1108	RLCA	
001920	07	B	1109	RLCA	
001921	E607	B	1110	AND	7
001923	3C	B	1111	INC	A ;Adj for 0 offset
001924	CD 3B 19	B	1112	CALL	CKDBLBIT
		B	1113		
		B	1114		; Divide register E by register A
		B	1115		
001927	C5	B	1116	zDIV8	PUSH BC
001928	4F	B	1117	LD	C,A
001929	0608	B	1118	LD	B,8
00192B	AF	B	1119	XOR	A
00192C	CB23	B	1120	DEA1	SLA E
00192E	17	B	1121	RLA	
00192F	B9	B	1122	CP	C
001930	38 02	B	1123	JR	C,DEA2
001932	91	B	1124	SUB	C
001933	1C	B	1125	INC	E
001934	10 F6	B	1126	DEA2	DJNZ DEA1
001936	4F	B	1127	LD	C,A
001937	7B	B	1128	LD	A,E
001938	59	B	1129	LD	E,C
001939	C1	B	1130	POP	BC
00193A	C9	B	1131	RET	
		B	1132		
		B	1133		; Routine to double the A register if DBL bit is set
		B	1134		
00193B	57	B	1135	CKDBLBIT	LD D,A ;Adjust for 2-sided &
00193C	3E04	B	1136	LD	A,4 ;calculate # of cyls
00193E	CD 2B 1A	B	1137	CALL	zDCTBYT
001941	CB6F	B	1138	BIT	5,A ;Test if 2-sided
001943	7A	B	1139	LD	A,D
001944	28 01	B	1140	JR	Z,\$+3 ;Double the grans if 2
001946	87	B	1141	ADD	A,A ;& fall thru to DIV8
001947	C9	B	1142	RET	

\*\*\*\*\* TRSDOS \*\*\*\*\*  
< File positioning subroutines >

PC	Object	I	Line	Source	..\SYSRES\SVC\SVC100-memory.s
		B	0	INCLUDE	"SVC100-memory.s"
		B	1	NEWPAGE	

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; File positioning subroutines &gt;

PC	Object	I	Line	Source	.. \SYSRES \SVC \SVC100-memory.s
		B	2	SUBTITLE	"< System memory manager and associated routines >"
		B	3	scope	
		B	4	; Update to LSDOS 6.3.1 2022-09-16 dpm	
		B	5		
		B	6	; Memory routines to set or retrieve HIGH\$/LOW\$	
		B	7		
		B	8	ORG	zspace.MEMORY\$
		B	9		
001948	7C	B	10	zHIGH	LD A,H ; Test if put or get.
001949	B5	B	11	OR	L
00194A	28 12	B	12	JR	Z,GETHILO ; Go if get.
00194C	3A 6C 00	B	13	LD	A,(CFLAG\$) ; Is HIGH\$ changeable?
00194F	0F	B	14	RRCA	
001950	3E2B	B	15	LD	A,43 ; Init SVC parm error.
001952	D8	B	16	RET	C ; Back with NZ.
001953	04	B	17	INC	B ; Test for HIGH\$/LOW\$.
001954	05	B	18	DEC	B
001955	20 0E	B	19	JR	NZ,PUTLO ; Go if LOW\$.
001957	22 0E 04	B	20	LD	(zHIGH\$),HL ; Set new HIGH\$.
00195A	2A 0E 04	B	21	GETHI	LD HL,(zHIGH\$) ; P/u the value &
00195D	C9	B	22	RET	; ret with Z-flag.
00195E	04	B	23	GETHILO	INC B ; Test for HIGH\$/LOW\$.
00195F	05	B	24	DEC	B
001960	28 F8	B	25	JR	Z,GETHI
001962	2A 1E 00	B	26	LD	HL,(zLOW\$) ; P/u LOW\$.
001965	22 1E 00	B	27	PUTLO	LD (zLOW\$),HL ; Get LOW\$.
001968	AF	B	28	XOR	A ; Set Z-flag.
001969	C9	B	29	RET	
		B	0	INCLUDE	"svc-handler.s"
		B	1	SUBTITLE	"< SVC handler routines. >"
		B	2	NEWPAGE	

\*\*\*\*\* TRSDOS \*\*\*\*\*  
< SVC handler routines. >

PC	Object	I	Line	Source	.. \SYSRES \SVC \svc-handler.s
		B	3	SCOPE	
		B	4	; Update to LSDOS 6.3.1 2022-09-16 dpm	
		B	5		
		B	6	ORG	zspace.SVCHANDLER\$
		B	7		
00196A	FD21 6A 00	B	8	zFLAGS	LD IY, FLGTAB\$
00196E	C9	B	9	RET	
		B	10		
00196F	E5	B	11	zBREAK	PUSH HL ;Save user vector
001970	2A 88 1C	B	12	LD	HL, (BRKVEC\$) ;P/u current vector
001973	E3	B	13	EX	(SP), HL ;Save current & get user
001974	22 88 1C	B	14	LD	(BRKVEC\$), HL ;Stuff new vector
001977	E1	B	15	POP	HL ;Recover old vector
001978	C9	B	16	RET	
		B	17		
001979	E1	B	18	zWHERE	POP HL
00197A	E9	B	19	JP	(HL)
		B	20		
		B	21	; Code for these SVCs is in system overlays	
		B	22		
00197B	3EA3	B	23	zCMNDR	LD A, 0A3h ;Interpret command & RET
00197D	EF	B	24	RST	40
00197E	3EB3	B	25	zCMNDI	LD A, 0B3h ;Interpret a command
001980	EF	B	26	RST	40
001981	3EC3	B	27	zFSPEC	LD A, 0C3h ;Parse a filespec
001983	EF	B	28	RST	40
001984	3ED3	B	29	zFEXT	LD A, 0D3h ;Optional default EXT
001986	EF	B	30	RST	40
001987	3EE3	B	31	zPARAM	LD A, 0E3h ;Parameter scanner
001989	EF	B	32	RST	40
00198A	3E94	B	33	zOPEN	LD A, 94h ;Open a file
00198C	EF	B	34	RST	40
00198D	3EA4	B	35	zINIT	LD A, 0A4h ;Initialize a file
00198F	EF	B	36	RST	40
001990	3EB4	B	37	zGTDCB	LD A, 0B4h ;Get a DCB vector
001992	EF	B	38	RST	40
001993	3EC4	B	39	zCKDRV	LD A, 0C4h ;Drive available?
001995	EF	B	40	RST	40
001996	3EF4	B	41	zRENAME	LD A, 0F4h ;Rename a file
001998	EF	B	42	RST	40
001999	3E95	B	43	zCLOSE	LD A, 95h ;Close a file
00199B	EF	B	44	RST	40
00199C	3EA5	B	45	zFNAME	LD A, 0A5h ;Recover filespec
00199E	EF	B	46	RST	40
00199F	C9	B	47	zDBGHK	RET ;Init DEBUG off (NOP=on)
0019A0	F5	B	48	zDEBUG	PUSH AF
0019A1	3E97	B	49	LD	A, 97h ;Enter system Debugger
0019A3	EF	B	50	RST	40
0019A4	0315	B	51	EXTDBG\$	DW 0RARETz ;Hook for extended DEBUG.
0019A6	3E9C	B	52	zREMOVE	LD A, 9Ch ;Remove a file/device
0019A8	EF	B	53	RST	40
0019A9	3ECD	B	54	zDOKEY	LD A, 0CDh ;D0 execution
0019AB	EF	B	55	RST	40
0019AC	3E9E	B	56	zRAMDIR	LD A, 09Eh ;Directory data
0019AE	EF	B	57	RST	40
0019AF	3EAE	B	58	zDODIR	LD A, 0AEh ;Directory data
0019B1	EF	B	59	RST	40
0019B2	3EBE	B	60	zGTMOD	LD A, 0BEh ;Get module address
0019B4	EF	B	61	RST	40
		B	62		

\*\*\*\*\* TRSDOS \*\*\*\*\*  
 < SVC handler routines. >

PC	Object	I	Line	Source	.. \SYSRES \SVC \svc-handler.s
		B	63		; These SVCs handle the disk primitive requests.
		B	64		
0019B5	AF	B	65	zDCSTAT	XOR A ; FDC status
0019B6	18 3E	B	66	JR	IOFUNC
0019B8	3A 23 00	B	67	TAPDRV	LD A, (LDRV\$) ; P/u drive #
0019BB	4F	B	68	LD	C, A
0019BC	3E01	B	69	zSLCT	LD A, 1 ; Select drive
0019BE	18 36	B	70	JR	IOFUNC
0019C0	3E02	B	71	zDCINIT	LD A, 2 ; FDC init
0019C2	18 32	B	72	JR	IOFUNC
0019C4	3E03	B	73	zDCRES	LD A, 3 ; FDC reset
0019C6	18 2E	B	74	JR	IOFUNC
0019C8	3E04	B	75	zRSTOR	LD A, 4 ; Restore to cyl 0
0019CA	18 2A	B	76	JR	IOFUNC
0019CC	3E05	B	77	zSTEPI	LD A, 5 ; Step in 1 cyl
0019CE	18 26	B	78	JR	IOFUNC
0019D0	3E06	B	79	zSEEK	LD A, 6 ; Seek a track/sector
0019D2	18 22	B	80	JR	IOFUNC
0019D4	3E07	B	81	zRSLCT	LD A, 7 ; Re-select drive
0019D6	18 1E	B	82	JR	IOFUNC
0019D8	3E08	B	83	zRDHDR	LD A, 8
0019DA	18 1A	B	84	JR	IOFUNC
0019DC	3E0A	B	85	zVRSEC	LD A, 10 ; Verify a sector
0019DE	18 16	B	86	JR	IOFUNC
0019E0	3E0B	B	87	zRDTRK	LD A, 11 ; Read a track.
0019E2	18 12	B	88	JR	IOFUNC
0019E4	3E0C	B	89	zHDFMT	LD A, 12 ; Format drive.
0019E6	18 0E	B	90	JR	IOFUNC
0019E8	3E0D	B	91	zWRSEC	LD A, 13 ; Write standard sector
0019EA	18 0A	B	92	JR	IOFUNC
0019EC	3E0E	B	93	zWRSSC	LD A, 14 ; Write a system sector
0019EE	18 06	B	94	JR	IOFUNC
0019F0	3E0F	B	95	zWRTRK	LD A, 15 ; Write a track
0019F2	18 02	B	96	JR	IOFUNC
0019F4	3E09	B	97	zRDSEC	LD A, 9 ; Read a sector.
		B	0		INCLUDE "iofunction.s"
		B	1		SUBTITLE "< IOFUN routines. >"
		B	2		NEWPAGE

\*\*\*\*\* TRSDOS \*\*\*\*\*  
< IOFUN routines. >

PC	Object	I	Line	Source	..\\SYSRES\\iofunction.s
		B	3		scope
		B	4		
		B	5	ORG	zspace.IOFUNCTION\$
		B	6		
0019F6	C5	B	7	IOFUNC	PUSH BC ;Save reg pair
0019F7	47	B	8	LD	B,A ;Xfer the function code
		B	9		
		B	10		; Bring up bank 0
		B	11		
0019F8	C5	B	12	PUSH	BC
0019F9	AF	B	13	xor	a
0019FA	47	B	14	LD	B,A ;Set bank function 0,
0019FB	4F	B	15	LD	C,A ;bank number 0
0019FC	CD 77 08	B	16	CALL	zBANK ;Bring up bank
0019FF	F1	B	17	POP	AF ;Perform EX (SP),BC
001A00	C5	B	18	PUSH	BC
001A01	F5	B	19	PUSH	AF
001A02	C1	B	20	POP	BC
		B	21		
		B	22		; Continue disk I/O setup
		B	23		
001A03	79	B	24	LD	A,C ;Xfer the drive code
001A04	32 23 00	B	25	LD	(LDRV\$),A
001A07	FDE5	B	26	PUSH	IY
001A09	CD 1E 1A	B	27	CALL	zGTDCT ;Get DCT address in IY
001A0C	3E20	B	28	LD	A,20h ;Set illegal drive #
001A0E	B7	B	29	OR	A ;if drive disabled
001A0F	CD 1C 1A	B	30	CALL	GOD0IO
001A12	FDE1	B	31	POP	IY
		B	32		
		B	33		; Bring back the old bank
		B	34		
001A14	C1	B	35	POP	BC
001A15	F5	B	36	PUSH	AF ;Save disk I/O retcod
001A16	3E66	B	37	LD	A,102 ;Set for zBANK
001A18	EF	B	38	RST	40 ;No need to ck for error
		B	39		;from zBANK
001A19	F1	B	40	POP	AF
001A1A	C1	B	41	POP	BC
001A1B	C9	B	42	RET	
		B	43		
001A1C	FDE9	B	44	GOD0IO	JP (IY)
		B	45		
001A1E	E5	B	46	zGTDCT	PUSH HL ;Get i/o routine addr
001A1F	CD 34 1A	B	47	CALL	DCTFLDz ;into IY
001A22	E3	B	48	EX	(SP),HL
001A23	FDE1	B	49	POP	IY
001A25	C9	B	50	RET	
		B	51		
		B	52		; Entry to get DCT+8 of FCB (IX) drive spec
		B	53		
		B	54		
001A26	DD4E06	B	55	DzFBYT8	LD C,(IX+6) ;P/u drive
		B	56		
		B	57		; Entry to get DCT+8 of Reg C drive spec
		B	58		
001A29	3E08	B	59	DCTBYT8z	LD A,8
		B	60		
		B	61		; Entry to get byte (Reg A) from DCT of Reg C drive
		B	62		; C => logical drive specification



\*\*\*\*\* TRSDOS \*\*\*\*\*  
 < IOFUN routines. >

PC	Object	I	Line	Source	.. \SYSRES\iofunction.s
		B	63	; A =>	relative byte requested from DCT
		B	64	; A <=	data at position requested
		B	65		
001A2B	E5	B	66	zDCTBYT	PUSH HL ;Save the register pair
001A2C	67	B	67		LD H,A
001A2D	CD 34 1A	B	68	CALL	DCTFLDz ; Get HL pointing to
001A30	6F	B	69	LD	L,A ; DCT position.
001A31	7E	B	70	LD	A,(HL) ;Get the byte
001A32	E1	B	71	POP	HL
001A33	C9	B	72		RET
		B	73		
		B	74	; Entry to get HL pointing to DCT byte Reg C, Reg A	
		B	75	; C =>	logical drive number
		B	76	; A =>	relative byte in DCT requested
		B	77	; HL <=	start of requested DCT for the drive
		B	78	; A <=	low order pointer to relative byte request
		B	79		
001A34	79	B	80	DCTFLDz	ld a,c ; Drive # to A.
001A35	E607	B	81	AND	7 ; strip any excess data.
001A37	87	B	82	ADD	A,A ;Times 2
001A38	6F	B	83	LD	L,A ; & saved
001A39	87	B	84	ADD	A,A ;Times 4
001A3A	87	B	85	ADD	A,A ;Times 8
001A3B	85	B	86	ADD	A,L ;Times 10
001A3C	C670	B	87	ADD	A,70h ;Add DCT offset from 0
001A3E	6F	B	88	LD	L,A ;Point L to DCT low order
001A3F	84	B	89	ADD	A,H ;Add in rel pos desired
001A40	26 04	B	90	LD	H,DCT\$>>8 ;Point H to DCT hi-order
001A42	C9	B	91		RET
		B	0	INCLUDE	"rst28.s"
		B	1	SUBTITLE	"< RST28 routine, process supervisory calls 0-127 >"
		B	2		NEWPAGE

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; RST28 routine, process supervisory calls 0-127 &gt;

PC	Object	I	Line	Source	.. \SYSRES\restarts\rst28.s
		B	3		SCOPE
		B	4		
		B	5	ORG	zspace.RST28\$
		B	6		
001A43	FE1A	B	7	SVCUSER	CP 26 ;Check for zERROR
001A45	28 08	B	8	JR	Z,ERRSVC ;Skip next if so
001A47	32 0D 00	B	9	LD	(LSVC\$),A ;Store SVC request
001A4A	E3	B	10	EX	(SP),HL ;P/u RET address
001A4B	22 0B 00	B	11	LD	(SVCRET\$),HL ;and save it
001A4E	E3	B	12	EX	(SP),HL ;Restore RET address
001A4F	E5	B	13	ERRSVC	PUSH HL ;Save HL
001A50	07	B	14	RLCA	;Multiply by two
001A51	26 01	B	15	LD	H,SVCTAB\$>>8 ;Base of table
001A53	6F	B	16	LD	L,A ;Set up the low order
001A54	7E	B	17	LD	A,(HL) ;P/u table entry
001A55	2C	B	18	INC	L
001A56	66	B	19	LD	H,(HL)
001A57	6F	B	20	LD	L,A
001A58	E3	B	21	EX	(SP),HL ;P/u HL & stuff vector
001A59	79	B	22	LD	A,C ;Xfer for PUT type ops
001A5A	C9	B	23	RET	
		B	24		
		B	25	; RST 28 vector - System & user SVCs	
		B	26		
001A5B	B7	B	27	RST28	OR A ;Test if bit 7 set
001A5C	F2 43 1A	B	28	JP	P,SVCUSER ;Jump on user SVC attempt
001A5F	E3	B	29	EX	(SP),HL ;Discard return addr &
001A60	F5	B	30	PUSH	AF ;save HL, AF
001A61	21 9F 19	B	31	LD	HL,zDBGHK ;Set up DEBUG linkage
001A64	7E	B	32	LD	A,(HL)
001A65	32 79 1A	B	33	LD	(SETzEXEC),A
001A68	36C9	B	34	LD	(HL),0C9h
001A6A	F1	B	35	POP	AF ;Restore AF, HL
001A6B	E1	B	36	POP	HL
001A6C	CD 7F 1A	B	37	HKRES\$	CALL CKMODz ;Get overlay if needed
001A6F	3E00	B	38	LD	A,0 ;P/u new overlay #
	00001A70	B	39	OVRLYOLD	EQU \$-1
001A71	32 69 00	B	40	LD	(OVRLY\$),A ;& update current
001A74	CD0000	B	41	TRANSFR	CALL 0 ;Traadr of SYSx
001A77	F5	B	42	PUSH	AF
001A78	3E00	B	43	LD	A,0 ;Set to C9 if EXEC only
	00001A79	B	44	SETzEXEC	EQU \$-1
001A7A	32 9F 19	B	45	LD	(zDBGHK),A
001A7D	F1	B	46	POP	AF
001A7E	C9	B	47	RET	
		B	48		
		B	49	; DOS command overlay request	
		B	50		
001A7F	E5	B	51	CKMODz	PUSH HL
001A80	67	B	52	LD	H,A ; Save command value
001A81	78	B	53	LD	A,B
001A82	32 D2 1A	B	54	LD	(EX0VR2+1),A ; Set overlay #
001A85	7C	B	55	LD	A,H
001A86	F601	B	56	OR	1 ; Set for SYS6 & SYS7
001A88	FE89	B	57	CP	89h ; Is it either?
001A8A	7C	B	58	LD	A,H ; Get back the correct #
001A8B	28 13	B	59	JR	Z,EX0VR ; Sys6/7 req? Use ISAM!
001A8D	FE8A	B	60	CP	8Ah ; Sys8 also ISAM
001A8F	28 0F	B	61	JR	Z,EX0VR
001A91	3A 69 00	B	62	LD	A,(OVRLY\$) ; P/u current overlay

***** TRSDOS *****				
< RST28 routine, process supervisory calls 0-127 >				
PC	Object	I	Line	Source ..\SYSRES\restarts\rst28.s
001A94	AC	B	63	XOR H ; Ck if it"s the one
001A95	E60F	B	64	AND 0Fh ; we need to execute
001A97	7C	B	65	LD A,H
001A98	32 70 1A	B	66	LD (OVRLY0LD),A ; Update current tempy
001A9B	21001E	B	67	LD HL,1E00h ; Init to SYSx entry
001A9E	28 3A	B	68	JR Z,EXOVR3 ; Go exec if resident
		B	0	INCLUDE "executeoverlay.s" ; Execute overlay.
		B	1	SUBTITLE "< Execute overlay requested routine. >"
		B	2	NEWPAGE

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; Execute overlay requested routine. &gt;

PC	Object	I	Line	Source	.. \SYSRES\executeoverlay.s
		B	3		scope
		B	4		
		B	5		; Execute a system overlay.
		B	6		
		B	7	ORG	zspace.EX0VR\$
		B	8		
001AA0	D5	B	9	EX0VR	PUSH DE
001AA1	C5	B	10		PUSH BC
001AA2	E60F	B	11		AND 0Fh ; Get right nybble.
001AA4	CB5F	B	12		BIT 3,A ; Check for SYS0-7.
001AA6	28 02	B	13		JR Z,EX0VR1 ; w/o changing carry.
001AA8	C618	B	14		ADD A,18h ; Adjust for sys8-15.
001AAA	32 93 00	B	15	EX0VR1	LD (SFCB\$+7),A
001AAD	47	B	16		LD B,A ; Set DEC for directory.
001AAE	3E20	B	17		LD A,20h ; Set bit 5 of FCB+1.
001AB0	32 8D 00	B	18		LD (SFCB\$+1),A
001AB3	ED62	B	19		SBC HL,HL ; Carry is clear here.
001AB5	22 96 00	B	20		LD (SFCB\$+10),HL ; Zero NRN.
001AB8	4C	B	21		LD C,H ; Init for drive 0.
001AB9	CD BB 18	B	22		CALL ZDIRRD ; Read dir entry.
001ABC	20 1A	B	23		JR NZ,EXERR ; Go if error.
001ABE	7E	B	24		LD A,(HL) ; Was overlay purged?
001ABF	E650	B	25		AND 50h ; or is it non-system?
001AC1	EE50	B	26		XOR 50h
001AC3	3E07	B	27		LD A,7 ; Init "deleted error".
001AC5	20 11	B	28		JR NZ,EXERR
001AC7	7D	B	29		LD A,L
001AC8	C616	B	30		ADD A,22 ; Point to 1st extent.
001ACA	6F	B	31		LD L,A
001ACB	11 9A 00	B	32		LD DE,SFCB\$+14 ; Extent field in FCB.
001ACE	CD E1 1A	B	33		CALL PAT1 ; Stuff 1st two extents.
001AD1	0600	B	34	EX0VR2	LD B,0 ; P/u ISAM # or zero.
001AD3	1E 8C	B	35		LD E,SFCB\$&0FFh
001AD5	CD 56 1B	B	36		CALL LOADER ; Read system overlay.
001AD8	C1	B	37	EXERR	POP BC
001AD9	D1	B	38		POP DE
001ADA	22 75 1A	B	39	EX0VR3	LD (TRANSFR+1),HL ; Stuff overlay entry point.
001ADD	E1	B	40		POP HL
001ADE	C8	B	41		RET Z
001ADF	18 16	B	42		JR SYSERR ; Go on read I/O error
		B	43		
		B	44		; Routine to calculate 1st two extents of SYS file.
		B	45		
001AE1	CD EC 1A	B	46	PAT1	CALL PAT1A ; Move first extent.
001AE4	E61F	B	47		AND 1Fh ; Compute # of granules.
001AE6	3C	B	48		INC A
001AE7	12	B	49		LD (DE),A ; And store in FCB.
001AE8	13	B	50		INC DE
001AE9	AF	B	51		XOR A
001AEA	12	B	52		LD (DE),A
001AEB	13	B	53		INC DE
001AEC	CD EF 1A	B	54	PAT1A	CALL PAT1B ; Move second extent.
001AEF	7E	B	55	PAT1B	LD A,(HL)
001AF0	12	B	56		LD (DE),A
001AF1	23	B	57		INC HL
001AF2	13	B	58		INC DE
001AF3	C9	B	59		RET
		B	0		INCLUDE "systemerror.s"
		B	1		SUBTITLE "< System Error display routine. >"
		B	2		NEWPAGE

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; System Error display routine. &gt;

PC	Object	I	Line	Source	.. \SYSRES\systemerror.s
		B	3		scope
		B	4		
		B	5	ORG	zspace.SYSERROR\$
		B	6		
		B	7		; System error display routine
		B	8		; The NOP is provided so an intercept routine vector
		B	9		; may be patched in during program development
		B	10		
001AF4	3E2B	B	11	SVCERR	LD A,43 ;SVC error
001AF6	00	B	12		NOP
001AF7	E63F	B	13	SYSERR	AND 3Fh ;Strip excess bits
001AF9	21 19 1B	B	14		LD HL,ERRNUM ;Pack error number
001AFC	CD C2 07	B	15		CALL ZHEX8 ;into message
001AFF	21 13 1B	B	16		LD HL,SYSERR\$
001B02	CD 00 05	B	17		CALL ZLOGOT ;Log the error & ABORT
001B05	31 80 03	B	18		LD SP,STACK\$ ;reset stack
001B08	21FFFF	B	19	ZABORT	LD HL,-1
001B0B	3E93	B	20	ZEXIT	LD A,93h ;Exit to DOS
001B0D	EF	B	21		RST 40
001B0E	E1	B	22	POPERR	POP HL ;Pop extended error
001B0F	F5	B	23	ZERROR	PUSH AF ;Save the error code
001B10	3E96	B	24		LD A,96h ;Display the error number
001B12	EF	B	25		RST 40
001B13	4572726F 7220	B	26	SYSERR\$	DB "Error "
001B19	7878480D	B	27	ERRNUM	DB "xxH",CR.asc
		B	28		
		B	0		include "svc77-run.s"
		B	1	SUBTITLE	"< RUN / LOAD SVC routines. >"
		B	2		NEWPAGE

\*\*\*\*\* TRSDOS \*\*\*\*\*  
 < RUN / LOAD SVC routines. >

PC	Object	I	Line	Source	.. \SYSRES \SVC \svc77-run.s
		B	3		scope
		B	4		
		B	5		; Update to LSDOS 6.3.1 2022-09-16 dpm
		B	6		
		B	7		; Routine to RUN a program.
		B	8		
		B	9		org 1B1DH
		B	10		
001B1D	E5	B	11	zRUN	PUSH HL ; Save register pair
001B1E	21 7C 00	B	12		LD HL,SFLAG\$
001B21	CBD6	B	13		SET 2,(HL) ; Turn on RUN flag bit
001B23	CD 38 1B	B	14		CALL zLOAD ; Load the program module
001B26	E3	B	15		EX (SP),HL ; Put traadr on the stack
		B	16		
		B	17		; Error code is set to NOT abort.
		B	18		; Errors will be passed back to calling module after zERROR.
		B	19		; HL will contain the error #.
		B	20		
001B27	20 E5	B	21		Jr NZ,POPERR
001B29	01 20 04	B	22		LD BC,INBUF\$ ; Place INBUF\$ pointer in BC, reflect buffer pointer.
		B	23		
		B	24		; Get TRAADR then test if we need to go to DEBUG.
		B	25		
001B2C	3A 7C 00	B	26		LD A,(SFLAG\$)
001B2F	CB4F	B	27		BIT 1,A ; Go to the program if
001B31	C0	B	28		RET NZ ; its EXEC only access
001B32	CB7F	B	29		BIT 7,A ; else test if DEBUG
001B34	C2 30 00	B	30		JP NZ,zRST30 ; is on & go to it
001B37	C9	B	31		RET ; else go to program
		B	32		
		B	0		include "svc76-load.s"
		B	1		
		B	2		; Update to LSDOS 6.3.1 2022-09-16 dpm
		B	3		
		B	4		; This routine LOADs a Load Module Format file.
		B	5		
		B	6		ORG zspace.LOAD\$
		B	7		
001B38	0600	B	8	zLOAD	LD B,0 ;LRL=256
001B3A	21 7C 00	B	9		LD HL,SFLAG\$
001B3D	CBC6	B	10		SET 0,(HL) ;Don't set "file open"
001B3F	21 00 1D	B	11		LD HL,SBUFF\$ ;Set buffer to system
001B42	CD 8A 19	B	12		CALL zOPEN ;Open the file
001B45	D5	B	13		PUSH DE ;Save FCB pointer
001B46	CC 56 1B	B	14		CALL Z,LOADER ;Load if no OPEN error
001B49	D1	B	15		POP DE ;Restore FCB pointer
001B4A	C8	B	16		RET Z ;Back if no error
001B4B	6F	B	17		LD L,A ;Xfer the error code
001B4C	2600	B	18		LD H,0
001B4E	F6C0	B	19		OR 0C0h ;Set RETurn & abbrev
001B50	FED8	B	20		CP 0D8h ;Change "file not in dir"
001B52	C0	B	21		RET NZ ;to "program not found"
001B53	C607	B	22		ADD A,7
001B55	C9	B	23		RET
		B	0		INCLUDE "loader.s" ; Get the system loader.
		B	1		SUBTITLE "< System Loader and associated routines >"
		B	2		NEWPAGE

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; System Loader and associated routines &gt;

PC	Object	I	Line	Source	.. \SYSRES \loader.s
		B	3		scope
		B	4		
		B	5		; System command file loader.
		B	6		
		B	7		ORG zspace.LOADER\$
		B	8		
001B56	78	B	9	LOADER	LD A,B ;Set overlay # (0 on non SYStem file).
001B57	32 B3 1B	B	10		(LDR14+1),A
001B5A	D5	B	11	PUSH	DE ;Save IX & xfer FCB to IX
001B5B	DDE3	B	12	EX	(SP),IX
001B5D	11 FF 1D	B	13	LD	DE,SBUFF\$+255 ;Init to end of buffer
001B60	CD 6F 1B	B	14	CALL	LDR01 ;Do the load
001B63	DDE1	B	15	POP	IX ;Recover IX
001B65	C9	B	16	RET	
		B	17		
		B	18		; Routine to ignore the LMF record.
		B	19		
001B66	CD D6 1B	B	20	LDR05	CALL LDR15 ;Get length of "comment"
001B69	47	B	21	LD	B,A
001B6A	CD D6 1B	B	22	LDR06	CALL LDR15 ;Read & ignore that many
001B6D	10 FB	B	23	DJNZ	LDR06 ;bytes then fall thru
		B	24		
		B	25		; Routine to parse LMF record types.
		B	26		
001B6F	CD D6 1B	B	27	LDR01	CALL LDR15 ;Get record type
001B72	FE01	B	28	LDR02	CP 1 ;Start of block?
001B74	28 1F	B	29	JR	Z,LDR08
001B76	FE02	B	30	CP	2 ;Start of TRAADR?
001B78	28 14	B	31	LDR03	JR Z,LDR07
001B7A	FE04	B	32	CP	4 ;End of LIB member?
001B7C	28 2A	B	33	JR	Z,LDR12
001B7E	FE08	B	34	CP	8 ;Begin ISAM table entry?
001B80	28 28	B	35	JR	Z,LDR13
001B82	FE0A	B	36	CP	10 ;End of ISAM map?
001B84	28 04	B	37	JR	Z,LDR04
001B86	FE20	B	38	CP	20h ;Ignore all other control
001B88	38 DC	B	39	JR	C,LDR05
001B8A	3E22	B	40	LDR04	LD A,22h ;Load file format err
001B8C	B7	B	41	OR	A
001B8D	C9	B	42	RET	
		B	43		
		B	44		; Grab transfer address.
		B	45		
001B8E	CD D6 1B	B	46	LDR07	CALL LDR15 ;Bypass 2nd X"02"
001B91	CD E8 1B	B	47	CALL	GETADR ;P/u transfer address
001B94	C9	B	48	RET	;Ret Z or NZ
		B	49		
		B	50		; Grab load block.
		B	51		
001B95	CD D6 1B	B	52	LDR08	CALL LDR15 ;P/u block len
001B98	47	B	53	LD	B,A
001B99	CD E8 1B	B	54	CALL	GETADR ;P/u load address
001B9C	C0	B	55	RET	NZ
001B9D	05	B	56	DEC	B ;Adj length for adr
001B9E	05	B	57	DEC	B
001B9F	CD D6 1B	B	58	LDR09	CALL LDR15 ;P/u block byte
001BA2	77	B	59	LD	(HL),A
001BA3	23	B	60	INC	HL
001BA4	10 F9	B	61	DJNZ	LDR09 ;Loop until block end
001BA6	18 C7	B	62	JR	LDR01

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; System Loader and associated routines &gt;

PC	Object	I	Line	Source	.. \SYSRES \loader.s
001BA8	E1	B	63	LDR12	POP HL
001BA9	C9	B	64		RET
		B	65		
		B	66	; Routine to check ISAM table match.	
		B	67		
001BAA	CD D6 1B	B	68	LDR13	CALL LDR15 ;Get record length
001BAD	47	B	69		LD B,A
001BAE	CD D6 1B	B	70		CALL LDR15 ;Get ISAM number
001BB1	05	B	71		DEC B ;& decrement counter
001BB2	FE00	B	72	LDR14	CP 0 ;Either ISAM# or 0
001BB4	20 B4	B	73		JR NZ,LDR06 ;Go if not a match
001BB6	CD E8 1B	B	74		CALL GETADR ;else get the TRAADR
001BB9	E5	B	75		PUSH HL ;& save it
001BBA	CC E8 1B	B	76		CALL Z,GETADR ;Get the NRN for member
001BBD	20 27	B	77		JR NZ,LODERR
001BBF	CD D6 1B	B	78		CALL LDR15 ;Get the sector offset
001BC2	5F	B	79		LD E,A ;Update pointer offset
001BC3	C5	B	80		PUSH BC
001BC4	44	B	81		LD B,H ;Xfer NRN position needed
001BC5	4D	B	82		LD C,L
001BC6	D5	B	83		PUSH DE ;Save buffer ptr offset
001BC7	DDE5	B	84		PUSH IX
001BC9	D1	B	85		POP DE ;P/u FCB into DE
001BCA	CD 5B 14	B	86		CALL ZPOSN ;Position to ISAM rec
001BCD	D1	B	87		POP DE ;Rcvr buffer ptr offset
001BCE	C1	B	88		POP BC
001BCF	20 15	B	89		JR NZ,LODERR
001BD1	CD DB 1B	B	90		CALL LDR17 ;Read the sector
001BD4	18 9C	B	91		JR LDR02 ;Now go read the member
		B	92		
		B	93	; Routine to get the next file byte.	
		B	94		
001BD6	1C	B	95	LDR15	INC E ;Bump buf pointer
001BD7	28 02	B	96		JR Z,LDR17 ;Read sector if needed
001BD9	1A	B	97	LDR16	LD A,(DE) ;P/U byte from buffer
001BDA	C9	B	98		RET
001BDB	E5	B	99	LDR17	PUSH HL ;Save regs
001BDC	D5	B	100		PUSH DE
001BDD	C5	B	101		PUSH BC
001BDE	CD 75 13	B	102		CALL NXTSECT ;Read next record
001BE1	C1	B	103		POP BC ;Restore regs
001BE2	D1	B	104		POP DE
001BE3	E1	B	105		POP HL
001BE4	28 F3	B	106		JR Z,LDR16 ;Bypass if no error
001BE6	C1	B	107	LODERR	POP BC ;Pop return address
001BE7	C9	B	108		RET
		B	109		
		B	110	; Routine to get an address field.	
		B	111		
001BE8	CD D6 1B	B	112	GETADR	CALL LDR15 ;Get low order byte
001BEB	6F	B	113		LD L,A
001BEC	CD D6 1B	B	114		CALL LDR15 ;Get hi order byte
001BEF	67	B	115		LD H,A
001BF0	BF	B	116		CP A
001BF1	C9	B	117		RET
		B	118		
		B	119	; End loader module	
		B	0	INCLUDE "interrupt_rst38.s" ;Interrupt 38h.	
		B	1	SUBTITLE "< Maskable interrupt routines & interrupt handler dispatch >"	
		B	2	NEWPAGE	



\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; Maskable interrupt routines &amp; interrupt handler dispatch &gt;

PC	Object	I	Line	Source	.. \SYSRES\interrupts\interrupt_rst38.s
		B	3		scope
		B	4		
		B	5		; Maskable interrupt has happened, we land here rst38.s.
		B	6		; Save PC for TRACE routine. TRACE will point where execution was happening when interrupt occurred.
		B	7		; Save flags & return address.
		B	8		; Mark system NETWORK flag as being in tasker/interrupt response.
		B	9		; Recover hardware MEMORY configuration & save so it can later be restored.
		B	10		; Switch hardware to standard 64k configuration.
		B	11		; Recover interrupt latch & strip out bits not enabled.
		B	12		; Execute handlers for devices enabled in MASK & hardware is interrupting.
		B	13		; Loop & test all 8 slots.
		B	14		; Test BREAK.
		B	15		; Restore memory.
		B	16		; Mark NETWORK flags we are no longer in tasker.
		B	17		; Restore registers.
		B	18		; Enable maskable interrupts.
		B	19		; Return from interrupt.
		B	20		
		B	21		ORG zspace.RST38\$
		B	22		
	00001BFF	B	23	RST38z	equ \$
001BFF	E3	B	24	EX	(SP),HL
001C00	22 AF 07	B	25	LD	(PCSAVE\$),HL ; Save for TRACE
001C03	E3	B	26	EX	(SP),HL
001C04	E5	B	27	PUSH	HL ; Save HL for now
001C05	F5	B	28	PUSH	AF ; Save AF for now
001C06	21 77 00	B	29	LD	HL,NFLAG\$ ; Show the system we
001C09	CBF6	B	30	SET	6,(HL) ; are in the TASKER
001C0B	210202	B	31	LD	HL,LBANK\$ ; P/U & save the current
001C0E	7E	B	32	LD	A,(HL) ; logical bank #
001C0F	3600	B	33	LD	(HL),0
001C11	F5	B	34	PUSH	AF
001C12	21 78 00	B	35	LD	HL,OPREG\$ ; Get current memory
001C15	7E	B	36	LD	A,(HL)
001C16	F5	B	37	PUSH	AF ; config & save
001C17	E68C	B	38	AND	8Ch ; Strip bits 0, 1, 4-6
001C19	F603	B	39	OR	3 ; Bring up regular 64K
001C1B	77	B	40	LD	(HL),A
001C1C	00	B	41	nop	
001C1D	00	B	42	nop	; OUT (084H),A
001C1E	3E00	B	43	ld	a,00h ; IN A,(INTLAT)
001C20	2F	B	44	cpl	; After CPL A is 0.
001C21	21 3C 00	B	45	ld	hl,INTIM\$ ; Address of interrupt latch.
001C24	7E	B	46	LD	a,(hl) ; Get who doin dis interrupin?
001C25	2C	B	47	INC	L ; Advance to int mask
001C26	A6	B	48	AND	(HL) ; Mask the latch bits
001C27	28 08	B	49	JR	Z,TSTBRK ; Go if nothing interrupted.
001C29	2C	B	50	NXTVCT	INC L ; Ck on INTVC\$.
001C2A	1F	B	51	RRA	; Ck if device interrupted.
001C2B	38 1C	B	52	JR	C,ACTVTSK
001C2D	2C	B	53	NXTMSK	INC L ; Ck all 8 bits of mask.
001C2E	B7	B	54	OR	A ; When fin, ck overhead
001C2F	20 F8	B	55	JR	NZ,NXTVCT ; task routine.
		B	56		
001C31	CD D6 07	B	57	TSTBRK	CALL KCKz ; Test <BREAK>, <SHIFT>
001C34	20 2A	B	58	JR	NZ,BREAKz ; Go if break
001C36	F1	B	59	TSKEXIT	POP AF ; Get previous mem config
001C37	32 78 00	B	60	LD	(OPREG\$),A ; & restore to it.
001C3A	00	B	61	nop	
001C3B	00	B	62	nop;	OUT (084H),A

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; Maskable interrupt routines &amp; interrupt handler dispatch &gt;

PC	Object	I	Line	Source	.. \SYSRES\interrupts\interrupt_rst38.s
001C3C	F1	B	63	POP	AF
001C3D	320202	B	64	LD	(LBANK\$),A
001C40	21 77 00	B	65	LD	HL,NFLAG\$ ; Now leaving the TASKER
001C43	CBB6	B	66	RES	6,(HL) ; show the system.
001C45	F1	B	67	POP	AF ; Restore previous regs
001C46	E1	B	68	POP	HL
001C47	FB	B	69	ei	
001C48	C9	B	70	RETINST	ret
		B	71		
		B	72	;	Found active INTVC\$
		B	73		
		B	74		
001C49	F5	B	75	ACTVTSK	PUSH AF ; Save the regs
001C4A	C5	B	76	PUSH	BC
001C4B	D5	B	77	PUSH	DE
001C4C	E5	B	78	PUSH	HL
001C4D	DDE5	B	79	PUSH	IX
001C4F	11 58 1C	B	80	LD	DE,POPREGS ; Stack return vector
001C52	D5	B	81	PUSH	DE
001C53	5E	B	82	LD	E,(HL) ; P/u INTVC pointer vector
001C54	2C	B	83	INC	L
001C55	56	B	84	LD	D,(HL)
001C56	EB	B	85	EX	DE,HL ; Shift it to HL
001C57	E9	B	86	JP	(HL) ; Go to service routine
		B	87		
		B	88		; Restore registers after service routine & loop to check next slot.
		B	89		
001C58	DDE1	B	90	POPREGS	POP IX
001C5A	E1	B	91	POP	HL
001C5B	D1	B	92	POP	DE
001C5C	C1	B	93	POP	BC
001C5D	F1	B	94	POP	AF
001C5E	18 CD	B	95	JR	NXTMSK ; Loop to next mask bit
		B	96		
		B	97		; BREAK key detected
		B	98		
001C60	30 08	B	99	BREAKz	JR NC,GOTBRK ; Go if <BREAK> only
001C62	C5	B	100	PUSH	BC ; Was <SHIFT-BREAK>
001C63	F3	B	101	DI	
001C64	CD B8 19	B	102	CALL	TAPDRV ; Reselect drive
001C67	C1	B	103	POP	BC
001C68	18 CC	B	104	JR	TSKEXIT
		B	105		
		B	106		; BREAK during tasking - enter DEBUG? - user BREAK?
		B	107		
001C6A	3A 7C 00	B	108	GOTBRK	LD A,(SFLAG\$) ; Check if BREAK key is
001C6D	E610	B	109	AND	10h ; disabled to inhibit
001C6F	20 C5	B	110	JR	NZ,TSKEXIT ; DEBUG or BREAK vector
001C71	21 9F 19	B	111	LD	HL,zDBGHK ; Merge DEBUG flag &
001C74	B6	B	112	OR	(HL) ; hook (X"00' or X"C9')
001C75	36C9	B	113	LD	(HL),0C9h ; Turn off DEBUG
001C77	23	B	114	INC	HL ; Point to zDEBUG vector &
001C78	28 14	B	115	JR	Z,EXITBRK ; go if DEBUG is active
001C7A	3A B0 07	B	116	LD	A,(PCSAVE\$+1) ; Don't allow vectored break
001C7D	FE24	B	117	CP	COREMAX\$>>8 ; if old PC is in SYSRES
001C7F	38 B5	B	118	JR	C,TSKEXIT
001C81	21 1D 00	B	119	LD	HL,PHIGH\$+1 ; or if old PC is
001C84	BE	B	120	CP	(HL) ; above HIGH\$
001C85	30 AF	B	121	JR	NC,TSKEXIT
001C87	210000	B	122	LD	HL,0 ; else ck if BREAK is

```

***** TRSDOS *****
< Maskable interrupt routines & interrupt handler dispatch >

PC      Object      I  Line      Source ..\SYSRES\interrupts\interrupt_rst38.s
      000001C88      B   123      BRKVEC$      EQU    $-2
001C8A 7C            B   124              LD      A,H          ; to be trapped by user
001C8B B5            B   125              OR      L
001C8C 28 A8         B   126              JR      Z,TSKEXIT
001C8E F1            B   127      EXITBRK      POP     AF          ; Discard old mem config
001C8F F1            B   128              POP     AF          ; Restore reg AF
001C90 F1            B   129              POP     AF
001C91 E3            B   130              EX      (SP),HL        ; P/u HL & stack vector.
001C92 FB            B   131              EI
001C93 C9            B   132              ret          ; To DEBUG or BREAK vector.
      0              B    0              INCLUDE "interrupt_TASKER.s"      ; System tasker.
      1              B    1              SUBTITLE "< TASKER interrupt processor >"
      2              B    2              NEWPAGE
```

\*\*\*\*\* TRSDOS \*\*\*\*\*  
 < TASKER interrupt processor >

PC	Object	I	Line	Source	.. \SYSRES\interrupts\interrupt_TASKER.s
		B	3		scope
		B	4		
		B	5	ORG	zspace.TASKER\$
		B	6		
001C94	00	B	7	TASKER	nop ; TASKER interrupt processor.
001C95	00	B	8		nop ; IN A,(0ECh).
		B	9		
001C96	3E0B	B	10		LD A,11 ; Task 11 executes every tick.
001C98	CD BB 1C	B	11		CALL RTCTASK ; RTC interrupt
001C9B	21 2B 00	B	12		LD HL,TIMSL\$
001C9E	CB06	B	13		RLC (HL) ; Ck on time slice
001CA0	D0	B	14		ret NC ; Ignore if nothing
001CA1	11 0F 07	B	15		LD DE,TIMTSK\$ ; on this interrupt
001CA4	D5	B	16		PUSH DE ; else init for clocker
001CA5	3E08	B	17		LD A,8 ; Task 8 at INT/2 if fast
001CA7	CD BB 1C	B	18		CALL RTCTASK
001CAA	3E09	B	19		LD A,9 ; Task 9 at INT/2 if fast
001CAC	CD BB 1C	B	20		CALL RTCTASK
001CAF	3E0A	B	21		LD A,10 ; Task 10 at INT/2 if fast
001CB1	CD BB 1C	B	22		CALL RTCTASK
001CB4	21 2C 00	B	23		LD HL,TIMER\$ ; Bump the timer at INT/2
001CB7	34	B	24		INC (HL) ; Beat the heart!
001CB8	7E	B	25		LD A,(HL) ; P/u the heart beat
001CB9	E607	B	26		AND 7 ; For this interrupt.
001CBB	07	B	27	RTCTASK	RLCA ; consider 0-7 only
001CBC	C6 4E	B	28		ADD A,TCB\$&0FFh ; Add offset to table.
001CBE	6F	B	29		LD L,A
001CBF	26 00	B	30		LD H,TCB\$>>8
001CC1	22 EC 1C	B	31		LD (zRPTSK+1),HL
001CC4	5E	B	32		LD E,(HL) ; P/u task vector addr
001CC5	2C	B	33		INC L
001CC6	56	B	34		LD D,(HL)
001CC7	D5	B	35		PUSH DE
001CC8	DDE1	B	36		POP IX ; Also to IX
001CCA	EB	B	37		EX DE,HL
001CCB	5E	B	38		LD E,(HL) ; P/u task entry point
001CCC	23	B	39		INC HL
001CCD	56	B	40		LD D,(HL)
001CCE	EB	B	41		EX DE,HL
001CCF	E9	B	42		jp (hl) ; Transfer control to task.
		B	43		
001CD0	D1	B	44	ZKLTSK	POP DE ; Remove ret
001CD1	3A EC 1C	B	45		LD A,(zRPTSK+1) ; Pt to task tbl entry
001CD4	D6 4E	B	46		SUB TCB\$&0FFh
001CD6	0F	B	47		RRCA ; of last task.
		B	48		
001CD7	11 E9 1C	B	49	ZRMTSK	LD DE,NOTASK ; Remove entry.
		B	50		
001CDA	FE0C	B	51	ZADTSK	CP 12 ; Too large a task?
001CDC	D0	B	52		RET NC ; Ret if too big else.
001CDD	07	B	53		RLCA ; add to task table
001CDE	C6 4E	B	54		ADD A,TCB\$&0FFh ; Add the offset
001CE0	6F	B	55		LD L,A ; Estab ptr to vector
001CE1	26 00	B	56		LD H,TCB\$>>8
001CE3	F3	B	57	CHGTASK	di ; Originally DI.
001CE4	73	B	58		LD (HL),E ; Vector adr to ptr tbl.
001CE5	2C	B	59		INC L
001CE6	72	B	60		LD (HL),D
001CE7	FB	B	61		ei ; Marker holding EI originally.
001CE8	C9	B	62		ret

\*\*\*\*\* TRSDOS \*\*\*\*\*  
 < TASKER interrupt processor >

PC	Object	I	Line	Source	.. \SYSRES\interrupts\interrupt_TASKER.s
001CE9	E81C	B	63	NOTASK	DW \$-1 ; Current task vector
		B	64		
001CEB		B	65	zRPTSK	; P/u last task done.
		B	66		
001CEB	210000	B	67	LD	HL,0
001CEE	5E	B	68	LD	E,(HL) ; P/u task vector addr
001CEF	23	B	69	INC	HL
001CF0	56	B	70	LD	D,(HL)
001CF1	EB	B	71	EX	DE,HL
001CF2	D1	B	72	POP	DE ;Pop ret addr
001CF3	18 EE	B	73	JR	CHGTASK
		B	74		
001CF5		B	75	zCKTSK	; Routine to check if task slot active.
		B	76		
001CF5	07	B	77	RLCA	;Task number * 2
001CF6	C6 4F	B	78	ADD	A,TCB\$&0FFh+1 ;Index into task table
001CF8	6F	B	79	LD	L,A
001CF9	26 00	B	80	LD	H,TCB\$>>8
001CFB	3E 1C	B	81	LD	A,NOTASK>>8 ;Check match of high order only.
001CFD	BE	B	82	CP	(HL)
001CFE	C9	B	83	RET	; Z or NZ result.
		A	135		
		A	136	ORG	zspace.SYSBUF\$ ; 0/S System buffer, 1D00h.
		A	137		
001D00	00 00 00 00 00 00	A	138	SBUF\$	BLKB 256,0 ; Page disk I/O buffer.
001D06	00 00 00 00 00 00				
001D0C	00 00 00 00 00 00				
001D12	00 00 00 00 00 00				
001D18	00 00 00 00 00 00				
001D1E	00 00 00 00 00 00				
001D24	00 00 00 00 00 00				
001D2A	00 00 00 00 00 00				
001D30	00 00 00 00 00 00				
001D36	00 00 00 00 00 00				
001D3C	00 00 00 00 00 00				
001D42	00 00 00 00 00 00				
001D48	00 00 00 00 00 00				
001D4E	00 00 00 00 00 00				
001D54	00 00 00 00 00 00				
001D5A	00 00 00 00 00 00				
001D60	00 00 00 00 00 00				
001D66	00 00 00 00 00 00				
001D6C	00 00 00 00 00 00				
001D72	00 00 00 00 00 00				
001D78	00 00 00 00 00 00				
001D7E	00 00 00 00 00 00				
001D84	00 00 00 00 00 00				
001D8A	00 00 00 00 00 00				
001D90	00 00 00 00 00 00				
001D96	00 00 00 00 00 00				
001D9C	00 00 00 00 00 00				
001DA2	00 00 00 00 00 00				
001DA8	00 00 00 00 00 00				
001DAE	00 00 00 00 00 00				
001DB4	00 00 00 00 00 00				
001DBA	00 00 00 00 00 00				
001DC0	00 00 00 00 00 00				
001DC6	00 00 00 00 00 00				
001DCC	00 00 00 00 00 00				
001DD2	00 00 00 00 00 00				

```

***** TRSDOS *****
< TASKER interrupt processor >

```

```

PC      Object
001DD8 00 00 00 00 00 00
001DDE 00 00 00 00 00 00
001DE4 00 00 00 00 00 00
001DEA 00 00 00 00 00 00
001DF0 00 00 00 00 00 00
001DF6 00 00 00 00 00 00
001DFC 00 00 00 00

I      Line      Source D:\TRS-OS\Jan2026\TRSDOS_7\TRSDOS.s
A      139
A      140
A      141      ; Begin IPL code @1E00h SYS0.
A      142      ; This code runs one time @ BOOT/RESET/IPL.
A      143
A      144      SUBTITLE      "< Beginning of SYS0/TRS-OS. >"
A      145
A      146      ORG      zspace.OVERLAY$                                ; SYS0 w/code to initialize CPU/registers i
A      147      ;      ORG      zspace.LIBRARY$                                ; LIBRARY$
A      148      ;      org      zspace.APPLICATION$                        ; TRSDOS applications start here.
A      149
B      0      include      "IPL_Code.s"
B      1      ; This code loads in system overlay region, @ 1E00h.
B      2      ; This module comprises SYS0.
B      3      ; It runs one time @IPL & is destroyed during boot to ready.
B      4
B      5
B      6      ; Next step, initialize CPU.
B      7
C      0      include "ipl-eZ80_CPU.s"
C      1      subtitle      "< SYS0 IPL - Minimum initialization for eZ80F91 & eZ80F92. >"
C      2      newpage

```

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - Minimum initialization for eZ80F91 &amp; eZ80F92. &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-eZ80_CPU.s
		C	3		scope
		C	4		
		C	5	; Copyright (C) 2022 - 2026 by Daniel Paul Martin. All Rights Reserved.	
		C	6		
		C	7		
001E00	F3	C	8	__init	di
001E01	ED5E	C	9		im 2 ; Prepare interrupt controller for mode 2.
		C	10		
001E03	3EC9	C	11		LD A, RETOPCODE ; Ignore stray interrupt requests.
001E05	32 66 00	C	12		LD (zNMI), A ; Ignore NMI, return to caller.
001E08	32 38 00	C	13		LD (zRST38), A ; Ignore RST 38h, return to caller.
		C	14		
001E0B	5B31FF00 01	C	15		ld.lil sp, 0100FFh ; Setup 24 bit stack pointer.
001E10	31 80 03	C	16		LD SP, STACK\$ ; Point SP to stack area.
		C	17		
001E13	AF	C	18		XOR A ; Initialize & clear stack.
001E14	21 81 03	C	19		LD HL, STACK\$+1 ; Start of Stack+1
001E17	2D	C	20	CLRLOOP	DEC L ; Move down a byte.
001E18	77	C	21		LD (HL), A ; Loop and fill.
001E19	20 FC	C	22		JR NZ, CLRLOOP ; Fill from stack\$ to bottom of stack\$'s page with 0"s.
		C	23		
		C	24		
001E1B	3A D9 10	C	25		ld a, (U0DVR)
001E1E	32 0A 87	C	26		ld (quiet_default+1), a
001E21	3A0300	C	27		ld a, (quiet_default_flg); Get quiet/default IPL code.
001E24	32 09 87	C	28		ld (quiet_default), a
001E27	CB7F	C	29		bit 7, a ; If bit 7 set, then go into quiet mode.
001E29	28 05	C	30		jr z, no_quiet
001E2B	3EC9	C	31		ld a, RETOPCODE ; Return op code.
001E2D	32 D9 10	C	32		ld (U0DVR), a ; Turn driver off.
		C	33		
001E30	C5	C	34	no_quiet	push bc ; Save pointer to 2nd file if any.
001E31	D5	C	35		push de ; Save pointer to 1st file if any.
		C	36		
		C	37	; Set PFLAG\$ in TRSDOS to 91h or 92h representing eZ80F91/eZ80f92.	
		C	38		
		C	39	MVC cpu_f91_msg, cpu_typ_msg, 7 ; Fill in splash screen <F91 detected> (default message).	
		C+	39	scope	
001E32	21 9E 85	C+	39		ld hl, cpu_f91_msg
001E35	11 88 71	C+	39		ld de, cpu_typ_msg
001E38	010700	C+	39		ld bc, 7
001E3B	EDB0	C+	39	LDIR	
		C+	39	ENDMAC MVC	
		C	40		
001E3D	3E01	C	41		ld a, 1h ; Processor type F92.
001E3F	CB27	C	42		sla a ; Shift it left one place.
001E41	32 79 00	C	43		ld (PFLAG\$), a ; Store in processor byte.
001E44	ED3800	C	44	in0	a, (ZDI_ID_L) ; Get product ID.
001E47	FE08	C	45	cp	08h ; 8=eZ80F91
001E49	28 12	C	46	jr	z, tst_initial
001E4B	3E02	C	47		ld a, 02h ; Processor type F92.
001E4D	CB27	C	48		sla a ; Pack it.
001E4F	32 79 00	C	49		ld (PFLAG\$), a ; If not 92h then set processor type to f92.
		C	50	MVC cpu_f92_msg, cpu_typ_msg, 7 ; Tell splash screen F92 detected.	
		C+	50	scope	
001E52	21 A5 85	C+	50		ld hl, cpu_f92_msg
001E55	11 88 71	C+	50		ld de, cpu_typ_msg
001E58	010700	C+	50		ld bc, 7
001E5B	EDB0	C+	50	LDIR	
		C+	50	ENDMAC MVC	

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - Minimum initialization for eZ80F91 &amp; eZ80F92. &gt;

PC	Object	I	Line	Source	..\\SYS0_IPL\\ipl-eZ80_CPU.s
		C	51		
		C	52	;	Test in entry to this module from an already running CPU.
		C	53		
001E5D	3EF6	C	54	tst_initial	ld    a,HIGH(INTERRUPT.handler)       ; Setup internal interrupt register to interrupt table.
001E5F	ED47	C	55		ld    i,a
		C	56		
001E61	ED38C7	C	57		in0   a,(UART0_SPR)                   ; Test if IPL came from other OS.
001E64	E60F	C	58		and   00001111b
001E66	32 FD 86	C	59		ld    (initialflg),a
		C	60		
001E69	CB27	C	61		sla   a
001E6B	CB27	C	62		sla   a
001E6D	CB27	C	63		sla   a
001E6F	CB27	C	64		sla   a                               ; Shift left 4 bits to pack platform into zTFLAG\$.
001E71	67	C	65		ld    h,a
001E72	3A 7D 00	C	66		ld    a,(zTFLAG\$)
001E75	B4	C	67		or    h
001E76	32 7D 00	C	68		ld    (zTFLAG\$),a                   ; Store platform bits into platform type byte.
		C	69		
001E79	CB2F	C	70		sra   a                               ; Unpack and get back platform nibble.
001E7B	CB2F	C	71		sra   a
001E7D	CB2F	C	72		sra   a
001E7F	CB2F	C	73		sra   a
		C	74		
001E81	B7	C	75		or    a                               ; Is it <>0?
001E82	C2 62 20	C	76		jp    nz,NOINITIAL                   ; If <>0 we came from CPM, do not initialize CPU.
		C	77		
001E85	C630	C	78		add   '0'
001E87	32 73 71	C	79		ld    (vendor\$-6),a                   ; Embed this initial flag in boot menu for diagnostics.
		C	80		
		C	81		MVC   vendor_z\$,vendor\$,8
		C+	81		scope
001E8A	21 C4 85	C+	81		ld    hl,vendor_z\$
001E8D	11 79 71	C+	81		ld    de,vendor\$
001E90	010800	C+	81		ld    bc,8
001E93	EDB0	C+	81		LDIR
		C+	81		ENDMAC       MVC
		C	82		MVC   from_por,ipl_from,4           ; Tell splash screen we are booting from Power On Reset.
		C+	82		scope
001E95	21 9A 85	C+	82		ld    hl,from_por
001E98	11 67 71	C+	82		ld    de,ipl_from
001E9B	010400	C+	82		ld    bc,4
001E9E	EDB0	C+	82		LDIR
		C+	82		ENDMAC       MVC
		C	83		
001EA0	D1	C	84		pop   de                               ; Discard pointer to 1st data file (it doesnt exist).
001EA1	C1	C	85		pop   bc                               ; Discard pointer to 2nd data file (it doesnt exist).
		C	86		
001EA2		C	87		init_f91f92f93;This section of code is common to both eZ80F91 & eZ80F92 processors.
		C	88		
		C	89		MVC   bootinitmsg,splashiplmode+1,bootinitmsglen-1
		C+	89		scope
001EA2	21 79 71	C+	89		ld    hl,bootinitmsg
001EA5	11 2B 79	C+	89		ld    de,splashiplmode+1
001EA8	01 17 00	C+	89		ld    bc,bootinitmsglen-1
001EAB	EDB0	C+	89		LDIR
		C+	89		ENDMAC       MVC
		C	90		
		C	91		;   Disable internal peripheral interrupt sources, this will help during a RAM debug session.
		C	92		



\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - Minimum initialization for eZ80F91 &amp; eZ80F92. &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-eZ80_CPU.s
001EAD	AF	C	93	xor	a
		C	94		
001EAE	ED399C	C	95	out0	(PB_ALT1), a
001EB1	ED39A0	C	96	out0	(PC_ALT1), a
001EB4	ED39A4	C	97	out0	(PD_ALT1), a
		C	98		
		C	99	;	OUT0 (PC_DR),A ; to enable RS232 Driver
		C	100		
		C	101	;	Port C = SF_HOLD, SF_WP, SF_MISO, SF_MOSI, CTS1#, RTS1#, RXD1, TXD1
		C	102		
001EB7	3EFC	C	103	LD	A,11111100b ; Port C default data
001EB9	ED399E	C	104	OUT0	(PC_DR),A ; and set it
001EBC	060F	C	105	LD	B,00001111b ; ALT 2
001EBE	ED01A1	C	106	OUT0	(PC_ALT2),B ; and set it
001EC1	162F	C	107	LD	D,00101111b ; Set bit directions
001EC3	ED119F	C	108	OUT0	(PC_DDR),D ; and "alternate functions"
		C	109		
001EC6	3E04	C	110	ld	a,%04
001EC8	ED39BA	C	111	out0	(SPI_CTL), a ; SPI
		C	112		
001ECB	ED38ED	C	113	in0	a,(RTC_CTRL) ; RTC,
001ECE	E6BE	C	114	and	a,%BE ; Writing to the RTC_CTRL register also resets the RTC coun
001ED0	ED39ED	C	115	out0	(RTC_CTRL), a ; prescaler allowing RTC to be synchronized to other.time s
		C	116		
001ED3	3E 10	C	117	ld	a, __CS0_LBR_INIT_PARAM ; Configure external memory & I/O.
001ED5	ED39A8	C	118	out0	(CS0_LBR), a
001ED8	3E 8F	C	119	ld	a, __CS0_UBR_INIT_PARAM
001EDA	ED39A9	C	120	out0	(CS0_UBR), a
001EDD	3E 00	C	121	ld	a, __CS0_BMC_INIT_PARAM
001EDF	ED39F0	C	122	out0	(CS0_BMC), a
001EE2	3E 88	C	123	ld	a, __CS0_CTL_INIT_PARAM
001EE4	ED39AA	C	124	out0	(CS0_CTL), a
001EE7	3E 00	C	125	ld	a, __CS1_LBR_INIT_PARAM
001EE9	ED39AB	C	126	out0	(CS1_LBR), a
001EEC	3E 07	C	127	ld	a, __CS1_UBR_INIT_PARAM
001EEE	ED39AC	C	128	out0	(CS1_UBR), a
001EF1	3E 00	C	129	ld	a, __CS1_BMC_INIT_PARAM
001EF3	ED39F1	C	130	out0	(CS1_BMC), a
001EF6	3E 28	C	131	ld	a, __CS1_CTL_INIT_PARAM
001EF8	ED39AD	C	132	out0	(CS1_CTL), a
001EFB	3E 08	C	133	ld	a, __CS2_LBR_INIT_PARAM
001EFD	ED39AE	C	134	out0	(CS2_LBR), a
001F00	3E 0F	C	135	ld	a, __CS2_UBR_INIT_PARAM
001F02	ED39AF	C	136	out0	(CS2_UBR), a
001F05	3E 00	C	137	ld	a, __CS2_BMC_INIT_PARAM
001F07	ED39F2	C	138	out0	(CS2_BMC), a
001F0A	3E 28	C	139	ld	a, __CS2_CTL_INIT_PARAM
001F0C	ED39B0	C	140	out0	(CS2_CTL), a
001F0F	3E 00	C	141	ld	a, __CS3_LBR_INIT_PARAM
001F11	ED39B1	C	142	out0	(CS3_LBR), a
001F14	3E 00	C	143	ld	a, __CS3_UBR_INIT_PARAM
001F16	ED39B2	C	144	out0	(CS3_UBR), a
001F19	3E 00	C	145	ld	a, __CS3_BMC_INIT_PARAM
001F1B	ED39F3	C	146	out0	(CS3_BMC), a
001F1E	3E 00	C	147	ld	a, __CS3_CTL_INIT_PARAM
001F20	ED39B3	C	148	out0	(CS3_CTL), a
		C	149		
		C	150	;	enable internal memory
		C	151		
001F23	3E 00	C	152	ld	a, __FLASH_ADDR_U_INIT_PARAM

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - Minimum initialization for eZ80F91 &amp; eZ80F92. &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-eZ80_CPU.s
001F25	ED39F7	C	153	out0	(FLASH_ADDR_U), a
001F28	3E 60	C	154	ld	a, __FLASH_CTL_INIT_PARAM
001F2A	ED39F8	C	155	out0	(FLASH_CTRL), a
001F2D	3E AF	C	156	ld	a, __RAM_ADDR_U_INIT_PARAM
001F2F	ED39B5	C	157	out0	(RAM_ADDR_U), a
001F32	3E 80	C	158	ld	a, __RAM_CTL_INIT_PARAM
001F34	ED39B4	C	159	out0	(RAM_CTL), a
		C	160		
		C	161		; End of common processor code.
		C	162		
		C	163		; Get processor model (91h,92h,93h) code.
		C	164		; Continue setup by executing processor specific routines.
		C	165		
001F37	3A 79 00	C	166	ld	a,(PFLAG\$)
001F3A	CB2F	C	167	sra	a
001F3C	FE01	C	168	cp	01h
001F3E	C2 AD 1F	C	169	jp	nz,init_f92
		C	170		
001F41		C	171	init_f91;	eZ80F91 Initialization.
		C	172		
001F41	AF	C	173	xor	a ; A=0.
		C	174		
001F42	ED394C	C	175	out0	(EMAC_IEN), a ;**** EMAC
001F45	ED39CB	C	176	out0	(I2C_CTL), a ; I2C
001F48	ED39FB	C	177	out0	(FLASH_IRQ), a ; Flash.
		C	178		
001F4B	ED3961	C	179	out0	(TMR0_IER), a ;**** timers
001F4E	ED3966	C	180	out0	(TMR1_IER), a ;****
001F51	ED3970	C	181	out0	(TMR2_IER), a ;****
001F54	ED3975	C	182	out0	(TMR3_IER), a ;****
		C	183		
001F57	ED3998	C	184	out0	(PA_ALT1), a ;***
001F5A	ED3999	C	185	out0	(PA_ALT2), a ;***
		C	186		
001F5D	ED399C	C	187	out0	(PB_ALT1), a ;***000000
001F60	ED399D	C	188	out0	(PB_ALT2), a
		C	189		
		C	190		
001F63	ED39A2	C	191	out0	(PD_DR),a
		C	192		
001F66	3E03	C	193	ld	a,3
001F68	ED39A5	C	194	OUT0	(PD_ALT2),A
		C	195		
001F6B	3EFF	C	196	ld	a,0FFh
001F6D	ED3997	C	197	out0	(PA_DDR), a ;*** ; GPIO
001F70	ED399B	C	198	out0	(PB_DDR), a
		C	199		
001F73	3EEB	C	200	ld	a,0EBh
001F75	ED39A3	C	201	out0	(PD_DDR), a
		C	202		
		C	203		; Set UART0 baud rate generator for eZ80F91. f91_baud is equated in eZ80.
		C	204		
001F78	3E83	C	205	LD	A,83h ; Set bit 7 to enable UART BRG register access.
001F7A	ED39C3	C	206	OUT0	(UART0_LCTL),A ; enable access to BRG.
		C	207		
001F7D	3E 0D	C	208	LD	A,LOW(f91_console_baud) ; LSB of BRG divisor
001F7F	ED39C0	C	209	OUT0	(BRG0_DLR_L),A ; load low byte of BRG.
		C	210		
001F82	3E 00	C	211	LD	A,HIGH(f91_console_baud) ; MSB of BRG divisor
001F84	ED39C1	C	212	OUT0	(BRG0_DLR_H),A ; load high byte of BRG

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - Minimum initialization for eZ80F91 &amp; eZ80F92. &gt;

PC	Object	I	Line	Source	..\\SYS0_IPL\\ipl-eZ80_CPU.s
		C	213		
001F87	3E03	C	214	LD A,03h	; select 8bits, no parity, 1 stop
001F89	ED39C3	C	215	OUT0 (UART0_LCTL),A	; and lock BRG.
		C	216		
001F8C	0600	C	217	LD B,0	
001F8E	10 FE	C	218	DJNZ \$	; WARMUP, let BRG stabilize?
		C	219		
		C	220		; Initialize real time interrupts here for f91 processor.
		C	221		; Setup 60HZ real time interrupts using TMR1, setup registers but do not enable.
		C	222		; Enable is done when background tasking is enabled.
		C	223		
001F90	AF	C	224	xor a	
		C	225	; out0 (TMR_ISS),a	; Clock is derived from divided systems clock.MISTAKE?
001F91	ED3965	C	226	out0 (TMR1_CTL),a	; Disable.
		C	227		
001F94	2173CB	C	228	ld hl,52083	; Get reload values.
001F97	ED2968	C	229	out0 (TMR1_RR_L),l	; Put low byte in register.
001F9A	ED2169	C	230	out0 (TMR1_RR_H),h	; Load upper byte in reload counter high.
		C	231		
		C	232		
001F9D	3E0C	C	233	ld a,00001100b	; Clock derrived from dividing system clock by 16.
001F9F	ED3965	C	234	out0 (TMR1_CTL),a	; Disable.
001FA2	ED3867	C	235	in0 a,(TMR1_IIR)	; Read to reset.
001FA5	3E01	C	236	ld a,01	; Enable interrupt at end of count.
001FA7	ED3966	C	237	out0 (TMR1_IER),a	; IRQ EOC EN.
		C	238		
001FAA	C3 26 20	C	239	jp START_UARTS	
		C	240		
001FAD		C	241	init_f92;	Initialize here for f92 processor.
		C	242		
001FAD	3EF4	C	243	LD A,11110100b	; PCB V1.1+ Port B default data
001FAF	06CC	C	244	LD B,11001100b	; ALT 2
001FB1	16CC	C	245	LD D,11001100b	; bit directions
		C	246		
001FB3	ED399A	C	247	OUT0 (PB_DR),A	; Set default data
001FB6	ED019D	C	248	OUT0 (PB_ALT2),B	; Set ALT 2.
001FB9	ED119B	C	249	OUT0 (PB_DDR),D	; Set bit directions
		C	250		
		C	251	; Port C = SF_HOLD, SF_WP, SF_MISO, SF_MOSI, CTS1#, RTS1#, RXD1, TXD1	
		C	252		
001FBC	3EFC	C	253	LD A,11111100b	; Port C default data
001FBE	060F	C	254	LD B,00001111b	; ALT 2
001FC0	162F	C	255	LD D,00101111b	; Set bit directions.
		C	256		
001FC2	ED399E	C	257	OUT0 (PC_DR),A	; and set it
001FC5	ED01A1	C	258	OUT0 (PC_ALT2),B	; and set it
001FC8	ED119F	C	259	OUT0 (PC_DDR),D	; and "alternate functions"
		C	260		
		C	261	; Port D = SD_CD#, N/A, SF_CE#, SF_CLK, CTS0#, RTS0#, RXD0, TXD0	
		C	262		
		C	263		
001FCB	3EAC	C	264	LD A,10101100b	; Set Port D default data
001FCD	ED39A2	C	265	OUT0 (PD_DR),A	; before changing direction
001FD0	3E0F	C	266	LD A,00001111b	; Allocate UART 0
001FD2	ED39A5	C	267	OUT0 (PD_ALT2),A	; bits to
		C	268		
001FD5	ED39C1	C	269	out0 (UART0_IER), a	; UARTs
001FD8	ED39D1	C	270	out0 (UART1_IER), a	
		C	271		
		C	272		

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - Minimum initialization for eZ80F91 &amp; eZ80F92. &gt;

PC	Object	I	Line	Source	..\\SYS0_IPL\\ipl-eZ80_CPU.s
		C	273		; Set baud rate generator. f92_baud rate is equated in eZ80.
		C	274		
001FDB	3E83	C	275	LD A,83h	; Set bit 7 to enable UART BRG register access.
001FDD	ED39C3	C	276	OUT0 (UART0_LCTL),A	; enable access to BRG.
		C	277		
001FE0	3E05	C	278	LD A,LOW(f92_console_baud)	; LSB of BRG divisor
001FE2	ED39C0	C	279	OUT0 (BRG0_DLR_L),A	; load low byte of BRG.
		C	280		
001FE5	3E00	C	281	LD A,HIGH(f92_console_baud)	; MSB of BRG divisor
001FE7	ED39C1	C	282	OUT0 (BRG0_DLR_H),A	; load high byte of BRG
		C	283		
001FEA	3E03	C	284	LD A,03h	; select 8bits, no parity, 1 stop
001FEC	ED39C3	C	285	OUT0 (UART0_LCTL),A	; and lock BRG.
		C	286		
001FEF	0600	C	287	LD B,0	
001FF1	10 FE	C	288	DJNZ \$	; WARMUP, let BRG stabilize?
		C	289		
		C	290		; Initialize Timers here for f92 processor.
		C	291		; Ensure all f92 timers are disabled & clear.
		C	292		
001FF3	AF	C	293	xor a	; Load 0's into following registers.
001FF4	ED3992	C	294	out0 (TMR_ISS),a	; Clock is derived from divided systems clock.
001FF7	ED3980	C	295	out0 (f92_TMR0_CTL),a	; Disable and procede with initial register load.
001FFA	ED3986	C	296	out0 (f92_TMR2_CTL),a	
001FFD	ED3989	C	297	out0 (f92_TMR3_CTL),a	
002000	ED398C	C	298	out0 (f92_TMR4_CTL),a	
002003	ED398F	C	299	out0 (f92_TMR5_CTL),a	
		C	300		
		C	301		; Now read these same registers to clear pending interrupts.
		C	302		
002006	ED3880	C	303	in0 a,(f92_TMR0_CTL)	
002009	ED3883	C	304	in0 a,(f92_TMR1_CTL)	
00200C	ED3886	C	305	in0 a,(f92_TMR2_CTL)	
00200F	ED3889	C	306	in0 a,(f92_TMR3_CTL)	
002012	ED388C	C	307	in0 a,(f92_TMR4_CTL)	
002015	ED388F	C	308	in0 a,(f92_TMR5_CTL)	
		C	309		
		C	310		; Setup 60HZ real time interrupts using TMR1.
		C	311		; Setup but do not enable timer.
		C	312		; Enable is done when background tasking enabled.
		C	313		
002018	01004B	C	314	ld bc,19200	; Get high/low byte of count down value.
00201B	ED0984	C	315	out0 (f92_TMR1_RR_L),c	; Put in register.
00201E	ED0185	C	316	out0 (f92_TMR1_RR_H),b	; Upper byte, load in reload counter high.
002021	3E14	C	317	ld a,00010100b	; Set divider in register.
002023	ED3983	C	318	out0 (f92_TMR1_CTL),a	
		C	319		
002026		C	320	START_UARTS;	*** START UART 0 SERIAL COMMUNICATIONS WITH HOST. ***
		C	321		
002026	3E01	C	322	LD A,01h	; Per Zilog manual, 1st set bit 0.
002028	ED39C2	C	323	OUT0 (UART0_FCTL),A	; Then set bits 1 & 2..
		C	324		
00202B	3E07	C	325	LD A,07h	; Activate receive and transmit FIFOs.
00202D	ED39C2	C	326	OUT0 (UART0_FCTL),A	; Transmit/Receive Enable.
		C	327		
		C	328		; Tell world UART's & hardware ready by setting DTR. CTS set later when driver is initialized.
		C	329		
002030	3E03	C	330	LD A,00000011b	; Set DTR active, RTS inactive.
002032	ED39C4	C	331	OUT0 (UART0_MCTL),A	; UART0 tell other end we are ready.
		C	332		

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - Minimum initialization for eZ80F91 &amp; eZ80F92. &gt;

PC	Object	I	Line	Source
		C	333	; *** START UART 1 SERIAL COMMUNICATIONS WITH HOST. ***
		C	334	
002035	3E83	C	335	LD A,83h ; Set bit 7 to enable UART BRG register access.
002037	ED39D3	C	336	OUT0 (UART1_LCTL),A ; enable access to BRG.
		C	337	
		C	338	; Set baud rate generator. Baud rate is equated in eZ80.
		C	339	
00203A	3E 0D	C	340	LD A,LOW(f91_serialnet_baud) ; LSB of BRG divisor
00203C	ED39D0	C	341	OUT0 (UART1_BRG_L),A ; load low byte of BRG.
		C	342	
00203F	3E 00	C	343	LD A,HIGH(f91_serialnet_baud) ; MSB of BRG divisor
002041	ED39D1	C	344	OUT0 (UART1_BRG_H),A ; load high byte of BRG
		C	345	
002044	3E03	C	346	LD A,03h ; select 8bits, no parity, 1 stop.
002046	ED39D3	C	347	OUT0 (UART1_LCTL),A ; and lock BRG.
		C	348	
002049	0600	C	349	LD B,0
00204B	10 FE	C	350	DJNZ \$ ; WARMUP, let BRG stabilize?
		C	351	
00204D	3E01	C	352	LD A,01h ; Per Zilog manual, 1st set bit 0.
00204F	ED39D2	C	353	OUT0 (UART1_FCTL),A ; Then set bits 1 & 2..
		C	354	
002052	3E07	C	355	LD A,07h ; Activate receive and transmit FIFOs.
002054	ED39D2	C	356	OUT0 (UART1_FCTL),A ; Transmit/Receive Enable.
		C	357	
		C	358	UART1.dtr.set
		C+	358	scope
002057	ED38D4	C+	358	in0 a,(UART1_MCTL) ; Get current register status.
00205A	F601	C+	358	or 1 ; Set bit 0 to 1.
00205C	ED39D4	C+	358	OUT0 (UART1_MCTL),A ; UART1 tell other end we are ready.
		C+	358	endmac UART1.dtr.set
		C	359	
00205F	C3 05 21	C	360	jp finish_init
		C	361	
		C	362	; We arrive here, IPL was warm started by other OS.
		C	363	; Test for optional data files passed.
		C	364	; If no file passed, skip altering storage map.
		C	365	; If file was loaded adjust map to reflect variable size files.
		C	366	
		C	367	; Tell world UART's & hardware ready by setting DTR. CTS set later when driver is initialized.
		C	368	
002062		C	369	NOINITIAL: MVC vendor_undef\$,vendor\$,8 ; Prepare for unrecognized vendor.
		C+	369	scope
002062	21 CC 85	C+	369	ld hl,vendor_undef\$
002065	11 79 71	C+	369	ld de,vendor\$
002068	010800	C+	369	ld bc,8
00206B	EDB0	C+	369	LDIR
		C+	369	ENDMAC MVC
00206D	3A FD 86	C	370	ld a,(initialflg)
002070	F5	C	371	push af
002071	C630	C	372	add '0'
002073	32 73 71	C	373	ld (vendor\$-6),a ; Embed this initial flag in boot menu for diagnostics.
002076	F1	C	374	pop af
		C	375	
002077	FE01	C	376	cp 1 ; Circle M platform?
		C	377	IF_NE_GOTO cir_tst
		C+	377	scope
002079	C2 8A 20	C+	377	jp nz,cir_tst
		C+	377	endmac IF_NE_GOTO
		C	378	MVC vendor_cirm\$,vendor\$,8

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - Minimum initialization for eZ80F91 &amp; eZ80F92. &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-eZ80_CPU.s
		C+	378		scope
00207C	21 AC 85	C+	378		ld hl,vendor_cirm\$
00207F	11 79 71	C+	378		ld de,vendor\$
002082	010800	C+	378		ld bc,8
002085	EDB0	C+	378		LDIR
		C+	378		ENDMAC MVC
		C	379		GOTO plat_set
		C+	379		scope
002087	C3 D6 20	C+	379		jp plat_set
		C+	379		endmac GOTO
		C	380		
00208A	FE02	C	381	cir_tst	cp 2 ; Circle M platform?
		C	382		IF_NE_GOTO agon_tst
		C+	382		scope
00208C	C2 9D 20	C+	382		jp nz,agon_tst
		C+	382		endmac IF_NE_GOTO
		C	383		MVC vendor_cirm\$,vendor\$,8
		C+	383		scope
00208F	21 AC 85	C+	383		ld hl,vendor_cirm\$
002092	11 79 71	C+	383		ld de,vendor\$
002095	010800	C+	383		ld bc,8
002098	EDB0	C+	383		LDIR
		C+	383		ENDMAC MVC
		C	384		GOTO plat_set
		C+	384		scope
00209A	C3 D6 20	C+	384		jp plat_set
		C+	384		endmac GOTO
		C	385		
00209D	FE03	C	386	agon_tst	cp 3 ; AGON platform?
		C	387		IF_NE_GOTO dinno_tst
		C+	387		scope
00209F	C2 B0 20	C+	387		jp nz,dinno_tst
		C+	387		endmac IF_NE_GOTO
		C	388		MVC vendor_agon\$,vendor\$,8 ; Tell operator we are on AGON platform.
		C+	388		scope
0020A2	21 BC 85	C+	388		ld hl,vendor_agon\$
0020A5	11 79 71	C+	388		ld de,vendor\$
0020A8	010800	C+	388		ld bc,8
0020AB	EDB0	C+	388		LDIR
		C+	388		ENDMAC MVC
		C	389		GOTO plat_set
		C+	389		scope
0020AD	C3 D6 20	C+	389		jp plat_set
		C+	389		endmac GOTO
		C	390		
0020B0	FE04	C	391	dinno_tst	cp 4 ; dinno platform?
		C	392		IF_NE_GOTO plat_set
		C+	392		scope
0020B2	C2 D6 20	C+	392		jp nz,plat_set
		C+	392		endmac IF_NE_GOTO
		C	393		MVC vendor_dinno\$,vendor\$,8 ; Tell operator we are on dinno platform.
		C+	393		scope
0020B5	21 B4 85	C+	393		ld hl,vendor_dinno\$
0020B8	11 79 71	C+	393		ld de,vendor\$
0020BB	010800	C+	393		ld bc,8
0020BE	EDB0	C+	393		LDIR
		C+	393		ENDMAC MVC
		C	394		GOTO plat_set
		C+	394		scope
0020C0	C3 D6 20	C+	394		jp plat_set

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - Minimum initialization for eZ80F91 &amp; eZ80F92. &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-eZ80_CPU.s
		C+	394	endmac	GOTO
		C	395		
0020C3	FE05	C	396	cp 5	; Circle M platform?
		C	397	IF_NE_GOTO plat_set	
		C+	397	scope	
0020C5	C2 D6 20	C+	397	jp nz,plat_set	
		C+	397	endmac	IF_NE_GOTO
		C	398	MVC vendor_cirm\$,vendor\$,8	
		C+	398	scope	
0020C8	21 AC 85	C+	398	ld hl,vendor_cirm\$	
0020CB	11 79 71	C+	398	ld de,vendor\$	
0020CE	010800	C+	398	ld bc,8	
0020D1	EDB0	C+	398	LDIR	
		C+	398	ENDMAC	MVC
		C	399	GOTO plat_set	
		C+	399	scope	
0020D3	C3 D6 20	C+	399	jp plat_set	
		C+	399	endmac	GOTO
		C	400		
0020D6		C	401	plat_set MVC from_cpm,ipl_from,4	; Tell splash screen we are booting from CPM.
		C+	401	scope	
0020D6	21 96 85	C+	401	ld hl,from_cpm	
0020D9	11 67 71	C+	401	ld de,ipl_from	
0020DC	010400	C+	401	ld bc,4	
0020DF	EDB0	C+	401	LDIR	
		C+	401	ENDMAC	MVC
		C	402		
		C	403	; If file was loaded, DE contained pointer to first. BC to 2nd file.	
		C	404		
0020E1	D1	C	405	pop de	; Recover pointer to 1st data file.
0020E2	C1	C	406	pop bc	; Recover pointer to 2nd data file.
		C	407		
0020E3	D5E1	C	408	ld hl,de	; No file loaded DE =0.
0020E5	AF	C	409	xor a	; Compare D to 0. 0=no file loaded.
0020E6	BA	C	410	cp d	; If no file loaded leave default storage map in place.
0020E7	28 1C	C	411	jr z,finish_init	; If B was 0 then file was <64k, cannot be <64k+1.
		C	412		
0020E9	110101	C	413	ld de,0101h	
0020EC	B7	C	414	or a	
0020ED	ED52	C	415	sbc hl,de	
0020EF	5B220101 01	C	416	ld.lil (010101h),hl	; Store offset from slice 1 to diskdisk file.
		C	417		
		C	418	; If file was loaded BC points to start of file. No file loaded B =0.	
		C	419		
0020F4	C5E1	C	420	ld hl,bc	
0020F6	AF	C	421	xor a	; Compare B to 0, it cannot be 0.
0020F7	B0	C	422	or B	; a zero if a file was loaded.
0020F8	28 0B	C	423	jr z,finish_init	; If B was 0 then file was less than 64k, cannot be <64k+1.
0020FA	110101	C	424	ld de,0101h	
0020FD	B7	C	425	or a	
0020FE	ED52	C	426	sbc hl,de	
002100	5B220401 01	C	427	ld.lil (010104h),hl	; Offset to drive 1 image file loaded by Bill@Circle-M.
		C	428		
		C	429	; Initialization of CPU complete, splash first welcome/copyright screen.	
		C	430		
002105		C	431	finish_init UART0.dtr.set	; Set DTR active tell other end we are powered on & ready.
		C+	431	scope	
002105	ED38C4	C+	431	in0 a,(UART0_MCTL)	; Get current register status.
002108	F601	C+	431	or 1	; Set bit 0 to 1.
00210A	ED39C4	C+	431	OUT0 (UART0_MCTL),A	; UART1 tell other end we are ready.

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - Minimum initialization for eZ80F91 &amp; eZ80F92. &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-eZ80_CPU.s
		C+	431	endmac	UART0.dtr.set
		C	432	UART1.dtr.set	; Set DTR active tell other end we are powered on & ready.
		C+	432	scope	
00210D	ED38D4	C+	432	in0 a,(UART1_MCTL)	; Get current register status.
002110	F601	C+	432	or 1	; Set bit 0 to 1.
002112	ED39D4	C+	432	OUT0 (UART1_MCTL),A	; UART1 tell other end we are ready.
		C+	432	endmac	UART1.dtr.set
002115	0E20	C	433	ld c,020h	; Get fill byte in accumulator.
002117	2100F8	C	434	ld hl,CRTBGN\$	
00211A	0608	C	435	ld b,8	; Fill 8 pages with spaces.
00211C	C5	C	436	more_fill push bc	
00211D	0600	C	437	ld b,0	; Setup for 256 chars.
00211F	71	C	438	fill_loop ld (hl),c	; Stuff a copy of fill byte.
002120	23	C	439	inc hl	
		C	440	LOOP fill_loop	
		C+	440	scope	
002121	10 FC	C+	440	djnz fill_loop	
		C+	440	endmac	LOOP
002123	C1	C	441	pop bc	
		C	442	LOOP more_fill	
		C+	442	scope	
002124	10 F6	C+	442	djnz more_fill	
		C+	442	endmac	LOOP
		C	443		
		C	444	MVC welcomemsg,CRTBGN\$,RDYMSGLEN	; Move welcome message to video RAM.
		C+	444	scope	
002126	21 52 77	C+	444	ld hl,welcomemsg	
002129	1100F8	C+	444	ld de,CRTBGN\$	
00212C	01 80 01	C+	444	ld bc,RDYMSGLEN	
00212F	EDB0	C+	444	LDIR	
		C+	444	ENDMAC	MVC
		C	445		
002131	C3 50 43	C	446	JP STARTBIOS	; Start BIOS.
		C	447		
		B	9		
		B	10		; Start TRS-OS.
		B	11		; Using TRS-OS, user starts TRSDOS using selected options - POST, RTC setup etc.
		B	12		; Turn control to TRSDOS/LS-DOS.
		B	13		
		B	14	org 4300h+80	; Protect data from sysgen residing @4300h.
		B	15		
		B	16		; Initialize BIOS.
		B	17		
		C	0	include "ipl-BIOS.s"	; Hardware initialization code for stack, b
		C	1	subtitle "< SYS0 IPL - Initialization code for BIOS, port hardware, stack, banks, etc >"	
		C	2	newpage	



\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - Initialization code for BIOS, port hardware, stack, banks, etc &gt;

PC	Object	I	Line	Source	..\\SYS0_IPL\\ipl-BIOS.s
		C	3		scope
		C	4		
004350	CD ED 6E	C	5	STARTBIOS	call scrolloff
004353	CD 85 12	C	6		call GETDATE
004356	CD B8 12	C	7		call GETTIME
		C	8		
		C	9		; Fill spare DCB area with 0's.
004359	AF	C	10		xor a
00435A	21 48 02	C	11		LD HL,S1DCB\$
00435D	77	C	12	ZERDCB	LD (HL),A ;Zero spare dcb area
00435E	2C	C	13		INC L
00435F	20 FC	C	14		JR NZ,ZERDCB
		C	15		
		C	16		; Setup high memory pointers.
004361	21FFFF	C	17		LD HL,0ffffh ; Set physical max memory to 64k.
004364	22 1C 00	C	18		LD (PHIGH\$),HL
		C	19		
		C	20		; This code is kept intact to notate need for bank initialization.
		C	21		; Initialize Bank Available Ram & Bank Used Ram. Check the BANKS.
		C	22		
004367	AF	C	23		XOR A
004368	320202	C	24		LD (LBANK\$),A ; Set logical bank #.
00436B	3EFE	C	25		LD A,0feh ; Init BAR\$ for bank 0-2.
00436D	320102	C	26		LD (BAR\$),A ; Load Bank Avail RAM.
004370	3EF8	C	27		ld a,0F8h
004372	320002	C	28		LD (BUR\$),A ; Load Bank Used RAM.
		C	29		
		C	30		; Initialize lowcore TIME\$ values.
		C	31		
		C	32		RTC.HOUR.get (TIME\$+2)
004375	ED38E2	C+	32		in0 a,(RTC_HRS) ;
004378	32 2F 00	C+	32		ld (TIME\$+2),a
		C+	32		ENDMAC
		C	33		RTC.MIN.get (TIME\$+1)
00437B	ED38E1	C+	33		in0 a,(RTC_MIN) ;
00437E	32 2E 00	C+	33		ld (TIME\$+1),a
		C+	33		ENDMAC
		C	34		RTC.SEC.get (TIME\$+0)
004381	ED38E0	C+	34		in0 a,(RTC_SEC) ;
004384	32 2D 00	C+	34		ld (TIME\$+0),a
		C+	34		ENDMAC
		C	35		PUMP DEVICE.console,CRTBGN\$ ; Pump out message to a fresh clear screen.
		C+	35		scope ; A tribute to Dave Shriner - Display Data Corp.
004387	2100F8	C+	35		LD HL,CRTBGN\$ ; Pump from this starting point until ETX.asc is found.
		C+	35		if DEVICE.console=0
00438A	7E	C+	35	__1150:	LD A,(HL) ; Get byte to send.
00438B	23	C+	35		INC HL ; pointer=pointer+1
		C+	35		; CP EOT.asc ; End Of Transmission (block)?
		C+	35		; jr Z,\$0a ; Return if finished.
00438C	FE03	C+	35		CP ETX.asc ; End Of Text?
00438E	28 16	C+	35		jr Z,__1150a ; Return if finished.
004390	FE0D	C+	35		CP CR.asc ; Was it a CR?
004392	CC 9C 43	C+	35		call z,__1151 ; 1st add a LF to stream.
004395	4F	C+	35		LD C,A ; Output byte in C.
004396	AF	C+	35		XOR A ; Set Z flag for put.
004397	CD D9 10	C+	35		CALL U0DVR ; Out the UART we go.
00439A	18 EE	C+	35		JR __1150
00439C	3E0A	C+	35	__1151:	ld a,LF.asc ; Get a LF.
00439E	4F	C+	35		LD C,A ; Output byte in C.
00439F	AF	C+	35		XOR A ; Set Z flag for put

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - Initialization code for BIOS, port hardware, stack, banks, etc &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-BIOS.s
0043A0	CD D9 10	C+	35	CALL	U0DVR ; Out the UART we go.
0043A3	3E0D	C+	35	ld	a,CR.asc ; Now put a CR back in stream.
0043A5	C9	C+	35	ret	; Return and print CR.
0043A6		C+	35	__1150a	endif
		C+	35	if	DEVICE.console=1
		C+	35	__1151:	LD A,(HL) ; Get byte to send.
		C+	35	INC	HL ; pointer=pointer+1
		C+	35	CP	EOT.asc ; End Of Transmission (block)?
		C+	35	;	
		C+	35	jr	Z,\$1a ; Return if finished.
		C+	35	CP	ETX.asc ; End Of Text?
		C+	35	jr	Z,__1151a ; Return if finished.
		C+	35	LD	C,A ; Output byte to C.
		C+	35	XOR	A ; Set Z flag for put.
		C+	35	CALL	U1DVR ; Out the UART we go.
		C+	35	JR	__1151
		C+	35	__1151a	equ \$
		C+	35	endif	
		C+	35	ENDMAC	PUMP
0043A6	3A FD 86	C	36	ld	a,(initialflg)
0043A9	B7	C	37	or	a
0043AA	20 1F	C	38	jr	nz,pastshawn
		C	39		
		C	40	PUMP	DEVICE.console,initialmsg
		C+	40	scope	; A tribute to Dave Shriner - Display Data Corp.
0043AC	21 D2 78	C+	40	LD	HL,initialmsg ; Pump from this starting point until ETX.asc is found.
		C+	40	if	DEVICE.console=0
0043AF	7E	C+	40	__1160:	LD A,(HL) ; Get byte to send.
0043B0	23	C+	40	INC	HL ; pointer=pointer+1
		C+	40	CP	EOT.asc ; End Of Transmission (block)?
		C+	40	;	
		C+	40	jr	Z,\$0a ; Return if finished.
0043B1	FE03	C+	40	CP	ETX.asc ; End Of Text?
0043B3	28 16	C+	40	jr	Z,__1160a ; Return if finished.
0043B5	FE0D	C+	40	CP	CR.asc ; Was it a CR?
0043B7	CC C1 43	C+	40	call	z,__1161 ; 1st add a LF to stream.
0043BA	4F	C+	40	LD	C,A ; Output byte in C.
0043BB	AF	C+	40	XOR	A ; Set Z flag for put.
0043BC	CD D9 10	C+	40	CALL	U0DVR ; Out the UART we go.
0043BF	18 EE	C+	40	JR	__1160
0043C1	3E0A	C+	40	__1161:	ld a,LF.asc ; Get a LF.
0043C3	4F	C+	40	LD	C,A ; Output byte in C.
0043C4	AF	C+	40	XOR	A ; Set Z flag for put
0043C5	CD D9 10	C+	40	CALL	U0DVR ; Out the UART we go.
0043C8	3E0D	C+	40	ld	a,CR.asc ; Now put a CR back in stream.
0043CA	C9	C+	40	ret	; Return and print CR.
0043CB		C+	40	__1160a	endif
		C+	40	if	DEVICE.console=1
		C+	40	__1161:	LD A,(HL) ; Get byte to send.
		C+	40	INC	HL ; pointer=pointer+1
		C+	40	CP	EOT.asc ; End Of Transmission (block)?
		C+	40	;	
		C+	40	jr	Z,\$1a ; Return if finished.
		C+	40	CP	ETX.asc ; End Of Text?
		C+	40	jr	Z,__1161a ; Return if finished.
		C+	40	LD	C,A ; Output byte to C.
		C+	40	XOR	A ; Set Z flag for put.
		C+	40	CALL	U1DVR ; Out the UART we go.
		C+	40	JR	__1161
		C+	40	__1161a	equ \$
		C+	40	endif	
		C+	40	ENDMAC	PUMP
		C	41		

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - Initialization code for BIOS, port hardware, stack, banks, etc &gt;

```

PC      Object      I  Line      Source ..\SYS0_IPL\ipl-BIOS.s
0043CB                                C    42      pastshawnPUMP  DEVICE.console,splashiplmode ; Report CPU registers initialized.
                                C+   42              scope                ; A tribute to Dave Shriner - Display Data Corp.
0043CB 21 2A 79          C+   42              LD      HL,splashiplmode ; Pump from this starting point until ETX.asc is found.
                                C+   42              if      DEVICE.console=0
0043CE 7E                C+   42      __1170:      LD      A,(HL)          ; Get byte to send.
0043CF 23                C+   42              INC      HL          ; pointer=pointer+1
                                C+   42              ;      CP      EOT.asc          ; End Of Transmission (block)?
                                C+   42              ;      jr      Z,$0a          ; Return if finished.
0043D0 FE03             C+   42              CP      ETX.asc          ; End Of Text?
0043D2 28 16            C+   42              jr      Z,__1170a        ; Return if finished.
0043D4 FE0D             C+   42              CP      CR.asc          ; Was it a CR?
0043D6 CC E0 43         C+   42              call   z,__1171        ; 1st add a LF to stream.
0043D9 4F               C+   42              LD      C,A          ; Output byte in C.
0043DA AF               C+   42              XOR      A          ; Set Z flag for put.
0043DB CD D9 10         C+   42              CALL   U0DVR          ; Out the UART we go.
0043DE 18 EE            C+   42              JR      __1170
0043E0 3E0A             C+   42      __1171:      ld      a,LF.asc          ; Get a LF.
0043E2 4F               C+   42              LD      C,A          ; Output byte in C.
0043E3 AF               C+   42              XOR      A          ; Set Z flag for put
0043E4 CD D9 10         C+   42              CALL   U0DVR          ; Out the UART we go.
0043E7 3E0D             C+   42              ld      a,CR.asc          ; Now put a CR back in stream.
0043E9 C9               C+   42              ret              ; Return and print CR.
0043EA                C+   42      __1170a      endif
                                C+   42              if      DEVICE.console=1
                                C+   42      __1171:      LD      A,(HL)          ; Get byte to send.
                                C+   42              INC      HL          ; pointer=pointer+1
                                C+   42              ;      CP      EOT.asc          ; End Of Transmission (block)?
                                C+   42              ;      jr      Z,$1a          ; Return if finished.
                                C+   42              CP      ETX.asc          ; End Of Text?
                                C+   42              jr      Z,__1171a        ; Return if finished.
                                C+   42              LD      C,A          ; Output byte to C.
                                C+   42              XOR      A          ; Set Z flag for put.
                                C+   42              CALL   U1DVR          ; Out the UART we go.
                                C+   42              JR      __1171
                                C+   42      __1171a      equ      $
                                C+   42              endif
                                C+   42              ENDMAC          PUMP
                                C    43
                                C    44      ; Get SLICE 0 @ 010000h. Slice 0 points to slices 1-9. Move slicetable in RAM for 9 slices (1-9).
                                C    45
0043EA 5B210000 01      C    46              LD.Lil    HL,010000h          ; HL now contains 24b address pointing to file info of RAM driv
0043EF 4911 26 87      C    47              LD.L    DE,slicebuf          ; Convert 16 bit destination address to 24 bit.
0043F3 49010001        C    48              LD.L    BC,256              ; Copy 256 bytes, HL contains 24b source & DE destination.
0043F7 49EDB0          C    49              LDIR.L          ; Copy now.
                                C    50              MVC slicebuf,slicetable,27 ; Move slice table to table in memory.
                                C+   50              scope
0043FA 21 26 87        C+   50              ld      hl,slicebuf
0043FD 11 0B 87        C+   50              ld      de,slicetable
004400 011B00          C+   50              ld      bc,27
004403 EDB0            C+   50              LDIR
                                C+   50              ENDMAC          MVC
                                C    51
                                C    52      ; Get SLICE 1 pointed to by SLICE 0.
                                C    53
004405 492A 0B 87      C    54              ld.l    hl,(slicetable)      ; HL now contains 24b address pointing to file info of RAM
004409 5B11 26 87 00   C    55              LD.lil    de,slicebuf          ; Convert 16 bit destination address to 24 bit.
00440E 49010001        C    56              LD.L    BC,256              ; Copy 256 Bytes, HL contains 24b source.
004412 49EDB0          C    57              LDIR.L          ; Copy now.
                                C    58
                                C    59      ; Slicebuf now contains SLICE 1.

```

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - Initialization code for BIOS, port hardware, stack, banks, etc &gt;

PC	Object	I	Line	Source	Comment
		C	60	; SLICE 1 first 3 bytes, is an offset to boot volume, VOLUME 0.	
		C	61		
		C	62	; Now we get volume 0 HEADER in volbuf.	
		C	63	; Using SLICE 1 offset to calculate physical address to VOLUME 0 (boot volume).	
		C	64		
004415	2A 27 87	C	65	ld hl,(slicebuf+1)	; Get offset to VOLUME 1.
004418	ED5B 0C 87	C	66	ld de,(slicetable+1)	; Get upper 16 bits of pointer to slice 1.
00441C	19	C	67	add hl,de	; offset = offset + base pointer of slice 1.
		C	68		
		C	69	; Using pointer to RAMDRIVE, copy first sector to volbuf.	
		C	70	; HL now points to slice 1.	
		C	71		
00441D	AF	C	72	xor A	; A = 0 for below code.
00441E	32 D3 86	C	73	LD (XXX),a	
004421	22 D4 86	C	74	ld (XXX+1),hl	; Store base of slice 1 + offset to volume 0.
		C	75		
		C	76	; But first store this pointer in driver for RAM drive.	
		C	77		
004424	23	C	78	inc hl	
004425	22 63 0E	C	79	ld (ramdrv\$),hl	; Store in 0 slot of RAM driver table of addresses.
		C	80		
		C	81	; Load volume 0 header/sector 0 into volbuf.	
		C	82		
004428	492A D3 86	C	83	LD.L HL,(XXX)	; Load HL with 24b address of RAM drive.
00442C	AF	C	84	xor a	
00442D	32 D5 86	C	85	LD (XXX+2),a	; Zero out upper 8 bits of 24b address.
004430	11 26 88	C	86	LD de,volbuf	; Convert 16 bit destination address to 24 bit.
004433	ED53 D3 86	C	87	LD (XXX),de	; Store 16 bit address of volbuf destination.
004437	49ED5B D3 86	C	88	LD.L DE,(XXX)	; DE contains 24b destination.
00443C	49010001	C	89	LD.L BC,256	; Copy 256 bytes, one sector, HL contains 24b source.
004440	49EDB0	C	90	LDIR.L	; HL contains pointer to RAM drive. Copy now.
		C	91		
		C	92	; Now test if first 8 bytes of buffer contain ASCII string 'DiskDISK'.	
		C	93	; If yes, this volume is in Misosys DiskDISK format.	
		C	94	; Else assume TRS-80 floppy disk format.	
		C	95		
004443	010800	C	96	ld bc,8	; Test a string length of 8 characters.
004446	21 AA 86	C	97	ld hl,string1\$	; Get pointer to string containing 'DiskDISK'
004449	11 26 88	C	98	ld de,volbuf	; Address of buffer to test.
00444C	1A	C	99	tstm0 ld a,(de)	; Get char pointed to by DE.
00444D	13	C	100	inc de	; Bump to next char.
00444E	EDA1	C	101	cpi	; Test and increment HL & DE while decrementing BC (char count).
004450	C2 97 6F	C	102	jp nz,mounterr0	; If they dont match we can stop now.
004453	EA 4C 44	C	103	jp v,tstm0	; If BC <> 0 after CPI then continue looping until it does.
		C	104		
		C	105	mov.BYT volbuf,boottyp\$,8	; Move boot volume name (DiskDisk) to menu.
		C+	105	scope	
		C+	105	; On entry, HL points to location containing bytes to move.	
		C+	105	; DE contains pointer to destination.	
		C+	105	; Length is maximum number of bytes to move (16 bits).	
		C+	105	; On exit.	
		C+	105	; BC=0	
		C+	105	; HL = Pointer next char to move.	
		C+	105	; DE = Points to next place in destination buffer to store a char.	
004456	21 26 88	C+	105	ld hl,volbuf	
004459	11 54 72	C+	105	ld de,boottyp\$	
00445C	010800	C+	105	ld bc,8	
00445F	EDB0	C+	105	LDIR	
		C+	105	ENDMAC	
004461		C	106	setup_dct0 MVC volbuf+11,DCT0\$+3,7	; Move DiskDisk DCT to drive 0 DCT.

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - Initialization code for BIOS, port hardware, stack, banks, etc &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-BIOS.s
		C+	106		scope
004461	21 31 88	C+	106		ld hl,volbuf+11
004464	11 73 04	C+	106		ld de,DCT0\$+3
004467	010700	C+	106		ld bc,7
00446A	EDB0	C+	106		LDIR
		C+	106		ENDMAC MVC
00446C	FD21 70 04	C	107		ld iy,DCT0\$ ; dct1.
004470	FDCB04C6	C	108		set 0,(iy+4) ; Tell world this is drive 0.
004474	FD5606	C	109		ld d,(iy+6) ; Get max cylinders.
004477	14	C	110		inc d ; Adjust +1.
004478	FD7E07	C	111		ld a,(iy+7) ; Get config, lower nibble contains sec per trk.
00447B	E63F	C	112		and 3Fh ; Get lower nibble containing sectors per trk.
00447D	3C	C	113		inc a
00447E	FDCB046E	C	114		bit 5,(IY+4) ; Sides = 2?
004482	28 02	C	115		jr z,pastdbl ; If so then we double sectors per track.
004484	CB27	C	116		sla a ; Sectors per track = sectors per track * 2.
004486	5F	C	117	pastdbl	ld e,a ; Setup to multiply track * sectors per track.
004487	ED5C	C	118		MLT DE ; we then multiply (# CYL * SEC per CYL) 16b result in DE.
004489	D5	C	119		push de ; Save a copy of MLT results.
00448A	13	C	120		inc de ; Bump one sector higher to adjust for diskdisk header.
00448B	01 5D 72	C	121		ld BC,bootsiz\$ ; Address to store ASCII result in string.
		C	122		
		C	123		; Convert DE to 24 bit by H=D, L=E, E=0 24 BIT--> E:HL.
		C	124		; Store ASCII representation of 24 bit value in main memu.
		C	125		; Convert to packed BCD.
		C	126		; Convert packed BCD to unpacked.
		C	127		; Convert unpacked to ASCII.
		C	128		; Trim leading 0's away.
		C	129		; Stuff all these results in main menu.
		C	130		
00448E	CD 41 70	C	131		call cvt_24_ascii
		C	132		
004491	D1	C	133		pop de ; Recover original MLT results.
004492	210200	C	134		ld hl,2 ; offset to VOLUME 1. Vol 0 length+diskdisk hdr sector VOLUME 0).
004495	19	C	135		add hl,de ; Add size of VOLUME 0 and this will be where VOL 1 starts.
004496	5B220401 01	C	136		ld.lil (010104h),hl ; Offset to volume 1 image.
00449B	22 2A 87	C	137		ld (slicebuf+4),hl ; Update table bc it has already been built.
		C	138		
		C	139		; Update current cylinder to 0.
		C	140		
00449E	AF	C	141		xor a ; Its way it is on real model 4.
00449F	32 75 04	C	142		ld (DCT0\$+5),a ; Update DCT current cylinder.
		C	143		
		C	144		; Now, get track 0, sector 0 of drive 0 image.
		C	145		
0044A2	21 26 88	C	146		LD HL,volbuf ; HL points to buffer of READ.
0044A5	1600	C	147		ld d,0 ; D=> Track 0.
0044A7	1E00	C	148		ld e,0 ; E=> Sector 0.
0044A9	0E00	C	149		ld c,0 ; C=> C contains drive #0.
0044AB	0609	C	150		ld b,9 ; B=> Function in register B.
0044AD	CD 58 0E	C	151		call FDGDVR ; Call driver.
		C	152		
		C	153		; Recover from sector 0, directory cylinder & store in DCT0\$.
		C	154		
0044B0	3A 28 88	C	155		ld a,(volbuf+2) ; Get pointer to directory.
0044B3	32 79 04	C	156		ld (DCT0\$+9),a ; Store directory cylinder in DCT.
		C	157		
		C	158		; Read directory sector GAT of drive 0, boot volume. Register A still has directory track.
		C	159		
0044B6	21 26 88	C	160		LD HL,volbuf ; HL points to buffer of READ.

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - Initialization code for BIOS, port hardware, stack, banks, etc &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-BIOS.s
0044B9	57	C	161	ld	d,a ; D=> A still contains directory track from last operation.
0044BA	1E00	C	162	ld	e,0 ; E=> sector desired.
0044BC	0E00	C	163	ld	c,0 ; C=> C contains drive #.
0044BE	0609	C	164	ld	b,9 ; B=> Function in register B.
0044C0	CD 58 0E	C	165	call	FDCDVR ; Call driver.
		C	166		
		C	167	MVC	volbuf+0D0h,bootvol\$,8 ; Copy volume name.
		C+	167	scope	
0044C3	21 F6 88	C+	167	ld	hl,volbuf+0D0h
0044C6	11 38 72	C+	167	ld	de,bootvol\$
0044C9	010800	C+	167	ld	bc,8
0044CC	EDB0	C+	167	LDIR	
		C+	167	ENDMAC	MVC
		C	168	MVC	volbuf+0D8h,bootvol\$+9,8 ; Copy volume creation time.
		C+	168	scope	
0044CE	21 FE 88	C+	168	ld	hl,volbuf+0D8h
0044D1	11 41 72	C+	168	ld	de,bootvol\$+9
0044D4	010800	C+	168	ld	bc,8
0044D7	EDB0	C+	168	LDIR	
		C+	168	ENDMAC	MVC
		C	169		
0044D9	3A F3 88	C	170	ld	a,(volbuf+0CDh) ; Get drive configuration.
0044DC	CB7F	C	171	bit	7,a ; Test if data disk?
0044DE	C4 DB 6F	C	172	call	nz,cant_ipl_data ; Report not systems disk.
		C	173		
0044E1	3A F1 88	C	174	ld	a,(volbuf+0cbh) ; Get version that formatted this volume.
0044E4	E6F0	C	175	and	0F0h ; Strip of right hex digit.
0044E6	FE60	C	176	cp	060h ; Make sure this volume created by a DOS 6.x version.
0044E8	C2 97 6F	C	177	jp	nz,mounterr0 ; If this volume not formatted by DOS 6.x we cannot use it.
		C	178		
0044EB	FDCB04E6	C	179	set	4,(iy+4) ; Tell world this is an alien controller.
		C	180		
0044EF	3A F2 88	C	181	ld	a,(volbuf+0cch) ; Get logical cylinders in excess of 35.
0044F2	C623	C	182	add	35
0044F4	DA 97 6F	C	183	jp	c,mounterr0 ; Cannot overflow.
0044F7	32 77 04	C	184	ld	(DCT0\$+7),a ; Make it highest sector per track.
		C	185		
0044FA	3A F3 88	C	186	ld	a,(volbuf+0CDh) ; Get drive configuration.
0044FD	CB7F	C	187	bit	7,a ; Test if data disk?
0044FF	C2 97 6F	C	188	jp	nz,mounterr0 ; Report not systems disk.
		C	189		
004502	010924	C	190	ld	bc,2409h ; Setup for single density.
004505	FD7107	C	191	ld	(iy+7),c
004508	FD7008	C	192	ld	(iy+8),b
00450B	CB77	C	193	bit	6,a ; Test if single or double density. 1=DDEN.
00450D	28 09	C	194	jr	z,tstsides
00450F	011145	C	195	ld	bc,4511h
004512	FD7107	C	196	ld	(iy+7),c
004515	FD7008	C	197	ld	(iy+8),b ; Tell OS this volume is formatted in single density.
		C	198		
004518	FDCB04EE	C	199	tstsides set	5,(IY+4) ; Set sides to 2.
00451C	CB6F	C	200	bit	5,a ; Test number sides when formatted.
00451E	20 04	C	201	jr	nz,getauto ; If <>0 then leave sides = 2.
004520	FDCB04AE	C	202	res	5,(IY+4) ; Make sides = 1.
		C	203		
		C	204		; Get if any AUTO command & status of SYSGEN flag. Information contained in sector 2 of track 0, dr
		C	205		
004524	FD21 70 04	C	206	getauto ld	iy,DCT0\$ ; dct0.
004528	21 26 88	C	207	LD	HL,volbuf ; HL points to buffer of READ.
00452B	1600	C	208	ld	d,0 ; D=> track desired.

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - Initialization code for BIOS, port hardware, stack, banks, etc &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-BIOS.s
00452D	1E02	C	209	ld	e,2 ; E=> sector desired.
00452F	0E00	C	210	ld	c,0 ; C=> C contains drive #.
004531	0609	C	211	ld	b,9 ; B=> Function in register B.
004533	CD 58 0E	C	212	call	FDCDVR ; Call driver.
		C	213		
		C	214	MVC	spaceout\$,autoflg\$+2,69 ; Make sure string is full of spaces.
		C+	214	scope	
004536	21 76 89	C+	214	ld	hl,spaceout\$
004539	11 BF 71	C+	214	ld	de,autoflg\$+2
00453C	014500	C+	214	ld	bc,69
00453F	EDB0	C+	214	LDIR	
		C+	214	ENDMAC	MVC
		C	215	mov.CHR	volbuf+20h,autoflg\$+2,71,CR.asc ; Move auto command if any to menu.
		C+	215	scope	; bytes or until stop byte is encountered.
		C+	215		; On entry, HL points to location containing bytes to move.
		C+	215		; Destination contains pointer of destination.
		C+	215		; Length is maximum number of bytes to move (16 bits).
		C+	215		; Move terminates before max number bytes if byte=stop value.
		C+	215		; On exit.
		C+	215		; BC=0 or BC= # chars not moved bc of termination character.
		C+	215		; HL = Pointer next char to move.
		C+	215		; DE = Points to next place in destination buffer to store a char.
		C+	215		; A = Termination char.
004541	21 46 88	C+	215	ld	hl,volbuf+20h ; Where we are moving from.
004544	11 BF 71	C+	215	ld	de,autoflg\$+2 ; Get destination in memory.
004547	014700	C+	215	ld	bc,71 ; Get how many characters to move.
00454A	1B	C+	215	dec	de ; Start back one byte.
00454B	3E0D	C+	215	__1240: ld	a,CR.asc ; What character stops moving operation.
00454D	EDA1	C+	215	cpi	; Compaere (HL) with A register. BC=BC-1
		C+	215	IF_EQ_BRANCH	__1241 ; Did not get termination character? Keep looping until done.
		C+	215	scope	
00454F	28 08	C+	215	jr	z,__1241
		C+	215	endmac	IF_EQ_BRANCH
004551	2B	C+	215	dec	hl
004552	13	C+	215	inc	de
004553	7E	C+	215	ld	a,(hl) ; Get character.
004554	12	C+	215	ld	(de),a ; Store in new destination.
004555	23	C+	215	inc	hl
004556	EA 4B 45	C+	215	jp	v,__1240 ; Keep looping until done (BC=0).
004559	13	C+	215	__1241: inc	de
00455A		C+	215	__1242: endmac	mov.CHR
		C	216	MVC	volbuf+20h,INBUF\$,79 ; Copy auto command to auto buffer.
		C+	216	scope	
00455A	21 46 88	C+	216	ld	hl,volbuf+20h
00455D	11 20 04	C+	216	ld	de,INBUF\$
004560	014F00	C+	216	ld	bc,79
004563	EDB0	C+	216	LDIR	
		C+	216	ENDMAC	MVC
004565	3E0D	C	217	ld	a,CR.asc
		C	218		
004567	3A 27 88	C	219	ld	a,(volbuf+1) ; Get sysgen flag.
00456A	32 01 04	C	220	ld	(ZERO\$),a ; Store it in flag.
00456D	B7	C	221	or	A ; Sysgen off or on?
00456E	20 0B	C	222	jr	nz,nosysgenmsg ; SYSGEN = N so do not put file name in menu.
		C	223	MVC	CFGFCB\$,SYSGN\$,10 ; Move name of sysgen file to menu.
		C+	223	scope	
004570	21 E0 00	C+	223	ld	hl,CFGFCB\$
004573	11 A9 71	C+	223	ld	de,SYSGN\$
004576	010A00	C+	223	ld	bc,10
004579	EDB0	C+	223	LDIR	

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - Initialization code for BIOS, port hardware, stack, banks, etc &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-BIOS.s
		C+	223	ENDMAC	MVC
		C	224		
00457B		C	225	nosysgenmsg	CKDRIVE 0,mounterr0 ; See if TRSDOS finds formatted media it can read. If not,
00457B	0E00	C+	225	ld	c,0
		C+	225	SVC	svc_ckdrv
00457D	3E21	C+	225	ld	a,svc_ckdrv
00457F	EF	C+	225	restart28	
		C+	225	ENDMAC	SVC
004580	C4 97 6F	C+	225	call	nz,mounterr0
		C+	225	endmac	CKDRIVE
		C	226		
		C	227		; slicebuf now contains SLICE 1. Bytes 3,4,5 is an offset to data VOLUME 1 (TRS-80 drive 1).
		C	228		; Get sector 0 of volume 1 in volbuf.
		C	229		
		C	230		; Using SLICE 1 offset to VOLUME 1, calculate physical address to ram drive.
		C	231		
004583	ED5B 0C 87	C	232	ld	de,(slicetable+1) ; Get upper 16 bits of pointer to slice 1.
004587	2A 2A 87	C	233	ld	hl,(slicebuf+4) ; Get offset to VOLUME 1.
00458A	19	C	234	add	hl,de ; offset = offset + base pointer of slice 1.
		C	235		
		C	236		; Using pointer to RAMDRIVE, copy first sector to volbuf.
		C	237		
00458B	AF	C	238	xor	A ; A = 0 for below code.
00458C	32 D3 86	C	239	LD	(XXX),a ; Zero lower 8 bits.
00458F	22 D4 86	C	240	ld	(XXX+1),hl ; Store upper 16 bit pointer to VOL1.
		C	241		
		C	242		; But first store this pointer in driver for RAM drive.
		C	243		
004592	23	C	244	inc	hl ; Advance pointer one page to actual RAMDRIVE contents.
004593	22 65 0E	C	245	ld	(ramdrv\$+2),hl ; Store in slot 1 of RAM driver table of addresses.
		C	246		
		C	247		; Now copy sector 0 of a drive image or diskdisk header to volbuf (either way, first record of volu
		C	248		
004596	492A D3 86	C	249	LD.L	HL,(XXX) ; Load HL with 24b address of RAM drive.
00459A	11 26 88	C	250	LD	de,volbuf ; Convert 16 bit destination address to 24 bit.
00459D	32 D5 86	C	251	LD	(XXX+2),a ; Zero out upper 8 bits of 24b address.
0045A0	ED53 D3 86	C	252	LD	(XXX),de ; Store address lower 16bits.
0045A4	49ED5B D3 86	C	253	LD.L	DE,(XXX) ; DE=24b destination address.
0045A9	49010001	C	254	LD.L	BC,256 ; Copy 256 Bytes, HL contains 24b source.
0045AD	49EDB0	C	255	LDIR.L	; Copy now.
		C	256		
0045B0	FD21 7A 04	C	257	ld	iy,DCT1\$
0045B4	FDCB04CE	C	258	set	1,(iy+4) ; Set 2nd physical drive (drive 1).
		C	259		
		C	260		; We test string1\$ which contains DiskDISK string against volbuff contents.
		C	261		
0045B8	010800	C	262	ld	bc,8 ; Test a string length of 8 characters.
0045BB	21 AA 86	C	263	ld	hl,string1\$ ; Get pointer to string containing 'DiskDISK'
0045BE	11 26 88	C	264	ld	de,volbuf ; Address of buffer to test.
0045C1	1A	C	265	tstmore1 ld	a,(de) ; Get char pointed to by DE.
0045C2	13	C	266	inc	de ; Bump to next char.
0045C3	EDA1	C	267	cpi	; Test and increment HL & DE while decrementing BC (char count).
0045C5	20 05	C	268	jr	nz,notdskdsk ; If they dont match we can stop now.
0045C7	EA C1 45	C	269	jp	v,tstmore1 ; If BC <> 0 after CPI then continue looping until it does.
0045CA	18 42	C	270	jr	dskdsk ; Finished testing, all match. Pass control to match code.
		C	271		
		C	272		; Arriving here we are not mounting a DiskDisk volume.
		C	273		; Change datatyp\$ to JV1
		C	274		; Recover pointer to RAM drive image and change from DD to JV1 by changing pointer to beginning. Be
		C	275		; JV1 image starts at xxxxxxh DiskDisk actual drive image starts at xxxxxxh+0100h.



\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - Initialization code for BIOS, port hardware, stack, banks, etc &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-BIOS.s
		C	276		
		C	277		; Adjust pointer for RAMDRIVE since this is not DiskDISK in Volume 1.
		C	278		
0045CC		C	279	notdskdskMVC	jv1\$,datatyp\$,8 ; Change data type to JV1.
		C+	279	scope	
0045CC 21 B2 86		C+	279	ld	hl,jv1\$
0045CF 11 94 72		C+	279	ld	de,datatyp\$
0045D2 010800		C+	279	ld	bc,8
0045D5 EDB0		C+	279	LDIR	
		C+	279	ENDMAC	MVC
		C	280		
0045D7 3E00		C	281	LD	A,0 ; Get Z80 NOP opcode.
0045D9 32 00 47		C	282	ld	(not_DD),a ; Change inc de to nop.
		C	283		
0045DC 2A 65 0E		C	284	ld	hl,(ramdrv\$+2) ; Slot for this Volume of RAM driver table of addresses.
0045DF 2B		C	285	dec	hl ; Pointer = pointer - one sector.
0045E0 22 65 0E		C	286	ld	(ramdrv\$+2),hl ; Store adjusted pointer back.
		C	287		
0045E3 AF		C	288	xor	a
0045E4 32 7F 04		C	289	ld	(DCT1\$+5),a ; Set track position of head.
		C	290		
		C	291		; Now read track=0 sector=0 of drive=1 & recover pointer to directory cylinder.
		C	292		
0045E7 21 26 88		C	293	LD	HL,volbuf ; HL points to buffer of READ.
0045EA 1600		C	294	ld	d,0 ; D=> Track 0.
0045EC 1E00		C	295	ld	e,0 ; E=> Sector 0.
0045EE 0E01		C	296	ld	c,1 ; C=> C contains drive #1.
0045F0 0609		C	297	ld	b,9 ; B=> Function in register B.
0045F2 CD 58 0E		C	298	call	FDCDVR ; Call driver.
		C	299		
0045F5 3A 28 88		C	300	ld	a,(volbuf+2) ; Sector 0 of a drive image is on volbuf.
0045F8 32 83 04		C	301	ld	(DCT1\$+9),a ; Point DCT to true directory cylinder.
		C	302		
0045FB FDCB04CE		C	303	set	1,(iy+4) ; Tell world this is drive 1.
0045FF FDCB04E6		C	304	set	4,(iy+4) ; Tell world this is an alien controller.
		C	305		
		C	306		; Set density for single and make first attempt @finding directory.
		C	307		
004603 010924		C	308	ld	bc,2409h ; Setup for single density.
004606 FD7107		C	309	ld	(iy+7),c
004609 FD7008		C	310	ld	(iy+8),b ; Tell OS this volume is formatted in single density.
		C	311		
00460C 18 40		C	312	jr	get1dir
		C	313		
		C	314		; Arriving here it verified we are mounting a DiskDISK for volume 1.
		C	315		; Move DCT from DD header to drive 1 DCT.
		C	316		; Move file type from volbuf to datatyp\$ in menu.
		C	317		
		C	318		; Pointers built in storage driver already point to correct place for diskdisk.
		C	319		; DiskDISK header still resides in volbuf.
		C	320		
		C	321		; Mounted drive has been determined to be DISKDisk.
		C	322		; Sector 0 of DISKDisk file is still in volbuf.
		C	323		; Recover DCT for this drive and copy to system DCT1\$.
		C	324		; Also set main menu to show DISKDisk type for this volume.
		C	325		
00460E		C	326	dskdsk	MVC volbuf,datatyp\$,8 ; Copy name of DiskDISK volume to menu.
		C+	326	scope	
00460E 21 26 88		C+	326	ld	hl,volbuf
004611 11 94 72		C+	326	ld	de,datatyp\$

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - Initialization code for BIOS, port hardware, stack, banks, etc &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-BIOS.s
004614	010800	C+	326	ld	bc,8
004617	EDB0	C+	326	LDIR	
		C+	326	ENDMAC	MVC
		C	327	MVC	volbuf+11,DCT1\$+3,7 ; Recover DCT settings for this DiskDISK volume to DCT 1.
		C+	327	scope	
004619	21 31 88	C+	327	ld	hl,volbuf+11
00461C	11 7D 04	C+	327	ld	de,DCT1\$+3
00461F	010700	C+	327	ld	bc,7
004622	EDB0	C+	327	LDIR	
		C+	327	ENDMAC	MVC
		C	328		
004624	3A 7E 04	C	329	ld	a,(DCT1\$+4) ; Drive 1.
004627	E6F0	C	330	and	0F0h ; Strip out drive select bits.
004629	CBC7	C	331	set	0,a ; Make this drive #1.
00462B	CBE7	C	332	set	4,a ; Tell world this is an alien controller.
00462D	CBF7	C	333	set	6,a ; Capable of double density.
00462F	32 7E 04	C	334	ld	(DCT1\$+4),a
		C	335		
004632	AF	C	336	xor	a ; Update DCT, its way it is on real model 4.
004633	32 7F 04	C	337	ld	(DCT1\$+5),a ; Cylinder=0.
		C	338		
		C	339		; Now read track=0 sector=0 of drive=1 & recover pointer to directory cylinder.
		C	340		
004636	FD21 7A 04	C	341	ld	iy,DCT1\$ ; dct1.
00463A	21 26 88	C	342	LD	HL,volbuf ; HL points to buffer of READ.
00463D	1600	C	343	ld	d,0 ; D=> Track 0.
00463F	1E00	C	344	ld	e,0 ; E=> Sector 0.
004641	0E01	C	345	ld	c,1 ; C=> C contains drive #1.
004643	0609	C	346	ld	b,9 ; B=> Function in register B.
004645	CD 58 0E	C	347	call	FDCDVR ; Call driver.
		C	348		
		C	349		; Get pointer to directory and place in DCT1 directory pointer.
		C	350		
004648	3A 28 88	C	351	ld	a,(volbuf+2) ; Sector 0 of a drive image is on volbuf.
00464B	32 83 04	C	352	ld	(DCT1\$+9),a ; Point DCT to true directory cylinder.
		C	353		
		C	354		; Now read directory as pointed to by recovered directory cylinder pointer.
		C	355		
		C	356		; Now read 1st sector of directory.
		C	357		
00464E	FD21 7A 04	C	358	get1dir	ld iy,DCT1\$ ; dct1.
004652	3A 83 04	C	359	ld	a,(DCT1\$+9)
004655	21 26 88	C	360	LD	HL,volbuf ; HL points to buffer of READ.
004658	57	C	361	ld	d,a ; D=> Track, A contains directory cylinder.
004659	1E00	C	362	ld	e,0 ; E=> Sector 0.
00465B	0E01	C	363	ld	c,1 ; C=> C contains drive #1.
00465D	0609	C	364	ld	b,9 ; B=> Function in register B.
00465F	CD 58 0E	C	365	call	FDCDVR ; Call driver.
		C	366		
		C	367		; Make sure driver reported error #6.
		C	368		; We are reading from directory thus driver should have generated error #6.
		C	369		; Error #6 is normal, no error is abnormal.
		C	370		
004662	28 FE	C	371	jr	z,\$ ; No error 6 means it was not a system sector.
004664	FE06	C	372	cp	6 ; Error on reading system sector.
004666	C4 BA 6F	C	373	call	nz,nodirectort1
		C	374		
		C	375		; volbuf now contains GAT,--- first sector of directory.
		C	376		
004669	3A F6 88	C	377	ld	a,(volbuf+0d0h) ; Get 1st char of volume name.

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - Initialization code for BIOS, port hardware, stack, banks, etc &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-BIOS.s
00466C	B7	C	378	or	a ; If filename starts with 00h then its bum steer.
00466D	CA 80 46	C	379	jp	z,switch_den ; Switch density then try getting again..
004670	3A F1 88	C	380	ld	a,(volbuf+0cbh) ; Get version that formatted this volume.
004673	E6F0	C	381	and	0F0h ; Strip of right hex digit.
004675	FE60	C	382	cp	060h ; Make sure this volume created by a DOS 6.x version.
004677	28 40	C	383	jr	z,setup_dct1 ; If this volume formatted by DOS 6.x skip to setting up DCT
004679	FE50	C	384	cp	050h ; If not formatted by LSDOS 6.x was it 5.x?
00467B	C4 FC 6F	C	385	call	nz,not56fmt ; Report no system directory.
00467E	18 39	C	386	jr	setup_dct1
		C	387		
004680	011145	C	388	switch_den	ld bc,4511h ; Values needed to set double density.
004683	FD7107	C	389	ld	(iy+7),c
004686	FD7008	C	390	ld	(iy+8),b ; Tell OS this volume is formatted in double density.
		C	391		
		C	392		; Try getting first sector of directory again using double density.
		C	393		
004689	FD21 7A 04	C	394	ld	iy,DCT1\$ ; dct1.
00468D	3A 83 04	C	395	ld	a,(DCT1\$+9)
004690	21 26 88	C	396	LD	HL,volbuf ; HL points to buffer of READ.
004693	57	C	397	ld	d,a ; D=> Track, A contains directory cylinder.
004694	1E00	C	398	ld	e,0 ; E=> Sector 0.
004696	0E01	C	399	ld	c,1 ; C=> C contains drive #1.
004698	0609	C	400	ld	b,9 ; B=> Function in register B.
00469A	CD 58 0E	C	401	call	FDCDVR ; Call driver.
		C	402		
		C	403		; Make sure driver reported error #6.
		C	404		; We are reading from directory thus driver should have generated error #6.
		C	405		; Error #6 is normal, no error is abnormal.
		C	406		
00469D	28 FE	C	407	jr	z,\$ ; No error 6 means it was not a system sector.
00469F	FE06	C	408	cp	6 ; Error on reading system sector.
0046A1	C4 BA 6F	C	409	call	nz,nodirectort1
		C	410		
		C	411		; volbuf now contains GAT,--- first sector of directory using double density settings.
		C	412		; Test and see if we found true directory.
		C	413		
0046A4	3A F6 88	C	414	ld	a,(volbuf+0d0h) ; Get 1st char of volume name.
0046A7	B7	C	415	or	a ; If filename starts with 00h then its bum steer.
0046A8	CC BA 6F	C	416	call	z,nodirectort1 ; Switch density then try getting again.
		C	417		
0046AB	3A F1 88	C	418	ld	a,(volbuf+0cbh) ; Get version that formatted this volume.
0046AE	E6F0	C	419	and	0F0h ; Strip of right hex digit.
0046B0	FE60	C	420	cp	060h ; Make sure this volume created by a DOS 6.x version.
0046B2	28 05	C	421	jr	z,setup_dct1 ; This was formatted by LSDOS 6.x
0046B4	FE50	C	422	cp	050h ; If not formatted by LSDOS 6.x was it 5.x?
0046B6	C4 FC 6F	C	423	call	nz,not56fmt ; Report if not formatted by DOS 5 or 6.
		C	424		
0046B9	FD21 7A 04	C	425	setup_dct1	ld iy,DCT1\$ ; dct1.
0046BD	FDCB0486	C	426	res	0,(iy+4) ; Not drive 0.
0046C1	FDCB04CE	C	427	set	1,(iy+4) ; Tell world this is drive 1.
0046C5	FDCB04E6	C	428	set	4,(iy+4) ; Tell world this is an alien controller.
		C	429		
0046C9	3A F2 88	C	430	ld	a,(volbuf+0cch) ; Get logical cylinders in excess of 35.
0046CC	C623	C	431	add	35 ; Add 35 to get true cylinder count.
0046CE	F5	C	432	push	af
0046CF	DC B8 6F	C	433	call	c,mounterr1 ; Cannot overflow.
0046D2	F1	C	434	pop	af
0046D3	3D	C	435	dec	a ; max_cyl=max_cyl - 1.
0046D4	F5	C	436	push	af
0046D5	DC B8 6F	C	437	call	c,mounterr1 ; Cannot overflow.

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - Initialization code for BIOS, port hardware, stack, banks, etc &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-BIOS.s
0046D8	F1	C	438	pop	af
0046D9	FD7706	C	439	ld	(iy+6),a ; Make it highest sector per track.
		C	440		
0046DC	3A F3 88	C	441	ld	a,(volbuf+0cdh) ; Get configuration byte.
0046DF	FDCB04AE	C	442	res	5,(IY+4) ; Set sides to 1.
0046E3	CB6F	C	443	bit	5,a ; Test number sides when formatted.
0046E5	28 04	C	444	jr	z,v0_mounted ; If <>0 then leave sides = 2.
0046E7	FDCB04EE	C	445	set	5,(IY+4) ; Make sides = 2.
		C	446		
0046EB	FD5606	C	447	v0_mounted ld	d,(iy+6) ; Get max cylinders.
0046EE	14	C	448	inc	d ; Adjust +1.
0046EF	FD7E07	C	449	ld	a,(iy+7) ; Get config, lower nibble contains sec per trk.
0046F2	E63F	C	450	and	3Fh ; Get lower nibble containing sectors per trk.
0046F4	3C	C	451	inc	a
0046F5	FDCB046E	C	452	bit	5,(IY+4) ; Sides = 2?
0046F9	28 02	C	453	jr	z,notdbl ; If so then we double sectors per track.
0046FB	CB27	C	454	sla	a ; Sectors per track = sectors per track * 2.
0046FD	5F	C	455	notdbl ld	e,a ; Setup to multiply track * sectors per track.
0046FE	ED5C	C	456	MLT	DE ; we then multiply (# CYL * SEC per CYL) 16b result in DE.
		C	457		
		C	458		; Following instruction changes to NOP if this is not a diskDISK image.
		C	459		
004700	13	C	460	not_DD inc	de ; Bump one sector higher to adjust for diskdisk header.
		C	461		
		C	462		
		C	463		
		C	464		; Stuff Volume 1 size in main menu.
		C	465		; Convert DE to 24 bit by H=D, L=E, E=0 24 BIT--> E:HL.
		C	466		; Store ASCII representation of 24 bit value in main memu.
		C	467		; Convert to packed BCD.
		C	468		; Convert packed BCD to unpacked.
		C	469		; Convert unpacked to ASCII.
		C	470		; Trim leading 0's away.
		C	471		; Stuff all these results in main menu.
		C	472		
004701	D5	C	473	push	de ; Save a copy of MLT results.
004702	01 9D 72	C	474	ld	BC,datasiz\$ ; Address to store ASCII result in string.
		C	475		
004705	CD 41 70	C	476	call	cvt_24_ascii ; Convert 24 bit value to ASCII & stuff in main menu.
		C	477		
		C	478		; Update slice 1 offset for volume 2.
		C	479		
004708	5B2A0401 01	C	480	ld.lil	hl,(010104h) ; Recover offset to vol1.
00470D	D1	C	481	pop	de ; Recover volume 1 length.
00470E	13	C	482	inc	de ; Bump one page forward to account for DiskDISK header.
00470F	19	C	483	add	hl,de ; New sum. offset to vol1+vol1 length+256 byte DD header.
004710	5B220801 01	C	484	ld.lil	(010108h),hl ; Offset to volume 2 image stored in slice table 1.
		C	485		
		C	486	MVC	volbuf+0D0h,datavol\$,8 ; Copy volume name.
		C+	486	scope	
004715	21 F6 88	C+	486	ld	hl,volbuf+0D0h
004718	11 78 72	C+	486	ld	de,datavol\$
00471B	010800	C+	486	ld	bc,8
00471E	EDB0	C+	486	LDIR	
		C+	486	ENDMAC	MVC
		C	487	MVC	volbuf+0D8h,datavol\$+9,8 ; Copy volume creation date.
		C+	487	scope	
004720	21 FE 88	C+	487	ld	hl,volbuf+0D8h
004723	11 81 72	C+	487	ld	de,datavol\$+9
004726	010800	C+	487	ld	bc,8

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - Initialization code for BIOS, port hardware, stack, banks, etc &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-BIOS.s
004729	EDB0	C+	487	LDIR	
		C+	487	ENDMAC	MVC
		C	488		
00472B	FD21 70 04	C	489	ld	iy,DCT0\$
		C	490	; set	7,(iy+3) ; WP vol 0. Force system/jcl to drive 1.
		C	491		
		C	492	; Last call for alcohol. Either we have 2 ready volumes or we dont.	
		C	493		
		C	494	CKDRIVE	0,mounterr0 ; Volume 0, u ready Freddie? This party not starting if you not
00472F	0E00	C+	494	ld	c,0
		C+	494	SVC	svc_ckdrv
004731	3E21	C+	494	ld	a,svc_ckdrv
004733	EF	C+	494	restart28	
		C+	494	ENDMAC	SVC
004734	C4 97 6F	C+	494	call	nz,mounterr0
		C+	494	endmac	CKDRIVE
		C	495	CKDRIVE 1,mounterr1	; Last test, drive 1 <> avail? Tell operator.....we can have a
004737	0E01	C+	495	ld	c,1
		C+	495	SVC	svc_ckdrv
004739	3E21	C+	495	ld	a,svc_ckdrv
00473B	EF	C+	495	restart28	
		C+	495	ENDMAC	SVC
00473C	C4 B8 6F	C+	495	call	nz,mounterr1
		C+	495	endmac	CKDRIVE
		C	496		
		C	497	; Continue IPL all drives mounted and accounted for.	
		C	498		
00473F	110802	C	499	flushkbd LD	DE,KIDCB\$ ; Flush keyboard, type,init ptrs.
004742	3E03	C	500	LD	A,3
004744	CD 23 06	C	501	CALL	zCTL
		C	502		
		C	503	; Test if server for TRSNET connected.	
		C	504		
		C	505	GOSUB ping_server+3	; Enter ping routine ahead of scroll off call instruction.
		C+	505	scope	
004747	CD 29 5A	C+	505	CALL ping_server+3	
		C+	505	endmac	GOSUB
		C	506		
		C	507	; Delay a bit and allow copyright splash to be read.	
		C	508		
00474A	CD 30 6E	C	509	call	cpu_dly ; Delay a bit.
00474D	21 8D F7	C	510	ld	hl,reboot ; Change RST00 vector to high memory routine.
004750	22 01 00	C	511	ld	(zRST00+1),hl ; Stuff that baby in there.
		C	512		
		C	513	GOTO	DOSINIT ; Off to see the wizard the wonderful wizard of TRSDOS!
		C+	513	scope	
004753	C3 56 47	C+	513	jp	DOSINIT
		C+	513	endmac	GOTO
		C	514		
		B	19		
		B	20	; IPL DOS contains main loop of TRS-OS startup. Main menu & input loop.	
		B	21		
		B	22	; Initialize DOS.	
		B	23		
		C	0	include "ipl-DOS.s"	; DOS initializaton, SYSGEN, interrupts, we
		C	1	subtitle	"< SYS0 IPL - DOS-initialization code & welcome message >"
		C	2	newpage	

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source
		C	3	scope
		C	4	
		C	5	; ORG zspace.LIBRARY\$
		C	6	
	00004756	C	7	DOSINIT: EQU \$
		C	8	
		C	9	; Get <SYSGEN> status and update menu. This menu item is then used as a flag in IPL.
		C	10	
004756	3E79	C	11	ld a, 'y'
004758	32 A9 71	C	12	ld (SYSGN\$), a
00475B	3A 01 04	C	13	ld a, (ZERO\$)
00475E	B7	C	14	or a ; Test if 0?
		C	15	IF_EQ_BRANCH CK_BRK
		C+	15	scope
00475F	28 05	C+	15	jr z, CK_BRK
		C+	15	endmac IF_EQ_BRANCH
004761	3E6E	C	16	ld a, 'n'
004763	32 A9 71	C	17	ld (SYSGN\$), a
		C	18	
		C	19	; Check is BREAK key disabled.
		C	20	
004766	3E79	C	21	CK_BRK ld a, 'y'
004768	32 A0 71	C	22	ld (brkflg\$), a
00476B	21 20 04	C	23	LD HL, INBUF\$
00476E	7E	C	24	LD A, (HL) ; Pt to 1st byte of AUTO
00476F	FE2A	C	25	CP '*' ; BREAK disable?
		C	26	IF_NE_BRANCH CKAUTO ; Go if BREAK enabled.
		C+	26	scope
004771	20 0E	C+	26	jr nz, CKAUTO
		C+	26	endmac IF_NE_BRANCH
004773	3E6E	C	27	ld a, 'n'
004775	32 A0 71	C	28	ld (brkflg\$), a
004778	3A 7C 00	C	29	ld a, (zSFLAG\$) ; Get systems flag.
00477B	F610	C	30	or 00010000b ; Disable break key.
00477D	32 7C 00	C	31	ld (zSFLAG\$), a ; Store it back.
004780	23	C	32	INC HL
		C	33	
		C	34	; Check on any AUTO command. ;
		C	35	
004781	3E79	C	36	CKAUTO ld a, 'y'
004783	32 BD 71	C	37	ld (autoflg\$), a
004786	7E	C	38	LD A, (HL) ; Any AUTO command?
004787	FE0D	C	39	CP CR.asc ; None if equal.
		C	40	IF_NE_BRANCH BERTDATE ; No AUTO command, skip setting flag to 'y'.
		C+	40	scope
004789	20 1B	C+	40	jr nz, BERTDATE
		C+	40	endmac IF_NE_BRANCH
00478B	3E6E	C	41	ld a, 'n'
00478D	32 BD 71	C	42	ld (autoflg\$), a
		C	43	MVC spaceout\$, autoflg\$+2, 70
		C+	43	scope
004790	21 76 89	C+	43	ld hl, spaceout\$
004793	11 BF 71	C+	43	ld de, autoflg\$+2
004796	014600	C+	43	ld bc, 70
004799	EDB0	C+	43	LDIR
		C+	43	ENDMAC MVC
		C	44	MVC no_auto\$, autoflg\$+2, 17
		C+	44	scope
00479B	21 61 86	C+	44	ld hl, no_auto\$
00479E	11 BF 71	C+	44	ld de, autoflg\$+2

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-DOS.s
0047A1	011100	C+	44	ld	bc,17
0047A4	EDB0	C+	44	LDIR	
		C+	44	ENDMAC	MVC
		C	45		
		C	46	; @@@@@@	Display welcome screen. @@@@@@
		C	47		
0047A6		C	48	BERTDATE	DATE.get BERTSAYS+7 ; Put current date in menu.
		C+	48	scope	
0047A6	21 29 71	C+	48	ld	hl,BERTSAYS+7
		C+	48	SVC	svc_date
0047A9	3E12	C+	48	ld	a,svc_date
0047AB	EF	C+	48	restart28	
		C+	48	ENDMAC	SVC
		C+	48	ENDMAC	DATE.get
		C	49	TIME.get	BERTSAYS+21 ; Put current time in menu.
		C+	49	scope	
0047AC	21 37 71	C+	49	ld	hl,BERTSAYS+21
		C+	49	SVC	svc_time
0047AF	3E13	C+	49	ld	a,svc_time
0047B1	EF	C+	49	restart28	
		C+	49	ENDMAC	SVC
		C+	49	ENDMAC	TIME.get
		C	50	MVC	PAKNAM\$,paknam\$,15 ; Move installation name from PAGE4.
		C+	50	scope	
0047B2	21 10 04	C+	50	ld	hl,PAKNAM\$
0047B5	11 49 71	C+	50	ld	de,paknam\$
0047B8	010F00	C+	50	ld	bc,15
0047BB	EDB0	C+	50	LDIR	
		C+	50	ENDMAC	MVC
		C	51		
		C	52	; Get AUTO command if any, test if BREAK flag enabled/disabled. If 1st char * then disable.	
		C	53		
0047BD	21 20 04	C	54	ld	hl,INBUF\$ ; Point to buffer with auto command if any.
0047C0	7E	C	55	ld	a,(hl) ; Get 1st byte.
0047C1	23	C	56	inc	hl
0047C2	E5	C	57	push	hl
0047C3	FE2A	C	58	cp	'*' ; Turn BREAK off?
		C	59	BREAK.disable	; Turn BREAK off, this macro only uses HL register.
		C+	59	scope	
0047C5	21 7C 00	C+	59	LD	hl,SFLAG\$
0047C8	CBE6	C+	59	set	4,(hl) ; Disable BREAK.
		C+	59	ENDMAC	BREAK.disable
		C	60	IF_EQ_BRANCH	sav_point ; If yes, skip turning on?
		C+	60	scope	
0047CA	28 12	C+	60	jr	z,sav_point
		C+	60	endmac	IF_EQ_BRANCH
		C	61	BREAK.enable	
		C+	61	scope	
0047CC	21 7C 00	C+	61	LD	hl,SFLAG\$
0047CF	CBA6	C+	61	res	4,(hl) ; Turn BREAK on.
		C+	61	endmac	BREAK.enable
		C	62	MVC	SYSGN0\$,CFGFCB\$,6 ; Setup default filename to load CONFIG/SYS.
		C+	62	scope	
0047D1	21 68 85	C+	62	ld	hl,SYSGN0\$
0047D4	11 E0 00	C+	62	ld	de,CFGFCB\$
0047D7	010600	C+	62	ld	bc,6
0047DA	EDB0	C+	62	LDIR	
		C+	62	ENDMAC	MVC
0047DC	E1	C	63	pop	hl
0047DD	2B	C	64	dec	hl ; Bump pointer +1.

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source
0047DE	22 B2 4C	C	65	sav_pointld (Hlsav),hl ; Save pointer to command buffer.
		C	66	
0047E1	3A0300	C	67	ld a,(quiet_default_flg) ; Get byte operator patched as starting terminal (default is 4)
0047E4	E607	C	68	and 00000111b ; Get lower nibble.
0047E6	C630	C	69	add '0' ; Convert to ASCII.
0047E8	32 29 73	C	70	ld (default_t),a ; Store default starting terminal in menu screen.
		C	71	
0047EB	3A0300	C	72	ld a,(quiet_default_flg) ; Get quiet/default IPL code.
0047EE	CB77	C	73	bit 6,a ; Is networking on?
		C	74	IF_EQ_BRANCH ipl_chk_quiet ; Bypass if quiet off.
		C+	74	scope
0047F0	28 06	C+	74	jr z,ipl_chk_quiet
		C+	74	endmac IF_EQ_BRANCH
		C	75	
0047F2	CD 03 66	C	76	call query_server ; Get server parameters.
0047F5	CD 39 60	C	77	call bind_server ; Bind server & client (me).
		C	78	
0047F8	3A0300	C	79	ipl_chk_quiet ld a,(quiet_default_flg) ; Get quiet/default IPL code.
0047FB	E680	C	80	and 10000000b ; Isolate quiet flag.
		C	81	IF_NE_GOTO default_ipl ; Bypass if quiet off.
		C+	81	scope
0047FD	C2 95 48	C+	81	jp nz,default_ipl
		C+	81	endmac IF_NE_GOTO
		C	82	
		C	83	
004800		C	84	main_ipl_menu DATE.get BERTSAYS+7 ; Put current date in menu.
		C+	84	scope
004800	21 29 71	C+	84	ld hl,BERTSAYS+7
		C+	84	SVC svc_date
004803	3E12	C+	84	ld a,svc_date
004805	EF	C+	84	restart28
		C+	84	ENDMAC SVC
		C+	84	ENDMAC DATE.get
		C	85	TIME.get BERTSAYS+21 ; Put current time in menu.
		C+	85	scope
004806	21 37 71	C+	85	ld hl,BERTSAYS+21
		C+	85	SVC svc_time
004809	3E13	C+	85	ld a,svc_time
00480B	EF	C+	85	restart28
		C+	85	ENDMAC SVC
		C+	85	ENDMAC TIME.get
		C	86	
		C	87	GOSUB scrolloff
		C+	87	scope
00480C	CD ED 6E	C+	87	CALL scrolloff
		C+	87	endmac GOSUB
		C	88	
		C	89	PUMP DEVICE.console,BOOTMENU ; Display main IPL menu.
		C+	89	scope ; A tribute to Dave Shriner - Display Data Corp.
00480F	21 BE 70	C+	89	LD HL,BOOTMENU ; Pump from this starting point until ETX.asc is found.
		C+	89	if DEVICE.console=0
004812	7E	C+	89	__1620: LD A,(HL) ; Get byte to send.
004813	23	C+	89	INC HL ; pointer=pointer+1
		C+	89	; CP EOT.asc ; End Of Transmission (block)?
		C+	89	; jr Z,\$0a ; Return if finished.
004814	FE03	C+	89	CP ETX.asc ; End Of Text?
004816	28 16	C+	89	jr Z,__1620a ; Return if finished.
004818	FE0D	C+	89	CP CR.asc ; Was it a CR?
00481A	CC 24 48	C+	89	call z,__1621 ; 1st add a LF to stream.
00481D	4F	C+	89	LD C,A ; Output byte in C.



\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-DOS.s
00481E	AF	C+	89	XOR A	; Set Z flag for put.
00481F	CD D9 10	C+	89	CALL U0DVR	; Out the UART we go.
004822	18 EE	C+	89	JR __1620	
004824	3E0A	C+	89	__1621: ld a,LF.asc	; Get a LF.
004826	4F	C+	89	LD C,A	; Output byte in C.
004827	AF	C+	89	XOR A	; Set Z flag for put
004828	CD D9 10	C+	89	CALL U0DVR	; Out the UART we go.
00482B	3E0D	C+	89	ld a,CR.asc	; Now put a CR back in stream.
00482D	C9	C+	89	ret	; Return and print CR.
00482E		C+	89	__1620a endif	
		C+	89	if DEVICE.console=1	
		C+	89	__1621: LD A,(HL)	; Get byte to send.
		C+	89	INC HL	; pointer=pointer+1
		C+	89	; CP EOT.asc	; End Of Transmission (block)?
		C+	89	; jr Z,\$1a	; Return if finished.
		C+	89	CP ETX.asc	; End Of Text?
		C+	89	jr Z,__1621a	; Return if finished.
		C+	89	LD C,A	; Output byte to C.
		C+	89	XOR A	; Set Z flag for put.
		C+	89	CALL U1DVR	; Out the UART we go.
		C+	89	JR __1621	
		C+	89	__1621a equ \$	
		C+	89	endif	
		C+	89	ENDMAC PUMP	
		C	90		
00482E		C	91	inqlp: GOSUB getchar	
		C+	91	scope	
00482E	CD 7E 6F	C+	91	CALL getchar	
		C+	91	endmac GOSUB	
		C	92		
004831	FE1B	C	93	cp ESC.asc	; Did they hit escape?
		C	94	IF_EQ_GOTO main_ipl_menu	; If YES refresh screen.
		C+	94	scope	
004833	CA 00 48	C+	94	jp z,main_ipl_menu	
		C+	94	endmac IF_EQ_GOTO	
		C	95		
004836	FE2F	C	96	cp '/'	
		C	97	IF_EQ_GOSUB test1	
		C+	97	scope	
004838	CC E5 8A	C+	97	call z,test1	
		C+	97	endmac IF_EQ_GOSUB	
		C	98		
00483B	FE2B	C	99	cp '+'	
		C	100	IF_EQ_GOSUB test2	
		C+	100	scope	
00483D	CC E5 8A	C+	100	call z,test2	
		C+	100	endmac IF_EQ_GOSUB	
		C	101		
004840	FE2A	C	102	cp '*'	
		C	103	IF_EQ_GOSUB test3	
		C+	103	scope	
004842	CC E5 8A	C+	103	call z,test3	
		C+	103	endmac IF_EQ_GOSUB	
		C	104		
004845	FE2D	C	105	cp '-'	
		C	106	IF_EQ_GOSUB test4	
		C+	106	scope	
004847	CC E5 8A	C+	106	call z,test4	
		C+	106	endmac IF_EQ_GOSUB	
		C	107		

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source
00484A	FE3D	C	108	cp '='
		C	109	IF_EQ_GOSUB test5
		C+	109	scope
00484C	CC 06 8B	C+	109	call z,test5
		C+	109	endmac IF_EQ_GOSUB
		C	110	
00484F	FE30	C	111	cp '0'
		C	112	IF_EQ_GOTO default_ipl
		C+	112	scope
004851	CA 95 48	C+	112	jp z,default_ipl
		C+	112	endmac IF_EQ_GOTO
		C	113	
		C	114	IF_GT_BRANCH inqlp ; If 0>key then it is not 0-9.
		C+	114	scope
004854	38 D8	C+	114	jr c,inqlp
		C+	114	endmac IF_GT_BRANCH
		C	115	
004856	FE31	C	116	cp '1' ; Run updates.
		C	117	IF_EQ_GOTO notimp
		C+	117	scope
004858	CA 1C 70	C+	117	jp z,notimp
		C+	117	endmac IF_EQ_GOTO
		C	118	
00485B	FE32	C	119	cp '2'
		C	120	IF_EQ_GOTO safe_ipl ; Safe mode prompts for load sysgen, auto command etc.
		C+	120	scope
00485D	CA A5 48	C+	120	jp z,safe_ipl
		C+	120	endmac IF_EQ_GOTO
		C	121	
004860	FE33	C	122	cp '3'
		C	123	IF_EQ_GOTO BIT_IPL ; BIT initialization.
		C+	123	scope
004862	CA 90 48	C+	123	jp z,BIT_IPL
		C+	123	endmac IF_EQ_GOTO
		C	124	
004865	FE34	C	125	cp '4'
		C	126	IF_EQ_GOTO RTC_IPL ; RTC initialization.
		C+	126	scope
004867	CA 11 4D	C+	126	jp z,RTC_IPL
		C+	126	endmac IF_EQ_GOTO
		C	127	
00486A	FE35	C	128	cp '5'
		C	129	IF_EQ_GOTO editslice ; Display/Edit Slice data.
		C+	129	scope
00486C	CA 11 6A	C+	129	jp z,editslice
		C+	129	endmac IF_EQ_GOTO
		C	130	
00486F	FE64	C	131	cp 'd'
		C	132	IF_EQ_GOSUB set_terminal ; User wants change default setting for
		C+	132	scope
004871	CC E7 82	C+	132	call z,set_terminal
		C+	132	endmac IF_EQ_GOSUB
004874	FE44	C	133	cp 'D'
		C	134	IF_EQ_GOSUB set_terminal ; default startup type.
		C+	134	scope
004876	CC E7 82	C+	134	call z,set_terminal
		C+	134	endmac IF_EQ_GOSUB
		C	135	
004879	FE6E	C	136	cp 'n'
		C	137	IF_EQ_GOSUB net_utils ; Network utils.

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-DOS.s
		C+	137	scope	
00487B	CC 37 63	C+	137	call	z,net_utils
		C+	137	endmac	IF_EQ_GOSUB
00487E	FE4E	C	138	cp	'N'
		C	139	IF_EQ_GOSUB	net_utils ; Network utils.
		C+	139	scope	
004880	CC 37 63	C+	139	call	z,net_utils
		C+	139	endmac	IF_EQ_GOSUB
		C	140		
004883	FE78	C	141	cp	'x'
		C	142	IF_EQ_GOTO	reboot ; Call in and IPL CP/M.
		C+	142	scope	
004885	CA 8D F7	C+	142	jp	z,reboot
		C+	142	endmac	IF_EQ_GOTO
004888	FE58	C	143	cp	'X'
		C	144	IF_EQ_GOTO	reboot ; Call in and IPL CP/M.
		C+	144	scope	
00488A	CA 8D F7	C+	144	jp	z,reboot
		C+	144	endmac	IF_EQ_GOTO
		C	145		
		C	146	GOTO	main_ipl_menu
		C+	146	scope	
00488D	C3 00 48	C+	146	jp	main_ipl_menu
		C+	146	endmac	GOTO
		C	147		
		C	148		; Final initialization code.
		C	149		
004890	3E35	C	150	BIT_IPL	ld a,'5'
004892	32 29 73	C	151		ld (default_t),a ; Set to 0, \$BIT.
		C	152		
004895	3E6E	C	153	default_ipl	ld a,'n' ; Tell program this is NOT a SAFE start.
004897	32 FF 86	C	154		ld (safeipl\$),a ; Initialize safe IPL flag to no.
		C	155		
		C	156	GOSUB	scrolloff
		C+	156	scope	
00489A	CD ED 6E	C+	156	CALL	scrolloff
		C+	156	endmac	GOSUB
		C	157		
00489D	3A 29 73	C	158		ld a,(default_t) ; Get default value for start...User can patch zRST00+3 to set defa
		C	159		
0048A0		C	160	set_term	GOSUB term_setup
		C+	160	scope	
0048A0	CD 11 83	C+	160	CALL	term_setup
		C+	160	endmac	GOSUB
		C	161		
		C	162	BRANCH	start_bkgnd ; Start background processing.
		C+	162	scope	
0048A3	18 51	C+	162	jr	start_bkgnd
		C+	162	endmac	BRANCH
		C	163		
		C	164		; Setup SAFE IPL session.
		C	165		; Set safeipl\$ to y.
		C	166		; GOSUB to query user for terminal @set_terminal.
		C	167		; set_terminal allows user to select terminal type for this session.
		C	168		
0048A5	3E79	C	169	safe_ipl	ld a,'y'
0048A7	32 FF 86	C	170		ld (safeipl\$),a ; Set safe IPL flag to YES.
		C	171		
		C	172		; User is requesting a SAFE start.
		C	173		; SAFE start allows operator to pick terminal type for session.

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source
		C	174	; GOSUB set_terminal asks operator to select which terminal.
		C	175	
		C	176	GOSUB set_terminal ; Get & set terminal type for connection session.
		C+	176	scope
0048AA	CD E7 82	C+	176	CALL set_terminal
		C+	176	endmac GOSUB
		C	177	
		C	178	; We are executing SAFE start, prompt if we should start background processing.
		C	179	
		C	180	GOSUB scrolloff ; Clean screen up.
		C+	180	scope
0048AD	CD ED 6E	C+	180	CALL scrolloff
		C+	180	endmac GOSUB
		C	181	
		C	182	PUMP DEVICE.console,crlf
		C+	182	scope ; A tribute to Dave Shriner - Display Data Corp.
0048B0	21 FA 86	C+	182	LD HL,crlf ; Pump from this starting point until ETX.asc is found.
		C+	182	if DEVICE.console=0
0048B3	7E	C+	182	__1890: LD A,(HL) ; Get byte to send.
0048B4	23	C+	182	INC HL ; pointer=pointer+1
		C+	182	; CP EOT.asc ; End Of Transmission (block)?
		C+	182	; jr Z,\$0a ; Return if finished.
0048B5	FE03	C+	182	CP ETX.asc ; End Of Text?
0048B7	28 16	C+	182	jr Z,__1890a ; Return if finished.
0048B9	FE0D	C+	182	CP CR.asc ; Was it a CR?
0048BB	CC C5 48	C+	182	call z,__1891 ; 1st add a LF to stream.
0048BE	4F	C+	182	LD C,A ; Output byte in C.
0048BF	AF	C+	182	XOR A ; Set Z flag for put.
0048C0	CD D9 10	C+	182	CALL U0DVR ; Out the UART we go.
0048C3	18 EE	C+	182	JR __1890
0048C5	3E0A	C+	182	__1891: ld a,LF.asc ; Get a LF.
0048C7	4F	C+	182	LD C,A ; Output byte in C.
0048C8	AF	C+	182	XOR A ; Set Z flag for put
0048C9	CD D9 10	C+	182	CALL U0DVR ; Out the UART we go.
0048CC	3E0D	C+	182	ld a,CR.asc ; Now put a CR back in stream.
0048CE	C9	C+	182	ret ; Return and print CR.
0048CF		C+	182	__1890a endif
		C+	182	if DEVICE.console=1
		C+	182	__1891: LD A,(HL) ; Get byte to send.
		C+	182	INC HL ; pointer=pointer+1
		C+	182	; CP EOT.asc ; End Of Transmission (block)?
		C+	182	; jr Z,\$1a ; Return if finished.
		C+	182	CP ETX.asc ; End Of Text?
		C+	182	jr Z,__1891a ; Return if finished.
		C+	182	LD C,A ; Output byte to C.
		C+	182	XOR A ; Set Z flag for put.
		C+	182	CALL U1DVR ; Out the UART we go.
		C+	182	JR __1891
		C+	182	__1891a equ \$
		C+	182	endif
		C+	182	ENDMAC PUMP
		C	183	PUMP DEVICE.console,stbgrnd\$ ; Ask operator if we start background processing.
		C+	183	scope ; A tribute to Dave Shriner - Display Data Corp.
0048CF	21 D4 85	C+	183	LD HL,stbgrnd\$ ; Pump from this starting point until ETX.asc is found.
		C+	183	if DEVICE.console=0
0048D2	7E	C+	183	__1900: LD A,(HL) ; Get byte to send.
0048D3	23	C+	183	INC HL ; pointer=pointer+1
		C+	183	; CP EOT.asc ; End Of Transmission (block)?
		C+	183	; jr Z,\$0a ; Return if finished.
0048D4	FE03	C+	183	CP ETX.asc ; End Of Text?

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-DOS.s
0048D6	28 16	C+	183	jr	Z, __1900a ; Return if finished.
0048D8	FE0D	C+	183	CP	CR.asc ; Was it a CR?
0048DA	CC E4 48	C+	183	call	z, __1901 ; 1st add a LF to stream.
0048DD	4F	C+	183	LD	C,A ; Output byte in C.
0048DE	AF	C+	183	XOR	A ; Set Z flag for put.
0048DF	CD D9 10	C+	183	CALL	U0DVR ; Out the UART we go.
0048E2	18 EE	C+	183	JR	__1900
0048E4	3E0A	C+	183	__1901:	ld a, LF.asc ; Get a LF.
0048E6	4F	C+	183	LD	C,A ; Output byte in C.
0048E7	AF	C+	183	XOR	A ; Set Z flag for put
0048E8	CD D9 10	C+	183	CALL	U0DVR ; Out the UART we go.
0048EB	3E0D	C+	183	ld	a, CR.asc ; Now put a CR back in stream.
0048ED	C9	C+	183	ret	; Return and print CR.
0048EE		C+	183	__1900a	endif
		C+	183		if DEVICE.console=1
		C+	183	__1901:	LD A, (HL) ; Get byte to send.
		C+	183	INC	HL ; pointer=pointer+1
		C+	183	;	CP EOT.asc ; End Of Transmission (block)?
		C+	183	;	jr Z, \$1a ; Return if finished.
		C+	183		CP ETX.asc ; End Of Text?
		C+	183		jr Z, __1901a ; Return if finished.
		C+	183		LD C,A ; Output byte to C.
		C+	183		XOR A ; Set Z flag for put.
		C+	183		CALL U1DVR ; Out the UART we go.
		C+	183		JR __1901
		C+	183	__1901a	equ \$
		C+	183		endif
		C+	183	ENDMAC	PUMP
		C	184	GOSUB	GET.yn ; Get response.
		C+	184	scope	
0048EE	CD 38 6F	C+	184	CALL	GET.yn
		C+	184	endmac	GOSUB
0048F1	FE79	C	185	cp	'y'
		C	186	IF_NE_GOTO	test_sysgen ; If answer if no, skip starting background tasks.
		C+	186	scope	
0048F3	C2 5F 49	C+	186	jp	nz, test_sysgen
		C+	186	endmac	IF_NE_GOTO
		C	187		
		C	188		
		C	189	;	< < < < ***** > > > > S E T U P I N T E R R U P T H A N D L E R < < < < ***** > > >
		C	190		
		C	191		
0048F6		C	192	start_bkgnd	PUMP DEVICE.console, crlf
		C+	192	scope	; A tribute to Dave Shriner - Display Data Corp.
0048F6	21 FA 86	C+	192	LD	HL, crlf ; Pump from this starting point until ETX.asc is found.
		C+	192	if	DEVICE.console=0
0048F9	7E	C+	192	__1930:	LD A, (HL) ; Get byte to send.
0048FA	23	C+	192	INC	HL ; pointer=pointer+1
		C+	192	;	CP EOT.asc ; End Of Transmission (block)?
		C+	192	;	jr Z, \$0a ; Return if finished.
0048FB	FE03	C+	192	CP	ETX.asc ; End Of Text?
0048FD	28 16	C+	192	jr	Z, __1930a ; Return if finished.
0048FF	FE0D	C+	192	CP	CR.asc ; Was it a CR?
004901	CC 0B 49	C+	192	call	z, __1931 ; 1st add a LF to stream.
004904	4F	C+	192	LD	C,A ; Output byte in C.
004905	AF	C+	192	XOR	A ; Set Z flag for put.
004906	CD D9 10	C+	192	CALL	U0DVR ; Out the UART we go.
004909	18 EE	C+	192	JR	__1930
00490B	3E0A	C+	192	__1931:	ld a, LF.asc ; Get a LF.
00490D	4F	C+	192	LD	C,A ; Output byte in C.

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-DOS.s
00490E	AF	C+	192	XOR	A ; Set Z flag for put
00490F	CD D9 10	C+	192	CALL	U0DVR ; Out the UART we go.
004912	3E0D	C+	192	ld	a,CR.asc ; Now put a CR back in stream.
004914	C9	C+	192	ret	; Return and print CR.
004915		C+	192	__1930a	endif
		C+	192		if DEVICE.console=1
		C+	192	__1931:	LD A,(HL) ; Get byte to send.
		C+	192	INC	HL ; pointer=pointer+1
		C+	192	CP	EOT.asc ; End Of Transmission (block)?
		C+	192	;	
		C+	192	jr	Z,\$1a ; Return if finished.
		C+	192	CP	ETX.asc ; End Of Text?
		C+	192	jr	Z,__1931a ; Return if finished.
		C+	192	LD	C,A ; Output byte to C.
		C+	192	XOR	A ; Set Z flag for put.
		C+	192	CALL	U1DVR ; Out the UART we go.
		C+	192	JR	__1931
		C+	192	__1931a	equ \$
		C+	192		endif
		C+	192	ENDMAC	PUMP
		C	193		
004915	3A 79 00	C	194	ld	a,(PFLAG\$) ; Determine which processor we are on.
004918	CB2F	C	195	sra	a ; Unpack it.
00491A	FE01	C	196	cp	01h ; Is it a 91?
		C	197	IF_NE_BRANCH	nota91 ; Bypass f91 code for PRT1 enable.
		C+	197	scope	
00491C	20 27	C+	197	jr	nz,nota91
		C+	197	endmac	IF_NE_BRANCH
		C	198		
		C	199		; Setup interrupt table for f91 processor.
		C	200		
		C	201	MVC	f91INTERRUPT.table,INTERRUPT.handler+40h,192 ; Move interrupt table for f91 model.
		C+	201	scope	
00491E	21 C5 89	C+	201	ld	hl,f91INTERRUPT.table
004921	1140F6	C+	201	ld	de,INTERRUPT.handler+40h
004924	01C000	C+	201	ld	bc,192
004927	EDB0	C+	201	LDIR	
		C+	201	ENDMAC	MVC
004929	21F600	C	202	ld	hl,INTERRUPT.handler >> 8 ; Get upper 8 bits of vector table.
00492C	EDC7	C	203	ld	i,hl ; Put upper 8 bits of table in I register.
		C	204		
		C	205		; Enable 60HZ real time interrupts using PRT1.INTERRUPT.handler+40h
		C	206		
00492E	ED3867	C	207	in0	a,(TMR1_IIR) ; Read to reset.
004931	ED3865	C	208	in0	a,(TMR1_CTL) ; Read to reset any pending interrupt.
		C	209		
004934	3E01	C	210	ld	a,01 ; Enable interrupt at end of count.
004936	ED3966	C	211	out0	(TMR1_IER),a ; IRQ EOC EN.
		C	212		
004939	ED3865	C	213	in0	a,(TMR1_CTL) ; Get platform specific bits 7,6,5,4,3.
00493C	E6F8	C	214	and	11111000b ; Mask out preset bits.
00493E	F607	C	215	or	00000111b ; TIM_CONT,RLD,TIM_EN, enable PRT1 timer for eZ80f91.
004940	ED3965	C	216	out0	(TMR1_CTL),a ; Timer is now loaded & counting. Waiting for EI.
		C	217	BRANCH	EN_IE ; Next step, interrupts and timwer on.
		C+	217	scope	
004943	18 19	C+	217	jr	EN_IE
		C+	217	endmac	BRANCH
		C	218		
004945		C	219	nota91	MVC f92INTERRUPT.table,INTERRUPT.handler,96 ; Move interrupt table for f92 model chip.
		C+	219	scope	
004945	21 85 8A	C+	219	ld	hl,f92INTERRUPT.table

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-DOS.s
004948	1100F6	C+	219	ld	de, INTERRUPT.handler
00494B	016000	C+	219	ld	bc, 96
00494E	EDB0	C+	219	LDIR	
		C+	219	ENDMAC	MVC
		C	220		
		C	221	; Setup PRT1 for eZ80f92 processor.	
		C	222		
004950	3EF6	C	223	ld	a, HIGH(INTERRUPT.handler) ; Setup internal interrupt register to interrupt table.
004952	ED47	C	224	ld	i, a
		C	225		
		C	226	; Winner winner time for a stripper.	
		C	227		
004954	ED3883	C	228	in0	a, (f92_TMR1_CTL) ; Read & reset any pending interrupts.
004957	E60C	C	229	and	00001100b ; Strip her bits marked wit 0.
004959	F653	C	230	or	01010011b ; Lets stuff da stripped-->Enbl, cont, using platform specific
00495B	ED3983	C	231	out0	(f92_TMR1_CTL), a ; Timer1 is now loaded & counting.
		C	232		
		C	233	; Setup to enable interrupts during final phases of IPL.	
		C	234	; This phase of code is common to all processor platforms.	
		C	235		
00495E	FB	C	236	EN_IE	ei ; Enable interrupts, start backgroud task processing.
		C	237		
00495F	3A FF 86	C	238	test_sysgen	ld a, (safeipl\$) ; Get safe\$ flag.
004962	FE79	C	239	cp	'y' ; Are we starting in safe mode?
		C	240	IF_NE_GOTO	load_sysgen ; Bypass asking if we load, just load.
		C+	240	scope	
004964	C2 38 4A	C+	240	jp	nz, load_sysgen
		C+	240	endmac	IF_NE_GOTO
		C	241		
		C	242	PUMP	DEVICE.console, ldSYSGN\$ ; Test if loading sysgen SYSGEN file needed.
		C+	242	scope	; A tribute to Dave Shriner - Display Data Corp.
004967	21 3A 85	C+	242	LD	HL, ldSYSGN\$ ; Pump from this starting point until ETX.asc is found.
		C+	242	if	DEVICE.console=0
00496A	7E	C+	242	LD	A, (HL) ; Get byte to send.
00496B	23	C+	242	INC	HL ; pointer=pointer+1
		C+	242	; CP	EOT.asc ; End Of Transmission (block)?
		C+	242	; jr	Z, \$0a ; Return if finished.
00496C	FE03	C+	242	CP	ETX.asc ; End Of Text?
00496E	28 16	C+	242	jr	Z, __1990a ; Return if finished.
004970	FE0D	C+	242	CP	CR.asc ; Was it a CR?
004972	CC 7C 49	C+	242	call	z, __1991 ; 1st add a LF to stream.
004975	4F	C+	242	LD	C, A ; Output byte in C.
004976	AF	C+	242	XOR	A ; Set Z flag for put.
004977	CD D9 10	C+	242	CALL	U0DVR ; Out the UART we go.
00497A	18 EE	C+	242	JR	__1990
00497C	3E0A	C+	242	__1991:	ld a, LF.asc ; Get a LF.
00497E	4F	C+	242	LD	C, A ; Output byte in C.
00497F	AF	C+	242	XOR	A ; Set Z flag for put
004980	CD D9 10	C+	242	CALL	U0DVR ; Out the UART we go.
004983	3E0D	C+	242	ld	a, CR.asc ; Now put a CR back in stream.
004985	C9	C+	242	ret	; Return and print CR.
004986		C+	242	__1990a	endif
		C+	242	if	DEVICE.console=1
		C+	242	__1991:	LD A, (HL) ; Get byte to send.
		C+	242	INC	HL ; pointer=pointer+1
		C+	242	; CP	EOT.asc ; End Of Transmission (block)?
		C+	242	; jr	Z, \$1a ; Return if finished.
		C+	242	CP	ETX.asc ; End Of Text?
		C+	242	jr	Z, __1991a ; Return if finished.
		C+	242	LD	C, A ; Output byte to C.

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-DOS.s
		C+	242	XOR A	; Set Z flag for put.
		C+	242	CALL U1DVR	; Out the UART we go.
		C+	242	JR __1991	
		C+	242	__1991a equ \$	
		C+	242	endif	
		C+	242	ENDMAC PUMP	
004986	CD 38 6F	C	243	call GET.yn	; Get response.
004989	FE79	C	244	cp 'y'	
		C	245	IF_NE_GOTO start_int	; If answer if no, skip loading system & move to next step.
		C+	245	scope	
00498B	C2 CB 4A	C+	245	jp nz,start_int	
		C+	245	endmac IF_NE_GOTO	
		C	246		
		C	247	; SAFE start will now prompt loading of SYSGEN?	
		C	248	; Prompt operator if SYSGEN should be loaded?	
		C	249	; If NO, proceed directly to testing if background processing should be started.	
		C	250		
		C	251	GOSUB scrolloff	
		C+	251	scope	
00498E	CD ED 6E	C+	251	CALL scrolloff	
		C+	251	endmac GOSUB	
		C	252	PUMP DEVICE.console,whichsysmsg\$	
		C+	252	scope	; A tribute to Dave Shriner - Display Data Corp.
004991	21 07 84	C+	252	LD HL,whichsysmsg\$	; Pump from this starting point until ETX.asc is found.
		C+	252	if DEVICE.console=0	
004994	7E	C+	252	__2020: LD A,(HL)	; Get byte to send.
004995	23	C+	252	INC HL	; pointer=pointer+1
		C+	252	; CP EOT.asc	; End Of Transmission (block)?
		C+	252	; jr Z,\$0a	; Return if finished.
004996	FE03	C+	252	CP ETX.asc	; End Of Text?
004998	28 16	C+	252	jr Z,__2020a	; Return if finished.
00499A	FE0D	C+	252	CP CR.asc	; Was it a CR?
00499C	CC A6 49	C+	252	call z,__2021	; 1st add a LF to stream.
00499F	4F	C+	252	LD C,A	; Output byte in C.
0049A0	AF	C+	252	XOR A	; Set Z flag for put.
0049A1	CD D9 10	C+	252	CALL U0DVR	; Out the UART we go.
0049A4	18 EE	C+	252	JR __2020	
0049A6	3E0A	C+	252	__2021: ld a,LF.asc	; Get a LF.
0049A8	4F	C+	252	LD C,A	; Output byte in C.
0049A9	AF	C+	252	XOR A	; Set Z flag for put
0049AA	CD D9 10	C+	252	CALL U0DVR	; Out the UART we go.
0049AD	3E0D	C+	252	ld a,CR.asc	; Now put a CR back in stream.
0049AF	C9	C+	252	ret	; Return and print CR.
0049B0		C+	252	__2020a endif	
		C+	252	if DEVICE.console=1	
		C+	252	__2021: LD A,(HL)	; Get byte to send.
		C+	252	INC HL	; pointer=pointer+1
		C+	252	; CP EOT.asc	; End Of Transmission (block)?
		C+	252	; jr Z,\$1a	; Return if finished.
		C+	252	CP ETX.asc	; End Of Text?
		C+	252	jr Z,__2021a	; Return if finished.
		C+	252	LD C,A	; Output byte to C.
		C+	252	XOR A	; Set Z flag for put.
		C+	252	CALL U1DVR	; Out the UART we go.
		C+	252	JR __2021	
		C+	252	__2021a equ \$	
		C+	252	endif	
		C+	252	ENDMAC PUMP	
		C	253	PUMP DEVICE.console,crlf	
		C+	253	scope	; A tribute to Dave Shriner - Display Data Corp.



\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source
0049B0	21 FA 86	C+	253	LD HL,crlf ; Pump from this starting point until ETX.asc is found.
		C+	253	if DEVICE.console=0
0049B3	7E	C+	253	__2030: LD A,(HL) ; Get byte to send.
0049B4	23	C+	253	INC HL ; pointer=pointer+1
		C+	253	; CP EOT.asc ; End Of Transmission (block)?
		C+	253	; jr Z,\$0a ; Return if finished.
0049B5	FE03	C+	253	CP ETX.asc ; End Of Text?
0049B7	28 16	C+	253	jr Z,__2030a ; Return if finished.
0049B9	FE0D	C+	253	CP CR.asc ; Was it a CR?
0049BB	CC C5 49	C+	253	call z,__2031 ; 1st add a LF to stream.
0049BE	4F	C+	253	LD C,A ; Output byte in C.
0049BF	AF	C+	253	XOR A ; Set Z flag for put.
0049C0	CD D9 10	C+	253	CALL U0DVR ; Out the UART we go.
0049C3	18 EE	C+	253	JR __2030
0049C5	3E0A	C+	253	__2031: ld a,LF.asc ; Get a LF.
0049C7	4F	C+	253	LD C,A ; Output byte in C.
0049C8	AF	C+	253	XOR A ; Set Z flag for put
0049C9	CD D9 10	C+	253	CALL U0DVR ; Out the UART we go.
0049CC	3E0D	C+	253	ld a,CR.asc ; Now put a CR back in stream.
0049CE	C9	C+	253	ret ; Return and print CR.
0049CF		C+	253	__2030a endif
		C+	253	if DEVICE.console=1
		C+	253	__2031: LD A,(HL) ; Get byte to send.
		C+	253	INC HL ; pointer=pointer+1
		C+	253	; CP EOT.asc ; End Of Transmission (block)?
		C+	253	; jr Z,\$1a ; Return if finished.
		C+	253	CP ETX.asc ; End Of Text?
		C+	253	jr Z,__2031a ; Return if finished.
		C+	253	LD C,A ; Output byte to C.
		C+	253	XOR A ; Set Z flag for put.
		C+	253	CALL U1DVR ; Out the UART we go.
		C+	253	JR __2031
		C+	253	__2031a equ \$
		C+	253	endif
		C+	253	ENDMAC PUMP
		C	254	PUMP DEVICE.console,pick_config\$
		C+	254	scope ; A tribute to Dave Shriner - Display Data Corp.
0049CF	21 1B 85	C+	254	LD HL,pick_config\$ ; Pump from this starting point until ETX.asc is found.
		C+	254	if DEVICE.console=0
0049D2	7E	C+	254	__2040: LD A,(HL) ; Get byte to send.
0049D3	23	C+	254	INC HL ; pointer=pointer+1
		C+	254	; CP EOT.asc ; End Of Transmission (block)?
		C+	254	; jr Z,\$0a ; Return if finished.
0049D4	FE03	C+	254	CP ETX.asc ; End Of Text?
0049D6	28 16	C+	254	jr Z,__2040a ; Return if finished.
0049D8	FE0D	C+	254	CP CR.asc ; Was it a CR?
0049DA	CC E4 49	C+	254	call z,__2041 ; 1st add a LF to stream.
0049DD	4F	C+	254	LD C,A ; Output byte in C.
0049DE	AF	C+	254	XOR A ; Set Z flag for put.
0049DF	CD D9 10	C+	254	CALL U0DVR ; Out the UART we go.
0049E2	18 EE	C+	254	JR __2040
0049E4	3E0A	C+	254	__2041: ld a,LF.asc ; Get a LF.
0049E6	4F	C+	254	LD C,A ; Output byte in C.
0049E7	AF	C+	254	XOR A ; Set Z flag for put
0049E8	CD D9 10	C+	254	CALL U0DVR ; Out the UART we go.
0049EB	3E0D	C+	254	ld a,CR.asc ; Now put a CR back in stream.
0049ED	C9	C+	254	ret ; Return and print CR.
0049EE		C+	254	__2040a endif
		C+	254	if DEVICE.console=1
		C+	254	__2041: LD A,(HL) ; Get byte to send.

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-DOS.s
		C+	254	INC	HL ; pointer=pointer+1
		C+	254	;	CP EOT.asc ; End Of Transmission (block)?
		C+	254	;	jz Z,\$1a ; Return if finished.
		C+	254	CP	ETX.asc ; End Of Text?
		C+	254	jz	Z,__2041a ; Return if finished.
		C+	254	LD	C,A ; Output byte to C.
		C+	254	XOR	A ; Set Z flag for put.
		C+	254	CALL	U1DVR ; Out the UART we go.
		C+	254	JR	__2041
		C+	254	__2041a	equ \$
		C+	254	endif	
		C+	254	ENDMAC	PUMP
		C	255	INPUT_U.str	CFGFCB\$,6 ; Get user defined CONFIG filename.
		C+	255	scope	
0049EE	21 E0 00	C+	255	ld	hl,CFGFCB\$
0049F1	0606	C+	255	ld	b,6 ; Max number bytes to input.
0049F3	18 17	C+	255	jz	__2051 ; Branch to beginning.
0049F5	4F	C+	255	__2052:	ld c,a ; Output BS just received.
0049F6	CD EB 11	C+	255	CALL	TXUART0 ; Send it.
0049F9	0E20	C+	255	ld	c,SP.asc ; Now space out char we are deleting.
0049FB	CD EB 11	C+	255	CALL	TXUART0
0049FE	0E08	C+	255	ld	c,BS.asc ; Backspace again to new location.
004A00	CD EB 11	C+	255	CALL	TXUART0
004A03	2B	C+	255	dec	hl ; Back pointer in buffer up by -1.
004A04	C1	C+	255	pop	bc ; Recover B counter.
004A05	04	C+	255	INC	B
004A06	FE06	C+	255	cp	6
004A08	30 02	C+	255	jz	nc,__2051
004A0A	0606	C+	255	ld	b,6
004A0C	C5	C+	255	__2051:	push bc ; Save counter as other routines may clobber it.
004A0D	CD 7E 6F	C+	255	CALL	getchar ; Using UART NOT TRSDOS get one byte from UART0.
004A10	FE08	C+	255	CP	BS.asc ; Is is backspace?
004A12	28 E1	C+	255	jz	z,__2052 ; If yes gp to backspace routine above.
004A14	77	C+	255	ld	(hl),a ; Put char in buff.
004A15	23	C+	255	inc	hl ; Bump pointer to next place in buffer.
004A16	FE0D	C+	255	cp	CR.asc ; Was it a CR?
004A18	28 0A	C+	255	jz	z,__2054 ; Exit input as terminalted by CR.
004A1A	4F	C+	255	ld	c,a
004A1B	CD EB 11	C+	255	CALL	TXUART0 ; Output character.
004A1E	C1	C+	255	__2053	pop bc ; Recover counter.
004A1F	10 EB	C+	255	djnz	__2051 ; Counter = counter - 1. If counter <> 0 branch \$1.
004A21	AF	C+	255	xor	a ; All is well set Z flag.
004A22	18 01	C+	255	jz	__2055 ; Finished, exit.
004A24	B7	C+	255	__2054:	or a ; Make it non zero.
004A25		C+	255	__2055:	endmac INPUT_U.str ; MACRO is done.
004A25	FE0D	C	256	cp	CR.asc ; Did it terminate with a CR?
		C	257	IF_EQ_BRANCH	ensysgen ; Operator pressed ENTER thus change.
		C+	257	scope	
004A27	28 0B	C+	257	jz	z,ensysgen
		C+	257	endmac	IF_EQ_BRANCH
		C	258	MVC	SYSGN0\$,CFGFCB\$,6 ; filename to default CONFIG/SYS.
		C+	258	scope	
004A29	21 68 85	C+	258	ld	hl,SYSGN0\$
004A2C	11 E0 00	C+	258	ld	de,CFGFCB\$
004A2F	010600	C+	258	ld	bc,6
004A32	EDB0	C+	258	LDIR	
		C+	258	ENDMAC	MVC
004A34	AF	C	259	ensysgen	xor a
004A35	32 01 04	C	260	ld	(ZERO\$),a ; Turn load sysgen on.
004A38	3A 01 04	C	261	load_sysgen	ld a,(ZERO\$)

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-DOS.s
004A3B	B7	C	262	or	a ; Test if 0?
		C	263	IF_NE_GOTO	start_int
		C+	263	scope	
004A3C	C2 CB 4A	C+	263	jp	nz, start_int
		C+	263	endmac	IF_NE_GOTO
		C	264		
		C	265	GOSUB	scrolloff
		C+	265	scope	
004A3F	CD ED 6E	C+	265	CALL	scrolloff
		C+	265	endmac	GOSUB
		C	266	MVC	sysgener, sysgmsg, 12 ; Setup to show error if it dont load.
		C+	266	scope	
004A42	21 5B 85	C+	266	ld	hl, sysgener
004A45	11 43 77	C+	266	ld	de, sysgmsg
004A48	010C00	C+	266	ld	bc, 12
004A4B	EDB0	C+	266	LDIR	
		C+	266	ENDMAC	MVC
		C	267	MVC	CFGFCB\$, sysgmsg, 6
		C+	267	scope	
004A4D	21 E0 00	C+	267	ld	hl, CFGFCB\$
004A50	11 43 77	C+	267	ld	de, sysgmsg
004A53	010600	C+	267	ld	bc, 6
004A56	EDB0	C+	267	LDIR	
		C+	267	ENDMAC	MVC
		C	268	MVC	CFGFCB\$, lding\$+10, 6
		C+	268	scope	
004A58	21 E0 00	C+	268	ld	hl, CFGFCB\$
004A5B	11 78 85	C+	268	ld	de, lding\$+10
004A5E	010600	C+	268	ld	bc, 6
004A61	EDB0	C+	268	LDIR	
		C+	268	ENDMAC	MVC
		C	269	MVC	CONFIG\$, sysgmsg, 12 ; Put sysgen message in LOGO splash screen.
		C+	269	scope	
004A63	21 4E 85	C+	269	ld	hl, CONFIG\$
004A66	11 43 77	C+	269	ld	de, sysgmsg
004A69	010C00	C+	269	ld	bc, 12
004A6C	EDB0	C+	269	LDIR	
		C+	269	ENDMAC	MVC
		C	270	PUMP	DEVICE.console, lding\$ ; Display sysgen file being loaded.
		C+	270	scope	; A tribute to Dave Shriner - Display Data Corp.
004A6E	21 6E 85	C+	270	LD	HL, lding\$ ; Pump from this starting point until ETX.asc is found.
		C+	270	if	DEVICE.console=0
004A71	7E	C+	270	LD	A, (HL) ; Get byte to send.
004A72	23	C+	270	INC	HL ; pointer=pointer+1
		C+	270	CP	EOT.asc ; End Of Transmission (block)?
		C+	270	jr	Z, \$0a ; Return if finished.
004A73	FE03	C+	270	CP	ETX.asc ; End Of Text?
004A75	28 16	C+	270	jr	Z, __2140a ; Return if finished.
004A77	FE0D	C+	270	CP	CR.asc ; Was it a CR?
004A79	CC 83 4A	C+	270	call	z, __2141 ; 1st add a LF to stream.
004A7C	4F	C+	270	LD	C, A ; Output byte in C.
004A7D	AF	C+	270	XOR	A ; Set Z flag for put.
004A7E	CD D9 10	C+	270	CALL	U0DVR ; Out the UART we go.
004A81	18 EE	C+	270	JR	__2140
004A83	3E0A	C+	270	LD	a, LF.asc ; Get a LF.
004A85	4F	C+	270	LD	C, A ; Output byte in C.
004A86	AF	C+	270	XOR	A ; Set Z flag for put
004A87	CD D9 10	C+	270	CALL	U0DVR ; Out the UART we go.
004A8A	3E0D	C+	270	ld	a, CR.asc ; Now put a CR back in stream.
004A8C	C9	C+	270	ret	; Return and print CR.

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-DOS.s
004A8D		C+	270	__2140a	endif
		C+	270		if DEVICE.console=1
		C+	270	__2141:	LD A,(HL) ; Get byte to send.
		C+	270		INC HL ; pointer=pointer+1
		C+	270	;	CP EOT.asc ; End Of Transmission (block)?
		C+	270	;	jr Z,\$1a ; Return if finished.
		C+	270		CP ETX.asc ; End Of Text?
		C+	270		jr Z,__2141a ; Return if finished.
		C+	270		LD C,A ; Output byte to C.
		C+	270		XOR A ; Set Z flag for put.
		C+	270		CALL U1DVR ; Out the UART we go.
		C+	270		JR __2141
		C+	270	__2141a	equ \$
		C+	270		endif
		C+	270		ENDMAC PUMP
		C	271		
		C	272		MVC DCT0\$,savedct,80 ; Save DCT before sysgen is loaded.
		C+	272		scope
004A8D 21 70 04		C+	272		ld hl,DCT0\$
004A90 11 26 89		C+	272		ld de,savedct
004A93 015000		C+	272		ld bc,80
004A96 EDB0		C+	272		LDIR
		C+	272		ENDMAC MVC
		C	273		
004A98 11 E0 00		C	274		LD DE,CFGFCB\$ ; Set up to load config.
		C	275		GOSUB zLOAD ; Go to load config.
		C+	275		scope
004A9B CD 38 1B		C+	275		CALL zLOAD
		C+	275		endmac GOSUB
		C	276		IF_NE_GOTO test_sysgen ; error loading sysgen - NZ means error loading.
		C+	276		scope
004A9E C2 5F 49		C+	276		jp nz,test_sysgen
		C+	276		endmac IF_NE_GOTO
		C	277		
		C	278		MVC savedct,DCT0\$,80
		C+	278		scope
004AA1 21 26 89		C+	278		ld hl,savedct
004AA4 11 70 04		C+	278		ld de,DCT0\$
004AA7 015000		C+	278		ld bc,80
004AAA EDB0		C+	278		LDIR
		C+	278		ENDMAC MVC
		C	279		PUMP DEVICE.console,ldingdn\$ ; Show file sysgen downloaded.
		C+	279		scope ; A tribute to Dave Shriner - Display Data Corp.
004AAC 21 83 85		C+	279		LD HL,ldingdn\$ ; Pump from this starting point until ETX.asc is found.
		C+	279		if DEVICE.console=0
004AAF 7E		C+	279	__2190:	LD A,(HL) ; Get byte to send.
004AB0 23		C+	279		INC HL ; pointer=pointer+1
		C+	279	;	CP EOT.asc ; End Of Transmission (block)?
		C+	279	;	jr Z,\$0a ; Return if finished.
004AB1 FE03		C+	279		CP ETX.asc ; End Of Text?
004AB3 28 16		C+	279		jr Z,__2190a ; Return if finished.
004AB5 FE0D		C+	279		CP CR.asc ; Was it a CR?
004AB7 CC C1 4A		C+	279		call z,__2191 ; 1st add a LF to stream.
004ABA 4F		C+	279		LD C,A ; Output byte in C.
004ABB AF		C+	279		XOR A ; Set Z flag for put.
004ABC CD D9 10		C+	279		CALL U0DVR ; Out the UART we go.
004ABF 18 EE		C+	279		JR __2190
004AC1 3E0A		C+	279	__2191:	ld a,LF.asc ; Get a LF.
004AC3 4F		C+	279		LD C,A ; Output byte in C.
004AC4 AF		C+	279		XOR A ; Set Z flag for put

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-DOS.s
004AC5	CD D9 10	C+	279	CALL	U0DVR ; Out the UART we go.
004AC8	3E0D	C+	279	ld	a,CR.asc ; Now put a CR back in stream.
004ACA	C9	C+	279	ret	; Return and print CR.
004ACB		C+	279	__2190a	endif
		C+	279	if	DEVICE.console=1
		C+	279	__2191:	LD A,(HL) ; Get byte to send.
		C+	279	INC	HL ; pointer=pointer+1
		C+	279	; CP	EOT.asc ; End Of Transmission (block)?
		C+	279	; jr	Z,\$1a ; Return if finished.
		C+	279	CP	ETX.asc ; End Of Text?
		C+	279	jr	Z,__2191a ; Return if finished.
		C+	279	LD	C,A ; Output byte to C.
		C+	279	XOR	A ; Set Z flag for put.
		C+	279	CALL	U1DVR ; Out the UART we go.
		C+	279	JR	__2191
		C+	279	__2191a	equ \$
		C+	279	endif	
		C+	279	ENDMAC	PUMP
		C	280		
004ACB		C	281	start_int	PUMP DEVICE.console,exinitial\$ ; Tell operator we are calling initializing cha
		C+	281	scope	; A tribute to Dave Shriner - Display Data Corp.
004ACB	21 2F 86	C+	281	LD	HL,exinitial\$ ; Pump from this starting point until ETX.asc is found.
		C+	281	if	DEVICE.console=0
004ACE	7E	C+	281	__2200:	LD A,(HL) ; Get byte to send.
004ACF	23	C+	281	INC	HL ; pointer=pointer+1
		C+	281	; CP	EOT.asc ; End Of Transmission (block)?
		C+	281	; jr	Z,\$0a ; Return if finished.
		C+	281	CP	ETX.asc ; End Of Text?
		C+	281	jr	Z,__2200a ; Return if finished.
		C+	281	CP	CR.asc ; Was it a CR?
		C+	281	call	z,__2201 ; 1st add a LF to stream.
		C+	281	LD	C,A ; Output byte in C.
		C+	281	XOR	A ; Set Z flag for put.
		C+	281	CALL	U0DVR ; Out the UART we go.
		C+	281	JR	__2200
		C+	281	__2201:	ld a,LF.asc ; Get a LF.
		C+	281	LD	C,A ; Output byte in C.
		C+	281	XOR	A ; Set Z flag for put
		C+	281	CALL	U0DVR ; Out the UART we go.
		C+	281	ld	a,CR.asc ; Now put a CR back in stream.
		C+	281	ret	; Return and print CR.
		C+	281	__2200a	endif
		C+	281	if	DEVICE.console=1
		C+	281	__2201:	LD A,(HL) ; Get byte to send.
		C+	281	INC	HL ; pointer=pointer+1
		C+	281	; CP	EOT.asc ; End Of Transmission (block)?
		C+	281	; jr	Z,\$1a ; Return if finished.
		C+	281	CP	ETX.asc ; End Of Text?
		C+	281	jr	Z,__2201a ; Return if finished.
		C+	281	LD	C,A ; Output byte to C.
		C+	281	XOR	A ; Set Z flag for put.
		C+	281	CALL	U1DVR ; Out the UART we go.
		C+	281	JR	__2201
		C+	281	__2201a	equ \$
		C+	281	endif	
		C+	281	ENDMAC	PUMP
		C	282	GOSUB	zICNFG ; Here we start initilization chain.
		C+	282	scope	
004AEA	CD 86 00	C+	282	CALL	zICNFG
		C+	282	endmac	GOSUB

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-DOS.s
		C	283		
		C	284		PUMP DEVICE.console,initialdne\$ ; Tell operator initialization done.
		C+	284		scope ; A tribute to Dave Shriner - Display Data Corp.
004AED	21 56 86	C+	284		LD HL,initialdne\$ ; Pump from this starting point until ETX.asc is found.
		C+	284		if DEVICE.console=0
004AF0	7E	C+	284	__2220:	LD A,(HL) ; Get byte to send.
004AF1	23	C+	284		INC HL ; pointer=pointer+1
		C+	284		CP EOT.asc ; End Of Transmission (block)?
		C+	284		; jr Z,\$0a ; Return if finished.
004AF2	FE03	C+	284		CP ETX.asc ; End Of Text?
004AF4	28 16	C+	284		jr Z,__2220a ; Return if finished.
004AF6	FE0D	C+	284		CP CR.asc ; Was it a CR?
004AF8	CC 02 4B	C+	284		call z,__2221 ; 1st add a LF to stream.
004AFB	4F	C+	284		LD C,A ; Output byte in C.
004AFC	AF	C+	284		XOR A ; Set Z flag for put.
004AFD	CD D9 10	C+	284		CALL U0DVR ; Out the UART we go.
004B00	18 EE	C+	284		JR __2220
004B02	3E0A	C+	284	__2221:	ld a,LF.asc ; Get a LF.
004B04	4F	C+	284		LD C,A ; Output byte in C.
004B05	AF	C+	284		XOR A ; Set Z flag for put
004B06	CD D9 10	C+	284		CALL U0DVR ; Out the UART we go.
004B09	3E0D	C+	284		ld a,CR.asc ; Now put a CR back in stream.
004B0B	C9	C+	284		ret ; Return and print CR.
004B0C		C+	284	__2220a	endif
		C+	284		if DEVICE.console=1
		C+	284	__2221:	LD A,(HL) ; Get byte to send.
		C+	284		INC HL ; pointer=pointer+1
		C+	284		CP EOT.asc ; End Of Transmission (block)?
		C+	284		; jr Z,\$1a ; Return if finished.
		C+	284		CP ETX.asc ; End Of Text?
		C+	284		jr Z,__2221a ; Return if finished.
		C+	284		LD C,A ; Output byte to C.
		C+	284		XOR A ; Set Z flag for put.
		C+	284		CALL U1DVR ; Out the UART we go.
		C+	284		JR __2221
		C+	284	__2221a	equ \$
		C+	284		endif
		C+	284		ENDMAC PUMP
		C	285		
		C	286		DELAY 75 ; Let this awesumness settle in a bit on user viewing display ;
		C+	286		scope
		C+	286		IFMA 75
		C+	286		call delay ; Call delay using default value.
		C+	286		ELSE
004B0C	064B	C+	286		ld b,75 ; Load delay value.
004B0E	CD 8C 6F	C+	286		call delay+2 ; Skip into delay routine, not to load default value.
		C+	286		ENDIF
		C+	286		ENDMAC DELAY ; End of.
		C	287		
		C	288		PUMP DEVICE.console,crlf
		C+	288		scope ; A tribute to Dave Shriner - Display Data Corp.
004B11	21 FA 86	C+	288		LD HL,crlf ; Pump from this starting point until ETX.asc is found.
		C+	288		if DEVICE.console=0
004B14	7E	C+	288	__2240:	LD A,(HL) ; Get byte to send.
004B15	23	C+	288		INC HL ; pointer=pointer+1
		C+	288		CP EOT.asc ; End Of Transmission (block)?
		C+	288		; jr Z,\$0a ; Return if finished.
004B16	FE03	C+	288		CP ETX.asc ; End Of Text?
004B18	28 16	C+	288		jr Z,__2240a ; Return if finished.
004B1A	FE0D	C+	288		CP CR.asc ; Was it a CR?

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-DOS.s
004B1C	CC 26 4B	C+	288	call	z, __2241 ; 1st add a LF to stream.
004B1F	4F	C+	288	LD	C,A ; Output byte in C.
004B20	AF	C+	288	XOR	A ; Set Z flag for put.
004B21	CD D9 10	C+	288	CALL	U0DVR ; Out the UART we go.
004B24	18 EE	C+	288	JR	__2240
004B26	3E0A	C+	288	__2241:	ld a, LF.asc ; Get a LF.
004B28	4F	C+	288	LD	C,A ; Output byte in C.
004B29	AF	C+	288	XOR	A ; Set Z flag for put
004B2A	CD D9 10	C+	288	CALL	U0DVR ; Out the UART we go.
004B2D	3E0D	C+	288	ld	a, CR.asc ; Now put a CR back in stream.
004B2F	C9	C+	288	ret	; Return and print CR.
004B30		C+	288	__2240a	endif
		C+	288		if DEVICE.console=1
		C+	288	__2241:	LD A, (HL) ; Get byte to send.
		C+	288		INC HL ; pointer=pointer+1
		C+	288		; CP EOT.asc ; End Of Transmission (block)?
		C+	288		; jr Z, \$1a ; Return if finished.
		C+	288		CP ETX.asc ; End Of Text?
		C+	288		jr Z, __2241a ; Return if finished.
		C+	288		LD C,A ; Output byte to C.
		C+	288		XOR A ; Set Z flag for put.
		C+	288		CALL U1DVR ; Out the UART we go.
		C+	288		JR __2241
		C+	288	__2241a	equ \$
		C+	288		endif
		C+	288		ENDMAC PUMP
		C	289	PUMP	DEVICE.console, bkgrd_start\$ ; Tell em starting task processor on console.
		C+	289	scope	; A tribute to Dave Shriner - Display Data Corp.
004B30	21 F7 85	C+	289	LD	HL, bkgrd_start\$ ; Pump from this starting point until ETX.asc is found.
		C+	289		if DEVICE.console=0
		C+	289	__2250:	LD A, (HL) ; Get byte to send.
		C+	289		INC HL ; pointer=pointer+1
		C+	289		; CP EOT.asc ; End Of Transmission (block)?
		C+	289		; jr Z, \$0a ; Return if finished.
		C+	289		CP ETX.asc ; End Of Text?
		C+	289		jr Z, __2250a ; Return if finished.
		C+	289		CP CR.asc ; Was it a CR?
		C+	289		call z, __2251 ; 1st add a LF to stream.
		C+	289		LD C,A ; Output byte in C.
		C+	289		XOR A ; Set Z flag for put.
		C+	289		CALL U0DVR ; Out the UART we go.
		C+	289		JR __2250
		C+	289	__2251:	ld a, LF.asc ; Get a LF.
		C+	289		LD C,A ; Output byte in C.
		C+	289		XOR A ; Set Z flag for put
		C+	289		CALL U0DVR ; Out the UART we go.
		C+	289		ld a, CR.asc ; Now put a CR back in stream.
		C+	289		ret ; Return and print CR.
		C+	289	__2250a	endif
		C+	289		if DEVICE.console=1
		C+	289	__2251:	LD A, (HL) ; Get byte to send.
		C+	289		INC HL ; pointer=pointer+1
		C+	289		; CP EOT.asc ; End Of Transmission (block)?
		C+	289		; jr Z, \$1a ; Return if finished.
		C+	289		CP ETX.asc ; End Of Text?
		C+	289		jr Z, __2251a ; Return if finished.
		C+	289		LD C,A ; Output byte to C.
		C+	289		XOR A ; Set Z flag for put.
		C+	289		CALL U1DVR ; Out the UART we go.
		C+	289		JR __2251

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-DOS.s
		C+	289	__2251a	equ \$
		C+	289		endif
		C+	289		ENDMAC PUMP
		C	290		
		C	291		DELAY 75
		C+	291		scope
		C+	291		IFMA 75
		C+	291		call delay ; Call delay using default value.
		C+	291		ELSE
004B4F	064B	C+	291		ld b,75 ; Load delay value.
004B51	CD 8C 6F	C+	291		call delay+2 ; Skip into delay routine, not to load default value.
		C+	291		ENDIF
		C+	291		ENDMAC DELAY ; End of.
		C	292		
004B54	1664	C	293		ld d,100 ; MAX retries testing if real-time running.
004B56	FB	C	294	ck_start	ei ; Will change to EI if background starts, see above.
004B57	010000	C	295		ld bc,0
004B5A	03	C	296	waitonint	inc bc ; BC = BC + 1.
004B5B	3A 80 F7	C	297		ld a,(int_tmr1_cnt) ; Get interrupt counter.
004B5E	CB7F	C	298		bit 7,a ; wait for about 128 iterations.
		C	299		IF_NE_BRANCH report_start ; Loop until we detect a 80 in LSB of counter.
		C+	299		scope
004B60	20 2B	C+	299		jr nz,report_start
		C+	299		endmac IF_NE_BRANCH
004B62	AF	C	300		xor a ; Check registers B & C.
004B63	B8	C	301		cp b ; When both registers = 0 at same time
		C	302		IF_NE_BRANCH waitonint ; we have looped and tested interrupt counter
		C+	302		scope
004B64	20 F4	C+	302		jr nz,waitonint
		C+	302		endmac IF_NE_BRANCH
004B66	B9	C	303		cp c ; 65535 times. If counter has not reached 80h by then
		C	304		IF_NE_BRANCH waitonint ; PRT1 must not be running.
		C+	304		scope
004B67	20 F1	C+	304		jr nz,waitonint
		C+	304		endmac IF_NE_BRANCH
004B69	15	C	305		dec d ; Loop testing d times.
		C	306		IF_NE_BRANCH ck_start ; Continue testing if PRT has started.
		C+	306		scope
004B6A	20 EA	C+	306		jr nz,ck_start
		C+	306		endmac IF_NE_BRANCH
		C	307		
004B6C		C	308	report_fail	PUMP DEVICE.console,bkgrd_fail\$ ; PRT1 failed to start, tell operator backgroun
		C+	308		scope ; A tribute to Dave Shriner - Display Data Corp.
004B6C	21 21 86	C+	308		LD HL,bkgrd_fail\$ ; Pump from this starting point until ETX.asc is found.
		C+	308		if DEVICE.console=0
004B6F	7E	C+	308	__2310:	LD A,(HL) ; Get byte to send.
004B70	23	C+	308		INC HL ; pointer=pointer+1
		C+	308		; CP EOT.asc ; End Of Transmission (block)?
		C+	308		; jr Z,\$0a ; Return if finished.
004B71	FE03	C+	308		CP ETX.asc ; End Of Text?
004B73	28 16	C+	308		jr Z,__2310a ; Return if finished.
004B75	FE0D	C+	308		CP CR.asc ; Was it a CR?
004B77	CC 81 4B	C+	308		call z,__2311 ; 1st add a LF to stream.
004B7A	4F	C+	308		LD C,A ; Output byte in C.
004B7B	AF	C+	308		XOR A ; Set Z flag for put.
004B7C	CD D9 10	C+	308		CALL U0DVR ; Out the UART we go.
004B7F	18 EE	C+	308		JR __2310
004B81	3E0A	C+	308	__2311:	ld a,LF.asc ; Get a LF.
004B83	4F	C+	308		LD C,A ; Output byte in C.
004B84	AF	C+	308		XOR A ; Set Z flag for put



\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-DOS.s
004B85	CD D9 10	C+	308	CALL	U0DVR ; Out the UART we go.
004B88	3E0D	C+	308	ld	a,CR.asc ; Now put a CR back in stream.
004B8A	C9	C+	308	ret	; Return and print CR.
004B8B		C+	308	__2310a	endif
		C+	308	if	DEVICE.console=1
		C+	308	__2311:	LD A,(HL) ; Get byte to send.
		C+	308	INC	HL ; pointer=pointer+1
		C+	308	; CP	EOT.asc ; End Of Transmission (block)?
		C+	308	; jr	Z,\$1a ; Return if finished.
		C+	308	CP	ETX.asc ; End Of Text?
		C+	308	jr	Z,__2311a ; Return if finished.
		C+	308	LD	C,A ; Output byte to C.
		C+	308	XOR	A ; Set Z flag for put.
		C+	308	CALL	U1DVR ; Out the UART we go.
		C+	308	JR	__2311
		C+	308	__2311a	equ \$
		C+	308	endif	
		C+	308	ENDMAC	PUMP
		C	309	BRANCH	start_trsdos
		C+	309	scope	
004B8B	18 24	C+	309	jr	start_trsdos
		C+	309	endmac	BRANCH
		C	310		
004B8D		C	311	report_start	PUMP DEVICE.console,bkgrd_run\$ ; Tell operator background running.
		C+	311	scope	; A tribute to Dave Shriner - Display Data Corp.
004B8D	21 13 86	C+	311	LD	HL,bkgrd_run\$ ; Pump from this starting point until ETX.asc is found.
		C+	311	if	DEVICE.console=0
004B90	7E	C+	311	__2330:	LD A,(HL) ; Get byte to send.
004B91	23	C+	311	INC	HL ; pointer=pointer+1
		C+	311	; CP	EOT.asc ; End Of Transmission (block)?
		C+	311	; jr	Z,\$0a ; Return if finished.
004B92	FE03	C+	311	CP	ETX.asc ; End Of Text?
004B94	28 16	C+	311	jr	Z,__2330a ; Return if finished.
004B96	FE0D	C+	311	CP	CR.asc ; Was it a CR?
004B98	CC A2 4B	C+	311	call	z,__2331 ; 1st add a LF to stream.
004B9B	4F	C+	311	LD	C,A ; Output byte in C.
004B9C	AF	C+	311	XOR	A ; Set Z flag for put.
004B9D	CD D9 10	C+	311	CALL	U0DVR ; Out the UART we go.
004BA0	18 EE	C+	311	JR	__2330
004BA2	3E0A	C+	311	__2331:	ld a,LF.asc ; Get a LF.
004BA4	4F	C+	311	LD	C,A ; Output byte in C.
004BA5	AF	C+	311	XOR	A ; Set Z flag for put
004BA6	CD D9 10	C+	311	CALL	U0DVR ; Out the UART we go.
004BA9	3E0D	C+	311	ld	a,CR.asc ; Now put a CR back in stream.
004BAB	C9	C+	311	ret	; Return and print CR.
004BAC		C+	311	__2330a	endif
		C+	311	if	DEVICE.console=1
		C+	311	__2331:	LD A,(HL) ; Get byte to send.
		C+	311	INC	HL ; pointer=pointer+1
		C+	311	; CP	EOT.asc ; End Of Transmission (block)?
		C+	311	; jr	Z,\$1a ; Return if finished.
		C+	311	CP	ETX.asc ; End Of Text?
		C+	311	jr	Z,__2331a ; Return if finished.
		C+	311	LD	C,A ; Output byte to C.
		C+	311	XOR	A ; Set Z flag for put.
		C+	311	CALL	U1DVR ; Out the UART we go.
		C+	311	JR	__2331
		C+	311	__2331a	equ \$
		C+	311	endif	
		C+	311	ENDMAC	PUMP

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	..\\SYS0_IPL\\ipl-DOS.s
		C	312		
		C	313	DELAY 75	; Let this awesumness settle in a bit on user viewing display ;
		C+	313	scope	
		C+	313	IFMA 75	
		C+	313	call delay	; Call delay using default value.
		C+	313	ELSE	
004BAC	064B	C+	313	ld b,75	; Load delay value.
004BAE	CD 8C 6F	C+	313	call delay+2	; Skip into delay routine, not to load default value.
		C+	313	ENDIF	
		C+	313	ENDMAC DELAY	; End of.
		B	25		
		C	0	include "ipl-trsdos.s"	; Final initialization of TRSDOS/auto comma
		C	1		
		C	2		
004BB1		C	3	start_trsdos PUMP DEVICE.console,ipldone\$	; Report IPL done & splash 6.3.1 startup sc
		C+	3	scope	; A tribute to Dave Shriner - Display Data Corp.
004BB1	21 8A 86	C+	3	LD HL,ipldone\$	; Pump from this starting point until ETX.asc is found.
		C+	3	if DEVICE.console=0	
004BB4	7E	C+	3	__2350: LD A,(HL)	; Get byte to send.
004BB5	23	C+	3	INC HL	; pointer=pointer+1
		C+	3	; CP EOT.asc	; End Of Transmission (block)?
		C+	3	; jr Z,\$0a	; Return if finished.
004BB6	FE03	C+	3	CP ETX.asc	; End Of Text?
004BB8	28 16	C+	3	jr Z,__2350a	; Return if finished.
004BBA	FE0D	C+	3	CP CR.asc	; Was it a CR?
004BBC	CC C6 4B	C+	3	call z,__2351	; 1st add a LF to stream.
004BBF	4F	C+	3	LD C,A	; Output byte in C.
004BC0	AF	C+	3	XOR A	; Set Z flag for put.
004BC1	CD D9 10	C+	3	CALL U0DVR	; Out the UART we go.
004BC4	18 EE	C+	3	JR __2350	
004BC6	3E0A	C+	3	__2351: ld a,LF.asc	; Get a LF.
004BC8	4F	C+	3	LD C,A	; Output byte in C.
004BC9	AF	C+	3	XOR A	; Set Z flag for put
004BCA	CD D9 10	C+	3	CALL U0DVR	; Out the UART we go.
004BCD	3E0D	C+	3	ld a,CR.asc	; Now put a CR back in stream.
004BCF	C9	C+	3	ret	; Return and print CR.
004BD0		C+	3	__2350a endif	
		C+	3	if DEVICE.console=1	
		C+	3	__2351: LD A,(HL)	; Get byte to send.
		C+	3	INC HL	; pointer=pointer+1
		C+	3	; CP EOT.asc	; End Of Transmission (block)?
		C+	3	; jr Z,\$1a	; Return if finished.
		C+	3	CP ETX.asc	; End Of Text?
		C+	3	jr Z,__2351a	; Return if finished.
		C+	3	LD C,A	; Output byte to C.
		C+	3	XOR A	; Set Z flag for put.
		C+	3	CALL U1DVR	; Out the UART we go.
		C+	3	JR __2351	
		C+	3	__2351a equ \$	
		C+	3	endif	
		C+	3	ENDMAC PUMP	
		C	4		
		C	5	DELAY 100	; Delay a bit, let this awesumness settle in a bit on user view
		C+	5	scope	
		C+	5	IFMA 100	
		C+	5	call delay	; Call delay using default value.
		C+	5	ELSE	
004BD0	0664	C+	5	ld b,100	; Load delay value.
004BD2	CD 8C 6F	C+	5	call delay+2	; Skip into delay routine, not to load default value.
		C+	5	ENDIF	

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-trsdos.s
		C+	5	ENDMAC	DELAY ; End of.
		C	6		
		C	7	DATE.get	lsdosdate ; Put current date in IPL screen.
		C+	7	scope	
004BD5	21 05 77	C+	7	ld	hl,lsdosdate
		C+	7	SVC	svc_date
004BD8	3E12	C+	7	ld	a,svc_date
004BDA	EF	C+	7	restart28	
		C+	7	ENDMAC	SVC
		C+	7	ENDMAC	DATE.get
		C	8	TIME.get	lsdostime ; Put current time in IPL screen.
		C+	8	scope	
004BDB	21 35 77	C+	8	ld	hl,lsdostime
		C+	8	SVC	svc_time
004BDE	3E13	C+	8	ld	a,svc_time
004BE0	EF	C+	8	restart28	
		C+	8	ENDMAC	SVC
		C+	8	ENDMAC	TIME.get
		C	9		
		C	10	PUMP	DEVICE.console,vt100_home\$ ; Cursor upper left corner.
		C+	10	scope	; A tribute to Dave Shriner - Display Data Corp.
004BE1	21 05 87	C+	10	LD	HL,vt100_home\$ ; Pump from this starting point until ETX.asc is found.
		C+	10	if	DEVICE.console=0
004BE4	7E	C+	10	LD	A,(HL) ; Get byte to send.
004BE5	23	C+	10	INC	HL ; pointer=pointer+1
		C+	10	CP	EOT.asc ; End Of Transmission (block)?
		C+	10	jr	Z,\$0a ; Return if finished.
004BE6	FE03	C+	10	CP	ETX.asc ; End Of Text?
004BE8	28 16	C+	10	jr	Z,__2410a ; Return if finished.
004BEA	FE0D	C+	10	CP	CR.asc ; Was it a CR?
004BEC	CC F6 4B	C+	10	call	z,__2411 ; 1st add a LF to stream.
004BEF	4F	C+	10	LD	C,A ; Output byte in C.
004BF0	AF	C+	10	XOR	A ; Set Z flag for put.
004BF1	CD D9 10	C+	10	CALL	U0DVR ; Out the UART we go.
004BF4	18 EE	C+	10	JR	__2410
004BF6	3E0A	C+	10	LD	a,LF.asc ; Get a LF.
004BF8	4F	C+	10	LD	C,A ; Output byte in C.
004BF9	AF	C+	10	XOR	A ; Set Z flag for put
004BFA	CD D9 10	C+	10	CALL	U0DVR ; Out the UART we go.
004BFD	3E0D	C+	10	ld	a,CR.asc ; Now put a CR back in stream.
004BFF	C9	C+	10	ret	; Return and print CR.
004C00		C+	10	__2410a	endif
		C+	10	if	DEVICE.console=1
		C+	10	__2411:	LD A,(HL) ; Get byte to send.
		C+	10	INC	HL ; pointer=pointer+1
		C+	10	CP	EOT.asc ; End Of Transmission (block)?
		C+	10	jr	Z,\$1a ; Return if finished.
		C+	10	CP	ETX.asc ; End Of Text?
		C+	10	jr	Z,__2411a ; Return if finished.
		C+	10	LD	C,A ; Output byte to C.
		C+	10	XOR	A ; Set Z flag for put.
		C+	10	CALL	U1DVR ; Out the UART we go.
		C+	10	JR	__2411
		C+	10	__2411a	equ \$
		C+	10	endif	
		C+	10	ENDMAC	PUMP
		C	11	PUMP	DEVICE.console,vt100_clear\$ ; Clear video screen on terminal...make nice clean.
		C+	11	scope	; A tribute to Dave Shriner - Display Data Corp.
004C00	21 01 87	C+	11	LD	HL,vt100_clear\$ ; Pump from this starting point until ETX.asc is found.
		C+	11	if	DEVICE.console=0

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-trsdos.s
004C03	7E	C+	11	__2420:	LD A,(HL) ; Get byte to send.
004C04	23	C+	11		INC HL ; pointer=pointer+1
		C+	11		; CP EOT.asc ; End Of Transmission (block)?
		C+	11		; jr Z,\$0a ; Return if finished.
004C05	FE03	C+	11		CP ETX.asc ; End Of Text?
004C07	28 16	C+	11		jr Z,__2420a ; Return if finished.
004C09	FE0D	C+	11		CP CR.asc ; Was it a CR?
004C0B	CC 15 4C	C+	11		call z,__2421 ; 1st add a LF to stream.
004C0E	4F	C+	11		LD C,A ; Output byte in C.
004C0F	AF	C+	11		XOR A ; Set Z flag for put.
004C10	CD D9 10	C+	11		CALL U0DVR ; Out the UART we go.
004C13	18 EE	C+	11		JR __2420
004C15	3E0A	C+	11	__2421:	ld a,LF.asc ; Get a LF.
004C17	4F	C+	11		LD C,A ; Output byte in C.
004C18	AF	C+	11		XOR A ; Set Z flag for put
004C19	CD D9 10	C+	11		CALL U0DVR ; Out the UART we go.
004C1C	3E0D	C+	11		ld a,CR.asc ; Now put a CR back in stream.
004C1E	C9	C+	11		ret ; Return and print CR.
004C1F		C+	11	__2420a	endif
		C+	11		if DEVICE.console=1
		C+	11	__2421:	LD A,(HL) ; Get byte to send.
		C+	11		INC HL ; pointer=pointer+1
		C+	11		; CP EOT.asc ; End Of Transmission (block)?
		C+	11		; jr Z,\$1a ; Return if finished.
		C+	11		CP ETX.asc ; End Of Text?
		C+	11		jr Z,__2421a ; Return if finished.
		C+	11		LD C,A ; Output byte to C.
		C+	11		XOR A ; Set Z flag for put.
		C+	11		CALL U1DVR ; Out the UART we go.
		C+	11		JR __2421
		C+	11	__2421a	equ \$
		C+	11		endif
		C+	11		ENDMAC PUMP
		C	12		PUMP DEVICE.console,lsdos631\$-4 ; Show LS-DOS 6.3.1 LOGO.
		C+	12		scope ; A tribute to Dave Shriner - Display Data Corp.
004C1F	21 7D 74	C+	12		LD HL,lsdos631\$-4 ; Pump from this starting point until ETX.asc is found.
		C+	12		if DEVICE.console=0
004C22	7E	C+	12	__2430:	LD A,(HL) ; Get byte to send.
004C23	23	C+	12		INC HL ; pointer=pointer+1
		C+	12		; CP EOT.asc ; End Of Transmission (block)?
		C+	12		; jr Z,\$0a ; Return if finished.
004C24	FE03	C+	12		CP ETX.asc ; End Of Text?
004C26	28 16	C+	12		jr Z,__2430a ; Return if finished.
004C28	FE0D	C+	12		CP CR.asc ; Was it a CR?
004C2A	CC 34 4C	C+	12		call z,__2431 ; 1st add a LF to stream.
004C2D	4F	C+	12		LD C,A ; Output byte in C.
004C2E	AF	C+	12		XOR A ; Set Z flag for put.
004C2F	CD D9 10	C+	12		CALL U0DVR ; Out the UART we go.
004C32	18 EE	C+	12		JR __2430
004C34	3E0A	C+	12	__2431:	ld a,LF.asc ; Get a LF.
004C36	4F	C+	12		LD C,A ; Output byte in C.
004C37	AF	C+	12		XOR A ; Set Z flag for put
004C38	CD D9 10	C+	12		CALL U0DVR ; Out the UART we go.
004C3B	3E0D	C+	12		ld a,CR.asc ; Now put a CR back in stream.
004C3D	C9	C+	12		ret ; Return and print CR.
004C3E		C+	12	__2430a	endif
		C+	12		if DEVICE.console=1
		C+	12	__2431:	LD A,(HL) ; Get byte to send.
		C+	12		INC HL ; pointer=pointer+1
		C+	12		; CP EOT.asc ; End Of Transmission (block)?

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-trsdos.s
		C+	12	; jr Z,\$1a	; Return if finished.
		C+	12	CP ETX.asc	; End Of Text?
		C+	12	jr Z,__2431a	; Return if finished.
		C+	12	LD C,A	; Output byte to C.
		C+	12	XOR A	; Set Z flag for put.
		C+	12	CALL U1DVR	; Out the UART we go.
		C+	12	JR __2431	
		C+	12	__2431a equ \$	
		C+	12	endif	
		C+	12	ENDMAC PUMP	
		C	13	MVC lsdos631\$,zspace.VIDEO\$(splash\$_len)	; Place a copy of this in video RAM also.
		C+	13	scope	
004C3E	21 81 74	C+	13	ld hl,lsdos631\$	
004C41	1100F8	C+	13	ld de,zspace.VIDEO\$	
004C44	ED4B 50 77	C+	13	ld bc,(splash\$_len)	
004C48	EDB0	C+	13	LDIR	
		C+	13	ENDMAC MVC	
		C	14		
		C	15	DELAY 125	
		C+	15	scope	
		C+	15	IFMA 125	
		C+	15	call delay	; Call delay using default value.
		C+	15	ELSE	
004C4A	067D	C+	15	ld b,125	; Load delay value.
004C4C	CD 8C 6F	C+	15	call delay+2	; Skip into delay routine, not to load default value.
		C+	15	ENDIF	
		C+	15	ENDMAC DELAY	; End of.
		C	16		
004C4F	3A 0A 87	C	17	ld a,(quiet_default+1)	; Get original instruction at beginning of UART driver.
004C52	32 D9 10	C	18	ld (U0DVR),a	; Stuff in driver to renable driver.
		C	19		
004C55		C	20	real_time_videocls	; I just cannot get it to work correctly if I dont CLS.
		C+	20	scope	
		C+	20	SVC svc_cls	
004C55	3E69	C+	20	ld a,svc_cls	
004C57	EF	C+	20	restart28	
		C+	20	ENDMAC SVC	
		C+	20	ENDMAC cls	
		C	21		
004C58	3A 85 F7	C	22	ld a,(pump_switch_value)	; Get value to set pump switch to.
004C5B	32 40 F7	C	23	ld (pump_switch),a	; When pump switch is updated it uses this value.
		C	24		
004C5E	3A BD 71	C	25	ld a,(autoflg\$)	; Get auto.
004C61	FE79	C	26	cp 'y'	; Auto command available?
		C	27	IF_NE_BRANCH no_auto	
		C+	27	scope	
004C63	20 52	C+	27	jr nz,no_auto	
		C+	27	endmac IF_NE_BRANCH	
		C	28		
004C65	3A FF 86	C	29	ld a,(safeipl\$)	; Get safe flag.
004C68	FE79	C	30	cp 'y'	; Are we in safe mode?
		C	31	IF_NE_BRANCH run_auto	; Handle AUTO command? If safe mode no.
		C+	31	scope	
004C6A	20 45	C+	31	jr nz,run_auto	
		C+	31	endmac IF_NE_BRANCH	
		C	32		
		C	33	PUMP DEVICE.console,crlf+1	
		C+	33	scope	; A tribute to Dave Shriner - Display Data Corp.
004C6C	21 FB 86	C+	33	LD HL,crlf+1	; Pump from this starting point until ETX.asc is found.
		C+	33	if DEVICE.console=0	

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-trsdos.s
004C6F	7E	C+	33	__2500:	LD A,(HL) ; Get byte to send.
004C70	23	C+	33		INC HL ; pointer=pointer+1
		C+	33		; CP EOT.asc ; End Of Transmission (block)?
		C+	33		; jr Z,\$0a ; Return if finished.
004C71	FE03	C+	33		CP ETX.asc ; End Of Text?
004C73	28 16	C+	33		jr Z,__2500a ; Return if finished.
004C75	FE0D	C+	33		CP CR.asc ; Was it a CR?
004C77	CC 81 4C	C+	33		call z,__2501 ; 1st add a LF to stream.
004C7A	4F	C+	33		LD C,A ; Output byte in C.
004C7B	AF	C+	33		XOR A ; Set Z flag for put.
004C7C	CD D9 10	C+	33		CALL U0DVR ; Out the UART we go.
004C7F	18 EE	C+	33		JR __2500
004C81	3E0A	C+	33	__2501:	ld a,LF.asc ; Get a LF.
004C83	4F	C+	33		LD C,A ; Output byte in C.
004C84	AF	C+	33		XOR A ; Set Z flag for put
004C85	CD D9 10	C+	33		CALL U0DVR ; Out the UART we go.
004C88	3E0D	C+	33		ld a,CR.asc ; Now put a CR back in stream.
004C8A	C9	C+	33		ret ; Return and print CR.
004C8B		C+	33	__2500a	endif
		C+	33		if DEVICE.console=1
		C+	33	__2501:	LD A,(HL) ; Get byte to send.
		C+	33		INC HL ; pointer=pointer+1
		C+	33		; CP EOT.asc ; End Of Transmission (block)?
		C+	33		; jr Z,\$1a ; Return if finished.
		C+	33		CP ETX.asc ; End Of Text?
		C+	33		jr Z,__2501a ; Return if finished.
		C+	33		LD C,A ; Output byte to C.
		C+	33		XOR A ; Set Z flag for put.
		C+	33		CALL U1DVR ; Out the UART we go.
		C+	33		JR __2501
		C+	33	__2501a	equ \$
		C+	33		endif
		C+	33		ENDMAC PUMP
		C	34		PUMP DEVICE.console,runauto\$ ; Prompt, run AUTO?
		C+	34		scope ; A tribute to Dave Shriner - Display Data Corp.
004C8B	21 72 86	C+	34		LD HL,runauto\$ ; Pump from this starting point until ETX.asc is found.
		C+	34		if DEVICE.console=0
004C8E	7E	C+	34	__2510:	LD A,(HL) ; Get byte to send.
004C8F	23	C+	34		INC HL ; pointer=pointer+1
		C+	34		; CP EOT.asc ; End Of Transmission (block)?
		C+	34		; jr Z,\$0a ; Return if finished.
004C90	FE03	C+	34		CP ETX.asc ; End Of Text?
004C92	28 16	C+	34		jr Z,__2510a ; Return if finished.
004C94	FE0D	C+	34		CP CR.asc ; Was it a CR?
004C96	CC A0 4C	C+	34		call z,__2511 ; 1st add a LF to stream.
004C99	4F	C+	34		LD C,A ; Output byte in C.
004C9A	AF	C+	34		XOR A ; Set Z flag for put.
004C9B	CD D9 10	C+	34		CALL U0DVR ; Out the UART we go.
004C9E	18 EE	C+	34		JR __2510
004CA0	3E0A	C+	34	__2511:	ld a,LF.asc ; Get a LF.
004CA2	4F	C+	34		LD C,A ; Output byte in C.
004CA3	AF	C+	34		XOR A ; Set Z flag for put
004CA4	CD D9 10	C+	34		CALL U0DVR ; Out the UART we go.
004CA7	3E0D	C+	34		ld a,CR.asc ; Now put a CR back in stream.
004CA9	C9	C+	34		ret ; Return and print CR.
004CAA		C+	34	__2510a	endif
		C+	34		if DEVICE.console=1
		C+	34	__2511:	LD A,(HL) ; Get byte to send.
		C+	34		INC HL ; pointer=pointer+1
		C+	34		; CP EOT.asc ; End Of Transmission (block)?

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-trsdos.s
		C+	34	; jr Z,\$1a	; Return if finished.
		C+	34	CP ETX.asc	; End Of Text?
		C+	34	jr Z,__2511a	; Return if finished.
		C+	34	LD C,A	; Output byte to C.
		C+	34	XOR A	; Set Z flag for put.
		C+	34	CALL U1DVR	; Out the UART we go.
		C+	34	JR __2511	
		C+	34	__2511a equ \$	
		C+	34	endif	
		C+	34	ENDMAC PUMP	
004CAA	CD 38 6F	C	35	call GET.yn	; Get response.
004CAD	FE79	C	36	cp 'y'	
		C	37	IF_NE_BRANCH no_auto	; If NO, skip AUTO COMMAND.
		C+	37	scope	
004CAF	20 06	C+	37	jr nz,no_auto	
		C+	37	endmac IF_NE_BRANCH	
		C	38		
004CB1	210000	C	39	run_auto ld hl,0000h	; Recover pointer to INBUF\$.
	00004CB2	C	40	HLSav equ \$-2	; For execution of INBUF\$ pointer adjusted for '*'. ; No return from zCMNDI -->* THE END *<-- were booted!
		C	41	GOTO zCMNDI	
		C+	41	scope	
004CB4	C3 7E 19	C+	41	jp zCMNDI	
		C+	41	endmac GOTO	
		C	42		
004CB7		C	43	no_auto PUMP DEVICE.console,vt100_home\$	; Cursor upper left corner.
		C+	43	scope	; A tribute to Dave Shriner - Display Data Corp.
004CB7	21 05 87	C+	43	LD HL,vt100_home\$	; Pump from this starting point until ETX.asc is found.
		C+	43	if DEVICE.console=0	
004CBA	7E	C+	43	__2540: LD A,(HL)	; Get byte to send.
004CBB	23	C+	43	INC HL	; pointer=pointer+1
		C+	43	; CP EOT.asc	; End Of Transmission (block)?
		C+	43	; jr Z,\$0a	; Return if finished.
004CBC	FE03	C+	43	CP ETX.asc	; End Of Text?
004CBE	28 16	C+	43	jr Z,__2540a	; Return if finished.
004CC0	FE0D	C+	43	CP CR.asc	; Was it a CR?
004CC2	CC CC 4C	C+	43	call z,__2541	; 1st add a LF to stream.
004CC5	4F	C+	43	LD C,A	; Output byte in C.
004CC6	AF	C+	43	XOR A	; Set Z flag for put.
004CC7	CD D9 10	C+	43	CALL U0DVR	; Out the UART we go.
004CCA	18 EE	C+	43	JR __2540	
004CCC	3E0A	C+	43	__2541: ld a,LF.asc	; Get a LF.
004CCE	4F	C+	43	LD C,A	; Output byte in C.
004CCF	AF	C+	43	XOR A	; Set Z flag for put
004CD0	CD D9 10	C+	43	CALL U0DVR	; Out the UART we go.
004CD3	3E0D	C+	43	ld a,CR.asc	; Now put a CR back in stream.
004CD5	C9	C+	43	ret	; Return and print CR.
004CD6		C+	43	__2540a endif	
		C+	43	if DEVICE.console=1	
		C+	43	__2541: LD A,(HL)	; Get byte to send.
		C+	43	INC HL	; pointer=pointer+1
		C+	43	; CP EOT.asc	; End Of Transmission (block)?
		C+	43	; jr Z,\$1a	; Return if finished.
		C+	43	CP ETX.asc	; End Of Text?
		C+	43	jr Z,__2541a	; Return if finished.
		C+	43	LD C,A	; Output byte to C.
		C+	43	XOR A	; Set Z flag for put.
		C+	43	CALL U1DVR	; Out the UART we go.
		C+	43	JR __2541	
		C+	43	__2541a equ \$	
		C+	43	endif	

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	..\\SYS0_IPL\\ipl-trsdos.s
		C+	43	ENDMAC	PUMP
		C	44	PUMP	DEVICE.console,vt100_clear\$ ; Clear video screen on terminal...make nice clean.
		C+	44	scope	; A tribute to Dave Shriner - Display Data Corp.
004CD6	21 01 87	C+	44	LD	HL,vt100_clear\$ ; Pump from this starting point until ETX.asc is found.
		C+	44	if	DEVICE.console=0
004CD9	7E	C+	44	LD	A,(HL) ; Get byte to send.
004CDA	23	C+	44	INC	HL ; pointer=pointer+1
		C+	44	CP	EOT.asc ; End Of Transmission (block)?
		C+	44	jr	Z,\$0a ; Return if finished.
004CDB	FE03	C+	44	CP	ETX.asc ; End Of Text?
004CDD	28 16	C+	44	jr	Z,__2550a ; Return if finished.
004CDF	FE0D	C+	44	CP	CR.asc ; Was it a CR?
004CE1	CC EB 4C	C+	44	call	z,__2551 ; 1st add a LF to stream.
004CE4	4F	C+	44	LD	C,A ; Output byte in C.
004CE5	AF	C+	44	XOR	A ; Set Z flag for put.
004CE6	CD D9 10	C+	44	CALL	U0DVR ; Out the UART we go.
004CE9	18 EE	C+	44	JR	__2550
004CEB	3E0A	C+	44	LD	a,LF.asc ; Get a LF.
004CED	4F	C+	44	LD	C,A ; Output byte in C.
004CEE	AF	C+	44	XOR	A ; Set Z flag for put
004CEF	CD D9 10	C+	44	CALL	U0DVR ; Out the UART we go.
004CF2	3E0D	C+	44	LD	a,CR.asc ; Now put a CR back in stream.
004CF4	C9	C+	44	ret	; Return and print CR.
004CF5		C+	44	endif	
		C+	44	if	DEVICE.console=1
		C+	44	LD	A,(HL) ; Get byte to send.
		C+	44	INC	HL ; pointer=pointer+1
		C+	44	CP	EOT.asc ; End Of Transmission (block)?
		C+	44	jr	Z,\$1a ; Return if finished.
		C+	44	CP	ETX.asc ; End Of Text?
		C+	44	jr	Z,__2551a ; Return if finished.
		C+	44	LD	C,A ; Output byte to C.
		C+	44	XOR	A ; Set Z flag for put.
		C+	44	CALL	U1DVR ; Out the UART we go.
		C+	44	JR	__2551
		C+	44	__2551a	equ \$
		C+	44	endif	
		C+	44	ENDMAC	PUMP
		C	45		
		C	46	locate	CURSOR 22,0
		C+	46	scope	
004CF5	0603	C+	46	LD	b,3 ; Function 3, relocate cursor.
004CF7	2616	C+	46	LD	h,22
004CF9	2E00	C+	46	LD	l,0 ; Get column.
		C+	46	SVC	svc_vdctl
004CFB	3E0F	C+	46	LD	a,svc_vdctl
004CFD	EF	C+	46	restart	28
		C+	46	ENDMAC	SVC
		C+	46	ENDMAC	locate.CURSOR
		C	47		
004CFE	3A F7 83	C	48	LD	a,(session_cmd\$) ; Get 1st char.
004D01	FE20	C	49	CP	' ' ; No command?
		C	50	IF_NE_BRANCH	session_cmd
		C+	50	scope	
004D03	20 06	C+	50	jr	nz,session_cmd
		C+	50	endmac	IF_NE_BRANCH
		C	51		
		C	52	SYS_EXIT	0 ; Normal system EXIT-->* THE END *<-- were booted!
		C+	52	scope	
004D05	210000	C+	52	LD	hl,0



\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-trsdos.s
		C+	52	SVC	svc_zexit
004D08	3E16	C+	52	ld	a,svc_zexit
004D0A	EF	C+	52	restart28	
		C+	52	ENDMAC	SVC
		C+	52	ENDMAC	SYS_EXIT
		C	53		
004D0B	21 F7 83	C	54	session_cmd	ld hl,session_cmd\$ ; Pointer to session driver.
		C	55		
		C	56	GOTO	zCMNDI ; No AUTO command, normal system EXIT-->* THE END *<-- were
		C+	56	scope	
004D0E	C3 7E 19	C+	56	jp	zCMNDI
		C+	56	endmac	GOTO
		C	57		
		C	58	; >* THE END *<	
		C	0	include	"ipl-RTC.s" ; Real time clock calander setup.
004D11	CD ED 6E	C	1	RTC_IPL	call scrolloff
		C	2	MVC	screenbuf,dowbuff+3,9 ; Always start with a fresh buffer by stuffing with spaces.
		C+	2	scope	
004D14	21 DE 54	C+	2	ld	hl,screenbuf
004D17	11 B2 54	C+	2	ld	de,dowbuff+3
004D1A	010900	C+	2	ld	bc,9
004D1D	EDB0	C+	2	LDIR	
		C+	2	ENDMAC	MVC
		C	3	DATE.get	DATEBUF
		C+	3	scope	
004D1F	21 2C 55	C+	3	ld	hl,DATEBUF
		C+	3	SVC	svc_date
004D22	3E12	C+	3	ld	a,svc_date
004D24	EF	C+	3	restart28	
		C+	3	ENDMAC	SVC
		C+	3	ENDMAC	DATE.get
		C	4	TIME.get	TIMEBUF
		C+	4	scope	
004D25	21 35 55	C+	4	ld	hl,TIMEBUF
		C+	4	SVC	svc_time
004D28	3E13	C+	4	ld	a,svc_time
004D2A	EF	C+	4	restart28	
		C+	4	ENDMAC	SVC
		C+	4	ENDMAC	TIME.get
004D2B	CD DA 53	C	5	call	getdow
		C	6	STRING.table.get	A,DOWTABLE,DOWBUF
		C+	6	scope	
004D2E	7F	C+	6	ld	a,A ; Index value of get (1-127) to accumulator.
004D2F	3D	C+	6	dec	a
004D30	87	C+	6	add	a,a ; index=index*2.
004D31	5F	C+	6	ld	e,a
004D32	1600	C+	6	ld	d,0 ; d=0 e=index*2.
004D34	21 8F 56	C+	6	ld	hl,DOWTABLE ; Get base address of table.
004D37	19	C+	6	add	hl,de ; Point hl to index table.
004D38	ED07	C+	6	ld	bc,(hl) ; Pickup indexed pointer to entry.
004D3A	C5E1	C+	6	ld	hl,bc
004D3C	ED07	C+	6	ld	bc,(hl) ; 1st word of entry which is length of entry.
004D3E	23	C+	6	inc	hl ; Bump past length.
004D3F	23	C+	6	inc	hl ; Bump past length and leave hl pointing to entry.
004D40	11 22 55	C+	6	ld	de,DOWBUF ; Point DE to receiver place for data.
004D43	EDB0	C+	6	LDIR	; Copy BC bytes from (HL) to (DE).
		C+	6	ENDMAC	STRING.table.get
		C	7	PUMP	DEVICE.console,screenbuf
		C+	7	scope	; A tribute to Dave Shriner - Display Data Corp.
004D45	21 DE 54	C+	7	LD	HL,screenbuf ; Pump from this starting point until ETX.asc is found.

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-RTC.s
		C+	7		if DEVICE.console=0
004D48	7E	C+	7	__2680:	LD A,(HL) ; Get byte to send.
004D49	23	C+	7		INC HL ; pointer=pointer+1
		C+	7	;	CP EOT.asc ; End Of Transmission (block)?
		C+	7	;	jr Z,\$0a ; Return if finished.
004D4A	FE03	C+	7		CP ETX.asc ; End Of Text?
004D4C	28 16	C+	7		jr Z,__2680a ; Return if finished.
004D4E	FE0D	C+	7		CP CR.asc ; Was it a CR?
004D50	CC 5A 4D	C+	7		call z,__2681 ; 1st add a LF to stream.
004D53	4F	C+	7		LD C,A ; Output byte in C.
004D54	AF	C+	7		XOR A ; Set Z flag for put.
004D55	CD D9 10	C+	7		CALL U0DVR ; Out the UART we go.
004D58	18 EE	C+	7		JR __2680
004D5A	3E0A	C+	7	__2681:	ld a,LF.asc ; Get a LF.
004D5C	4F	C+	7		LD C,A ; Output byte in C.
004D5D	AF	C+	7		XOR A ; Set Z flag for put
004D5E	CD D9 10	C+	7		CALL U0DVR ; Out the UART we go.
004D61	3E0D	C+	7		ld a,CR.asc ; Now put a CR back in stream.
004D63	C9	C+	7		ret ; Return and print CR.
004D64		C+	7	__2680a	endif
		C+	7		if DEVICE.console=1
		C+	7	__2681:	LD A,(HL) ; Get byte to send.
		C+	7		INC HL ; pointer=pointer+1
		C+	7	;	CP EOT.asc ; End Of Transmission (block)?
		C+	7	;	jr Z,\$1a ; Return if finished.
		C+	7		CP ETX.asc ; End Of Text?
		C+	7		jr Z,__2681a ; Return if finished.
		C+	7		LD C,A ; Output byte to C.
		C+	7		XOR A ; Set Z flag for put.
		C+	7		CALL U1DVR ; Out the UART we go.
		C+	7		JR __2681
		C+	7	__2681a	equ \$
		C+	7		endif
		C+	7		ENDMAC PUMP
004D64	CD 7E 6F	C	8	prompt	call getchar
004D67	FE1B	C	9		cp ESC.asc ; If <CR> only return to main menu.
004D69	CA 00 48	C	10		jp z,main_ipl_menu
004D6C	FE31	C	11		cp '1' ; Check is operator wants to change DATE.
004D6E	CA 7E 4D	C	12		jp z,cngdat
004D71	FE32	C	13		cp '2' ; Check is operator wants to change day of week.
004D73	CA 6D 50	C	14		jp z, cngdow
004D76	FE33	C	15		cp '3' ; Test of operator wants to change time.
004D78	CA 25 52	C	16		jp z,cngtim
		C	17		
004D7B	C3 64 4D	C	18		jp prompt
		C	19		
004D7E	CD ED 6E	C	20	cngdat	call scrolloff
		C	21		DATE.get cdat
		C+	21		scope
004D81	21 E8 55	C+	21		ld hl,cdat
		C+	21		SVC svc_date
004D84	3E12	C+	21		ld a,svc_date
004D86	EF	C+	21		restart28
		C+	21		ENDMAC SVC
		C+	21		ENDMAC DATE.get
		C	22		PUMP DEVICE.console,edtdatscn
		C+	22		scope ; A tribute to Dave Shriner - Display Data Corp.
004D87	21 99 55	C+	22		LD HL,edtdatscn ; Pump from this starting point until ETX.asc is found.
		C+	22		if DEVICE.console=0
004D8A	7E	C+	22	__2710:	LD A,(HL) ; Get byte to send.

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-RTC.s
004D8B	23	C+	22	INC	HL ; pointer=pointer+1
		C+	22	;	CP EOT.asc ; End Of Transmission (block)?
		C+	22	;	jr Z,\$0a ; Return if finished.
004D8C	FE03	C+	22	CP	ETX.asc ; End Of Text?
004D8E	28 16	C+	22	jr	Z,__2710a ; Return if finished.
004D90	FE0D	C+	22	CP	CR.asc ; Was it a CR?
004D92	CC 9C 4D	C+	22	call	z,__2711 ; 1st add a LF to stream.
004D95	4F	C+	22	LD	C,A ; Output byte in C.
004D96	AF	C+	22	XOR	A ; Set Z flag for put.
004D97	CD D9 10	C+	22	CALL	U0DVR ; Out the UART we go.
004D9A	18 EE	C+	22	JR	__2710
004D9C	3E0A	C+	22	__2711:	ld a,LF.asc ; Get a LF.
004D9E	4F	C+	22	LD	C,A ; Output byte in C.
004D9F	AF	C+	22	XOR	A ; Set Z flag for put
004DA0	CD D9 10	C+	22	CALL	U0DVR ; Out the UART we go.
004DA3	3E0D	C+	22	ld	a,CR.asc ; Now put a CR back in stream.
004DA5	C9	C+	22	ret	; Return and print CR.
004DA6		C+	22	__2710a	endif
		C+	22	if	DEVICE.console=1
		C+	22	__2711:	LD A,(HL) ; Get byte to send.
		C+	22	INC	HL ; pointer=pointer+1
		C+	22	;	CP EOT.asc ; End Of Transmission (block)?
		C+	22	;	jr Z,\$1a ; Return if finished.
		C+	22	CP	ETX.asc ; End Of Text?
		C+	22	jr	Z,__2711a ; Return if finished.
		C+	22	LD	C,A ; Output byte to C.
		C+	22	XOR	A ; Set Z flag for put.
		C+	22	CALL	U1DVR ; Out the UART we go.
		C+	22	JR	__2711
		C+	22	__2711a	equ \$
		C+	22	endif	
		C+	22	ENDMAC	PUMP
004DA6		C	23	cngmon	PUMP DEVICE.console,gtmon
		C+	23	scope	; A tribute to Dave Shriner - Display Data Corp.
004DA6	21 25 56	C+	23	LD	HL,gtmon ; Pump from this starting point until ETX.asc is found.
		C+	23	if	DEVICE.console=0
004DA9	7E	C+	23	__2720:	LD A,(HL) ; Get byte to send.
004DAA	23	C+	23	INC	HL ; pointer=pointer+1
		C+	23	;	CP EOT.asc ; End Of Transmission (block)?
		C+	23	;	jr Z,\$0a ; Return if finished.
004DAB	FE03	C+	23	CP	ETX.asc ; End Of Text?
004DAD	28 16	C+	23	jr	Z,__2720a ; Return if finished.
004DAF	FE0D	C+	23	CP	CR.asc ; Was it a CR?
004DB1	CC BB 4D	C+	23	call	z,__2721 ; 1st add a LF to stream.
004DB4	4F	C+	23	LD	C,A ; Output byte in C.
004DB5	AF	C+	23	XOR	A ; Set Z flag for put.
004DB6	CD D9 10	C+	23	CALL	U0DVR ; Out the UART we go.
004DB9	18 EE	C+	23	JR	__2720
004DBB	3E0A	C+	23	__2721:	ld a,LF.asc ; Get a LF.
004DBD	4F	C+	23	LD	C,A ; Output byte in C.
004DBE	AF	C+	23	XOR	A ; Set Z flag for put
004DBF	CD D9 10	C+	23	CALL	U0DVR ; Out the UART we go.
004DC2	3E0D	C+	23	ld	a,CR.asc ; Now put a CR back in stream.
004DC4	C9	C+	23	ret	; Return and print CR.
004DC5		C+	23	__2720a	endif
		C+	23	if	DEVICE.console=1
		C+	23	__2721:	LD A,(HL) ; Get byte to send.
		C+	23	INC	HL ; pointer=pointer+1
		C+	23	;	CP EOT.asc ; End Of Transmission (block)?
		C+	23	;	jr Z,\$1a ; Return if finished.

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-RTC.s
		C+	23	CP	ETX.asc ; End Of Text?
		C+	23	jr	Z, __2721a ; Return if finished.
		C+	23	LD	C,A ; Output byte to C.
		C+	23	XOR	A ; Set Z flag for put.
		C+	23	CALL	U1DVR ; Out the UART we go.
		C+	23	JR	__2721
		C+	23	__2721a	equ \$
		C+	23		endif
		C+	23	ENDMAC	PUMP
		C	24	INPUT_U.str	editbuffer,2
		C+	24		scope
004DC5	21 D8 54	C+	24	ld	hl,editbuffer
004DC8	0602	C+	24	ld	b,2 ; Max number bytes to input.
004DCA	18 17	C+	24	jr	__2731 ; Branch to beginning.
004DCC	4F	C+	24	__2732:	ld c,a ; Output BS just received.
004DCD	CD EB 11	C+	24	CALL	TXUART0 ; Send it.
004DD0	0E20	C+	24	ld	c,SP.asc ; Now space out char we are deleting.
004DD2	CD EB 11	C+	24	CALL	TXUART0
004DD5	0E08	C+	24	ld	c,BS.asc ; Backspace again to new location.
004DD7	CD EB 11	C+	24	CALL	TXUART0
004DDA	2B	C+	24	dec	hl ; Back pointer in buffer up by -1.
004DDB	C1	C+	24	pop	bc ; Recover B counter.
004DDC	04	C+	24	INC	B
004DDD	FE02	C+	24	cp	2
004DDF	30 02	C+	24	jr	nc, __2731
004DE1	0602	C+	24	ld	b,2
004DE3	C5	C+	24	__2731:	push bc ; Save counter as other routines may clobber it.
004DE4	CD 7E 6F	C+	24	CALL	getchar ; Using UART NOT TRSDOS get one byte from UART0.
004DE7	FE08	C+	24	CP	BS.asc ; Is is backspace?
004DE9	28 E1	C+	24	jr	z, __2732 ; If yes gp to backspace routine above.
004DEB	77	C+	24	ld	(hl),a ; Put char in buff.
004DEC	23	C+	24	inc	hl ; Bump pointer to next place in buffer.
004DED	FE0D	C+	24	cp	CR.asc ; Was it a CR?
004DEF	28 0A	C+	24	jr	z, __2734 ; Exit input as terminalted by CR.
004DF1	4F	C+	24	ld	c,a
004DF2	CD EB 11	C+	24	CALL	TXUART0 ; Output character.
004DF5	C1	C+	24	__2733	pop bc ; Recover counter.
004DF6	10 EB	C+	24	djnz	__2731 ; Counter = counter - 1. If counter <> 0 branch \$1.
004DF8	AF	C+	24	xor	a ; All is well set Z flag.
004DF9	18 01	C+	24	jr	__2735 ; Finished, exit.
004DFB	B7	C+	24	__2734:	or a ; Make it non zero.
004DFC		C+	24	__2735:	endmac INPUT_U.str ; MACRO is done.
		C	25		PUMP DEVICE.console,CR\$
		C+	25	scope	; A tribute to Dave Shriner - Display Data Corp.
004DFC	21 24 5A	C+	25	LD	HL,CR\$ ; Pump from this starting point until ETX.asc is found.
		C+	25	if	DEVICE.console=0
004DFF	7E	C+	25	__2740:	LD A,(HL) ; Get byte to send.
004E00	23	C+	25	INC	HL ; pointer=pointer+1
		C+	25	;	CP EOT.asc ; End Of Transmission (block)?
		C+	25	;	jr Z,\$0a ; Return if finished.
004E01	FE03	C+	25	CP	ETX.asc ; End Of Text?
004E03	28 16	C+	25	jr	Z, __2740a ; Return if finished.
004E05	FE0D	C+	25	CP	CR.asc ; Was it a CR?
004E07	CC 11 4E	C+	25	call	z, __2741 ; 1st add a LF to stream.
004E0A	4F	C+	25	LD	C,A ; Output byte in C.
004E0B	AF	C+	25	XOR	A ; Set Z flag for put.
004E0C	CD D9 10	C+	25	CALL	U0DVR ; Out the UART we go.
004E0F	18 EE	C+	25	JR	__2740
004E11	3E0A	C+	25	__2741:	ld a,LF.asc ; Get a LF.
004E13	4F	C+	25	LD	C,A ; Output byte in C.

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-RTC.s
004E14	AF	C+	25	XOR	A ; Set Z flag for put
004E15	CD D9 10	C+	25	CALL	U0DVR ; Out the UART we go.
004E18	3E0D	C+	25	ld	a,CR.asc ; Now put a CR back in stream.
004E1A	C9	C+	25	ret	; Return and print CR.
004E1B		C+	25	__2740a	endif
		C+	25	if	DEVICE.console=1
		C+	25	__2741:	LD A,(HL) ; Get byte to send.
		C+	25	INC	HL ; pointer=pointer+1
		C+	25	CP	EOT.asc ; End Of Transmission (block)?
		C+	25	;	
		C+	25	jr	Z,\$1a ; Return if finished.
		C+	25	CP	ETX.asc ; End Of Text?
		C+	25	jr	Z,__2741a ; Return if finished.
		C+	25	LD	C,A ; Output byte to C.
		C+	25	XOR	A ; Set Z flag for put.
		C+	25	CALL	U1DVR ; Out the UART we go.
		C+	25	JR	__2741
		C+	25	__2741a	equ \$
		C+	25	endif	
		C+	25	ENDMAC	PUMP
		C	26	DECHEX	editbuffer
004E1B	21 D8 54	C+	26	ld	hl,editbuffer
		C+	26	SVC	svc_dechex
004E1E	3E60	C+	26	ld	a,svc_dechex
004E20	EF	C+	26	restart28	
		C+	26	ENDMAC	SVC
		C+	26	ENDMAC	DECHEX
004E21	79	C	27	ld	a,c
004E22	B7	C	28	or	a
004E23	CA A6 4D	C	29	jp	z,cngmon
004E26	FE0D	C	30	cp	13
004E28	D2 A6 4D	C	31	jp	nc,cngmon
004E2B	32 88 56	C	32	ld	(savmon),a
		C	33	HEXDEC	BC,cngmo
004E2E	C5E1	C+	33	ld	hl,BC
004E30	11 E5 56	C+	33	ld	de,cngmo
		C+	33	SVC	svc_hexdec
004E33	3E61	C+	33	ld	a,svc_hexdec
004E35	EF	C+	33	restart28	
		C+	33	ENDMAC	SVC
		C+	33	ENDMAC	HEXDEC
004E36		C	34	cngday	PUMP DEVICE.console,gtday
		C+	34	scope	; A tribute to Dave Shriner - Display Data Corp.
004E36	21 3A 56	C+	34	LD	HL,gtday ; Pump from this starting point until ETX.asc is found.
		C+	34	if	DEVICE.console=0
004E39	7E	C+	34	__2790:	LD A,(HL) ; Get byte to send.
004E3A	23	C+	34	INC	HL ; pointer=pointer+1
		C+	34	CP	EOT.asc ; End Of Transmission (block)?
		C+	34	;	
		C+	34	jr	Z,\$0a ; Return if finished.
004E3B	FE03	C+	34	CP	ETX.asc ; End Of Text?
004E3D	28 16	C+	34	jr	Z,__2790a ; Return if finished.
004E3F	FE0D	C+	34	CP	CR.asc ; Was it a CR?
004E41	CC 4B 4E	C+	34	call	z,__2791 ; 1st add a LF to stream.
004E44	4F	C+	34	LD	C,A ; Output byte in C.
004E45	AF	C+	34	XOR	A ; Set Z flag for put.
004E46	CD D9 10	C+	34	CALL	U0DVR ; Out the UART we go.
004E49	18 EE	C+	34	JR	__2790
004E4B	3E0A	C+	34	__2791:	ld a,LF.asc ; Get a LF.
004E4D	4F	C+	34	LD	C,A ; Output byte in C.
004E4E	AF	C+	34	XOR	A ; Set Z flag for put
004E4F	CD D9 10	C+	34	CALL	U0DVR ; Out the UART we go.

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-RTC.s
004E52	3E0D	C+	34	ld	a,CR.asc ; Now put a CR back in stream.
004E54	C9	C+	34	ret	; Return and print CR.
004E55		C+	34	__2790a	endif
		C+	34		if DEVICE.console=1
		C+	34	__2791:	LD A,(HL) ; Get byte to send.
		C+	34		INC HL ; pointer=pointer+1
		C+	34	;	CP EOT.asc ; End Of Transmission (block)?
		C+	34	;	jr Z,\$1a ; Return if finished.
		C+	34		CP ETX.asc ; End Of Text?
		C+	34		jr Z,__2791a ; Return if finished.
		C+	34		LD C,A ; Output byte to C.
		C+	34		XOR A ; Set Z flag for put.
		C+	34		CALL U1DVR ; Out the UART we go.
		C+	34		JR __2791
		C+	34	__2791a	equ \$
		C+	34		endif
		C+	34		ENDMAC PUMP
		C	35		INPUT_U.str editbuffer,2
		C+	35		scope
004E55	21 D8 54	C+	35		ld hl,editbuffer
004E58	0602	C+	35		ld b,2 ; Max number bytes to input.
004E5A	18 17	C+	35		jr __2801 ; Branch to beginning.
004E5C	4F	C+	35	__2802:	ld c,a ; Output BS just received.
004E5D	CD EB 11	C+	35		CALL TXUART0 ; Send it.
004E60	0E20	C+	35		ld c,SP.asc ; Now space out char we are deleting.
004E62	CD EB 11	C+	35		CALL TXUART0
004E65	0E08	C+	35		ld c,BS.asc ; Backspace again to new location.
004E67	CD EB 11	C+	35		CALL TXUART0
004E6A	2B	C+	35		dec hl ; Back pointer in buffer up by -1.
004E6B	C1	C+	35		pop bc ; Recover B counter.
004E6C	04	C+	35		INC B
004E6D	FE02	C+	35		cp 2
004E6F	30 02	C+	35		jr nc,__2801
004E71	0602	C+	35		ld b,2
004E73	C5	C+	35	__2801:	push bc ; Save counter as other routines may clobber it.
004E74	CD 7E 6F	C+	35		CALL getchar ; Using UART NOT TRSDOS get one byte from UART0.
004E77	FE08	C+	35		CP BS.asc ; Is is backspace?
004E79	28 E1	C+	35		jr z,__2802 ; If yes gp to backspace routine above.
004E7B	77	C+	35		ld (hl),a ; Put char in buff.
004E7C	23	C+	35		inc hl ; Bump pointer to next place in buffer.
004E7D	FE0D	C+	35		cp CR.asc ; Was it a CR?
004E7F	28 0A	C+	35		jr z,__2804 ; Exit input as terminalted by CR.
004E81	4F	C+	35		ld c,a
004E82	CD EB 11	C+	35		CALL TXUART0 ; Output character.
004E85	C1	C+	35	__2803	pop bc ; Recover counter.
004E86	10 EB	C+	35		djnz __2801 ; Counter = counter - 1. If counter <> 0 branch \$1.
004E88	AF	C+	35		xor a ; All is well set Z flag.
004E89	18 01	C+	35		jr __2805 ; Finished, exit.
004E8B	B7	C+	35	__2804:	or a ; Make it non zero.
004E8C		C+	35	__2805:	endmac INPUT_U.str ; MACRO is done.
		C	36		PUMP DEVICE.console,CR\$
		C+	36		scope ; A tribute to Dave Shriner - Display Data Corp.
004E8C	21 24 5A	C+	36		LD HL,CR\$ ; Pump from this starting point until ETX.asc is found.
		C+	36		if DEVICE.console=0
004E8F	7E	C+	36	__2810:	LD A,(HL) ; Get byte to send.
004E90	23	C+	36		INC HL ; pointer=pointer+1
		C+	36	;	CP EOT.asc ; End Of Transmission (block)?
		C+	36	;	jr Z,\$0a ; Return if finished.
004E91	FE03	C+	36		CP ETX.asc ; End Of Text?
004E93	28 16	C+	36		jr Z,__2810a ; Return if finished.

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-RTC.s
004E95	FE0D	C+	36	CP	CR.asc ; Was it a CR?
004E97	CC A1 4E	C+	36	call	z,__2811 ; 1st add a LF to stream.
004E9A	4F	C+	36	LD	C,A ; Output byte in C.
004E9B	AF	C+	36	XOR	A ; Set Z flag for put.
004E9C	CD D9 10	C+	36	CALL	U0DVR ; Out the UART we go.
004E9F	18 EE	C+	36	JR	__2810
004EA1	3E0A	C+	36	__2811:	ld a,LF.asc ; Get a LF.
004EA3	4F	C+	36	LD	C,A ; Output byte in C.
004EA4	AF	C+	36	XOR	A ; Set Z flag for put
004EA5	CD D9 10	C+	36	CALL	U0DVR ; Out the UART we go.
004EA8	3E0D	C+	36	ld	a,CR.asc ; Now put a CR back in stream.
004EAA	C9	C+	36	ret	; Return and print CR.
004EAB		C+	36	__2810a	endif
		C+	36		if DEVICE.console=1
		C+	36	__2811:	LD A,(HL) ; Get byte to send.
		C+	36	INC	HL ; pointer=pointer+1
		C+	36	;	CP EOT.asc ; End Of Transmission (block)?
		C+	36	;	jr Z,\$1a ; Return if finished.
		C+	36		CP ETX.asc ; End Of Text?
		C+	36		jr Z,__2811a ; Return if finished.
		C+	36	LD	C,A ; Output byte to C.
		C+	36	XOR	A ; Set Z flag for put.
		C+	36	CALL	U1DVR ; Out the UART we go.
		C+	36	JR	__2811
		C+	36	__2811a	equ \$
		C+	36		endif
		C+	36	ENDMAC	PUMP
		C	37	DECHEX	editbuffer
004EAB	21 D8 54	C+	37	ld	hl,editbuffer
		C+	37	SVC	svc_dechex
004EAE	3E60	C+	37	ld	a,svc_dechex
004EB0	EF	C+	37		restart28
		C+	37	ENDMAC	SVC
		C+	37	ENDMAC	DECHEX
004EB1	79	C	38	ld	a,c
004EB2	B7	C	39	or	a
004EB3	CA 36 4E	C	40	jp	z,cngday
004EB6	FE20	C	41	cp	32
004EB8	D2 36 4E	C	42	jp	nc,cngday
004EBB	32 89 56	C	43	ld	(savday),a
		C	44	HEXDEC	BC,cngdy
004EBE	C5E1	C+	44	ld	hl,BC
004EC0	11 EF 56	C+	44	ld	de,cngdy
		C+	44	SVC	svc_hexdec
004EC3	3E61	C+	44	ld	a,svc_hexdec
004EC5	EF	C+	44		restart28
		C+	44	ENDMAC	SVC
		C+	44	ENDMAC	HEXDEC
004EC6		C	45	cngcen	PUMP DEVICE.console,gtcen
		C+	45		scope ; A tribute to Dave Shriner - Display Data Corp.
004EC6	21 56 56	C+	45	LD	HL,gtcen ; Pump from this starting point until ETX.asc is found.
		C+	45		if DEVICE.console=0
004EC9	7E	C+	45	__2860:	LD A,(HL) ; Get byte to send.
004ECA	23	C+	45	INC	HL ; pointer=pointer+1
		C+	45	;	CP EOT.asc ; End Of Transmission (block)?
		C+	45	;	jr Z,\$0a ; Return if finished.
004ECB	FE03	C+	45	CP	ETX.asc ; End Of Text?
004ECD	28 16	C+	45	jr	Z,__2860a ; Return if finished.
004ECF	FE0D	C+	45	CP	CR.asc ; Was it a CR?
004ED1	CC DB 4E	C+	45	call	z,__2861 ; 1st add a LF to stream.

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-RTC.s
004ED4	4F	C+	45	LD	C,A ; Output byte in C.
004ED5	AF	C+	45	XOR	A ; Set Z flag for put.
004ED6	CD D9 10	C+	45	CALL	U0DVR ; Out the UART we go.
004ED9	18 EE	C+	45	JR	__2860
004EDB	3E0A	C+	45	__2861:	ld a,LF.asc ; Get a LF.
004EDD	4F	C+	45	LD	C,A ; Output byte in C.
004EDE	AF	C+	45	XOR	A ; Set Z flag for put
004EDF	CD D9 10	C+	45	CALL	U0DVR ; Out the UART we go.
004EE2	3E0D	C+	45	ld	a,CR.asc ; Now put a CR back in stream.
004EE4	C9	C+	45	ret	; Return and print CR.
004EE5		C+	45	__2860a	endif
		C+	45		if DEVICE.console=1
		C+	45	__2861:	LD A,(HL) ; Get byte to send.
		C+	45	INC	HL ; pointer=pointer+1
		C+	45	; CP	EOT.asc ; End Of Transmission (block)?
		C+	45	; jr	Z,\$1a ; Return if finished.
		C+	45	CP	ETX.asc ; End Of Text?
		C+	45	jr	Z,__2861a ; Return if finished.
		C+	45	LD	C,A ; Output byte to C.
		C+	45	XOR	A ; Set Z flag for put.
		C+	45	CALL	U1DVR ; Out the UART we go.
		C+	45	JR	__2861
		C+	45	__2861a	equ \$
		C+	45		endif
		C+	45	ENDMAC	PUMP
		C	46	INPUT_U.str	editbuffer,2
		C+	46		scope
004EE5	21 D8 54	C+	46	ld	hl,editbuffer
004EE8	0602	C+	46	ld	b,2 ; Max number bytes to input.
004EEA	18 17	C+	46	jr	__2871 ; Branch to beginning.
004EEC	4F	C+	46	__2872:	ld c,a ; Output BS just received.
004EED	CD EB 11	C+	46	CALL	TXUART0 ; Send it.
004EF0	0E20	C+	46	ld	c,SP.asc ; Now space out char we are deleting.
004EF2	CD EB 11	C+	46	CALL	TXUART0
004EF5	0E08	C+	46	ld	c,BS.asc ; Backspace again to new location.
004EF7	CD EB 11	C+	46	CALL	TXUART0
004EFA	2B	C+	46	dec	hl ; Back pointer in buffer up by -1.
004EFB	C1	C+	46	pop	bc ; Recover B counter.
004EFC	04	C+	46	INC	B
004EFD	FE02	C+	46	cp	2
004EFF	30 02	C+	46	jr	nc,__2871
004F01	0602	C+	46	ld	b,2
004F03	C5	C+	46	__2871:	push bc ; Save counter as other routines may clobber it.
004F04	CD 7E 6F	C+	46	CALL	getchar ; Using UART NOT TRSDOS get one byte from UART0.
004F07	FE08	C+	46	CP	BS.asc ; Is is backspace?
004F09	28 E1	C+	46	jr	z,__2872 ; If yes gp to backspace routine above.
004F0B	77	C+	46	ld	(hl),a ; Put char in buff.
004F0C	23	C+	46	inc	hl ; Bump pointer to next place in buffer.
004F0D	FE0D	C+	46	cp	CR.asc ; Was it a CR?
004F0F	28 0A	C+	46	jr	z,__2874 ; Exit input as terminalted by CR.
004F11	4F	C+	46	ld	c,a
004F12	CD EB 11	C+	46	CALL	TXUART0 ; Output character.
004F15	C1	C+	46	__2873	pop bc ; Recover counter.
004F16	10 EB	C+	46	djnz	__2871 ; Counter = counter - 1. If counter <> 0 branch \$1.
004F18	AF	C+	46	xor	a ; All is well set Z flag.
004F19	18 01	C+	46	jr	__2875 ; Finished, exit.
004F1B	B7	C+	46	__2874:	or a ; Make it non zero.
004F1C		C+	46	__2875:	endmac INPUT_U.str ; MACRO is done.
		C	47	PUMP	DEVICE.console,CR\$
		C+	47	scope	; A tribute to Dave Shriner - Display Data Corp.



\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source
				..\SYS0_IPL\ipl-RTC.s
004F1C	21 24 5A	C+	47	LD HL,CR\$ ; Pump from this starting point until ETX.asc is found.
		C+	47	if DEVICE.console=0
004F1F	7E	C+	47	__2880: LD A,(HL) ; Get byte to send.
004F20	23	C+	47	INC HL ; pointer=pointer+1
		C+	47	; CP EOT.asc ; End Of Transmission (block)?
		C+	47	; jr Z,\$0a ; Return if finished.
004F21	FE03	C+	47	CP ETX.asc ; End Of Text?
004F23	28 16	C+	47	jr Z,__2880a ; Return if finished.
004F25	FE0D	C+	47	CP CR.asc ; Was it a CR?
004F27	CC 31 4F	C+	47	call z,__2881 ; 1st add a LF to stream.
004F2A	4F	C+	47	LD C,A ; Output byte in C.
004F2B	AF	C+	47	XOR A ; Set Z flag for put.
004F2C	CD D9 10	C+	47	CALL U0DVR ; Out the UART we go.
004F2F	18 EE	C+	47	JR __2880
004F31	3E0A	C+	47	__2881: ld a,LF.asc ; Get a LF.
004F33	4F	C+	47	LD C,A ; Output byte in C.
004F34	AF	C+	47	XOR A ; Set Z flag for put
004F35	CD D9 10	C+	47	CALL U0DVR ; Out the UART we go.
004F38	3E0D	C+	47	ld a,CR.asc ; Now put a CR back in stream.
004F3A	C9	C+	47	ret ; Return and print CR.
004F3B		C+	47	__2880a endif
		C+	47	if DEVICE.console=1
		C+	47	__2881: LD A,(HL) ; Get byte to send.
		C+	47	INC HL ; pointer=pointer+1
		C+	47	; CP EOT.asc ; End Of Transmission (block)?
		C+	47	; jr Z,\$1a ; Return if finished.
		C+	47	CP ETX.asc ; End Of Text?
		C+	47	jr Z,__2881a ; Return if finished.
		C+	47	LD C,A ; Output byte to C.
		C+	47	XOR A ; Set Z flag for put.
		C+	47	CALL U1DVR ; Out the UART we go.
		C+	47	JR __2881
		C+	47	__2881a equ \$
		C+	47	endif
		C+	47	ENDMAC PUMP
		C	48	DECHEX editbuffer
004F3B	21 D8 54	C+	48	ld hl,editbuffer
		C+	48	SVC svc_dechex
004F3E	3E60	C+	48	ld a,svc_dechex
004F40	EF	C+	48	restart28
		C+	48	ENDMAC SVC
		C+	48	ENDMAC DECHEX
004F41	79	C	49	ld a,c
004F42	B7	C	50	or a
004F43	CA C6 4E	C	51	jp z,cngcn
004F46	FE64	C	52	cp 100
004F48	D2 C6 4E	C	53	jp nc,cngcn
004F4B	32 8A 56	C	54	ld (savcn),a
		C	55	HEXDEC BC,cngcn
004F4E	C5E1	C+	55	ld hl,BC
004F50	11 FD 56	C+	55	ld de,cngcn
		C+	55	SVC svc_hexdec
004F53	3E61	C+	55	ld a,svc_hexdec
004F55	EF	C+	55	restart28
		C+	55	ENDMAC SVC
		C+	55	ENDMAC HEXDEC
004F56		C	56	cngyear PUMP DEVICE.console,gtyear
		C+	56	scope ; A tribute to Dave Shriner - Display Data Corp.
004F56	21 6E 56	C+	56	LD HL,gtyear ; Pump from this starting point until ETX.asc is found.
		C+	56	if DEVICE.console=0

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-RTC.s
004F59	7E	C+	56	__2930:	LD A,(HL) ; Get byte to send.
004F5A	23	C+	56		INC HL ; pointer=pointer+1
		C+	56		; CP EOT.asc ; End Of Transmission (block)?
		C+	56		; jr Z,\$0a ; Return if finished.
004F5B	FE03	C+	56		CP ETX.asc ; End Of Text?
004F5D	28 16	C+	56		jr Z,__2930a ; Return if finished.
004F5F	FE0D	C+	56		CP CR.asc ; Was it a CR?
004F61	CC 6B 4F	C+	56		call z,__2931 ; 1st add a LF to stream.
004F64	4F	C+	56		LD C,A ; Output byte in C.
004F65	AF	C+	56		XOR A ; Set Z flag for put.
004F66	CD D9 10	C+	56		CALL U0DVR ; Out the UART we go.
004F69	18 EE	C+	56		JR __2930
004F6B	3E0A	C+	56	__2931:	ld a,LF.asc ; Get a LF.
004F6D	4F	C+	56		LD C,A ; Output byte in C.
004F6E	AF	C+	56		XOR A ; Set Z flag for put
004F6F	CD D9 10	C+	56		CALL U0DVR ; Out the UART we go.
004F72	3E0D	C+	56		ld a,CR.asc ; Now put a CR back in stream.
004F74	C9	C+	56		ret ; Return and print CR.
004F75		C+	56	__2930a	endif
		C+	56		if DEVICE.console=1
		C+	56	__2931:	LD A,(HL) ; Get byte to send.
		C+	56		INC HL ; pointer=pointer+1
		C+	56		; CP EOT.asc ; End Of Transmission (block)?
		C+	56		; jr Z,\$1a ; Return if finished.
		C+	56		CP ETX.asc ; End Of Text?
		C+	56		jr Z,__2931a ; Return if finished.
		C+	56		LD C,A ; Output byte to C.
		C+	56		XOR A ; Set Z flag for put.
		C+	56		CALL U1DVR ; Out the UART we go.
		C+	56		JR __2931
		C+	56	__2931a	equ \$
		C+	56		endif
		C+	56		ENDMAC PUMP
		C	57		INPUT_U.str editbuffer,2
		C+	57		scope
004F75	21 D8 54	C+	57		ld hl,editbuffer
004F78	0602	C+	57		ld b,2 ; Max number bytes to input.
004F7A	18 17	C+	57		jr __2941 ; Branch to beginning.
004F7C	4F	C+	57	__2942:	ld c,a ; Output BS just received.
004F7D	CD EB 11	C+	57		CALL TXUART0 ; Send it.
004F80	0E20	C+	57		ld c,SP.asc ; Now space out char we are deleting.
004F82	CD EB 11	C+	57		CALL TXUART0
004F85	0E08	C+	57		ld c,BS.asc ; Backspace again to new location.
004F87	CD EB 11	C+	57		CALL TXUART0
004F8A	2B	C+	57		dec hl ; Back pointer in buffer up by -1.
004F8B	C1	C+	57		pop bc ; Recover B counter.
004F8C	04	C+	57		INC B
004F8D	FE02	C+	57		cp 2
004F8F	30 02	C+	57		jr nc,__2941
004F91	0602	C+	57		ld b,2
004F93	C5	C+	57	__2941:	push bc ; Save counter as other routines may clobber it.
004F94	CD 7E 6F	C+	57		CALL getchar ; Using UART NOT TRSDOS get one byte from UART0.
004F97	FE08	C+	57		CP BS.asc ; Is is backspace?
004F99	28 E1	C+	57		jr z,__2942 ; If yes gp to backspace routine above.
004F9B	77	C+	57		ld (hl),a ; Put char in buff.
004F9C	23	C+	57		inc hl ; Bump pointer to next place in buffer.
004F9D	FE0D	C+	57		cp CR.asc ; Was it a CR?
004F9F	28 0A	C+	57		jr z,__2944 ; Exit input as terminalted by CR.
004FA1	4F	C+	57		ld c,a
004FA2	CD EB 11	C+	57		CALL TXUART0 ; Output character.

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-RTC.s
004FA5	C1	C+	57	__2943	pop bc ; Recover counter.
004FA6	10 EB	C+	57		djnz __2941 ; Counter = counter - 1. If counter <> 0 branch \$1.
004FA8	AF	C+	57		xor a ; All is well set Z flag.
004FA9	18 01	C+	57		jr __2945 ; Finished, exit.
004FAB	B7	C+	57	__2944:	or a ; Make it non zero.
004FAC		C+	57	__2945:	endmac INPUT_U.str ; MACRO is done.
		C	58		PUMP DEVICE.console,CR\$
		C+	58		scope ; A tribute to Dave Shriner - Display Data Corp.
004FAC	21 24 5A	C+	58		LD HL,CR\$ ; Pump from this starting point until ETX.asc is found.
		C+	58		if DEVICE.console=0
004FAF	7E	C+	58	__2950:	LD A,(HL) ; Get byte to send.
004FB0	23	C+	58		INC HL ; pointer=pointer+1
		C+	58		; CP EOT.asc ; End Of Transmission (block)?
		C+	58		; jr Z,\$0a ; Return if finished.
004FB1	FE03	C+	58		CP ETX.asc ; End Of Text?
004FB3	28 16	C+	58		jr Z,__2950a ; Return if finished.
004FB5	FE0D	C+	58		CP CR.asc ; Was it a CR?
004FB7	CC C1 4F	C+	58		call z,__2951 ; 1st add a LF to stream.
004FBA	4F	C+	58		LD C,A ; Output byte in C.
004FBB	AF	C+	58		XOR A ; Set Z flag for put.
004FBC	CD D9 10	C+	58		CALL U0DVR ; Out the UART we go.
004FBF	18 EE	C+	58		JR __2950
004FC1	3E0A	C+	58	__2951:	ld a,LF.asc ; Get a LF.
004FC3	4F	C+	58		LD C,A ; Output byte in C.
004FC4	AF	C+	58		XOR A ; Set Z flag for put
004FC5	CD D9 10	C+	58		CALL U0DVR ; Out the UART we go.
004FC8	3E0D	C+	58		ld a,CR.asc ; Now put a CR back in stream.
004FCA	C9	C+	58		ret ; Return and print CR.
004FCB		C+	58	__2950a	endif
		C+	58		if DEVICE.console=1
		C+	58	__2951:	LD A,(HL) ; Get byte to send.
		C+	58		INC HL ; pointer=pointer+1
		C+	58		; CP EOT.asc ; End Of Transmission (block)?
		C+	58		; jr Z,\$1a ; Return if finished.
		C+	58		CP ETX.asc ; End Of Text?
		C+	58		jr Z,__2951a ; Return if finished.
		C+	58		LD C,A ; Output byte to C.
		C+	58		XOR A ; Set Z flag for put.
		C+	58		CALL U1DVR ; Out the UART we go.
		C+	58		JR __2951
		C+	58	__2951a	equ \$
		C+	58		endif
		C+	58		ENDMAC PUMP
		C	59		DECHEX editbuffer
004FCB	21 D8 54	C+	59		ld hl,editbuffer
		C+	59		SVC svc_dechex
004FCE	3E60	C+	59		ld a,svc_dechex
004FD0	EF	C+	59		restart28
		C+	59		ENDMAC SVC
		C+	59		ENDMAC DECHEX
004FD1	79	C	60		ld a,c
004FD2	B7	C	61		or a
004FD3	CA 56 4F	C	62		jp z,cngyear
004FD6	FE64	C	63		cp 100
004FD8	D2 56 4F	C	64		jp nc,cngyear
004FDB	32 87 56	C	65		ld (savyr),a
		C	66		HEXDEC BC,cngyr
004FDE	C5E1	C+	66		ld hl,BC
004FE0	11 08 57	C+	66		ld de,cngyr
		C+	66		SVC svc_hexdec

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-RTC.s
004FE3	3E61	C+	66	ld	a,svc_hexdec
004FE5	EF	C+	66	restart28	
		C+	66	ENDMAC	SVC
		C+	66	ENDMAC	HEXDEC
		C	67		
004FE6		C	68	cngpromptPUMP	DEVICE.console,cngmsg
		C+	68	scope	; A tribute to Dave Shriner - Display Data Corp.
004FE6	21 DD 56	C+	68	LD	HL,cngmsg ; Pump from this starting point until ETX.asc is found.
		C+	68	if	DEVICE.console=0
004FE9	7E	C+	68	__3000:	LD A,(HL) ; Get byte to send.
004FEA	23	C+	68	INC	HL ; pointer=pointer+1
		C+	68	; CP	EOT.asc ; End Of Transmission (block)?
		C+	68	; jr	Z,\$0a ; Return if finished.
004FEB	FE03	C+	68	CP	ETX.asc ; End Of Text?
004FED	28 16	C+	68	jr	Z,__3000a ; Return if finished.
004FEF	FE0D	C+	68	CP	CR.asc ; Was it a CR?
004FF1	CC FB 4F	C+	68	call	z,__3001 ; 1st add a LF to stream.
004FF4	4F	C+	68	LD	C,A ; Output byte in C.
004FF5	AF	C+	68	XOR	A ; Set Z flag for put.
004FF6	CD D9 10	C+	68	CALL	U0DVR ; Out the UART we go.
004FF9	18 EE	C+	68	JR	__3000
004FFB	3E0A	C+	68	__3001:	ld a,LF.asc ; Get a LF.
004FFD	4F	C+	68	LD	C,A ; Output byte in C.
004FFE	AF	C+	68	XOR	A ; Set Z flag for put
004FFF	CD D9 10	C+	68	CALL	U0DVR ; Out the UART we go.
005002	3E0D	C+	68	ld	a,CR.asc ; Now put a CR back in stream.
005004	C9	C+	68	ret	; Return and print CR.
005005		C+	68	__3000a	endif
		C+	68	if	DEVICE.console=1
		C+	68	__3001:	LD A,(HL) ; Get byte to send.
		C+	68	INC	HL ; pointer=pointer+1
		C+	68	; CP	EOT.asc ; End Of Transmission (block)?
		C+	68	; jr	Z,\$1a ; Return if finished.
		C+	68	CP	ETX.asc ; End Of Text?
		C+	68	jr	Z,__3001a ; Return if finished.
		C+	68	LD	C,A ; Output byte to C.
		C+	68	XOR	A ; Set Z flag for put.
		C+	68	CALL	U1DVR ; Out the UART we go.
		C+	68	JR	__3001
		C+	68	__3001a	equ \$
		C+	68	endif	
		C+	68	ENDMAC	PUMP
005005		C	69	cngget	KEY_U.get
005005	CD 7E 6F	C+	69	call	getchar
		C+	69	IFMA	1
		C+	69	ld	,a
		C+	69	else	
		C+	69	ENDIF	
		C+	69	ENDMAC	KEY_U.get
005008	FE1B	C	70	cp	ESC.asc
00500A	CA 11 4D	C	71	jp	z,RTC_IPL
00500D	FE04	C	72	cp	EOT.asc
00500F	20 F4	C	73	jr	nz,cngget
		C	74	RTC.unlock	
005011	3E21	C+	74	ld	a,00100001b ; Unlock & disable interrupts.
005013	ED39ED	C+	74	OUT0	(RTC_CTRL),a
		C+	74	endmac	RTC.unlock
005016	3A 89 56	C	75	ld	a,(savday)
005019	CD DA 12	C	76	call	bin2bcd
		C	77	RTC.DOM.put	a

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-RTC.s
00501C	7F	C+	77	ld	a,a
00501D	ED39E4	C+	77	out0	(RTC_DOM),A
		C+	77	endmac	
005020	3A 88 56	C	78	ld	a,(savmon)
005023	CD DA 12	C	79	call	bin2bcd
		C	80	RTC.MON.put	a
005026	7F	C+	80	ld	a,a
005027	ED39E5	C+	80	out0	(RTC_MON),A
		C+	80	ENDMAC	
00502A	3A 87 56	C	81	ld	a,(savyr)
00502D	CD DA 12	C	82	call	bin2bcd
		C	83	RTC.YEAR.put	a
005030	7F	C+	83	ld	a,a
005031	ED39E6	C+	83	out0	(RTC_YR),A ;
		C+	83	ENDMAC	
005034	3A 8A 56	C	84	ld	a,(savcen)
005037	CD DA 12	C	85	call	bin2bcd
		C	86	RTC.CEN.put	a
00503A	7F	C+	86	ld	a,a
00503B	ED39E7	C+	86	out0	(RTC_CEN),A
		C+	86	ENDMAC	
		C	87	RTC.lock	
00503E	3E20	C+	87	ld	a,00100000b ; Lock clock & disable interrupts.
005040	ED39ED	C+	87	OUT0	(RTC_CTRL),a ; Execute.
		C+	87	ENDMAC	RTC.lock
		C	88		
		C	89	PUMP	DEVICE.console,cngcomplete
		C+	89	scope	; A tribute to Dave Shriner - Display Data Corp.
005043	21 F6 55	C+	89	LD	HL,cngcomplete ; Pump from this starting point until ETX.asc is found.
		C+	89	if	DEVICE.console=0
005046	7E	C+	89	LD	A,(HL) ; Get byte to send.
005047	23	C+	89	INC	HL ; pointer=pointer+1
		C+	89	CP	EOT.asc ; End Of Transmission (block)?
		C+	89	jr	Z,\$0a ; Return if finished.
005048	FE03	C+	89	CP	ETX.asc ; End Of Text?
00504A	28 16	C+	89	jr	Z,__3080a ; Return if finished.
00504C	FE0D	C+	89	CP	CR.asc ; Was it a CR?
00504E	CC 58 50	C+	89	call	z,__3081 ; 1st add a LF to stream.
005051	4F	C+	89	LD	C,A ; Output byte in C.
005052	AF	C+	89	XOR	A ; Set Z flag for put.
005053	CD D9 10	C+	89	CALL	U0DVR ; Out the UART we go.
005056	18 EE	C+	89	JR	__3080
005058	3E0A	C+	89	LD	a,LF.asc ; Get a LF.
00505A	4F	C+	89	LD	C,A ; Output byte in C.
00505B	AF	C+	89	XOR	A ; Set Z flag for put
00505C	CD D9 10	C+	89	CALL	U0DVR ; Out the UART we go.
00505F	3E0D	C+	89	ld	a,CR.asc ; Now put a CR back in stream.
005061	C9	C+	89	ret	; Return and print CR.
005062		C+	89	endif	
		C+	89	if	DEVICE.console=1
		C+	89	LD	A,(HL) ; Get byte to send.
		C+	89	INC	HL ; pointer=pointer+1
		C+	89	CP	EOT.asc ; End Of Transmission (block)?
		C+	89	jr	Z,\$1a ; Return if finished.
		C+	89	CP	ETX.asc ; End Of Text?
		C+	89	jr	Z,__3081a ; Return if finished.
		C+	89	LD	C,A ; Output byte to C.
		C+	89	XOR	A ; Set Z flag for put.
		C+	89	CALL	U1DVR ; Out the UART we go.
		C+	89	JR	__3081

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-RTC.s
		C+	89	__3081a	equ \$
		C+	89		endif
		C+	89		ENDMAC PUMP
005062		C	90	regetrsp1	KEY_U.get
005062	CD 7E 6F	C+	90		call getchar
		C+	90		IFMA 1
		C+	90		ld ,a
		C+	90		else
		C+	90		ENDIF
		C+	90		ENDMAC KEY_U.get
005065	FE0D	C	91	cp	CR.asc
005067	CA 11 4D	C	92	JP	z,RTC_IPL
00506A	C3 62 50	C	93	jp	regetrsp1
		C	94		
00506D	CD ED 6E	C	95	cngdow	call scrolloff
		C	96	MVC	screenbuf,dowbuff+3,9 ; Always start with a fresh buffer by stuffing with spa
		C+	96		scope
005070	21 DE 54	C+	96	ld	hl,screenbuf
005073	11 B2 54	C+	96	ld	de,dowbuff+3
005076	010900	C+	96	ld	bc,9
005079	EDB0	C+	96	LDIR	
		C+	96	ENDMAC	MVC
00507B	CD DA 53	C	97	call	getdow
		C	98	PUMP	DEVICE.console,dowprompt
		C+	98	scope	; A tribute to Dave Shriner - Display Data Corp.
00507E	21 8E 54	C+	98	LD	HL,dowprompt ; Pump from this starting point until ETX.asc is found.
		C+	98		
		C+	98	if	DEVICE.console=0
005081	7E	C+	98	__3110:	LD A,(HL) ; Get byte to send.
005082	23	C+	98		
		C+	98	INC	HL ; pointer=pointer+1
		C+	98	; CP	EOT.asc ; End Of Transmission (block)?
		C+	98	; jr	Z,\$0a ; Return if finished.
005083	FE03	C+	98	CP	ETX.asc ; End Of Text?
005085	28 16	C+	98	jr	Z,__3110a ; Return if finished.
005087	FE0D	C+	98	CP	CR.asc ; Was it a CR?
005089	CC 93 50	C+	98	call	z,__3111 ; 1st add a LF to stream.
00508C	4F	C+	98	LD	C,A ; Output byte in C.
00508D	AF	C+	98	XOR	A ; Set Z flag for put.
00508E	CD D9 10	C+	98	CALL	U0DVR ; Out the UART we go.
005091	18 EE	C+	98	JR	__3110
005093	3E0A	C+	98	__3111:	ld a,LF.asc ; Get a LF.
005095	4F	C+	98	LD	C,A ; Output byte in C.
005096	AF	C+	98	XOR	A ; Set Z flag for put
005097	CD D9 10	C+	98	CALL	U0DVR ; Out the UART we go.
00509A	3E0D	C+	98	ld	a,CR.asc ; Now put a CR back in stream.
00509C	C9	C+	98	ret	; Return and print CR.
00509D		C+	98	__3110a	endif
		C+	98		if DEVICE.console=1
		C+	98	__3111:	LD A,(HL) ; Get byte to send.
		C+	98		
		C+	98	INC	HL ; pointer=pointer+1
		C+	98	; CP	EOT.asc ; End Of Transmission (block)?
		C+	98	; jr	Z,\$1a ; Return if finished.
		C+	98	CP	ETX.asc ; End Of Text?
		C+	98	jr	Z,__3111a ; Return if finished.
		C+	98	LD	C,A ; Output byte to C.
		C+	98	XOR	A ; Set Z flag for put.
		C+	98	CALL	U1DVR ; Out the UART we go.
		C+	98	JR	__3111
		C+	98	__3111a	equ \$
		C+	98		endif
		C+	98		ENDMAC PUMP

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-RTC.s
		C	99	; PUMP	DEVICE.console,cngmsg
		C	100		
00509D		C	101	dspdw	FOR 1,7,1 ; Loop thru all 7 days of week.
		C+	101	SCOPE	
00509D	ED4B A9 50	C+	101	ld	bc,(__3121)
0050A1	ED43 A7 50	C+	101	ld	(__3124),bc
0050A5	18 08	C+	101	JR	__3125
0050A7	0000	C+	101	__3124	DW 0 ; I hold count when running (+8).
0050A9	0100	C+	101	__3121	DW 1 ; Start count from xx (+10).
0050AB	0700	C+	101	__3122	DW 7 ; Stop when equal or greater than STOP (+12).
0050AD	0100	C+	101	__3123	DW 1 ; Step by (+14).
	000050AF	C+	101	__3125	equ \$
		C+	101	ENDMAC	FOR
0050AF	79	C	102	LD	a,c
0050B0	32 BD 54	C	103	ld	(savcntr),a
0050B3	C630	C	104	add	'0' ; Convert to ASCII.
0050B5	32 AF 54	C	105	ld	(dowbuff),a ; Stuff ASCII character (1-7) in buffer.
0050B8	3E2E	C	106	ld	a,'.' ; Follow number with a .
0050BA	32 B0 54	C	107	ld	(dowbuff+1),a
0050BD	3E20	C	108	ld	a,' '
0050BF	32 B1 54	C	109	ld	(dowbuff+2),a ; Place a space in buffer after number and period.
		C	110	MVC	screenbuf,dowbuff+3,9 ; Always start with a fresh buffer by stuffing with spa
		C+	110	scope	
0050C2	21 DE 54	C+	110	ld	hl,screenbuf
0050C5	11 B2 54	C+	110	ld	de,dowbuff+3
0050C8	010900	C+	110	ld	bc,9
0050CB	EDB0	C+	110	LDIR	
		C+	110	ENDMAC	MVC
		C	111	STRING.table.get	(savcntr),DOWTABLE,dowbuff+3
		C+	111	scope	
0050CD	3A BD 54	C+	111	ld	a,(savcntr) ; Index value of get (1-127) to accumulator.
0050D0	3D	C+	111	dec	a
0050D1	87	C+	111	add	a,a ; index=index*2.
0050D2	5F	C+	111	ld	e,a
0050D3	1600	C+	111	ld	d,0 ; d=0 e=index*2.
0050D5	21 8F 56	C+	111	ld	hl,DOWTABLE ; Get base address of table.
0050D8	19	C+	111	add	hl,de ; Point hl to index table.
0050D9	ED07	C+	111	ld	bc,(hl) ; Pickup indexed pointer to entry.
0050DB	C5E1	C+	111	ld	hl,bc
0050DD	ED07	C+	111	ld	bc,(hl) ; 1st word of entry which is length of entry.
0050DF	23	C+	111	inc	hl ; Bump past length.
0050E0	23	C+	111	inc	hl ; Bump past length and leave hl pointing to entry.
0050E1	11 B2 54	C+	111	ld	de,dowbuff+3 ; Point DE to receiver place for data.
0050E4	EDB0	C+	111	LDIR	; Copy BC bytes from (HL) to (DE).
		C+	111	ENDMAC	STRING.table.get
		C	112	PUMP	DEVICE.console,dowbuff
		C+	112	scope	; A tribute to Dave Shriner - Display Data Corp.
0050E6	21 AF 54	C+	112	LD	HL,dowbuff ; Pump from this starting point until ETX.asc is found.
		C+	112	if	DEVICE.console=0
0050E9	7E	C+	112	__3150:	LD A,(HL) ; Get byte to send.
0050EA	23	C+	112	INC	HL ; pointer=pointer+1
		C+	112	; CP	EOT.asc ; End Of Transmission (block)?
		C+	112	; jr	Z,\$0a ; Return if finished.
0050EB	FE03	C+	112	CP	ETX.asc ; End Of Text?
0050ED	28 16	C+	112	jr	Z,__3150a ; Return if finished.
0050EF	FE0D	C+	112	CP	CR.asc ; Was it a CR?
0050F1	CC FB 50	C+	112	call	z,__3151 ; 1st add a LF to stream.
0050F4	4F	C+	112	LD	C,A ; Output byte in C.
0050F5	AF	C+	112	XOR	A ; Set Z flag for put.
0050F6	CD D9 10	C+	112	CALL	U0DVR ; Out the UART we go.

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-RTC.s
0050F9	18 EE	C+	112	JR	__3150
0050FB	3E0A	C+	112	__3151:	ld a,LF.asc ; Get a LF.
0050FD	4F	C+	112	LD	C,A ; Output byte in C.
0050FE	AF	C+	112	XOR	A ; Set Z flag for put
0050FF	CD D9 10	C+	112	CALL	U0DVR ; Out the UART we go.
005102	3E0D	C+	112	ld	a,CR.asc ; Now put a CR back in stream.
005104	C9	C+	112	ret	; Return and print CR.
005105		C+	112	__3150a	endif
		C+	112	if	DEVICE.console=1
		C+	112	__3151:	LD A,(HL) ; Get byte to send.
		C+	112	INC	HL ; pointer=pointer+1
		C+	112	CP	EOT.asc ; End Of Transmission (block)?
		C+	112	jr	Z,\$1a ; Return if finished.
		C+	112	CP	ETX.asc ; End Of Text?
		C+	112	jr	Z,__3151a ; Return if finished.
		C+	112	LD	C,A ; Output byte to C.
		C+	112	XOR	A ; Set Z flag for put.
		C+	112	CALL	U1DVR ; Out the UART we go.
		C+	112	JR	__3151
		C+	112	__3151a	equ \$
		C+	112	endif	
		C+	112	ENDMAC	PUMP
		C	113	NEXT	dspdown
		C+	113	scope	
005105	2A A7 50	C+	113	ld	hl,(dspdown+10) ; Counter.
005108	ED4B AD 50	C+	113	ld	bc,(dspdown+16) ; STEP
00510C	09	C+	113	add	hl,bc
00510D	38 0F	C+	113	JR	C,__3160 ; Overflow, exceeds limit of counter, were done.
00510F	E5	C+	113	PUSH	HL
005110	ED4B AB 50	C+	113	LD	BC,(dspdown+14) ; Get limit.
005114	A7	C+	113	and	A ; Clear carry flag.
005115	ED42	C+	113	SBC	HL,BC ; count-limit.
005117	C1	C+	113	POP	bc
005118	CA A1 50	C+	113	JP	z,dspdown+4 ; limit=counter? Go around again.
00511B	FA A1 50	C+	113	JP	m,dspdown+4 ; limit>counter? Yes, go around again.
	0000511E	C+	113	__3160:	equ \$
		C+	113	endmac	NEXT
		C	114	PUMP	DEVICE.console,getadow\$
		C+	114	scope	; A tribute to Dave Shriner - Display Data Corp.
00511E	21 EB 53	C+	114	LD	HL,getadow\$ ; Pump from this starting point until ETX.asc is found.
		C+	114	if	DEVICE.console=0
005121	7E	C+	114	__3170:	LD A,(HL) ; Get byte to send.
005122	23	C+	114	INC	HL ; pointer=pointer+1
		C+	114	CP	EOT.asc ; End Of Transmission (block)?
		C+	114	jr	Z,\$0a ; Return if finished.
005123	FE03	C+	114	CP	ETX.asc ; End Of Text?
005125	28 16	C+	114	jr	Z,__3170a ; Return if finished.
005127	FE0D	C+	114	CP	CR.asc ; Was it a CR?
005129	CC 33 51	C+	114	call	z,__3171 ; 1st add a LF to stream.
00512C	4F	C+	114	LD	C,A ; Output byte in C.
00512D	AF	C+	114	XOR	A ; Set Z flag for put.
00512E	CD D9 10	C+	114	CALL	U0DVR ; Out the UART we go.
005131	18 EE	C+	114	JR	__3170
005133	3E0A	C+	114	__3171:	ld a,LF.asc ; Get a LF.
005135	4F	C+	114	LD	C,A ; Output byte in C.
005136	AF	C+	114	XOR	A ; Set Z flag for put
005137	CD D9 10	C+	114	CALL	U0DVR ; Out the UART we go.
00513A	3E0D	C+	114	ld	a,CR.asc ; Now put a CR back in stream.
00513C	C9	C+	114	ret	; Return and print CR.
00513D		C+	114	__3170a	endif



\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-RTC.s
		C+	114	if	DEVICE.console=1
		C+	114	__3171:	LD A,(HL) ; Get byte to send.
		C+	114	INC	HL ; pointer=pointer+1
		C+	114	;	CP EOT.asc ; End Of Transmission (block)?
		C+	114	;	jr Z,\$1a ; Return if finished.
		C+	114	CP	ETX.asc ; End Of Text?
		C+	114	jr	Z,__3171a ; Return if finished.
		C+	114	LD	C,A ; Output byte to C.
		C+	114	XOR	A ; Set Z flag for put.
		C+	114	CALL	U1DVR ; Out the UART we go.
		C+	114	JR	__3171
		C+	114	__3171a	equ \$
		C+	114	endif	
		C+	114	ENDMAC	PUMP
		C	115		
		C	116	KEY_U.get	A
00513D	CD 7E 6F	C+	116	call	getchar
		C+	116	IFMA	1
005140	7F	C+	116	ld	A,a
		C+	116	else	
		C+	116	ENDIF	
		C+	116	ENDMAC	KEY_U.get
005141	FE1B	C	117	cp	ESC.asc
005143	CA 11 4D	C	118	jp	z,RTC_IPL
005146	32 AF 54	C	119	ld	(dowbuff),a
005149	D630	C	120	sub	'0' ; Convert number from ASCII to binary.
00514B	32 8B 56	C	121	ld	(savdow),a
00514E	CA 6D 50	C	122	jp	z,cngdow
005151	B7	C	123	or	A ; Zero not allowed.
005152	FE08	C	124	cp	8 ; Test if greater than 7.
005154	CA 6D 50	C	125	jp	z,cngdow
005157	D2 6D 50	C	126	jp	nc,cngdow
00515A	CD ED 6E	C	127	call	scrolloff
		C	128	MVC	screenbuf,dowbuff+3,9 ; Always start with a fresh buffer by stuffing with spa
		C+	128	scope	
00515D	21 DE 54	C+	128	ld	hl,screenbuf
005160	11 B2 54	C+	128	ld	de,dowbuff+3
005163	010900	C+	128	ld	bc,9
005166	EDB0	C+	128	LDIR	
		C+	128	ENDMAC	MVC
		C	129	STRING.table.get	(savdow),DOWTABLE,dowbuff+3
		C+	129	scope	
005168	3A 8B 56	C+	129	ld	a,(savdow) ; Index value of get (1-127) to accumulator.
00516B	3D	C+	129	dec	a
00516C	87	C+	129	add	a,a ; index=index*2.
00516D	5F	C+	129	ld	e,a
00516E	1600	C+	129	ld	d,0 ; d=0 e=index*2.
005170	21 8F 56	C+	129	ld	hl,DOWTABLE ; Get base address of table.
005173	19	C+	129	add	hl,de ; Point hl to index table.
005174	ED07	C+	129	ld	bc,(hl) ; Pickup indexed pointer to entry.
005176	C5E1	C+	129	ld	hl,bc
005178	ED07	C+	129	ld	bc,(hl) ; 1st word of entry which is length of entry.
00517A	23	C+	129	inc	hl ; Bump past length.
00517B	23	C+	129	inc	hl ; Bump past length and leave hl pointing to entry.
00517C	11 B2 54	C+	129	ld	de,dowbuff+3 ; Point DE to receiver place for data.
00517F	EDB0	C+	129	LDIR	; Copy BC bytes from (HL) to (DE).
		C+	129	ENDMAC	STRING.table.get
		C	130	PUMP	DEVICE.console,downew
		C+	130	scope	; A tribute to Dave Shriner - Display Data Corp.
005181	21 BE 54	C+	130	LD	HL,downew ; Pump from this starting point until ETX.asc is found.

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-RTC.s
		C+	130	if	DEVICE.console=0
005184	7E	C+	130	__3210:	LD A,(HL) ; Get byte to send.
005185	23	C+	130	INC	HL ; pointer=pointer+1
		C+	130	;	CP EOT.asc ; End Of Transmission (block)?
		C+	130	;	jr Z,\$0a ; Return if finished.
005186	FE03	C+	130	CP	ETX.asc ; End Of Text?
005188	28 16	C+	130	jr	Z,__3210a ; Return if finished.
00518A	FE0D	C+	130	CP	CR.asc ; Was it a CR?
00518C	CC 96 51	C+	130	call	z,__3211 ; 1st add a LF to stream.
00518F	4F	C+	130	LD	C,A ; Output byte in C.
005190	AF	C+	130	XOR	A ; Set Z flag for put.
005191	CD D9 10	C+	130	CALL	U0DVR ; Out the UART we go.
005194	18 EE	C+	130	JR	__3210
005196	3E0A	C+	130	__3211:	ld a,LF.asc ; Get a LF.
005198	4F	C+	130	LD	C,A ; Output byte in C.
005199	AF	C+	130	XOR	A ; Set Z flag for put
00519A	CD D9 10	C+	130	CALL	U0DVR ; Out the UART we go.
00519D	3E0D	C+	130	ld	a,CR.asc ; Now put a CR back in stream.
00519F	C9	C+	130	ret	; Return and print CR.
0051A0		C+	130	__3210a	endif
		C+	130	if	DEVICE.console=1
		C+	130	__3211:	LD A,(HL) ; Get byte to send.
		C+	130	INC	HL ; pointer=pointer+1
		C+	130	;	CP EOT.asc ; End Of Transmission (block)?
		C+	130	;	jr Z,\$1a ; Return if finished.
		C+	130	CP	ETX.asc ; End Of Text?
		C+	130	jr	Z,__3211a ; Return if finished.
		C+	130	LD	C,A ; Output byte to C.
		C+	130	XOR	A ; Set Z flag for put.
		C+	130	CALL	U1DVR ; Out the UART we go.
		C+	130	JR	__3211
		C+	130	__3211a	equ \$
		C+	130	endif	
		C+	130	ENDMAC	PUMP
		C	131	PUMP	DEVICE.console,dowbuff+3
		C+	131	scope	; A tribute to Dave Shriner - Display Data Corp.
0051A0	21 B2 54	C+	131	LD	HL,dowbuff+3 ; Pump from this starting point until ETX.asc is found.
		C+	131	if	DEVICE.console=0
0051A3	7E	C+	131	__3220:	LD A,(HL) ; Get byte to send.
0051A4	23	C+	131	INC	HL ; pointer=pointer+1
		C+	131	;	CP EOT.asc ; End Of Transmission (block)?
		C+	131	;	jr Z,\$0a ; Return if finished.
0051A5	FE03	C+	131	CP	ETX.asc ; End Of Text?
0051A7	28 16	C+	131	jr	Z,__3220a ; Return if finished.
0051A9	FE0D	C+	131	CP	CR.asc ; Was it a CR?
0051AB	CC B5 51	C+	131	call	z,__3221 ; 1st add a LF to stream.
0051AE	4F	C+	131	LD	C,A ; Output byte in C.
0051AF	AF	C+	131	XOR	A ; Set Z flag for put.
0051B0	CD D9 10	C+	131	CALL	U0DVR ; Out the UART we go.
0051B3	18 EE	C+	131	JR	__3220
0051B5	3E0A	C+	131	__3221:	ld a,LF.asc ; Get a LF.
0051B7	4F	C+	131	LD	C,A ; Output byte in C.
0051B8	AF	C+	131	XOR	A ; Set Z flag for put
0051B9	CD D9 10	C+	131	CALL	U0DVR ; Out the UART we go.
0051BC	3E0D	C+	131	ld	a,CR.asc ; Now put a CR back in stream.
0051BE	C9	C+	131	ret	; Return and print CR.
0051BF		C+	131	__3220a	endif
		C+	131	if	DEVICE.console=1
		C+	131	__3221:	LD A,(HL) ; Get byte to send.
		C+	131	INC	HL ; pointer=pointer+1

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	..\SYS0_IPL\ipl-RTC.s
		C+	131	;	CP EOT.asc ; End Of Transmission (block)?
		C+	131	;	jr Z,\$1a ; Return if finished.
		C+	131		CP ETX.asc ; End Of Text?
		C+	131		jr Z,__3221a ; Return if finished.
		C+	131		LD C,A ; Output byte to C.
		C+	131		XOR A ; Set Z flag for put.
		C+	131		CALL U1DVR ; Out the UART we go.
		C+	131		JR __3221
		C+	131	__3221a	equ \$
		C+	131		endif
		C+	131		ENDMAC PUMP
		C	132		PUMP DEVICE.console,cngdowverify
		C+	132		scope ; A tribute to Dave Shriner - Display Data Corp.
0051BF	21 19 54	C+	132		LD HL,cngdowverify ; Pump from this starting point until ETX.asc is found.
		C+	132		if DEVICE.console=0
0051C2	7E	C+	132	__3230:	LD A,(HL) ; Get byte to send.
0051C3	23	C+	132		INC HL ; pointer=pointer+1
		C+	132	;	CP EOT.asc ; End Of Transmission (block)?
		C+	132	;	jr Z,\$0a ; Return if finished.
		C+	132		CP ETX.asc ; End Of Text?
0051C4	FE03	C+	132		jr Z,__3230a ; Return if finished.
0051C6	28 16	C+	132		CP CR.asc ; Was it a CR?
0051C8	FE0D	C+	132		call z,__3231 ; 1st add a LF to stream.
0051CA	CC D4 51	C+	132		LD C,A ; Output byte in C.
0051CD	4F	C+	132		XOR A ; Set Z flag for put.
0051CE	AF	C+	132		CALL U0DVR ; Out the UART we go.
0051CF	CD D9 10	C+	132		JR __3230
0051D2	18 EE	C+	132		
0051D4	3E0A	C+	132	__3231:	ld a,LF.asc ; Get a LF.
0051D6	4F	C+	132		LD C,A ; Output byte in C.
0051D7	AF	C+	132		XOR A ; Set Z flag for put
0051D8	CD D9 10	C+	132		CALL U0DVR ; Out the UART we go.
0051DB	3E0D	C+	132		ld a,CR.asc ; Now put a CR back in stream.
0051DD	C9	C+	132		ret ; Return and print CR.
0051DE		C+	132	__3230a	endif
		C+	132		if DEVICE.console=1
		C+	132	__3231:	LD A,(HL) ; Get byte to send.
		C+	132		INC HL ; pointer=pointer+1
		C+	132	;	CP EOT.asc ; End Of Transmission (block)?
		C+	132	;	jr Z,\$1a ; Return if finished.
		C+	132		CP ETX.asc ; End Of Text?
		C+	132		jr Z,__3231a ; Return if finished.
		C+	132		LD C,A ; Output byte to C.
		C+	132		XOR A ; Set Z flag for put.
		C+	132		CALL U1DVR ; Out the UART we go.
		C+	132		JR __3231
		C+	132	__3231a	equ \$
		C+	132		endif
		C+	132		ENDMAC PUMP
		C	133		
		C	134		KEY_U.get
0051DE	CD 7E 6F	C+	134		call getchar
		C+	134		IFMA 1
		C+	134		ld ,a
		C+	134		else
		C+	134		ENDIF
		C+	134		ENDMAC KEY_U.get
0051E1	FE1B	C	135		cp ESC.asc
0051E3	CA 11 4D	C	136		jp z,RTC_IPL
0051E6	FE04	C	137		cp EOT.asc
0051E8	C2 6D 50	C	138		jp nz,cngdow

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source
		C	139	Source ..\SYS0_IPL\ipl-RTC.s
		C	139	RTC.unlock
0051EB	3E21	C+	139	ld a,00100001b ; Unlock & disable interrupts.
0051ED	ED39ED	C+	139	OUT0 (RTC_CTRL),a
		C+	139	endmac RTC.unlock
		C	140	RTC.DOW.put (savdow)
0051F0	3A 8B 56	C+	140	ld a,(savdow)
0051F3	ED39E3	C+	140	out0 (RTC_DOW),A
		C+	140	endmac
		C	141	RTC.lock
0051F6	3E20	C+	141	ld a,00100000b ; Lock clock & disable interrupts.
0051F8	ED39ED	C+	141	OUT0 (RTC_CTRL),a ; Execute.
		C+	141	ENDMAC RTC.lock
		C	142	PUMP DEVICE.console,cngcomplete
		C+	142	scope ; A tribute to Dave Shriner - Display Data Corp.
0051FB	21 F6 55	C+	142	LD HL,cngcomplete ; Pump from this starting point until ETX.asc is found.
		C+	142	if DEVICE.console=0
0051FE	7E	C+	142	__3280: LD A,(HL) ; Get byte to send.
0051FF	23	C+	142	INC HL ; pointer=pointer+1
		C+	142	; CP EOT.asc ; End Of Transmission (block)?
		C+	142	; jr Z,\$0a ; Return if finished.
005200	FE03	C+	142	CP ETX.asc ; End Of Text?
005202	28 16	C+	142	jr Z,__3280a ; Return if finished.
005204	FE0D	C+	142	CP CR.asc ; Was it a CR?
005206	CC 10 52	C+	142	call z,__3281 ; 1st add a LF to stream.
005209	4F	C+	142	LD C,A ; Output byte in C.
00520A	AF	C+	142	XOR A ; Set Z flag for put.
00520B	CD D9 10	C+	142	CALL U0DVR ; Out the UART we go.
00520E	18 EE	C+	142	JR __3280
005210	3E0A	C+	142	__3281: ld a,LF.asc ; Get a LF.
005212	4F	C+	142	LD C,A ; Output byte in C.
005213	AF	C+	142	XOR A ; Set Z flag for put
005214	CD D9 10	C+	142	CALL U0DVR ; Out the UART we go.
005217	3E0D	C+	142	ld a,CR.asc ; Now put a CR back in stream.
005219	C9	C+	142	ret ; Return and print CR.
00521A		C+	142	__3280a endif
		C+	142	if DEVICE.console=1
		C+	142	__3281: LD A,(HL) ; Get byte to send.
		C+	142	INC HL ; pointer=pointer+1
		C+	142	; CP EOT.asc ; End Of Transmission (block)?
		C+	142	; jr Z,\$1a ; Return if finished.
		C+	142	CP ETX.asc ; End Of Text?
		C+	142	jr Z,__3281a ; Return if finished.
		C+	142	LD C,A ; Output byte to C.
		C+	142	XOR A ; Set Z flag for put.
		C+	142	CALL U1DVR ; Out the UART we go.
		C+	142	JR __3281
		C+	142	__3281a equ \$
		C+	142	endif
		C+	142	ENDMAC PUMP
00521A		C	143	regetrsp2KEY_U.get
00521A	CD 7E 6F	C+	143	call getchar
		C+	143	IFMA 1
		C+	143	ld ,a
		C+	143	else
		C+	143	ENDIF
		C+	143	ENDMAC KEY_U.get
00521D	FE0D	C	144	cp CR.asc
00521F	CA 11 4D	C	145	JP z,RTC_IPL
005222	C3 1A 52	C	146	jp regetrsp2
		C	147	

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source
005225	CD ED 6E	C	148	.. \SYS0_IPL\ipl-RTC.s cngtim call scrolloff
		C	149	PUMP DEVICE.console,cngtimscrn
		C+	149	scope ; A tribute to Dave Shriner - Display Data Corp.
005228	21 A9 58	C+	149	LD HL,cngtimscrn ; Pump from this starting point until ETX.asc is found.
		C+	149	if DEVICE.console=0
00522B	7E	C+	149	__3300: LD A,(HL) ; Get byte to send.
00522C	23	C+	149	INC HL ; pointer=pointer+1
		C+	149	; CP EOT.asc ; End Of Transmission (block)?
		C+	149	; jr Z,\$0a ; Return if finished.
00522D	FE03	C+	149	CP ETX.asc ; End Of Text?
00522F	28 16	C+	149	jr Z,__3300a ; Return if finished.
005231	FE0D	C+	149	CP CR.asc ; Was it a CR?
005233	CC 3D 52	C+	149	call z,__3301 ; 1st add a LF to stream.
005236	4F	C+	149	LD C,A ; Output byte in C.
005237	AF	C+	149	XOR A ; Set Z flag for put.
005238	CD D9 10	C+	149	CALL U0DVR ; Out the UART we go.
00523B	18 EE	C+	149	JR __3300
00523D	3E0A	C+	149	__3301: ld a,LF.asc ; Get a LF.
00523F	4F	C+	149	LD C,A ; Output byte in C.
005240	AF	C+	149	XOR A ; Set Z flag for put
005241	CD D9 10	C+	149	CALL U0DVR ; Out the UART we go.
005244	3E0D	C+	149	ld a,CR.asc ; Now put a CR back in stream.
005246	C9	C+	149	ret ; Return and print CR.
005247		C+	149	__3300a endif
		C+	149	if DEVICE.console=1
		C+	149	__3301: LD A,(HL) ; Get byte to send.
		C+	149	INC HL ; pointer=pointer+1
		C+	149	; CP EOT.asc ; End Of Transmission (block)?
		C+	149	; jr Z,\$1a ; Return if finished.
		C+	149	CP ETX.asc ; End Of Text?
		C+	149	jr Z,__3301a ; Return if finished.
		C+	149	LD C,A ; Output byte to C.
		C+	149	XOR A ; Set Z flag for put.
		C+	149	CALL U1DVR ; Out the UART we go.
		C+	149	JR __3301
		C+	149	__3301a equ \$
		C+	149	endif
		C+	149	ENDMAC PUMP
		C	150	PUMP DEVICE.console,cnghrsprompt
		C+	150	scope ; A tribute to Dave Shriner - Display Data Corp.
005247	21 7F 58	C+	150	LD HL,cnghrsprompt ; Pump from this starting point until ETX.asc is found.
		C+	150	if DEVICE.console=0
00524A	7E	C+	150	__3310: LD A,(HL) ; Get byte to send.
00524B	23	C+	150	INC HL ; pointer=pointer+1
		C+	150	; CP EOT.asc ; End Of Transmission (block)?
		C+	150	; jr Z,\$0a ; Return if finished.
00524C	FE03	C+	150	CP ETX.asc ; End Of Text?
00524E	28 16	C+	150	jr Z,__3310a ; Return if finished.
005250	FE0D	C+	150	CP CR.asc ; Was it a CR?
005252	CC 5C 52	C+	150	call z,__3311 ; 1st add a LF to stream.
005255	4F	C+	150	LD C,A ; Output byte in C.
005256	AF	C+	150	XOR A ; Set Z flag for put.
005257	CD D9 10	C+	150	CALL U0DVR ; Out the UART we go.
00525A	18 EE	C+	150	JR __3310
00525C	3E0A	C+	150	__3311: ld a,LF.asc ; Get a LF.
00525E	4F	C+	150	LD C,A ; Output byte in C.
00525F	AF	C+	150	XOR A ; Set Z flag for put
005260	CD D9 10	C+	150	CALL U0DVR ; Out the UART we go.
005263	3E0D	C+	150	ld a,CR.asc ; Now put a CR back in stream.
005265	C9	C+	150	ret ; Return and print CR.

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-RTC.s
005266		C+	150	__3310a	endif
		C+	150		if DEVICE.console=1
		C+	150	__3311:	LD A,(HL) ; Get byte to send.
		C+	150		INC HL ; pointer=pointer+1
		C+	150	;	CP EOT.asc ; End Of Transmission (block)?
		C+	150	;	jz \$1a ; Return if finished.
		C+	150		CP ETX.asc ; End Of Text?
		C+	150		jz __3311a ; Return if finished.
		C+	150		LD C,A ; Output byte to C.
		C+	150		XOR A ; Set Z flag for put.
		C+	150		CALL U1DVR ; Out the UART we go.
		C+	150		JR __3311
		C+	150	__3311a	equ \$
		C+	150		endif
		C+	150		ENDMAC PUMP
		C	151		INPUT_U.str editbuffer,2
		C+	151		scope
005266 21 D8 54		C+	151		ld hl,editbuffer
005269 0602		C+	151		ld b,2 ; Max number bytes to input.
00526B 18 17		C+	151		jz __3321 ; Branch to beginning.
00526D 4F		C+	151	__3322:	ld c,a ; Output BS just received.
00526E CD EB 11		C+	151		CALL TXUART0 ; Send it.
005271 0E20		C+	151		ld c,SP.asc ; Now space out char we are deleting.
005273 CD EB 11		C+	151		CALL TXUART0
005276 0E08		C+	151		ld c,BS.asc ; Backspace again to new location.
005278 CD EB 11		C+	151		CALL TXUART0
00527B 2B		C+	151		dec hl ; Back pointer in buffer up by -1.
00527C C1		C+	151		pop bc ; Recover B counter.
00527D 04		C+	151		INC B
00527E FE02		C+	151		cp 2
005280 30 02		C+	151		jz nc,__3321
005282 0602		C+	151		ld b,2
005284 C5		C+	151	__3321:	push bc ; Save counter as other routines may clobber it.
005285 CD 7E 6F		C+	151		CALL getchar ; Using UART NOT TRSDOS get one byte from UART0.
005288 FE08		C+	151		CP BS.asc ; Is is backspace?
00528A 28 E1		C+	151		jz z,__3322 ; If yes go to backspace routine above.
00528C 77		C+	151		ld (hl),a ; Put char in buff.
00528D 23		C+	151		inc hl ; Bump pointer to next place in buffer.
00528E FE0D		C+	151		cp CR.asc ; Was it a CR?
005290 28 0A		C+	151		jz z,__3324 ; Exit input as terminalted by CR.
005292 4F		C+	151		ld c,a
005293 CD EB 11		C+	151		CALL TXUART0 ; Output character.
005296 C1		C+	151	__3323	pop bc ; Recover counter.
005297 10 EB		C+	151		djnz __3321 ; Counter = counter - 1. If counter <> 0 branch \$1.
005299 AF		C+	151		xor a ; All is well set Z flag.
00529A 18 01		C+	151		jz __3325 ; Finished, exit.
00529C B7		C+	151	__3324:	or a ; Make it non zero.
00529D		C+	151	__3325:	endmac INPUT_U.str ; MACRO is done.
00529D 78		C	152		ld a,b
		C	153		PUMP DEVICE.console,CR\$
		C+	153		scope ; A tribute to Dave Shriner - Display Data Corp.
00529E 21 24 5A		C+	153		LD HL,CR\$ ; Pump from this starting point until ETX.asc is found.
		C+	153		if DEVICE.console=0
0052A1 7E		C+	153	__3330:	LD A,(HL) ; Get byte to send.
0052A2 23		C+	153		INC HL ; pointer=pointer+1
		C+	153	;	CP EOT.asc ; End Of Transmission (block)?
		C+	153	;	jz \$0a ; Return if finished.
0052A3 FE03		C+	153		CP ETX.asc ; End Of Text?
0052A5 28 16		C+	153		jz z,__3330a ; Return if finished.
0052A7 FE0D		C+	153		CP CR.asc ; Was it a CR?

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-RTC.s
0052A9	CC B3 52	C+	153	call	z, __3331 ; 1st add a LF to stream.
0052AC	4F	C+	153	LD	C,A ; Output byte in C.
0052AD	AF	C+	153	XOR	A ; Set Z flag for put.
0052AE	CD D9 10	C+	153	CALL	U0DVR ; Out the UART we go.
0052B1	18 EE	C+	153	JR	__3330
0052B3	3E0A	C+	153	__3331:	ld a,LF.asc ; Get a LF.
0052B5	4F	C+	153	LD	C,A ; Output byte in C.
0052B6	AF	C+	153	XOR	A ; Set Z flag for put
0052B7	CD D9 10	C+	153	CALL	U0DVR ; Out the UART we go.
0052BA	3E0D	C+	153	ld	a,CR.asc ; Now put a CR back in stream.
0052BC	C9	C+	153	ret	; Return and print CR.
0052BD		C+	153	__3330a	endif
		C+	153		if DEVICE.console=1
		C+	153	__3331:	LD A,(HL) ; Get byte to send.
		C+	153		INC HL ; pointer=pointer+1
		C+	153		CP EOT.asc ; End Of Transmission (block)?
		C+	153		; jr Z,\$1a ; Return if finished.
		C+	153		CP ETX.asc ; End Of Text?
		C+	153		; jr Z,__3331a ; Return if finished.
		C+	153		LD C,A ; Output byte to C.
		C+	153		XOR A ; Set Z flag for put.
		C+	153		CALL U1DVR ; Out the UART we go.
		C+	153		JR __3331
		C+	153	__3331a	equ \$
		C+	153		endif
		C+	153	ENDMAC	PUMP
		C	154		
		C	155	DECHEX	editbuffer ; Convert ASCII in buffer to binary.
0052BD	21 D8 54	C+	155	ld	hl,editbuffer
		C+	155	SVC	svc_dechex
0052C0	3E60	C+	155	ld	a,svc_dechex
0052C2	EF	C+	155	restart28	
		C+	155	ENDMAC	SVC
		C+	155	ENDMAC	DECHEX
0052C3	79	C	156	ld	a,c ; Binary results are in BC. We only need lower 8 bits.
0052C4	FE18	C	157	cp	24 ; Make sure hours <24.
0052C6	D2 25 52	C	158	jp	nc,cngtim ; Branch if 24 or >.
0052C9	32 8C 56	C	159	ld	(savhrs),a ; We have a new hours so save in hour buffer.
		C	160	HEXDEC	BC,cnghr ; BC still has binary, convert ascii & stuff in menu.
0052CC	C5E1	C+	160	ld	hl,BC
0052CE	11 DC 59	C+	160	ld	de,cnghr
		C+	160	SVC	svc_hexdec
0052D1	3E61	C+	160	ld	a,svc_hexdec
0052D3	EF	C+	160	restart28	
		C+	160	ENDMAC	SVC
		C+	160	ENDMAC	HEXDEC
		C	161		
		C	162		; Prompt change minutes.
		C	163		
0052D4		C	164	cngmn	PUMP DEVICE.console,cngminprompt ; Get minutes in ascii
		C+	164	scope	; A tribute to Dave Shriner - Display Data Corp.
0052D4	21 93 58	C+	164	LD	HL,cngminprompt ; Pump from this starting point until ETX.asc is found.
		C+	164		if DEVICE.console=0
0052D7	7E	C+	164	__3380:	LD A,(HL) ; Get byte to send.
0052D8	23	C+	164	INC	HL ; pointer=pointer+1
		C+	164		; CP EOT.asc ; End Of Transmission (block)?
		C+	164		; jr Z,\$0a ; Return if finished.
0052D9	FE03	C+	164	CP	ETX.asc ; End Of Text?
0052DB	28 16	C+	164	jr	Z,__3380a ; Return if finished.
0052DD	FE0D	C+	164	CP	CR.asc ; Was it a CR?

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-RTC.s
0052DF	CC E9 52	C+	164	call	z, __3381 ; 1st add a LF to stream.
0052E2	4F	C+	164	LD	C,A ; Output byte in C.
0052E3	AF	C+	164	XOR	A ; Set Z flag for put.
0052E4	CD D9 10	C+	164	CALL	U0DVR ; Out the UART we go.
0052E7	18 EE	C+	164	JR	__3380
0052E9	3E0A	C+	164	__3381:	ld a, LF.asc ; Get a LF.
0052EB	4F	C+	164	LD	C,A ; Output byte in C.
0052EC	AF	C+	164	XOR	A ; Set Z flag for put
0052ED	CD D9 10	C+	164	CALL	U0DVR ; Out the UART we go.
0052F0	3E0D	C+	164	ld	a, CR.asc ; Now put a CR back in stream.
0052F2	C9	C+	164	ret	; Return and print CR.
0052F3		C+	164	__3380a	endif
		C+	164		if DEVICE.console=1
		C+	164	__3381:	LD A, (HL) ; Get byte to send.
		C+	164		INC HL ; pointer=pointer+1
		C+	164		CP EOT.asc ; End Of Transmission (block)?
		C+	164		; jr Z, \$1a ; Return if finished.
		C+	164		CP ETX.asc ; End Of Text?
		C+	164		; jr Z, __3381a ; Return if finished.
		C+	164		LD C,A ; Output byte to C.
		C+	164		XOR A ; Set Z flag for put.
		C+	164		CALL U1DVR ; Out the UART we go.
		C+	164		JR __3381
		C+	164	__3381a	equ \$
		C+	164		endif
		C+	164	ENDMAC	PUMP
		C	165	INPUT_U.str	editbuffer, 2 ; into editbuffer.
		C+	165	scope	
0052F3	21 D8 54	C+	165	ld	hl, editbuffer
0052F6	0602	C+	165	ld	b, 2 ; Max number bytes to input.
0052F8	18 17	C+	165	jr	__3391 ; Branch to beginning.
0052FA	4F	C+	165	__3392:	ld c, a ; Output BS just received.
0052FB	CD EB 11	C+	165	CALL	TXUART0 ; Send it.
0052FE	0E20	C+	165	ld	c, SP.asc ; Now space out char we are deleting.
005300	CD EB 11	C+	165	CALL	TXUART0
005303	0E08	C+	165	ld	c, BS.asc ; Backspace again to new location.
005305	CD EB 11	C+	165	CALL	TXUART0
005308	2B	C+	165	dec	hl ; Back pointer in buffer up by -1.
005309	C1	C+	165	pop	bc ; Recover B counter.
00530A	04	C+	165	INC	B
00530B	FE02	C+	165	cp	2
00530D	30 02	C+	165	jr	nc, __3391
00530F	0602	C+	165	ld	b, 2
005311	C5	C+	165	__3391:	push bc ; Save counter as other routines may clobber it.
005312	CD 7E 6F	C+	165	CALL	getchar ; Using UART NOT TRSDOS get one byte from UART0.
005315	FE08	C+	165	CP	BS.asc ; Is is backspace?
005317	28 E1	C+	165	jr	z, __3392 ; If yes gp to backspace routine above.
005319	77	C+	165	ld	(hl), a ; Put char in buff.
00531A	23	C+	165	inc	hl ; Bump pointer to next place in buffer.
00531B	FE0D	C+	165	cp	CR.asc ; Was it a CR?
00531D	28 0A	C+	165	jr	z, __3394 ; Exit input as terminalted by CR.
00531F	4F	C+	165	ld	c, a
005320	CD EB 11	C+	165	CALL	TXUART0 ; Output character.
005323	C1	C+	165	__3393	pop bc ; Recover counter.
005324	10 EB	C+	165	djnz	__3391 ; Counter = counter - 1. If counter <> 0 branch \$1.
005326	AF	C+	165	xor	a ; All is well set Z flag.
005327	18 01	C+	165	jr	__3395 ; Finished, exit.
005329	B7	C+	165	__3394:	or a ; Make it non zero.
00532A		C+	165	__3395:	endmac INPUT_U.str ; MACRO is done.
00532A	78	C	166	ld	a, b ; I think this does nothing.



\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	..\\SYS0_IPL\\ipl-RTC.s
		C	167	PUMP	DEVICE.console,CR\$
		C+	167	scope	; A tribute to Dave Shriner - Display Data Corp.
00532B	21 24 5A	C+	167	LD	HL,CR\$ ; Pump from this starting point until ETX.asc is found.
		C+	167	if	DEVICE.console=0
00532E	7E	C+	167	__3400:	LD A,(HL) ; Get byte to send.
00532F	23	C+	167	INC	HL ; pointer=pointer+1
		C+	167	; CP	EOT.asc ; End Of Transmission (block)?
		C+	167	; jr	Z,\$0a ; Return if finished.
005330	FE03	C+	167	CP	ETX.asc ; End Of Text?
005332	28 16	C+	167	jr	Z,__3400a ; Return if finished.
005334	FE0D	C+	167	CP	CR.asc ; Was it a CR?
005336	CC 40 53	C+	167	call	z,__3401 ; 1st add a LF to stream.
005339	4F	C+	167	LD	C,A ; Output byte in C.
00533A	AF	C+	167	XOR	A ; Set Z flag for put.
00533B	CD D9 10	C+	167	CALL	U0DVR ; Out the UART we go.
00533E	18 EE	C+	167	JR	__3400
005340	3E0A	C+	167	__3401:	ld a,LF.asc ; Get a LF.
005342	4F	C+	167	LD	C,A ; Output byte in C.
005343	AF	C+	167	XOR	A ; Set Z flag for put
005344	CD D9 10	C+	167	CALL	U0DVR ; Out the UART we go.
005347	3E0D	C+	167	ld	a,CR.asc ; Now put a CR back in stream.
005349	C9	C+	167	ret	; Return and print CR.
00534A		C+	167	__3400a	endif
		C+	167	if	DEVICE.console=1
		C+	167	__3401:	LD A,(HL) ; Get byte to send.
		C+	167	INC	HL ; pointer=pointer+1
		C+	167	; CP	EOT.asc ; End Of Transmission (block)?
		C+	167	; jr	Z,\$1a ; Return if finished.
		C+	167	CP	ETX.asc ; End Of Text?
		C+	167	jr	Z,__3401a ; Return if finished.
		C+	167	LD	C,A ; Output byte to C.
		C+	167	XOR	A ; Set Z flag for put.
		C+	167	CALL	U1DVR ; Out the UART we go.
		C+	167	JR	__3401
		C+	167	__3401a	equ \$
		C+	167	endif	
		C+	167	ENDMAC	PUMP
		C	168		
		C	169	DECHEX	editbuffer ; Convert 2 digit buffer to binary.
00534A	21 D8 54	C+	169	ld	hl,editbuffer
		C+	169	SVC	svc_dechex
00534D	3E60	C+	169	ld	a,svc_dechex
00534F	EF	C+	169	restart28	
		C+	169	ENDMAC	SVC
		C+	169	ENDMAC	DECHEX
005350	79	C	170	ld	a,c ; Results in BC.
005351	FE3C	C	171	cp	60 ; > 59?
005353	D2 D4 52	C	172	jp	nc,cngmn ; Jump if so.
005356	32 8D 56	C	173	ld	(savmin),a ; Else save into minutes.
		C	174	HEXDEC	BC,cngmin ; Convert binary BC to ASCII & stuff in menu.
005359	C5E1	C+	174	ld	hl,BC
00535B	11 EF 59	C+	174	ld	de,cngmin
		C+	174	SVC	svc_hexdec
00535E	3E61	C+	174	ld	a,svc_hexdec
005360	EF	C+	174	restart28	
		C+	174	ENDMAC	SVC
		C+	174	ENDMAC	HEXDEC
		C	175		
		C	176		; This will display all changes in a small menu.
		C	177		; Menu prompts for ^D confirm or ESC abort.

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-RTC.s
		C	178		
		C	179		PUMP DEVICE.console,cnghrsnew
		C+	179		scope ; A tribute to Dave Shriner - Display Data Corp.
005361	21 D1 59	C+	179		LD HL,cnghrsnew ; Pump from this starting point until ETX.asc is found.
		C+	179		if DEVICE.console=0
005364	7E	C+	179	__3450:	LD A,(HL) ; Get byte to send.
005365	23	C+	179		INC HL ; pointer=pointer+1
		C+	179		; CP EOT.asc ; End Of Transmission (block)?
		C+	179		; jr Z,\$0a ; Return if finished.
005366	FE03	C+	179		CP ETX.asc ; End Of Text?
005368	28 16	C+	179		jr Z,__3450a ; Return if finished.
00536A	FE0D	C+	179		CP CR.asc ; Was it a CR?
00536C	CC 76 53	C+	179		call z,__3451 ; 1st add a LF to stream.
00536F	4F	C+	179		LD C,A ; Output byte in C.
005370	AF	C+	179		XOR A ; Set Z flag for put.
005371	CD D9 10	C+	179		CALL U0DVR ; Out the UART we go.
005374	18 EE	C+	179		JR __3450
005376	3E0A	C+	179	__3451:	ld a,LF.asc ; Get a LF.
005378	4F	C+	179		LD C,A ; Output byte in C.
005379	AF	C+	179		XOR A ; Set Z flag for put
00537A	CD D9 10	C+	179		CALL U0DVR ; Out the UART we go.
00537D	3E0D	C+	179		ld a,CR.asc ; Now put a CR back in stream.
00537F	C9	C+	179		ret ; Return and print CR.
005380		C+	179	__3450a	endif
		C+	179		if DEVICE.console=1
		C+	179	__3451:	LD A,(HL) ; Get byte to send.
		C+	179		INC HL ; pointer=pointer+1
		C+	179		; CP EOT.asc ; End Of Transmission (block)?
		C+	179		; jr Z,\$1a ; Return if finished.
		C+	179		CP ETX.asc ; End Of Text?
		C+	179		jr Z,__3451a ; Return if finished.
		C+	179		LD C,A ; Output byte to C.
		C+	179		XOR A ; Set Z flag for put.
		C+	179		CALL U1DVR ; Out the UART we go.
		C+	179		JR __3451
		C+	179	__3451a	equ \$
		C+	179		endif
		C+	179		ENDMAC PUMP
005380	CD 7E 6F	C	180		KEY_U.get
		C+	180		call getchar
		C+	180		IFMA 1
		C+	180		ld ,a
		C+	180		else
		C+	180		ENDIF
		C+	180		ENDMAC KEY_U.get
005383	FE1B	C	181		CP ESC.asc ; ABORT?
005385	CA 11 4D	C	182		JP z,RTC_IPL ; Yes? Return to main IPL menu.
005388	FE04	C	183		cp EOT.asc ; ^D?
00538A	C2 25 52	C	184		jp nz,cngtim ; If not keep prompting for new time until abort or save.
		C	185		
		C	186		; Code to change physical time clock is here.
		C	187		
		C	188		RTC.unlock
00538D	3E21	C+	188		ld a,00100001b ; Unlock & disable interrupts.
00538F	ED39ED	C+	188		OUT0 (RTC_CTRL),a
		C+	188		endmac RTC.unlock
005392	3A 8C 56	C	189		ld a,(savhrs)
005395	CD DA 12	C	190		call bin2bcd
		C	191		RTC.HOUR.put a
005398	7F	C+	191		ld a,a

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source
005399	ED39E2	C+	191	out0 (RTC_HRS),a ;
		C+	191	ENDMAC
00539C	3A 8D 56	C	192	ld a,(savmin)
00539F	CD DA 12	C	193	call bin2bcd
		C	194	RTC.MIN.put a
0053A2	7F	C+	194	ld a,a
0053A3	ED39E1	C+	194	out0 (RTC_MIN),a ;
		C+	194	ENDMAC
0053A6	3A 8E 56	C	195	ld a,(savsec)
0053A9	CD DA 12	C	196	call bin2bcd
		C	197	RTC.SEC.put a
0053AC	7F	C+	197	ld a,a
0053AD	ED39E0	C+	197	out0 (RTC_SEC),a ;
		C+	197	ENDMAC
		C	198	RTC.lock
0053B0	3E20	C+	198	ld a,00100000b ; Lock clock & disable interrupts.
0053B2	ED39ED	C+	198	OUT0 (RTC_CTRL),a ; Execute.
		C+	198	ENDMAC RTC.lock
		C	199	
		C	200	PUMP DEVICE.console,cngcomplete
		C+	200	scope ; A tribute to Dave Shriner - Display Data Corp.
0053B5	21 F6 55	C+	200	LD HL,cngcomplete ; Pump from this starting point until ETX.asc is found.
		C+	200	if DEVICE.console=0
0053B8	7E	C+	200	__3520: LD A,(HL) ; Get byte to send.
0053B9	23	C+	200	INC HL ; pointer=pointer+1
		C+	200	; CP EOT.asc ; End Of Transmission (block)?
		C+	200	; jr Z,\$0a ; Return if finished.
0053BA	FE03	C+	200	CP ETX.asc ; End Of Text?
0053BC	28 16	C+	200	jr Z,__3520a ; Return if finished.
0053BE	FE0D	C+	200	CP CR.asc ; Was it a CR?
0053C0	CC CA 53	C+	200	call z,__3521 ; 1st add a LF to stream.
0053C3	4F	C+	200	LD C,A ; Output byte in C.
0053C4	AF	C+	200	XOR A ; Set Z flag for put.
0053C5	CD D9 10	C+	200	CALL U0DVR ; Out the UART we go.
0053C8	18 EE	C+	200	JR __3520
0053CA	3E0A	C+	200	__3521: ld a,LF.asc ; Get a LF.
0053CC	4F	C+	200	LD C,A ; Output byte in C.
0053CD	AF	C+	200	XOR A ; Set Z flag for put
0053CE	CD D9 10	C+	200	CALL U0DVR ; Out the UART we go.
0053D1	3E0D	C+	200	ld a,CR.asc ; Now put a CR back in stream.
0053D3	C9	C+	200	ret ; Return and print CR.
0053D4		C+	200	__3520a endif
		C+	200	if DEVICE.console=1
		C+	200	__3521: LD A,(HL) ; Get byte to send.
		C+	200	INC HL ; pointer=pointer+1
		C+	200	; CP EOT.asc ; End Of Transmission (block)?
		C+	200	; jr Z,\$1a ; Return if finished.
		C+	200	CP ETX.asc ; End Of Text?
		C+	200	jr Z,__3521a ; Return if finished.
		C+	200	LD C,A ; Output byte to C.
		C+	200	XOR A ; Set Z flag for put.
		C+	200	CALL U1DVR ; Out the UART we go.
		C+	200	JR __3521
		C+	200	__3521a equ \$
		C+	200	endif
		C+	200	ENDMAC PUMP
0053D4		C	201	regetrsp3KEY_U.get
0053D4	CD 7E 6F	C+	201	call getchar
		C+	201	IFMA 1
		C+	201	ld ,a

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-RTC.s
		C+	201	else	
		C+	201	ENDIF	
		C+	201	ENDMAC	KEY_U.get
0053D7	C3 11 4D	C	202	JP	RTC_IPL
		C	203		
0053DA		C	204	getdow	RTC.DOW.get A
0053DA	ED38E3	C+	204	in0	a, (RTC_DOW)
0053DD	B7	C+	204	or	a
0053DE	20 02	C+	204	jr	nz, __3541
0053E0	3E02	C+	204	ld	a, 02h
0053E2	7F	C+	204	__3541	ld A, a
		C+	204	ENDMAC	
0053E3	E607	C	205	and	7 ; Strip all but lower 3 bits (0-7).
0053E5	B7	C	206	or	a ; Test if not initialized and happens to have a 0.
0053E6	20 02	C	207	jr	nz, itsnot0 ; It is not a 0, thus between 1-7 (Sun-Sat).
0053E8	3E01	C	208	ld	a, 1 ; Make it Sunday.
0053EA	C9	C	209	itsnot0	ret
		C	0	include	"ipl_rtc_screens.s" ; Menu & screen data for RTC routines.
0053EB	0A	C	1	getadow\$	db LF.asc
0053EC	44617920 6F662077	C	2	db	"Day of week (1-7) you want to change RTC to."
0053F4	65656B20 28312D37				
0053FC	2920796F 75207761				
005404	6E742074 6F206368				
00540C	616E6765 20525443				
005414	20746F2E				
005418	03	C	3	db	ETX.asc
		C	4		
005419	0A	C	5	cngdowverify	db LF.asc
00541A	56657269 66792063	C	6	db	"Verify changes by sending a CONTROL^D now or ESC to ABORT.", CR.asc
005422	68616E67 65732062				
00542A	79207365 6E64696E				
005432	67206120 434F4E54				
00543A	524F4C5E 44206E6F				
005442	77206F72 20455343				
00544A	20746F20 41424F52				
005452	542E0D				
005455	416E7920 6F746865	C	7	cngdowverify1	db "Any other key prompts for day again", CR.asc
00545D	72206B65 79207072				
005465	6F6D7074 7320666F				
00546D	72206461 79206167				
005475	61696E0D				
005479	50726573 73204553	C	8	cngdowverify2	db "Press ESC to ABORT.", CR.asc
005481	4320746F 2041424F				
005489	52542E0D				
00548D	03	C	9	db	ETX.asc
		C	10		
00548E	43757272 656E746C	C	11	dowprompt	db "Currently, day of week is set to ", CR.asc
005496	792C2064 6179206F				
00549E	66207765 656B2069				
0054A6	73207365 7420746F				
0054AE	20202020 20202020				
0054B6	20202020 200D				
0054BC	03	C	12	db	ETX.asc
	000054AF	C	13	dowbuff	equ \$-14
		C	14		
0054BD	00	C	15	savcntr	db 0
0054BE	0A0A4368 616E6765	C	16	downew	db LF.asc, LF.asc, "Change day of week to: ", ETX.asc
0054C6	20646179 206F6620				
0054CE	7765656B 20746F3A				
0054D6	2003				

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\screens\ipl_rtc_screens.s
0054D8	202020	C	17	editbuffer	db 20h,20h,20h
0054DB	0A0D03	C	18		db LF.asc,CR.asc,ETX.asc
		C	19		
0054DE	20202020 20202020	C	20	screenbuf	db " eZ80 BIOS clock / calander settings"
0054E6	20202020 20655A38				
0054EE	30204249 4F532063				
0054F6	6C6F636B 202F2063				
0054FE	616C616E 64657220				
005506	73657474 696E6773				
00550E	0D0A	C	21		db CR.asc,LF.asc
005510	20202020 20202020	C	22		db " "
005518	20202020 20202020				
005520	2020				
005522	20202020 20202020	C	23	DOWBUF	db " "
00552A	20				
00552B	20	C	24		db 20h ; Space.
00552C	20202020 20202020	C	25	DATEBUF	db " " ; Place holder of 8 bytes.
005534	20	C	26		DB 20h ; Space.
005535	20202020 20202020	C	27	TIMEBUF	db " " ; Place holder of 8 bytes.
00553D	0D0A	C	28		db CR.asc,LF.asc
00553F	0D	C	29		db CR.asc
005540	0A	C	30		db LF.asc
005541	312E2043 68616E67	C	31		db "1. Change DATE.",CR.asc
005549	65204441 54452E0D				
005551	322E2043 68616E67	C	32		db "2. Change DAY of WEEK.",CR.asc
005559	65204441 59206F66				
005561	20574545 4B2E0D				
005568	332E2043 68616E67	C	33		db "3. Change TIME.",CR.asc
005570	65205449 4D452E0D				
005578	0A0A	C	34		db LF.asc,LF.asc
00557A	4974656D 20746F20	C	35		db "Item to change or ESC aborts. ",ETX.asc
005582	6368616E 6765206F				
00558A	72204553 43206162				
005592	6F727473 2E2003				
		C	36		
005599	20202020 20202020	C	37	edtdatscnd	db " >- Change BIOS calander DATE -<",CR.asc,LF.asc
0055A1	20202020 203E2D20				
0055A9	4368616E 67652042				
0055B1	494F5320 63616C61				
0055B9	6E646572 20444154				
0055C1	45202D3C 0D0A				
0055C7	436C6F63 6B2F4361	C	38		db "Clock/Calander chip date set to: ",CR.asc,LF.asc,ETX.asc
0055CF	6C616E64 65722063				
0055D7	68697020 64617465				
0055DF	20736574 20746F3A				
0055E7	20202020 20202020				
0055EF	20202020 0D0A03				
	000055E8	C	39	cdat	equ \$-14
0055F6	0D0A4368 616E6765	C	40	cngcomplete	db CR.asc,LF.asc,"Changes complete, press ENTER to continue.",ETX.asc
0055FE	7320636F 6D706C65				
005606	74652C20 70726573				
00560E	7320454E 54455220				
005616	746F2063 6F6E7469				
00561E	6E75652E 03				
		C	41		
		C	42		
005623	0320	C	43	inbuff	db 3,32
005625	496E7075 74204D4F	C	44	gtmon	db "Input MONTH (1-12): ",ETX.asc
00562D	4E544820 28312D31				
005635	32293A20 03				

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\screens\ipl_rtc_screens.s
00563A	496E7075 74204441	C	45	gtday	db "Input DAY of month (1-31): ",ETX.asc
005642	59206F66 206D6F6E				
00564A	74682028 312D3331				
005652	293A2003				
005656	496E7075 74204345	C	46	gtcen	db "Input CENTURY (00-99): ",ETX.asc
00565E	4E545552 59202830				
005666	302D3939 293A2003				
00566E	496E7075 74205945	C	47	gtyear	db "Input YEAR (00-99): ",ETX.asc
005676	41522028 30302D39				
00567E	39293A20 03				
005683	444F573D	C	48	doweq	db "DOW="
		C	49		
		C	50		; Following bytes are written to RTC when updateing date/time etc.
		C	51		
005687	00	C	52	savyr	db 0 ; When editing this byte will hold in binary YEAR.
005688	00	C	53	savmon	db 0 ; When editing this byte will hold in binary MONTH.
005689	00	C	54	savday	db 0 ; When editing this byte will hold in binary DAY.
00568A	00	C	55	savcen	db 0 ; When editing this byte will hold in binary CENTURY.
00568B	00	C	56	savdow	db 0 ; When editing this byte will hold in binary DAY OF WEEK.
00568C	00	C	57	savhrs	db 0 ; When editing this byte will hold in binary HOURS.
00568D	00	C	58	savmin	db 0 ; When editing this byte will hold in binary MINUTES.
00568E	00	C	59	savsec	db 0 ; When editing this byte will hold in binary SECONDS.
		C	60		
		C	61		
	0000568F	C	62	DOWTABLE equ	\$
00568F	9D56	C	63		dw dow1st
005691	A556	C	64		dw dow2nd
005693	AD56	C	65		dw dow3rd
005695	B656	C	66		dw dow4th
005697	C156	C	67		dw dow5th
005699	CB56	C	68		dw dow6th
00569B	D356	C	69		dw dow7th
		C	70		
00569D	0600	C	71	dow1st	dw 6
00569F	53756E64 6179	C	72		db "Sunday"
0056A5	0600	C	73	dow2nd	dw 6
0056A7	4D6F6E64 6179	C	74		db "Monday"
0056AD	0700	C	75	dow3rd	dw 7
0056AF	54756573 646179	C	76		db "Tuesday"
0056B6	0900	C	77	dow4th	dw 9
0056B8	5765646E 65736461	C	78		db "Wednesday"
0056C0	79				
0056C1	0800	C	79	dow5th	dw 8
0056C3	54687572 73646179	C	80		db "Thursday"
0056CB	0600	C	81	dow6th	dw 6
0056CD	46726964 6179	C	82		db "Friday"
0056D3	0800	C	83	dow7th	dw 8
0056D5	53617475 72646179	C	84		db "Saturday"
		C	85		
0056DD	0D0D	C	86	cngmsg	db CR.asc,CR.asc
0056DF	4D6F6E74 683D	C	87		db "Month="
0056E5	20202020 200D	C	88	cngmo	db " ",CR.asc
0056EB	4461793D	C	89		db "Day="
0056EF	20202020 200D	C	90	cngdy	db " ",CR.asc
0056F5	43656E74 7572793D	C	91		db "Century="
0056FD	20202020 200D	C	92	cngcn	db " ",CR.asc
005703	59656172 3D	C	93		db "Year="
005708	20202020 200D	C	94	cngyr	db " ",CR.asc
00570E	596F7520 68617665	C	95		db "You have elected to change calander.",CR.asc
005716	20656C65 63746564				

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\screens\ipl_rtc_screens.s
00571E	20746F20 6368616E				
005726	67652063 616C616E				
00572E	6465722E 0D				
005733	0D	C	96	db	CR.asc
005734	45534320 77696C6C	C	97	db	"ESC will return to previous menu, losing changes.",CR.asc
00573C	20726574 75726E20				
005744	746F2070 72657669				
00574C	6F757320 6D656E75				
005754	2C206C6F 73696E67				
00575C	20636861 6E676573				
005764	2E0D				
005766	0D	C	98	db	CR.asc
005767	0D	C	99	db	CR.asc
005768	53656E64 696E6720	C	100	db	"Sending ^D from your terminal (CONTROL D), verifys making changes.",CR.asc
005770	5E442066 726F6D20				
005778	796F7572 20746572				
005780	6D696E61 6C202843				
005788	4F4E5452 4F4C2044				
005790	292C2076 65726966				
005798	7973206D 616B696E				
0057A0	67206368 616E6765				
0057A8	732E0D				
0057AB	0A	C	101	db	LF.asc
0057AC	56657269 6679696E	C	102	db	"Verifying changes to take place will:",CR.asc
0057B4	67206368 616E6765				
0057BC	7320746F 2074616B				
0057C4	6520706C 61636520				
0057CC	77696C6C 3A0D				
0057D2	0A	C	103	db	LF.asc
0057D3	3E205374 6F702061	C	104	db	"> Stop and unlock clock.",CR.asc
0057DB	6E642075 6E6C6F63				
0057E3	6B20636C 6F636B2E				
0057EB	0D				
0057EC	3E20436C 6F636B20	C	105	db	"> Clock will be updated to new values.",CR.asc
0057F4	77696C6C 20626520				
0057FC	75706461 74656420				
005804	746F206E 65772076				
00580C	616C7565 732E0D				
005813	3E20436C 6F636B20	C	106	db	"> Clock will lock.",CR.asc
00581B	77696C6C 206C6F63				
005823	6B2E0D				
005826	3E20436C 6F636B20	C	107	db	"> Clock will start counting from new setting.",CR.asc
00582E	77696C6C 20737461				
005836	72742063 6F756E74				
00583E	696E6720 66726F6D				
005846	206E6577 20736574				
00584E	74696E67 2E0D				
005854	0A	C	108	db	LF.asc
005855	0A	C	109	db	LF.asc
005856	205E4420 746F2076	C	110	db	" ^D to verify changes or ESC to discard."
00585E	65726966 79206368				
005866	616E6765 73206F72				
00586E	20455343 20746F20				
005876	64697363 6172642E				
00587E	03	C	111	db	ETX.asc
		C	112		
00587F	456E7465 7220686F	C	113	cnghrsprompt	db "Enter hours (0-23):",ETX.asc
005887	75727320 28302D32				
00588F	33293A03				
005893	456E7465 72206D69	C	114	cngminprompt	db "Enter minutes (0-59):",ETX.asc

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\screens\ipl_rtc_screens.s
00589B	6E757465 73202830				
0058A3	2D353929 3A03				
0058A9	4368616E 67652074	C	115	cngtimscrn	db "Change time.",CR.asc
0058B1	696D652E 0D				
0058B6	0D	C	116		db CR.asc
0058B7	456E7465 72206120	C	117		db "Enter a time 1 minute ahead of current time but do not confirm.",CR.asc
0058BF	74696D65 2031206D				
0058C7	696E7574 65206168				
0058CF	65616420 6F662063				
0058D7	75727265 6E742074				
0058DF	696D6520 62757420				
0058E7	646F206E 6F742063				
0058EF	6F6E6669 726D2E0D				
0058F7	0D	C	118		db CR.asc
0058F8	456E7465 72206D69	C	119		db "Enter minutes & hours in military. You will be prompted to confirm clock update.",CR.a
005900	6E757465 73202620				
005908	686F7572 7320696E				
005910	206D696C 69746172				
005918	792E2059 6F752077				
005920	696C6C20 62652070				
005928	726F6D70 74656420				
005930	746F2063 6F6E6669				
005938	726D2063 6C6F636B				
005940	20757064 6174652E				
005948	0D				
005949	0D	C	120		db CR.asc
00594A	49662079 6F752077	C	121		db "If you wish to save changes do a ^D @ stroke real time matches = entered time.",CR.asc
005952	69736820 746F2073				
00595A	61766520 6368616E				
005962	67657320 646F2061				
00596A	205E4420 40207374				
005972	726F6B65 20726561				
00597A	6C207469 6D65206D				
005982	61746368 6573203D				
00598A	20656E74 65726564				
005992	2074696D 652E0D				
005999	0D	C	122		db CR.asc
00599A	4368616E 67657320	C	123		db "Changes will take effect within milliseconds of ^D.",CR.asc
0059A2	77696C6C 2074616B				
0059AA	65206566 66656374				
0059B2	20776974 68696E20				
0059BA	6D696C6C 69736563				
0059C2	6F6E6473 206F6620				
0059CA	5E442E0D				
0059CE	0A	C	124		db LF.asc
0059CF	0A	C	125		db LF.asc
0059D0	03	C	126		db ETX.asc
		C	127		
0059D1	4E657720 686F7572	C	128	cnghrsnewdb	"New hours: ",CR.asc
0059D9	733A2020 20202020				
0059E1	0D				
	000059DC	C	129	cnghr	equ \$-6
0059E2	4E657720 6D696E75	C	130	cngminnewdb	"New minutes: ",LF.asc,CR.asc
0059EA	7465733A 20202020				
0059F2	20200A0D				
	000059EF	C	131	cngmin	equ \$-7
0059F6	5E442074 6F207570	C	132		db "^D to update clock or ESC to discard changes.",CR.asc
0059FE	64617465 20636C6F				
005A06	636B206F 72204553				
005A0E	4320746F 20646973				



\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\screens\ipl_rtc_screens.s
005A16	63617264 20636861				
005A1E	6E676573 2E0D				
005A24	0D	C	133	CR\$	db CR.asc
005A25	03	C	134		db ETX.asc
		C	0		include "ipl_net_ping.s" ; Wiggle i/o line and test if server there.
		C	1		; This section of code tests if server available.
		C	2		
		C	3		; After sending "@ping<CR>" this code tests for reply "@pong<CR>"
		C	4		
		C	5		; Only tests if server available, connected & responding.
		C	6		
005A26		C	7	ping_server	GOSUB scrolloff
		C+	7		scope
005A26	CD ED 6E	C+	7		CALL scrolloff
		C+	7		endmac GOSUB
		C	8		
		C	9	PUMP	DEVICE.console,ping_screen\$ ; Tell operator this is ping program.
		C+	9	scope	; A tribute to Dave Shriner - Display Data Corp.
005A29	21 4B 5B	C+	9	LD	HL,ping_screen\$ ; Pump from this starting point until ETX.asc is found.
		C+	9	if	DEVICE.console=0
005A2C	7E	C+	9	__3560:	LD A,(HL) ; Get byte to send.
005A2D	23	C+	9	INC	HL ; pointer=pointer+1
		C+	9	; CP	EOT.asc ; End Of Transmission (block)?
		C+	9	; jr	Z,\$0a ; Return if finished.
005A2E	FE03	C+	9	CP	ETX.asc ; End Of Text?
005A30	28 16	C+	9	jr	Z,__3560a ; Return if finished.
005A32	FE0D	C+	9	CP	CR.asc ; Was it a CR?
005A34	CC 3E 5A	C+	9	call	z,__3561 ; 1st add a LF to stream.
005A37	4F	C+	9	LD	C,A ; Output byte in C.
005A38	AF	C+	9	XOR	A ; Set Z flag for put.
005A39	CD D9 10	C+	9	CALL	U0DVR ; Out the UART we go.
005A3C	18 EE	C+	9	JR	__3560
005A3E	3E0A	C+	9	__3561:	ld a,LF.asc ; Get a LF.
005A40	4F	C+	9	LD	C,A ; Output byte in C.
005A41	AF	C+	9	XOR	A ; Set Z flag for put
005A42	CD D9 10	C+	9	CALL	U0DVR ; Out the UART we go.
005A45	3E0D	C+	9	ld	a,CR.asc ; Now put a CR back in stream.
005A47	C9	C+	9	ret	; Return and print CR.
005A48		C+	9	__3560a	endif
		C+	9	if	DEVICE.console=1
		C+	9	__3561:	LD A,(HL) ; Get byte to send.
		C+	9	INC	HL ; pointer=pointer+1
		C+	9	; CP	EOT.asc ; End Of Transmission (block)?
		C+	9	; jr	Z,\$1a ; Return if finished.
		C+	9	CP	ETX.asc ; End Of Text?
		C+	9	jr	Z,__3561a ; Return if finished.
		C+	9	LD	C,A ; Output byte to C.
		C+	9	XOR	A ; Set Z flag for put.
		C+	9	CALL	U1DVR ; Out the UART we go.
		C+	9	JR	__3561
		C+	9	__3561a	equ \$
		C+	9	endif	
		C+	9	ENDMAC	PUMP
		C	10	PUMP	DEVICE.console,ping_da_ping\$ ; Report pinging server.
		C+	10	scope	; A tribute to Dave Shriner - Display Data Corp.
005A48	21 66 5B	C+	10	LD	HL,ping_da_ping\$ ; Pump from this starting point until ETX.asc is found.
		C+	10	if	DEVICE.console=0
005A4B	7E	C+	10	__3570:	LD A,(HL) ; Get byte to send.
005A4C	23	C+	10	INC	HL ; pointer=pointer+1
		C+	10	; CP	EOT.asc ; End Of Transmission (block)?

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl_net_ping.s
		C+	10	; jr	Z,\$0a ; Return if finished.
005A4D	FE03	C+	10	CP	ETX.asc ; End Of Text?
005A4F	28 16	C+	10	jr	Z,__3570a ; Return if finished.
005A51	FE0D	C+	10	CP	CR.asc ; Was it a CR?
005A53	CC 5D 5A	C+	10	call	z,__3571 ; 1st add a LF to stream.
005A56	4F	C+	10	LD	C,A ; Output byte in C.
005A57	AF	C+	10	XOR	A ; Set Z flag for put.
005A58	CD D9 10	C+	10	CALL	U0DVR ; Out the UART we go.
005A5B	18 EE	C+	10	JR	__3570
005A5D	3E0A	C+	10	__3571:	ld a,LF.asc ; Get a LF.
005A5F	4F	C+	10	LD	C,A ; Output byte in C.
005A60	AF	C+	10	XOR	A ; Set Z flag for put
005A61	CD D9 10	C+	10	CALL	U0DVR ; Out the UART we go.
005A64	3E0D	C+	10	ld	a,CR.asc ; Now put a CR back in stream.
005A66	C9	C+	10	ret	; Return and print CR.
005A67		C+	10	__3570a	endif
		C+	10	if	DEVICE.console=1
		C+	10	__3571:	LD A,(HL) ; Get byte to send.
		C+	10	INC	HL ; pointer=pointer+1
		C+	10	; CP	EOT.asc ; End Of Transmission (block)?
		C+	10	; jr	Z,\$1a ; Return if finished.
		C+	10	CP	ETX.asc ; End Of Text?
		C+	10	jr	Z,__3571a ; Return if finished.
		C+	10	LD	C,A ; Output byte to C.
		C+	10	XOR	A ; Set Z flag for put.
		C+	10	CALL	U1DVR ; Out the UART we go.
		C+	10	JR	__3571
		C+	10	__3571a	equ \$
		C+	10	endif	
		C+	10	ENDMAC	PUMP
		C	11		
		C	12	; UART1.dtr.set	; Turn UART 1 DTR ON.
		C	13	; UART1.rts.set	
		C	14	PUMP	DEVICE.serialnet,ping_string\$ ; Send ping string to server @ping LF.asc.
		C+	14	scope	; A tribute to Dave Shriner - Display Data Corp.
005A67	21 24 5B	C+	14	LD	HL,ping_string\$ ; Pump from this starting point until ETX.asc is found.
		C+	14	if	DEVICE.serialnet=0
		C+	14	__3580:	LD A,(HL) ; Get byte to send.
		C+	14	INC	HL ; pointer=pointer+1
		C+	14	; CP	EOT.asc ; End Of Transmission (block)?
		C+	14	; jr	Z,\$0a ; Return if finished.
		C+	14	CP	ETX.asc ; End Of Text?
		C+	14	jr	Z,__3580a ; Return if finished.
		C+	14	CP	CR.asc ; Was it a CR?
		C+	14	call	z,__3581 ; 1st add a LF to stream.
		C+	14	LD	C,A ; Output byte in C.
		C+	14	XOR	A ; Set Z flag for put.
		C+	14	CALL	U0DVR ; Out the UART we go.
		C+	14	JR	__3580
		C+	14	__3581:	ld a,LF.asc ; Get a LF.
		C+	14	LD	C,A ; Output byte in C.
		C+	14	XOR	A ; Set Z flag for put
		C+	14	CALL	U0DVR ; Out the UART we go.
		C+	14	ld	a,CR.asc ; Now put a CR back in stream.
		C+	14	ret	; Return and print CR.
		C+	14	__3580a	endif
		C+	14	if	DEVICE.serialnet=1
005A6A	7E	C+	14	__3581:	LD A,(HL) ; Get byte to send.
005A6B	23	C+	14	INC	HL ; pointer=pointer+1
		C+	14	; CP	EOT.asc ; End Of Transmission (block)?

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl_net_ping.s
		C+	14	; jr Z,\$1a	; Return if finished.
005A6C	FE03	C+	14	CP ETX.asc	; End Of Text?
005A6E	28 07	C+	14	jr Z,__3581a	; Return if finished.
005A70	4F	C+	14	LD C,A	; Output byte to C.
005A71	AF	C+	14	XOR A	; Set Z flag for put.
005A72	CD F5 11	C+	14	CALL U1DVR	; Out the UART we go.
005A75	18 F3	C+	14	JR __3581	
	00005A77	C+	14	__3581a equ \$	
		C+	14	endif	
		C+	14	ENDMAC PUMP	
		C	15	; UART1.rts.reset	
		C	16		
		C	17	PUMP DEVICE.console,ping_sent\$	; Report starting ping to server.
		C+	17	scope	; A tribute to Dave Shriner - Display Data Corp.
005A77	21 2B 5B	C+	17	LD HL,ping_sent\$	; Pump from this starting point until ETX.asc is found.
		C+	17	if DEVICE.console=0	
005A7A	7E	C+	17	__3590: LD A,(HL)	; Get byte to send.
005A7B	23	C+	17	INC HL	; pointer=pointer+1
		C+	17	; CP EOT.asc	; End Of Transmission (block)?
		C+	17	; jr Z,\$0a	; Return if finished.
005A7C	FE03	C+	17	CP ETX.asc	; End Of Text?
005A7E	28 16	C+	17	jr Z,__3590a	; Return if finished.
005A80	FE0D	C+	17	CP CR.asc	; Was it a CR?
005A82	CC 8C 5A	C+	17	call z,__3591	; 1st add a LF to stream.
005A85	4F	C+	17	LD C,A	; Output byte in C.
005A86	AF	C+	17	XOR A	; Set Z flag for put.
005A87	CD D9 10	C+	17	CALL U0DVR	; Out the UART we go.
005A8A	18 EE	C+	17	JR __3590	
005A8C	3E0A	C+	17	__3591: ld a,LF.asc	; Get a LF.
005A8E	4F	C+	17	LD C,A	; Output byte in C.
005A8F	AF	C+	17	XOR A	; Set Z flag for put
005A90	CD D9 10	C+	17	CALL U0DVR	; Out the UART we go.
005A93	3E0D	C+	17	ld a,CR.asc	; Now put a CR back in stream.
005A95	C9	C+	17	ret	; Return and print CR.
005A96		C+	17	__3590a endif	
		C+	17	if DEVICE.console=1	
		C+	17	__3591: LD A,(HL)	; Get byte to send.
		C+	17	INC HL	; pointer=pointer+1
		C+	17	; CP EOT.asc	; End Of Transmission (block)?
		C+	17	; jr Z,\$1a	; Return if finished.
		C+	17	CP ETX.asc	; End Of Text?
		C+	17	jr Z,__3591a	; Return if finished.
		C+	17	LD C,A	; Output byte to C.
		C+	17	XOR A	; Set Z flag for put.
		C+	17	CALL U1DVR	; Out the UART we go.
		C+	17	JR __3591	
		C+	17	__3591a equ \$	
		C+	17	endif	
		C+	17	ENDMAC PUMP	
		C	18		
005A96		C	19	test_reply UART1.get.byte no_pong	; Get byte from UART1. On timeout or error goto no_pong
		C+	19	scope	
005A96	3E01	C+	19	ld a,1	
005A98	B7	C+	19	or a	; On timeout or error goto vector.
005A99	37	C+	19	scf	
		C+	19	GOSUB U1DVR	; C flag set & NZ set means PUT.
		C+	19	scope	
005A9A	CD F5 11	C+	19	CALL U1DVR	
		C+	19	endmac GOSUB	
		C+	19	IF_NE_GOSUB no_pong	; If nz we had a timeout or other error.

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl_net_ping.s
		C+	19		scope
005A9D	C4 F6 5A	C+	19		call nz,no_pong
		C+	19		endmac IF_NE_GOSUB
		C+	19		endmac UART1.get.byte
005AA0	3E40	C	20	ld a,'@'	; Is 1st char @ control code?
005AA2	B9	C	21	cp c	; Compare to C, C has char from GET_BYTE_UART1.
005AA3	20 F1	C	22	JR NZ,test_reply	; Ignore if not control char & continue getting chars.
		C	23		
		C	24	; UART1.rts.set	
		C	25	UART1.get.bytes 5,pong_buff+1,no_pong	; get 5 bytes into pong_buff, if timeout or error go
		C+	25	scope	
005AA5	0605	C+	25	ld b,5	; Input 5 bytes.
005AA7	21 A9 5B	C+	25	ld hl,pong_buff+1	; Get address of buffer.
005AAA		C+	25	__3631 UART1.get.byte no_pong	; Get byte from UART1. If timeout goto no_pong.
		C+	25	scope	
005AAA	3E01	C+	25	ld a,1	
005AAC	B7	C+	25	or a	; On timeout or error goto vector.
005AAD	37	C+	25	scf	
		C+	25	GOSUB U1DVR	; C flag set & NZ set means PUT.
		C+	25	scope	
005AAE	CD F5 11	C+	25	CALL U1DVR	
		C+	25	endmac GOSUB	
		C+	25	IF_NE_GOSUB no_pong	; If nz we had a timeout or other error.
		C+	25	scope	
005AB1	C4 F6 5A	C+	25	call nz,no_pong	
		C+	25	endmac IF_NE_GOSUB	
		C+	25	endmac UART1.get.byte	
005AB4	71	C+	25	ld (hl),c	; Store char.
005AB5	23	C+	25	inc hl	
		C+	25	LOOP __3631	; Loop for count bytes.
		C+	25	scope	
005AB6	10 F2	C+	25	djnz __3631	
		C+	25	endmac LOOP	
		C+	25	endmac UART1.get.bytes	
		C	26	; UART1.rts.reset	
		C	27		
		C	28	CMP.bytes 6,pong\$,pong_buff,no_pong	; Compare 'pong' to pong_buffer. If <> goto no_pong.
		C+	28	scope	
005AB8	0606	C+	28	ld b,6	; Setup to compare length bytes.
005ABA	21 A2 5B	C+	28	ld hl,pong\$	; Get address of first buffer.
005ABD	11 A8 5B	C+	28	ld de,pong_buff	; Get address of second buffer.
005AC0	1A	C+	28	__3681 ld a,(de)	; Get a byte from rbuffer.
005AC1	BE	C+	28	cp (hl)	; Compare to other buffer.
		C+	28	IF_NE_GOTO no_pong	; If compare fails match.
		C+	28	scope	
005AC2	C2 F6 5A	C+	28	jp nz,no_pong	
		C+	28	endmac IF_NE_GOTO	
005AC5	23	C+	28	inc hl	; Advance pointers for
005AC6	13	C+	28	inc de	; both buffers.
		C+	28	LOOP __3681	; Loop until B=0
		C+	28	scope	
005AC7	10 F7	C+	28	djnz __3681	
		C+	28	endmac LOOP	
		C+	28	endmac CMP.bytes	
		C	29	MVC server_avail,server_stat,12	; Change server to show available.
		C+	29	scope	
005AC9	21 AF 5B	C+	29	ld hl,server_avail	
005ACC	11 17 72	C+	29	ld de,server_stat	
005ACF	010C00	C+	29	ld bc,12	
005AD2	EDB0	C+	29	LDIR	

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source
		C+	29	ENDMAC MVC
		C	30	PUMP DEVICE.console,reply_rx\$ ; Report received @pong from server.
		C+	30	scope ; A tribute to Dave Shriner - Display Data Corp.
005AD4	21 7A 5B	C+	30	LD HL,reply_rx\$ ; Pump from this starting point until ETX.asc is found.
		C+	30	if DEVICE.console=0
005AD7	7E	C+	30	__3720: LD A,(HL) ; Get byte to send.
005AD8	23	C+	30	INC HL ; pointer=pointer+1
		C+	30	; CP EOT.asc ; End Of Transmission (block)?
		C+	30	; jr Z,\$0a ; Return if finished.
005AD9	FE03	C+	30	CP ETX.asc ; End Of Text?
005ADB	28 16	C+	30	jr Z,__3720a ; Return if finished.
005ADD	FE0D	C+	30	CP CR.asc ; Was it a CR?
005ADF	CC E9 5A	C+	30	call z,__3721 ; 1st add a LF to stream.
005AE2	4F	C+	30	LD C,A ; Output byte in C.
005AE3	AF	C+	30	XOR A ; Set Z flag for put.
005AE4	CD D9 10	C+	30	CALL U0DVR ; Out the UART we go.
005AE7	18 EE	C+	30	JR __3720
005AE9	3E0A	C+	30	__3721: ld a,LF.asc ; Get a LF.
005AEB	4F	C+	30	LD C,A ; Output byte in C.
005AEC	AF	C+	30	XOR A ; Set Z flag for put
005AED	CD D9 10	C+	30	CALL U0DVR ; Out the UART we go.
005AF0	3E0D	C+	30	ld a,CR.asc ; Now put a CR back in stream.
005AF2	C9	C+	30	ret ; Return and print CR.
005AF3		C+	30	__3720a endif
		C+	30	if DEVICE.console=1
		C+	30	__3721: LD A,(HL) ; Get byte to send.
		C+	30	INC HL ; pointer=pointer+1
		C+	30	; CP EOT.asc ; End Of Transmission (block)?
		C+	30	; jr Z,\$1a ; Return if finished.
		C+	30	CP ETX.asc ; End Of Text?
		C+	30	jr Z,__3721a ; Return if finished.
		C+	30	LD C,A ; Output byte to C.
		C+	30	XOR A ; Set Z flag for put.
		C+	30	CALL U1DVR ; Out the UART we go.
		C+	30	JR __3721
		C+	30	__3721a equ \$
		C+	30	endif
		C+	30	ENDMAC PUMP
		C	31	GOTO ping_exit
		C+	31	scope
005AF3	C3 21 5B	C+	31	jp ping_exit
		C+	31	endmac GOTO
		C	32	
005AF6	E1	C	33	no_pong pop hl ; Discard error trap address.
		C	34	MVC server_not,server_stat,12 ; Change server to show available.
		C+	34	scope
005AF7	21 12 62	C+	34	ld hl,server_not
005AFA	11 17 72	C+	34	ld de,server_stat
005AFD	010C00	C+	34	ld bc,12
005B00	EDB0	C+	34	LDIR
		C+	34	ENDMAC MVC
		C	35	PUMP DEVICE.console,no_pong_rec\$ ; Tell operator no reply recieved.
		C+	35	scope ; A tribute to Dave Shriner - Display Data Corp.
005B02	21 8D 5B	C+	35	LD HL,no_pong_rec\$ ; Pump from this starting point until ETX.asc is found.
		C+	35	if DEVICE.console=0
005B05	7E	C+	35	__3750: LD A,(HL) ; Get byte to send.
005B06	23	C+	35	INC HL ; pointer=pointer+1
		C+	35	; CP EOT.asc ; End Of Transmission (block)?
		C+	35	; jr Z,\$0a ; Return if finished.
005B07	FE03	C+	35	CP ETX.asc ; End Of Text?

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl_net_ping.s
005B09	28 16	C+	35	jr	Z, __3750a ; Return if finished.
005B0B	FE0D	C+	35	CP	CR.asc ; Was it a CR?
005B0D	CC 17 5B	C+	35	call	z, __3751 ; 1st add a LF to stream.
005B10	4F	C+	35	LD	C, A ; Output byte in C.
005B11	AF	C+	35	XOR	A ; Set Z flag for put.
005B12	CD D9 10	C+	35	CALL	U0DVR ; Out the UART we go.
005B15	18 EE	C+	35	JR	__3750
005B17	3E0A	C+	35	__3751:	ld a, LF.asc ; Get a LF.
005B19	4F	C+	35	LD	C, A ; Output byte in C.
005B1A	AF	C+	35	XOR	A ; Set Z flag for put
005B1B	CD D9 10	C+	35	CALL	U0DVR ; Out the UART we go.
005B1E	3E0D	C+	35	ld	a, CR.asc ; Now put a CR back in stream.
005B20	C9	C+	35	ret	; Return and print CR.
005B21		C+	35	__3750a	endif
		C+	35		if DEVICE.console=1
		C+	35	__3751:	LD A, (HL) ; Get byte to send.
		C+	35	INC	HL ; pointer=pointer+1
		C+	35	;	CP EOT.asc ; End Of Transmission (block)?
		C+	35	;	jr Z, \$1a ; Return if finished.
		C+	35		CP ETX.asc ; End Of Text?
		C+	35		jr Z, __3751a ; Return if finished.
		C+	35		LD C, A ; Output byte to C.
		C+	35		XOR A ; Set Z flag for put.
		C+	35		CALL U1DVR ; Out the UART we go.
		C+	35		JR __3751
		C+	35	__3751a	equ \$
		C+	35		endif
		C+	35		ENDMAC PUMP
		C	36		
005B21	3EFF	C	37	ping_exit	ld a, 0ffh
		C	38		RETURN
		C+	38		scope
005B23	C9	C+	38	ret	; Return from call.
		C+	38	endmac	RETURN
		C	39		
		C	40		; Strings & data used by ping command.
		C	41		
005B24	4070696E 670A03	C	42	ping_string\$	db "@ping", LF.asc, ETX.asc
		C	43		
005B2B	2050696E 67207365	C	44	ping_sent\$	db " Ping sent, waiting for reply.", CR.asc, ETX.asc
005B33	6E742C20 77616974				
005B3B	696E6720 666F7220				
005B43	7265706C 792E0D03				
		C	45		
005B4B	0D202D2D 2D2D3E20	C	46	ping_screen\$	db CR.asc, " ----> Ping server <----", CR.asc, ETX.asc
005B53	50696E67 20736572				
005B5B	76657220 3C2D2D2D				
005B63	2D0D03				
		C	47		
005B66	2050696E 67696E67	C	48	ping_da_ping\$	db " Pinging server.", CR.asc, ETX.asc
005B6E	20736572 7665722E				
005B76	0D03				
		C	49		
005B78	0D03	C	50	ping_end_line\$	db CR.asc, ETX.asc
		C	51		
005B7A	0D205265 706C7920	C	52	reply_rx\$db	CR.asc, " Reply received.", CR.asc, ETX.asc
005B82	72656365 69766564				
005B8A	2E0D03				
		C	53		
005B8D	0D204E6F 20726570	C	54	no_pong_rec\$	db CR.asc, " No reply to ping.", CR.asc, ETX.asc

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl_net_ping.s
005B95	6C792074 6F207069				
005B9D	6E672E0D 03				
		C	55		
005BA2	40706F6E 670A	C	56	pong\$	db "@pong",LF.asc
		C	57		
005BA8	40313233 343520	C	58	pong_buffdb	"@", "12345 "
		C	59		
005BAF	41766169 6C61626C	C	60	server_avail	db "Available "
005BB7	65202020				
		C	0		include "ipl_net_echo.s" ; Send 256 byte to server, wait for server
		C	1		; This section of code sends a 256 byte pack to server.
		C	2		; After sending packet this app waits for reply & compares packet received to packet sent.
		C	3		; Program then reports if compare was equal or not.
		C	4		
		C	5		; This is only used to test if server connected, running and responding.
		C	6		
005BBB		C	7	echo_server	GOSUB scrolloff
		C+	7		scope
005BBB	CD ED 6E	C+	7		CALL scrolloff
		C+	7		endmac GOSUB
		C	8		
		C	9		
		C	10	PUMP	DEVICE.console,echo_screen\$ ; Report to operator this program performs @echo.
		C+	10	scope	; A tribute to Dave Shriner - Display Data Corp.
005BBE	21 0B 5D	C+	10	LD	HL,echo_screen\$ ; Pump from this starting point until ETX.asc is found.
		C+	10	if	DEVICE.console=0
005BC1	7E	C+	10	LD	A,(HL) ; Get byte to send.
005BC2	23	C+	10	INC	HL ; pointer=pointer+1
		C+	10	CP	EOT.asc ; End Of Transmission (block)?
		C+	10	jr	Z,\$0a ; Return if finished.
005BC3	FE03	C+	10	CP	ETX.asc ; End Of Text?
005BC5	28 16	C+	10	jr	Z,__3780a ; Return if finished.
005BC7	FE0D	C+	10	CP	CR.asc ; Was it a CR?
005BC9	CC D3 5B	C+	10	call	z,__3781 ; 1st add a LF to stream.
005BCC	4F	C+	10	LD	C,A ; Output byte in C.
005BCD	AF	C+	10	XOR	A ; Set Z flag for put.
005BCE	CD D9 10	C+	10	CALL	U0DVR ; Out the UART we go.
005BD1	18 EE	C+	10	JR	__3780
005BD3	3E0A	C+	10	LD	a,LF.asc ; Get a LF.
005BD5	4F	C+	10	LD	C,A ; Output byte in C.
005BD6	AF	C+	10	XOR	A ; Set Z flag for put
005BD7	CD D9 10	C+	10	CALL	U0DVR ; Out the UART we go.
005BDA	3E0D	C+	10	ld	a,CR.asc ; Now put a CR back in stream.
005BDC	C9	C+	10	ret	; Return and print CR.
005BDD		C+	10	endif	
		C+	10	if	DEVICE.console=1
		C+	10	LD	A,(HL) ; Get byte to send.
		C+	10	INC	HL ; pointer=pointer+1
		C+	10	CP	EOT.asc ; End Of Transmission (block)?
		C+	10	jr	Z,\$1a ; Return if finished.
		C+	10	CP	ETX.asc ; End Of Text?
		C+	10	jr	Z,__3781a ; Return if finished.
		C+	10	LD	C,A ; Output byte to C.
		C+	10	XOR	A ; Set Z flag for put.
		C+	10	CALL	U1DVR ; Out the UART we go.
		C+	10	JR	__3781
		C+	10	equ	\$
		C+	10	endif	
		C+	10	ENDMAC	PUMP
		C	11	PUMP	DEVICE.console,try_echo\$ ; Tell operator sending @echo packet to server.

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl_net_echo.s
		C+	11	scope	; A tribute to Dave Shriner - Display Data Corp.
005BDD	21 3D 5D	C+	11	LD HL,try_echo\$	; Pump from this starting point until ETX.asc is found.
		C+	11	if DEVICE.console=0	
005BE0	7E	C+	11	__3790: LD A,(HL)	; Get byte to send.
005BE1	23	C+	11	INC HL	; pointer=pointer+1
		C+	11	; CP EOT.asc	; End Of Transmission (block)?
		C+	11	; jr Z,\$0a	; Return if finished.
005BE2	FE03	C+	11	CP ETX.asc	; End Of Text?
005BE4	28 16	C+	11	jr __3790a	; Return if finished.
005BE6	FE0D	C+	11	CP CR.asc	; Was it a CR?
005BE8	CC F2 5B	C+	11	call z,__3791	; 1st add a LF to stream.
005BEB	4F	C+	11	LD C,A	; Output byte in C.
005BEC	AF	C+	11	XOR A	; Set Z flag for put.
005BED	CD D9 10	C+	11	CALL U0DVR	; Out the UART we go.
005BF0	18 EE	C+	11	JR __3790	
005BF2	3E0A	C+	11	__3791: ld a,LF.asc	; Get a LF.
005BF4	4F	C+	11	LD C,A	; Output byte in C.
005BF5	AF	C+	11	XOR A	; Set Z flag for put
005BF6	CD D9 10	C+	11	CALL U0DVR	; Out the UART we go.
005BF9	3E0D	C+	11	ld a,CR.asc	; Now put a CR back in stream.
005BFB	C9	C+	11	ret	; Return and print CR.
005BFC		C+	11	__3790a endif	
		C+	11	if DEVICE.console=1	
		C+	11	__3791: LD A,(HL)	; Get byte to send.
		C+	11	INC HL	; pointer=pointer+1
		C+	11	; CP EOT.asc	; End Of Transmission (block)?
		C+	11	; jr Z,\$1a	; Return if finished.
		C+	11	CP ETX.asc	; End Of Text?
		C+	11	jr __3791a	; Return if finished.
		C+	11	LD C,A	; Output byte to C.
		C+	11	XOR A	; Set Z flag for put.
		C+	11	CALL U1DVR	; Out the UART we go.
		C+	11	JR __3791	
		C+	11	__3791a equ \$	
		C+	11	endif	
		C+	11	ENDMAC PUMP	
		C	12		
		C	13	PUMP DEVICE.serialnet,echo_string\$	; Send @echo LF.asc packet.
		C+	13	scope	; A tribute to Dave Shriner - Display Data Corp.
005BFC	21 04 5D	C+	13	LD HL,echo_string\$	; Pump from this starting point until ETX.asc is found.
		C+	13	if DEVICE.serialnet=0	
		C+	13	__3800: LD A,(HL)	; Get byte to send.
		C+	13	INC HL	; pointer=pointer+1
		C+	13	; CP EOT.asc	; End Of Transmission (block)?
		C+	13	; jr Z,\$0a	; Return if finished.
		C+	13	CP ETX.asc	; End Of Text?
		C+	13	jr __3800a	; Return if finished.
		C+	13	CP CR.asc	; Was it a CR?
		C+	13	call z,__3801	; 1st add a LF to stream.
		C+	13	LD C,A	; Output byte in C.
		C+	13	XOR A	; Set Z flag for put.
		C+	13	CALL U0DVR	; Out the UART we go.
		C+	13	JR __3800	
		C+	13	__3801: ld a,LF.asc	; Get a LF.
		C+	13	LD C,A	; Output byte in C.
		C+	13	XOR A	; Set Z flag for put
		C+	13	CALL U0DVR	; Out the UART we go.
		C+	13	ld a,CR.asc	; Now put a CR back in stream.
		C+	13	ret	; Return and print CR.
		C+	13	__3800a endif	



\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl_net_echo.s
		C+	13		if DEVICE.serialnet=1
005BFF	7E	C+	13	__3801:	LD A,(HL) ; Get byte to send.
005C00	23	C+	13		INC HL ; pointer=pointer+1
		C+	13		; CP EOT.asc ; End Of Transmission (block)?
		C+	13		; jr Z,\$1a ; Return if finished.
005C01	FE03	C+	13		CP ETX.asc ; End Of Text?
005C03	28 07	C+	13		jr Z,__3801a ; Return if finished.
005C05	4F	C+	13		LD C,A ; Output byte to C.
005C06	AF	C+	13		XOR A ; Set Z flag for put.
005C07	CD F5 11	C+	13		CALL U1DVR ; Out the UART we go.
005C0A	18 F3	C+	13		JR __3801
	00005C0C	C+	13	__3801a	equ \$
		C+	13		endif
		C+	13		ENDMAC PUMP
		C	14		
		C	15		; UART1.mctl.put 00000001b ; Set DTR active, RTS inactive telling over end were ready.
		C	16		
		C	17		UART1.put.bytes 0,echo_xmit_buff,echo_no_reply ; @echo & 256 byte packet has been sent.
		C+	17		scope
005C0C	0600	C+	17		ld b,0 ; Were sending "count" bytes.
005C0E	21 39 5E	C+	17		ld hl,echo_xmit_buff
005C11	4E	C+	17	__3811	ld c,(hl)
005C12	23	C+	17		inc hl
		C+	17		UART1.put.byte echo_no_reply ; If nz we had a timeout or other error.
		C+	17		scope
005C13	AF	C+	17		xor a ; Set Z flag set means PUT.
		C+	17		GOSUB U1DVR ; Call UART driver.
		C+	17		scope
005C14	CD F5 11	C+	17		CALL U1DVR
		C+	17		endmac GOSUB
		C+	17		IF_NE_GOSUB echo_no_reply ; If error goto error routine.
		C+	17		scope
005C17	C4 E1 5C	C+	17		call nz,echo_no_reply
		C+	17		endmac IF_NE_GOSUB
		C+	17		endmac UART1.put.byte
		C+	17		LOOP __3811
		C+	17		scope
005C1A	10 F5	C+	17		djnz __3811
		C+	17		endmac LOOP
		C+	17		endmac UART1.put.bytes
		C	18		
		C	19		; UART1.mctl.put 00000001b ; Set DTR inactive, RTS inactive, telling server we are rea
		C	20		
		C	21		PUMP DEVICE.console,packet_sent\$ ; Tell operator packet sent.
		C+	21		scope ; A tribute to Dave Shriner - Display Data Corp.
005C1C	21 59 5D	C+	21		LD HL,packet_sent\$ ; Pump from this starting point until ETX.asc is found.
		C+	21		if DEVICE.console=0
005C1F	7E	C+	21	__3860:	LD A,(HL) ; Get byte to send.
005C20	23	C+	21		INC HL ; pointer=pointer+1
		C+	21		; CP EOT.asc ; End Of Transmission (block)?
		C+	21		; jr Z,\$0a ; Return if finished.
005C21	FE03	C+	21		CP ETX.asc ; End Of Text?
005C23	28 16	C+	21		jr Z,__3860a ; Return if finished.
005C25	FE0D	C+	21		CP CR.asc ; Was it a CR?
005C27	CC 31 5C	C+	21		call z,__3861 ; 1st add a LF to stream.
005C2A	4F	C+	21		LD C,A ; Output byte in C.
005C2B	AF	C+	21		XOR A ; Set Z flag for put.
005C2C	CD D9 10	C+	21		CALL U0DVR ; Out the UART we go.
005C2F	18 EE	C+	21		JR __3860
005C31	3E0A	C+	21	__3861:	ld a,LF.asc ; Get a LF.

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl_net_echo.s
005C33	4F	C+	21	LD	C,A ; Output byte in C.
005C34	AF	C+	21	XOR	A ; Set Z flag for put
005C35	CD D9 10	C+	21	CALL	U0DVR ; Out the UART we go.
005C38	3E0D	C+	21	ld	a,CR.asc ; Now put a CR back in stream.
005C3A	C9	C+	21	ret	; Return and print CR.
005C3B		C+	21	__3860a	endif
		C+	21		if DEVICE.console=1
		C+	21	__3861:	LD A,(HL) ; Get byte to send.
		C+	21	INC	HL ; pointer=pointer+1
		C+	21	; CP	EOT.asc ; End Of Transmission (block)?
		C+	21	; jr	Z,\$1a ; Return if finished.
		C+	21	CP	ETX.asc ; End Of Text?
		C+	21	jr	Z,__3861a ; Return if finished.
		C+	21	LD	C,A ; Output byte to C.
		C+	21	XOR	A ; Set Z flag for put.
		C+	21	CALL	U1DVR ; Out the UART we go.
		C+	21	JR	__3861
		C+	21	__3861a	equ \$
		C+	21		endif
		C+	21		ENDMAC PUMP
		C	22		
		C	23		
		C	24		; Server echos back packet we sent. Receive that packet now.
		C	25		
		C	26		
		C	27		
		C	28		UART1.get.bytes 0,echo_rec_buff,echo_no_reply
		C+	28		scope
005C3B	0600	C+	28	ld	b,0 ; Input 5 bytes.
005C3D	21 39 5F	C+	28	ld	hl,echo_rec_buff ; Get address of buffer.
005C40		C+	28	__3871	UART1.get.byte echo_no_reply ; Get byte from UART1. If timeout goto no_pong.
		C+	28		scope
005C40	3E01	C+	28	ld	a,1
005C42	B7	C+	28	or	a ; On timeout or error goto vector.
005C43	37	C+	28	scf	
		C+	28	GOSUB	U1DVR ; C flag set & NZ set means PUT.
		C+	28		scope
005C44	CD F5 11	C+	28	CALL	U1DVR
		C+	28	endmac	GOSUB
		C+	28	IF_NE_GOSUB	echo_no_reply ; If nz we had a timeout or other error.
		C+	28		scope
005C47	C4 E1 5C	C+	28	call	nz,echo_no_reply
		C+	28	endmac	IF_NE_GOSUB
		C+	28	endmac	UART1.get.byte
005C4A	71	C+	28	ld	(hl),c ; Store char.
005C4B	23	C+	28	inc	hl
		C+	28	LOOP	__3871 ; Loop for count bytes.
		C+	28		scope
005C4C	10 F2	C+	28	djnz	__3871
		C+	28	endmac	LOOP
		C+	28	endmac	UART1.get.bytes
		C	29		
		C	30	; ld	b,0
		C	31	; ld	hl,echo_rec_buff
		C	32	;echo_get_byte;UART1.mctl.put	00000011b ; Set DTR active, RTS active tell we are ready.
		C	33		
		C	34	;echo_RXUART1WAIT UART1.tst.rx	echo_RXUART1WAIT ; loop until byte avail.
		C	35		
		C	36	; UART1.mctl.put	00000001b ; Set DTR active, RTS inactive telling other end wait.
		C	37		

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl_net_echo.s
		C	38	; IN0	c,(UART1_RBR) ; GET char waiting.
		C	39	; ld	(hl),c ; Store char.
		C	40	; inc	hl
		C	41	; LOOP	echo_get_byte ;
		C	42		
		C	43	; UART1.mctl.put	00000001b ; Set DTR active, RTS inactive telling server we are ready.
		C	44		
		C	45	; 256 byte packet has been received back from server.	
		C	46		
		C	47	PUMP	DEVICE.console,echo_rx\$ ; Report to operator packet received
		C+	47	scope	; A tribute to Dave Shriner - Display Data Corp.
005C4E	21 6F 5D	C+	47	LD	HL,echo_rx\$ ; Pump from this starting point until ETX.asc is found.
		C+	47	if	DEVICE.console=0
005C51	7E	C+	47	LD	A,(HL) ; Get byte to send.
005C52	23	C+	47	INC	HL ; pointer=pointer+1
		C+	47	; CP	EOT.asc ; End Of Transmission (block)?
		C+	47	; jr	Z,\$0a ; Return if finished.
005C53	FE03	C+	47	CP	ETX.asc ; End Of Text?
005C55	28 16	C+	47	jr	Z,__3920a ; Return if finished.
005C57	FE0D	C+	47	CP	CR.asc ; Was it a CR?
005C59	CC 63 5C	C+	47	call	z,__3921 ; 1st add a LF to stream.
005C5C	4F	C+	47	LD	C,A ; Output byte in C.
005C5D	AF	C+	47	XOR	A ; Set Z flag for put.
005C5E	CD D9 10	C+	47	CALL	U0DVR ; Out the UART we go.
005C61	18 EE	C+	47	JR	__3920
005C63	3E0A	C+	47	LD	a,LF.asc ; Get a LF.
005C65	4F	C+	47	LD	C,A ; Output byte in C.
005C66	AF	C+	47	XOR	A ; Set Z flag for put
005C67	CD D9 10	C+	47	CALL	U0DVR ; Out the UART we go.
005C6A	3E0D	C+	47	ld	a,CR.asc ; Now put a CR back in stream.
005C6C	C9	C+	47	ret	; Return and print CR.
005C6D		C+	47	__3920a	endif
		C+	47	if	DEVICE.console=1
		C+	47	__3921:	LD A,(HL) ; Get byte to send.
		C+	47	INC	HL ; pointer=pointer+1
		C+	47	; CP	EOT.asc ; End Of Transmission (block)?
		C+	47	; jr	Z,\$1a ; Return if finished.
		C+	47	CP	ETX.asc ; End Of Text?
		C+	47	jr	Z,__3921a ; Return if finished.
		C+	47	LD	C,A ; Output byte to C.
		C+	47	XOR	A ; Set Z flag for put.
		C+	47	CALL	U1DVR ; Out the UART we go.
		C+	47	JR	__3921
		C+	47	__3921a	equ \$
		C+	47	endif	
		C+	47	ENDMAC	PUMP
		C	48	PUMP	DEVICE.console,echo_compare\$ ; Tell operator beginning comparing sent packet to
		C+	48	scope	; A tribute to Dave Shriner - Display Data Corp.
005C6D	21 89 5D	C+	48	LD	HL,echo_compare\$ ; Pump from this starting point until ETX.asc is found.
		C+	48	if	DEVICE.console=0
005C70	7E	C+	48	LD	A,(HL) ; Get byte to send.
005C71	23	C+	48	INC	HL ; pointer=pointer+1
		C+	48	; CP	EOT.asc ; End Of Transmission (block)?
		C+	48	; jr	Z,\$0a ; Return if finished.
005C72	FE03	C+	48	CP	ETX.asc ; End Of Text?
005C74	28 16	C+	48	jr	Z,__3930a ; Return if finished.
005C76	FE0D	C+	48	CP	CR.asc ; Was it a CR?
005C78	CC 82 5C	C+	48	call	z,__3931 ; 1st add a LF to stream.
005C7B	4F	C+	48	LD	C,A ; Output byte in C.
005C7C	AF	C+	48	XOR	A ; Set Z flag for put.

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	..\\SYS0_IPL\\ipl_net_echo.s
005C7D	CD D9 10	C+	48	CALL	U0DVR ; Out the UART we go.
005C80	18 EE	C+	48	JR	__3930
005C82	3E0A	C+	48	__3931:	ld a,LF.asc ; Get a LF.
005C84	4F	C+	48	LD	C,A ; Output byte in C.
005C85	AF	C+	48	XOR	A ; Set Z flag for put
005C86	CD D9 10	C+	48	CALL	U0DVR ; Out the UART we go.
005C89	3E0D	C+	48	ld	a,CR.asc ; Now put a CR back in stream.
005C8B	C9	C+	48	ret	; Return and print CR.
005C8C		C+	48	__3930a	endif
		C+	48	if	DEVICE.console=1
		C+	48	__3931:	LD A,(HL) ; Get byte to send.
		C+	48	INC	HL ; pointer=pointer+1
		C+	48	; CP	EOT.asc ; End Of Transmission (block)?
		C+	48	; jr	Z,\$1a ; Return if finished.
		C+	48	CP	ETX.asc ; End Of Text?
		C+	48	jr	Z,__3931a ; Return if finished.
		C+	48	LD	C,A ; Output byte to C.
		C+	48	XOR	A ; Set Z flag for put.
		C+	48	CALL	U1DVR ; Out the UART we go.
		C+	48	JR	__3931
		C+	48	__3931a	equ \$
		C+	48	endif	
		C+	48	ENDMAC	PUMP
		C	49		
		C	50		CMP.bytes 0,echo_xmit_buff,echo_rec_buff,echo_no_match ; CMP 256 byte buffers, if <> then g
		C+	50	scope	
005C8C	0600	C+	50	ld	b,0 ; Setup to compare length bytes.
005C8E	21 39 5E	C+	50	ld	hl,echo_xmit_buff ; Get address of first buffer.
005C91	11 39 5F	C+	50	ld	de,echo_rec_buff ; Get address of second buffer.
005C94	1A	C+	50	__3941	ld a,(de) ; Get a byte from rbuffer.
005C95	BE	C+	50	cp	(hl) ; Compare to other buffer.
		C+	50	IF_NE_GOTO	echo_no_match ; If compare fails match.
		C+	50	scope	
005C96	C2 BF 5C	C+	50	jp	nz,echo_no_match
		C+	50	endmac	IF_NE_GOTO
005C99	23	C+	50	inc	hl ; Advance pointers for
005C9A	13	C+	50	inc	de ; both buffers.
		C+	50	LOOP	__3941 ; Loop until B=0
		C+	50	scope	
005C9B	10 F7	C+	50	djnz	__3941
		C+	50	endmac	LOOP
		C+	50	endmac	CMP.bytes
		C	51		
		C	52		; If we arrive here, both buffers match.
		C	53		
005C9D		C	54	echo_match	PUMP DEVICE.console,echo_match\$ ; Report to operator contents of both packets match
		C+	54	scope	; A tribute to Dave Shriner - Display Data Corp.
005C9D	21 B5 5D	C+	54	LD	HL,echo_match\$ ; Pump from this starting point until ETX.asc is found.
		C+	54	if	DEVICE.console=0
005CA0	7E	C+	54	__3970:	LD A,(HL) ; Get byte to send.
005CA1	23	C+	54	INC	HL ; pointer=pointer+1
		C+	54	; CP	EOT.asc ; End Of Transmission (block)?
		C+	54	; jr	Z,\$0a ; Return if finished.
005CA2	FE03	C+	54	CP	ETX.asc ; End Of Text?
005CA4	28 16	C+	54	jr	Z,__3970a ; Return if finished.
005CA6	FE0D	C+	54	CP	CR.asc ; Was it a CR?
005CA8	CC B2 5C	C+	54	call	z,__3971 ; 1st add a LF to stream.
005CAB	4F	C+	54	LD	C,A ; Output byte in C.
005CAC	AF	C+	54	XOR	A ; Set Z flag for put.
005CAD	CD D9 10	C+	54	CALL	U0DVR ; Out the UART we go.

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl_net_echo.s
005CB0	18 EE	C+	54	JR	__3970
005CB2	3E0A	C+	54	__3971:	ld a,LF.asc ; Get a LF.
005CB4	4F	C+	54	LD	C,A ; Output byte in C.
005CB5	AF	C+	54	XOR	A ; Set Z flag for put
005CB6	CD D9 10	C+	54	CALL	U0DVR ; Out the UART we go.
005CB9	3E0D	C+	54	ld	a,CR.asc ; Now put a CR back in stream.
005CBB	C9	C+	54	ret	; Return and print CR.
005CBC		C+	54	__3970a	endif
		C+	54	if	DEVICE.console=1
		C+	54	__3971:	LD A,(HL) ; Get byte to send.
		C+	54	INC	HL ; pointer=pointer+1
		C+	54	; CP	EOT.asc ; End Of Transmission (block)?
		C+	54	; jr	Z,\$1a ; Return if finished.
		C+	54	CP	ETX.asc ; End Of Text?
		C+	54	jr	Z,__3971a ; Return if finished.
		C+	54	LD	C,A ; Output byte to C.
		C+	54	XOR	A ; Set Z flag for put.
		C+	54	CALL	U1DVR ; Out the UART we go.
		C+	54	JR	__3971
		C+	54	__3971a	equ \$
		C+	54	endif	
		C+	54	ENDMAC	PUMP
		C	55	GOTO	askforenter ; After operator presses enter, exit this program.
		C+	55	scope	
005CBC	C3 01 5D	C+	55	jp	askforenter
		C+	55	endmac	GOTO
		C	56		
005CBF		C	57	echo_no_match	PUMP DEVICE.console,echo_no_match\$; Report to operator contents of packets DO
		C+	57	scope	; A tribute to Dave Shriner - Display Data Corp.
005CBF	21 E7 5D	C+	57	LD	HL,echo_no_match\$; Pump from this starting point until ETX.asc is found.
		C+	57	if	DEVICE.console=0
005CC2	7E	C+	57	__3990:	LD A,(HL) ; Get byte to send.
005CC3	23	C+	57	INC	HL ; pointer=pointer+1
		C+	57	; CP	EOT.asc ; End Of Transmission (block)?
		C+	57	; jr	Z,\$0a ; Return if finished.
005CC4	FE03	C+	57	CP	ETX.asc ; End Of Text?
005CC6	28 16	C+	57	jr	Z,__3990a ; Return if finished.
005CC8	FE0D	C+	57	CP	CR.asc ; Was it a CR?
005CCA	CC D4 5C	C+	57	call	z,__3991 ; 1st add a LF to stream.
005CCD	4F	C+	57	LD	C,A ; Output byte in C.
005CCE	AF	C+	57	XOR	A ; Set Z flag for put.
005CCF	CD D9 10	C+	57	CALL	U0DVR ; Out the UART we go.
005CD2	18 EE	C+	57	JR	__3990
005CD4	3E0A	C+	57	__3991:	ld a,LF.asc ; Get a LF.
005CD6	4F	C+	57	LD	C,A ; Output byte in C.
005CD7	AF	C+	57	XOR	A ; Set Z flag for put
005CD8	CD D9 10	C+	57	CALL	U0DVR ; Out the UART we go.
005CDB	3E0D	C+	57	ld	a,CR.asc ; Now put a CR back in stream.
005CDD	C9	C+	57	ret	; Return and print CR.
005CDE		C+	57	__3990a	endif
		C+	57	if	DEVICE.console=1
		C+	57	__3991:	LD A,(HL) ; Get byte to send.
		C+	57	INC	HL ; pointer=pointer+1
		C+	57	; CP	EOT.asc ; End Of Transmission (block)?
		C+	57	; jr	Z,\$1a ; Return if finished.
		C+	57	CP	ETX.asc ; End Of Text?
		C+	57	jr	Z,__3991a ; Return if finished.
		C+	57	LD	C,A ; Output byte to C.
		C+	57	XOR	A ; Set Z flag for put.
		C+	57	CALL	U1DVR ; Out the UART we go.

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl_net_echo.s
		C+	57	JR	__3991
		C+	57	__3991a	equ \$
		C+	57		endif
		C+	57	ENDMAC	PUMP
		C	58	GOTO	askforenter ; After operator presses enter, exit this program.
		C+	58	scope	
005CDE	C3 01 5D	C+	58	jp	askforenter
		C+	58	endmac	GOTO
		C	59		
005CE1		C	60	echo_no_reply	PUMP DEVICE.console,echo_no_reply\$; Tell operator no reply.
		C+	60	scope	; A tribute to Dave Shriner - Display Data Corp.
005CE1	21 20 5E	C+	60	LD	HL,echo_no_reply\$; Pump from this starting point until ETX.asc is found.
		C+	60	if	DEVICE.console=0
005CE4	7E	C+	60	__4010:	LD A,(HL) ; Get byte to send.
005CE5	23	C+	60	INC	HL ; pointer=pointer+1
		C+	60	;	CP EOT.asc ; End Of Transmission (block)?
		C+	60	;	jr Z,\$0a ; Return if finished.
005CE6	FE03	C+	60	CP	ETX.asc ; End Of Text?
005CE8	28 16	C+	60	jr	Z,__4010a ; Return if finished.
005CEA	FE0D	C+	60	CP	CR.asc ; Was it a CR?
005CEC	CC F6 5C	C+	60	call	z,__4011 ; 1st add a LF to stream.
005CEF	4F	C+	60	LD	C,A ; Output byte in C.
005CF0	AF	C+	60	XOR	A ; Set Z flag for put.
005CF1	CD D9 10	C+	60	CALL	U0DVR ; Out the UART we go.
005CF4	18 EE	C+	60	JR	__4010
005CF6	3E0A	C+	60	__4011:	ld a,LF.asc ; Get a LF.
005CF8	4F	C+	60	LD	C,A ; Output byte in C.
005CF9	AF	C+	60	XOR	A ; Set Z flag for put
005CFA	CD D9 10	C+	60	CALL	U0DVR ; Out the UART we go.
005CFD	3E0D	C+	60	ld	a,CR.asc ; Now put a CR back in stream.
005CFF	C9	C+	60	ret	; Return and print CR.
005D00		C+	60	__4010a	endif
		C+	60	if	DEVICE.console=1
		C+	60	__4011:	LD A,(HL) ; Get byte to send.
		C+	60	INC	HL ; pointer=pointer+1
		C+	60	;	CP EOT.asc ; End Of Transmission (block)?
		C+	60	;	jr Z,\$1a ; Return if finished.
		C+	60	CP	ETX.asc ; End Of Text?
		C+	60	jr	Z,__4011a ; Return if finished.
		C+	60	LD	C,A ; Output byte to C.
		C+	60	XOR	A ; Set Z flag for put.
		C+	60	CALL	U1DVR ; Out the UART we go.
		C+	60	JR	__4011
		C+	60	__4011a	equ \$
		C+	60		endif
		C+	60	ENDMAC	PUMP
005D00	E1	C	61	pop	hl ; Discard return vector where error happened.
005D01	3EFF	C	62	askforenter	ld a,0ffh
		C	63	RETURN	; Return to calling main IPL menu.
		C+	63	scope	
005D03	C9	C+	63	ret	; Return from call.
		C+	63	endmac	RETURN
		C	64		
		C	65		; Strings used by network echo command.
		C	66		
		C	67		
005D04	40656368 6F0A03	C	68	echo_string\$	db "@echo",LF.asc,ETX.asc
		C	69		
005D0B	20436F6D 6D616E64	C	70	echo_screen\$	db " Command server to echo packet back from client.",CR.asc,ETX.asc
005D13	20736572 76657220				

\*\*\*\*\* TRSDOS \*\*\*\*\*  
< SYS0 IPL - DOS-initialization code & welcome message >

PC	Object		I	Line	Source ..\SYS0_IPL\ipl_net_echo.s	
005D1B	746F2065	63686F20				
005D23	7061636B	65742062				
005D2B	61636B20	66726F6D				
005D33	20636C69	656E742E				
005D3B	0D03					
			C	71		
005D3D	2053656E	64696E67	C	72	try_echo\$db	" Sending packet to server.",CR.asc,ETX.asc
005D45	20706163	6B657420				
005D4D	746F2073	65727665				
005D55	722E0D03					
			C	73		
005D59	20406563	686F2070	C	74	packet_sent\$db	" @echo packet sent.",CR.asc,CR.asc,ETX.asc
005D61	61636B65	74207365				
005D69	6E742E0D	0D03				
			C	75		
005D6F	20526570	6C792070	C	76	echo_rx\$db	" Reply packet received.",CR.asc,CR.asc,ETX.asc
005D77	61636B65	74207265				
005D7F	63656976	65642E0D				
005D87	0D03					
			C	77		
005D89	20436F6D	70617269	C	78	echo_compare\$db	" Comparing packet sent to packet received.",CR.asc,ETX.asc
005D91	6E672070	61636B65				
005D99	74207365	6E742074				
005DA1	6F207061	636B6574				
005DA9	20726563	65697665				
005DB1	642E0D03					
			C	79		
005DB5	20242424	50415353	C	80	echo_match\$db	" \$\$\$PASS\$\$\$ Packet sent matches packet received.",CR.asc,ETX.asc
005DBD	24242420	5061636B				
005DC5	65742073	656E7420				
005DCD	6D617463	68657320				
005DD5	7061636B	65742072				
005DDD	65636569	7665642E				
005DE5	0D03					
			C	81		
005DE7	20242424	4641494C	C	82	echo_no_match\$db	" \$\$\$FAIL\$\$\$ Packet sent does NOT match packet received.",CR.asc,ETX.asc
005DEF	24242420	5061636B				
005DF7	65742073	656E7420				
005DFF	646F6573	204E4F54				
005E07	206D6174	63682070				
005E0F	61636B65	74207265				
005E17	63656976	65642E0D				
005E1F	03					
			C	83		
005E20	0D204E6F	20726570	C	84	echo_no_reply\$db	CR.asc, " No reply from server.",CR.asc,ETX.asc
005E28	6C792066	726F6D20				
005E30	73657276	65722E0D				
005E38	03					
			C	85		
005E39	54686534	35363738	C	86	echo_xmit_buff\$db	"The456789a1quick789b12brown89c123fox789d1234jumped12345overf123456theg1234567l
005E41	39613171	7569636B				
005E49	37383962	31326272				
005E51	6F776E38	39633132				
005E59	33666F78	37383964				
005E61	31323334	6A756D70				
005E69	65643132	3334356F				
005E71	76657266	31323334				
005E79	35367468	65673132				
005E81	33343536	376C617A				
005E89	79323334	35363738				

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl_net_echo.s
005E91	646F6773	33343536			
005E99	37383962	61636B34			
005EA1	35363738	396B3132			
005EA9	33343536	3738396C			
005EB1	31323334	35363738			
005EB9	396D3132	33343536			
005EC1	3738396E	31323334			
005EC9	35363738	396F3132			
005ED1	33343536	37383970			
005ED9	31323334	35363738			
005EE1	39713132	33343536			
005EE9	37383972	31323334			
005EF1	35363738	39733132			
005EF9	33343536	37383974			
005F01	31323334	35363738			
005F09	39753132	33343536			
005F11	37383976	31323334			
005F19	35363738	39773132			
005F21	33343536	37383978			
005F29	31323334	35363738			
005F31	39793132	33343536			
		C	87		
005F39		C	88	echo_rec_buff ds 256	; Buffer to hold ping/RX/TX data.
		C	0	include "ipl_net_bind.s"	; Bind TRSDOS volume x to network driver.
		C	1	; This section of code tests if server available.	
		C	2		
		C	3	; After sending "@ping<CR>" this code tests for reply "@pong<CR>"	
		C	4		
		C	5	; Only tests if server available, connected & responding.	
		C	6		
006039		C	7	bind_server GOSUB scrolloff	
		C+	7	scope	
006039	CD ED 6E	C+	7	CALL scrolloff	
		C+	7	endmac GOSUB	
		C	8		
00603C	3E06	C	9	ld a,6	
00603E	32 76 0E	C	10	ld (net_volume),a	; Update flag in FDC driver.
		C	11		
006041	C630	C	12	add '0'	; Convert to ASCII.
006043	32 8A 61	C	13	ld (bind_who),a	
006046	32 EE 61	C	14	ld (bound_x),a	
006049	32 14 72	C	15	ld (serv_vol_num),a	; Main menu server volume #.
		C	16		
		C	17	PUMP DEVICE.console,bind_screen\$	; Tell operator this is bind program.
		C+	17	scope	; A tribute to Dave Shriner - Display Data Corp.
00604C	21 61 61	C+	17	LD HL,bind_screen\$	; Pump from this starting point until ETX.asc is found.
		C+	17	if DEVICE.console=0	
00604F	7E	C+	17	LD A,(HL)	; Get byte to send.
006050	23	C+	17	INC HL	; pointer=pointer+1
		C+	17	; CP EOT.asc	; End Of Transmission (block)?
		C+	17	; jr Z,\$0a	; Return if finished.
006051	FE03	C+	17	CP ETX.asc	; End Of Text?
006053	28 16	C+	17	jr Z,__4040a	; Return if finished.
006055	FE0D	C+	17	CP CR.asc	; Was it a CR?
006057	CC 61 60	C+	17	call z,__4041	; 1st add a LF to stream.
00605A	4F	C+	17	LD C,A	; Output byte in C.
00605B	AF	C+	17	XOR A	; Set Z flag for put.
00605C	CD D9 10	C+	17	CALL U0DVR	; Out the UART we go.
00605F	18 EE	C+	17	JR __4040	
006061	3E0A	C+	17	LD a,LF.asc	; Get a LF.



\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	..\SYS0_IPL\ipl_net_bind.s
006063	4F	C+	17	LD	C,A ; Output byte in C.
006064	AF	C+	17	XOR	A ; Set Z flag for put
006065	CD D9 10	C+	17	CALL	U0DVR ; Out the UART we go.
006068	3E0D	C+	17	ld	a,CR.asc ; Now put a CR back in stream.
00606A	C9	C+	17	ret	; Return and print CR.
00606B		C+	17	__4040a	endif
		C+	17	if	DEVICE.console=1
		C+	17	__4041:	LD A,(HL) ; Get byte to send.
		C+	17	INC	HL ; pointer=pointer+1
		C+	17	; CP	EOT.asc ; End Of Transmission (block)?
		C+	17	; jr	Z,\$1a ; Return if finished.
		C+	17	CP	ETX.asc ; End Of Text?
		C+	17	jr	Z,__4041a ; Return if finished.
		C+	17	LD	C,A ; Output byte to C.
		C+	17	XOR	A ; Set Z flag for put.
		C+	17	CALL	U1DVR ; Out the UART we go.
		C+	17	JR	__4041
		C+	17	__4041a	equ \$
		C+	17	endif	
		C+	17	ENDMAC	PUMP
		C	18	PUMP	DEVICE.console,bind_who\$ ; Report which vol to bind to server.
		C+	18	scope	; A tribute to Dave Shriner - Display DataCorp.
00606B	21 7D 61	C+	18	LD	HL,bind_who\$ ; Pump from this starting point until ETX.asc is found.
		C+	18	if	DEVICE.console=0
00606E	7E	C+	18	__4050:	LD A,(HL) ; Get byte to send.
00606F	23	C+	18	INC	HL ; pointer=pointer+1
		C+	18	; CP	EOT.asc ; End Of Transmission (block)?
		C+	18	; jr	Z,\$0a ; Return if finished.
		C+	18	CP	ETX.asc ; End Of Text?
006070	FE03	C+	18	jr	Z,__4050a ; Return if finished.
006072	28 16	C+	18	CP	CR.asc ; Was it a CR?
006074	FE0D	C+	18	call	z,__4051 ; 1st add a LF to stream.
006076	CC 80 60	C+	18	LD	C,A ; Output byte in C.
006079	4F	C+	18	XOR	A ; Set Z flag for put.
00607A	AF	C+	18	CALL	U0DVR ; Out the UART we go.
00607B	CD D9 10	C+	18	JR	__4050
00607E	18 EE	C+	18	__4051:	ld a,LF.asc ; Get a LF.
006080	3E0A	C+	18	LD	C,A ; Output byte in C.
006082	4F	C+	18	XOR	A ; Set Z flag for put
006083	AF	C+	18	CALL	U0DVR ; Out the UART we go.
006084	CD D9 10	C+	18	ld	a,CR.asc ; Now put a CR back in stream.
006087	3E0D	C+	18	ret	; Return and print CR.
006089	C9	C+	18	__4050a	endif
00608A		C+	18	if	DEVICE.console=1
		C+	18	__4051:	LD A,(HL) ; Get byte to send.
		C+	18	INC	HL ; pointer=pointer+1
		C+	18	; CP	EOT.asc ; End Of Transmission (block)?
		C+	18	; jr	Z,\$1a ; Return if finished.
		C+	18	CP	ETX.asc ; End Of Text?
		C+	18	jr	Z,__4051a ; Return if finished.
		C+	18	LD	C,A ; Output byte to C.
		C+	18	XOR	A ; Set Z flag for put.
		C+	18	CALL	U1DVR ; Out the UART we go.
		C+	18	JR	__4051
		C+	18	__4051a	equ \$
		C+	18	endif	
		C+	18	ENDMAC	PUMP
		C	19	; UART1.mctl.put	00000001b ; Set DTR active, RTS active tell we are ready.
		C	20		
		C	21	PUMP	DEVICE.serialnet,bind_command\$ ; Send ping string to server @ping LF.asc.

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl_net_bind.s
		C+	21	scope	; A tribute to Dave Shriner - Display Data Corp.
00608A	21 4C 61	C+	21	LD HL,bind_command\$	; Pump from this starting point until ETX.asc is found.
		C+	21	if DEVICE.serialnet=0	
		C+	21	__4060: LD A,(HL)	; Get byte to send.
		C+	21	INC HL	; pointer=pointer+1
		C+	21	; CP EOT.asc	; End Of Transmission (block)?
		C+	21	; jr Z,\$0a	; Return if finished.
		C+	21	CP ETX.asc	; End Of Text?
		C+	21	jr __4060a	; Return if finished.
		C+	21	CP CR.asc	; Was it a CR?
		C+	21	call z,__4061	; 1st add a LF to stream.
		C+	21	LD C,A	; Output byte in C.
		C+	21	XOR A	; Set Z flag for put.
		C+	21	CALL U0DVR	; Out the UART we go.
		C+	21	JR __4060	
		C+	21	__4061: ld a,LF.asc	; Get a LF.
		C+	21	LD C,A	; Output byte in C.
		C+	21	XOR A	; Set Z flag for put
		C+	21	CALL U0DVR	; Out the UART we go.
		C+	21	ld a,CR.asc	; Now put a CR back in stream.
		C+	21	ret	; Return and print CR.
		C+	21	__4060a endif	
		C+	21	if DEVICE.serialnet=1	
00608D	7E	C+	21	__4061: LD A,(HL)	; Get byte to send.
00608E	23	C+	21	INC HL	; pointer=pointer+1
		C+	21	; CP EOT.asc	; End Of Transmission (block)?
		C+	21	; jr Z,\$1a	; Return if finished.
		C+	21	CP ETX.asc	; End Of Text?
		C+	21	jr __4061a	; Return if finished.
		C+	21	LD C,A	; Output byte to C.
		C+	21	XOR A	; Set Z flag for put.
		C+	21	CALL U1DVR	; Out the UART we go.
		C+	21	JR __4061	
	0000609A	C+	21	__4061a equ \$	
		C+	21	endif	
		C+	21	ENDMAC PUMP	
		C	22		
00609A		C	23	get_bind_byte; UART1.mctl.put 00000011b	; Set DTR active, RTS active tell we are ready.
		C	24	UART1.get.byte bind_timeout	; Get byte from UART1. On timeout or error goto no_pong
		C+	24	scope	
00609A	3E01	C+	24	ld a,1	
00609C	B7	C+	24	or a	; On timeout or error goto vector.
00609D	37	C+	24	scf	
		C+	24	GOSUB U1DVR	; C flag set & NZ set means PUT.
		C+	24	scope	
00609E	CD F5 11	C+	24	CALL U1DVR	
		C+	24	endmac GOSUB	
		C+	24	IF_NE_GOSUB bind_timeout	; If nz we had a timeout or other error.
		C+	24	scope	
0060A1	C4 45 61	C+	24	call nz,bind_timeout	
		C+	24	endmac IF_NE_GOSUB	
		C+	24	endmac UART1.get.byte	
		C	25	; UART1.mctl.put 00000001b	; Set DTR active, RTS active tell we are ready.
		C	26		
0060A4	3E40	C	27	ld a,'@'	; Is 1st char @ control code?
0060A6	B9	C	28	cp c	; Compare to C, C has char from GET_BYTE_UART1.
0060A7	20 F1	C	29	JR NZ,get_bind_byte	; Ignore if not control char & continue getting chars.
		C	30		
		C	31	UART1.get.bytes 6,bind_buff+1,bind_timeout	; get 5 bytes into bind_buff, if timeout or error
		C+	31	scope	

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl_net_bind.s
0060A9	0606	C+	31	ld	b,6 ; Input 5 bytes.
0060AB	21 5B 61	C+	31	ld	hl,bind_buff+1 ; Get address of buffer.
0060AE		C+	31	__4101	UART1.get.byte bind_timeout ; Get byte from UART1. If timeout goto no_pong.
		C+	31		scope
0060AE	3E01	C+	31	ld	a,1
0060B0	B7	C+	31	or	a ; On timeout or error goto vector.
0060B1	37	C+	31	scf	
		C+	31	GOSUB	U1DVR ; C flag set & NZ set means PUT.
		C+	31		scope
0060B2	CD F5 11	C+	31	CALL	U1DVR
		C+	31	endmac	GOSUB
		C+	31	IF_NE_GOSUB	bind_timeout ; If nz we had a timeout or other error.
		C+	31		scope
0060B5	C4 45 61	C+	31	call	nz,bind_timeout
		C+	31	endmac	IF_NE_GOSUB
		C+	31	endmac	UART1.get.byte
0060B8	71	C+	31	ld	(hl),c ; Store char.
0060B9	23	C+	31	inc	hl
		C+	31	LOOP	__4101 ; Loop for count bytes.
		C+	31		scope
0060BA	10 F2	C+	31	djnz	__4101
		C+	31	endmac	LOOP
		C+	31	endmac	UART1.get.bytes
		C	32		
		C	33		CMP.bytes 6,bound\$,bind_buff,no_bind ; Compare '@bound' to pong_buffer. If <> goto no_po
		C+	33		scope
0060BC	0606	C+	33	ld	b,6 ; Setup to compare length bytes.
0060BE	21 53 61	C+	33	ld	hl,bound\$ ; Get address of first buffer.
0060C1	11 5A 61	C+	33	ld	de,bind_buff ; Get address of second buffer.
0060C4	1A	C+	33	__4151	ld a,(de) ; Get a byte from rbuffer.
0060C5	BE	C+	33	cp	(hl) ; Compare to other buffer.
		C+	33	IF_NE_GOTO	no_bind ; If compare fails match.
		C+	33		scope
0060C6	C2 18 61	C+	33	jp	nz,no_bind
		C+	33	endmac	IF_NE_GOTO
0060C9	23	C+	33	inc	hl ; Advance pointers for
0060CA	13	C+	33	inc	de ; both buffers.
		C+	33	LOOP	__4151 ; Loop until B=0
		C+	33		scope
0060CB	10 F7	C+	33	djnz	__4151
		C+	33	endmac	LOOP
		C+	33	endmac	CMP.bytes
		C	34	MVC	bound_server,server_stat,12 ; Change server to show available.
		C+	34		scope
0060CD	21 99 61	C+	34	ld	hl,bound_server
0060D0	11 17 72	C+	34	ld	de,server_stat
0060D3	010C00	C+	34	ld	bc,12
0060D6	EDB0	C+	34	LDIR	
		C+	34	ENDMAC	MVC
		C	35	PUMP	DEVICE.console,bind_reply_rx\$ ; Report received @bound from server.
		C+	35		scope ; A tribute to Dave Shriner - Display Data Corp.
0060D8	21 D6 61	C+	35	LD	HL,bind_reply_rx\$ ; Pump from this starting point until ETX.asc is found.
		C+	35	if	DEVICE.console=0
0060DB	7E	C+	35	__4190:	LD A,(HL) ; Get byte to send.
0060DC	23	C+	35	INC	HL ; pointer=pointer+1
		C+	35	;	CP EOT.asc ; End Of Transmission (block)?
		C+	35	;	jr Z,\$0a ; Return if finished.
0060DD	FE03	C+	35	CP	ETX.asc ; End Of Text?
0060DF	28 16	C+	35	jr	Z,__4190a ; Return if finished.
0060E1	FE0D	C+	35	CP	CR.asc ; Was it a CR?

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl_net_bind.s
0060E3	CC ED 60	C+	35	call	z, __4191 ; 1st add a LF to stream.
0060E6	4F	C+	35	LD	C,A ; Output byte in C.
0060E7	AF	C+	35	XOR	A ; Set Z flag for put.
0060E8	CD D9 10	C+	35	CALL	U0DVR ; Out the UART we go.
0060EB	18 EE	C+	35	JR	__4190
0060ED	3E0A	C+	35	__4191:	ld a,LF.asc ; Get a LF.
0060EF	4F	C+	35	LD	C,A ; Output byte in C.
0060F0	AF	C+	35	XOR	A ; Set Z flag for put
0060F1	CD D9 10	C+	35	CALL	U0DVR ; Out the UART we go.
0060F4	3E0D	C+	35	ld	a,CR.asc ; Now put a CR back in stream.
0060F6	C9	C+	35	ret	; Return and print CR.
0060F7		C+	35	__4190a	endif
		C+	35		if DEVICE.console=1
		C+	35	__4191:	LD A,(HL) ; Get byte to send.
		C+	35		INC HL ; pointer=pointer+1
		C+	35		; CP EOT.asc ; End Of Transmission (block)?
		C+	35		; jr Z,\$1a ; Return if finished.
		C+	35		; CP ETX.asc ; End Of Text?
		C+	35		; jr Z,__4191a ; Return if finished.
		C+	35		LD C,A ; Output byte to C.
		C+	35		XOR A ; Set Z flag for put.
		C+	35		CALL U1DVR ; Out the UART we go.
		C+	35		JR __4191
		C+	35	__4191a	equ \$
		C+	35		endif
		C+	35		ENDMAC PUMP
		C	36		
		C	37		UART1.get.bytes 0,bind_buffer,bind_timeout ; Place 256 bytes in buffer & vector on timeout.
		C+	37		scope
0060F7	0600	C+	37		ld b,0 ; Input 5 bytes.
0060F9	21 1E 62	C+	37		ld hl,bind_buffer ; Get address of buffer.
0060FC		C+	37	__4201	UART1.get.byte bind_timeout ; Get byte from UART1. If timeout goto no_pong.
		C+	37		scope
0060FC	3E01	C+	37		ld a,1
0060FE	B7	C+	37		or a ; On timeout or error goto vector.
0060FF	37	C+	37		scf
		C+	37		GOSUB U1DVR ; C flag set & NZ set means PUT.
		C+	37		scope
006100	CD F5 11	C+	37		CALL U1DVR
		C+	37		endmac GOSUB
		C+	37		IF_NE_GOSUB bind_timeout ; If nz we had a timeout or other error.
		C+	37		scope
006103	C4 45 61	C+	37		call nz,bind_timeout
		C+	37		endmac IF_NE_GOSUB
		C+	37		endmac UART1.get.byte
006106	71	C+	37		ld (hl),c ; Store char.
006107	23	C+	37		inc hl
		C+	37		LOOP __4201 ; Loop for count bytes.
		C+	37		scope
006108	10 F2	C+	37		djnz __4201
		C+	37		endmac LOOP
		C+	37		endmac UART1.get.bytes
		C	38		
		C	39		get.DCT 6 ; Get address of DCT #6 in IY.
00610A	0E06	C+	39		ld c,6
		C+	39		SVC svc_gtdct
00610C	3E51	C+	39		ld a,svc_gtdct
00610E	EF	C+	39		restart28
		C+	39		ENDMAC SVC
		C+	39		endmac get.DCT ; Return address of DCT in IY register.

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl_net_bind.s
00610F	3A 49 61	C	40	ld	a,(jp_ins)
006112	FD7700	C	41	ld	(IY),a ; Enable driver DCT6.
		C	42		
006115	3EFF	C	43	bind_exit	ld a,0ffh
006117		C	44	bind_ret	RETURN
		C+	44	scope	
006117	C9	C+	44	ret	; Return from call.
		C+	44	endmac	RETURN
		C	45		
006118		C	46	no_bind	MVC server_not,server_stat,12 ; Change server to show available.
		C+	46	scope	
006118	21 12 62	C+	46	ld	hl,server_not
00611B	11 17 72	C+	46	ld	de,server_stat
00611E	010C00	C+	46	ld	bc,12
006121	EDB0	C+	46	LDIR	
		C+	46	ENDMAC	MVC
		C	47	PUMP	DEVICE.console,bind_not_rec\$ ; Tell operator no reply recieved.
		C+	47	scope	; A tribute to Dave Shriner - Display Data Corp.
006123	21 A5 61	C+	47	LD	HL,bind_not_rec\$ ; Pump from this starting point until ETX.asc is found.
		C+	47	if	DEVICE.console=0
006126	7E	C+	47	LD	A,(HL) ; Get byte to send.
006127	23	C+	47	INC	HL ; pointer=pointer+1
		C+	47	CP	EOT.asc ; End Of Transmission (block)?
		C+	47	jr	Z,\$0a ; Return if finished.
006128	FE03	C+	47	CP	ETX.asc ; End Of Text?
00612A	28 16	C+	47	jr	Z,__4290a ; Return if finished.
00612C	FE0D	C+	47	CP	CR.asc ; Was it a CR?
00612E	CC 38 61	C+	47	call	z,__4291 ; 1st add a LF to stream.
006131	4F	C+	47	LD	C,A ; Output byte in C.
006132	AF	C+	47	XOR	A ; Set Z flag for put.
006133	CD D9 10	C+	47	CALL	U0DVR ; Out the UART we go.
006136	18 EE	C+	47	JR	__4290
006138	3E0A	C+	47	LD	a,LF.asc ; Get a LF.
00613A	4F	C+	47	LD	C,A ; Output byte in C.
00613B	AF	C+	47	XOR	A ; Set Z flag for put
00613C	CD D9 10	C+	47	CALL	U0DVR ; Out the UART we go.
00613F	3E0D	C+	47	ld	a,CR.asc ; Now put a CR back in stream.
006141	C9	C+	47	ret	; Return and print CR.
006142		C+	47	__4290a	endif
		C+	47	if	DEVICE.console=1
		C+	47	LD	A,(HL) ; Get byte to send.
		C+	47	INC	HL ; pointer=pointer+1
		C+	47	CP	EOT.asc ; End Of Transmission (block)?
		C+	47	jr	Z,\$1a ; Return if finished.
		C+	47	CP	ETX.asc ; End Of Text?
		C+	47	jr	Z,__4291a ; Return if finished.
		C+	47	LD	C,A ; Output byte to C.
		C+	47	XOR	A ; Set Z flag for put.
		C+	47	CALL	U1DVR ; Out the UART we go.
		C+	47	JR	__4291
		C+	47	__4291a	equ \$
		C+	47	endif	
		C+	47	ENDMAC	PUMP
		C	48	GOTO	bind_exit
		C+	48	scope	
006142	C3 15 61	C+	48	jp	bind_exit
		C+	48	endmac	GOTO
		C	49		
006145	E1	C	50	bind_timeout	pop hl ; Discard return address.
		C	51	GOTO	no_bind

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl_net_bind.s
		C+	51	scope	
006146	C3 18 61	C+	51	jp no_bind	
		C+	51	endmac	GOTO
		C	52		
006149	C30000	C	53	jp_ins	jp 0
		C	54		
00614C	4062696E 640A03	C	55	bind_command\$ db	"@bind",LF.asc,ETX.asc ; This is command sent to server.
		C	56		
006153	40626F75 6E640A	C	57	bound\$ db	"@bound",LF.asc
		C	58		
00615A	40313233 343520	C	59	bind_buff db	"@12345 "
		C	60		
006161	2042696E 6420766F	C	61	bind_screen\$ db	" Bind volumes to network.",CR.asc,CR.asc,ETX.asc
006169	6C756D65 7320746F				
006171	206E6574 776F726B				
006179	2E0D0D03				
		C	62		
00617D	2042696E 6420766F	C	63	bind_who\$db	" Bind volume x to network.",CR.asc,ETX.asc
006185	6C756D65 20782074				
00618D	6F206E65 74776F72				
006195	6B2E0D03				
		C	64		
	0000618A	C	65	bind_who equ	\$-15 ; Pointer to storage holding ascii of vol #.
		C	66		
006199	636F6E6E 65637465	C	67	bound_server db	"connected "
0061A1	64202020				
		C	68		
0061A5	0D205365 72766572	C	69	bind_not_rec\$ db	CR.asc," Server did not reply or accept bind request.",CR.asc,CR.asc,ETX.asc
0061AD	20646964 206E6F74				
0061B5	20726570 6C79206F				
0061BD	72206163 63657074				
0061C5	2062696E 64207265				
0061CD	71756573 742E0D0D				
0061D5	03				
		C	70		
	000061EE	C	71	bound_x equ	+\$+24
0061D6	20536572 76657220	C	72	bind_reply_rx\$ db	" Server replied, volume x is now bound to network server.",CR.asc,CR.asc,ETX.a
0061DE	7265706C 6965642C				
0061E6	20766F6C 756D6520				
0061EE	78206973 206E6F77				
0061F6	20626F75 6E642074				
0061FE	6F206E65 74776F72				
006206	6B207365 72766572				
00620E	2E0D0D03				
		C	73		
006212	756E6176 61696C61	C	74	server_not db	"unavailable "
00621A	626C6520				
		C	75		
00621E		C	76	bind_buffer ds	256
		C	77		
		C	78		
		C	79		
		C	0	include "ipl_net_stats.s"	; Code to display server status on server.
00631E	00	C	1	stat_server nop	
		C	2		
		C	3	PUMP DEVICE.serialnet,stat_string\$	; Command server to display statistics.
		C+	3	scope	; A tribute to Dave Shriner - Display Data Corp.
00631F	21 30 63	C+	3	LD HL,stat_string\$	; Pump from this starting point until ETX.asc is found.
		C+	3	if DEVICE.serialnet=0	
		C+	3	__4320: LD A,(HL)	; Get byte to send.

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl_net_stats.s
		C+	3	INC	HL ; pointer=pointer+1
		C+	3	;	CP EOT.asc ; End Of Transmission (block)?
		C+	3	;	jz Z,\$0a ; Return if finished.
		C+	3	CP	ETX.asc ; End Of Text?
		C+	3	jz	Z,__4320a ; Return if finished.
		C+	3	CP	CR.asc ; Was it a CR?
		C+	3	call	z,__4321 ; 1st add a LF to stream.
		C+	3	LD	C,A ; Output byte in C.
		C+	3	XOR	A ; Set Z flag for put.
		C+	3	CALL	U0DVR ; Out the UART we go.
		C+	3	JR	__4320
		C+	3	__4321:	ld a,LF.asc ; Get a LF.
		C+	3	LD	C,A ; Output byte in C.
		C+	3	XOR	A ; Set Z flag for put
		C+	3	CALL	U0DVR ; Out the UART we go.
		C+	3	ld	a,CR.asc ; Now put a CR back in stream.
		C+	3	ret	; Return and print CR.
		C+	3	__4320a	endif
		C+	3	if	DEVICE.serialnet=1
006322	7E	C+	3	__4321:	LD A,(HL) ; Get byte to send.
006323	23	C+	3	INC	HL ; pointer=pointer+1
		C+	3	;	CP EOT.asc ; End Of Transmission (block)?
		C+	3	;	jz Z,\$1a ; Return if finished.
006324	FE03	C+	3	CP	ETX.asc ; End Of Text?
006326	28 07	C+	3	jz	Z,__4321a ; Return if finished.
006328	4F	C+	3	LD	C,A ; Output byte to C.
006329	AF	C+	3	XOR	A ; Set Z flag for put.
00632A	CD F5 11	C+	3	CALL	U1DVR ; Out the UART we go.
00632D	18 F3	C+	3	JR	__4321
	0000632F	C+	3	__4321a	equ \$
		C+	3	endif	
		C+	3	ENDMAC	PUMP
		C	4	RETURN	
		C+	4	scope	
00632F	C9	C+	4	ret	; Return from call.
		C+	4	endmac	RETURN
		C	5		
006330	40737461 740A03	C	6	stat_string\$	db "@stat",LF.asc,ETX.asc
		C	0	include	"ipl_net_utils.s" ; Menu containing network utilities.
		C	1		; This section of code tests if server available.
		C	2		
		C	3		
		C	4		
006337		C	5	net_utils	GOSUB scrolloff
		C+	5	scope	
006337	CD ED 6E	C+	5	CALL	scrolloff
		C+	5	endmac	GOSUB
		C	6	PUMP	DEVICE.console,ipl_net_utils ; Tell operator this is network utils.
		C+	6	scope	; A tribute to Dave Shriner - Display Data Corp.
00633A	21 E3 63	C+	6	LD	HL,ipl_net_utils ; Pump from this starting point until ETX.asc is found.
		C+	6	if	DEVICE.console=0
00633D	7E	C+	6	__4350:	LD A,(HL) ; Get byte to send.
00633E	23	C+	6	INC	HL ; pointer=pointer+1
		C+	6	;	CP EOT.asc ; End Of Transmission (block)?
		C+	6	;	jz Z,\$0a ; Return if finished.
00633F	FE03	C+	6	CP	ETX.asc ; End Of Text?
006341	28 16	C+	6	jz	Z,__4350a ; Return if finished.
006343	FE0D	C+	6	CP	CR.asc ; Was it a CR?
006345	CC 4F 63	C+	6	call	z,__4351 ; 1st add a LF to stream.
006348	4F	C+	6	LD	C,A ; Output byte in C.

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	..\SYS0_IPL\ipl_net_utils.s
006349	AF	C+	6	XOR A	; Set Z flag for put.
00634A	CD D9 10	C+	6	CALL U0DVR	; Out the UART we go.
00634D	18 EE	C+	6	JR __4350	
00634F	3E0A	C+	6	__4351: ld a,LF.asc	; Get a LF.
006351	4F	C+	6	LD C,A	; Output byte in C.
006352	AF	C+	6	XOR A	; Set Z flag for put
006353	CD D9 10	C+	6	CALL U0DVR	; Out the UART we go.
006356	3E0D	C+	6	ld a,CR.asc	; Now put a CR back in stream.
006358	C9	C+	6	ret	; Return and print CR.
006359		C+	6	__4350a endif	
		C+	6	if DEVICE.console=1	
		C+	6	__4351: LD A,(HL)	; Get byte to send.
		C+	6	INC HL	; pointer=pointer+1
		C+	6	; CP EOT.asc	; End Of Transmission (block)?
		C+	6	; jr Z,\$1a	; Return if finished.
		C+	6	CP ETX.asc	; End Of Text?
		C+	6	jr Z,__4351a	; Return if finished.
		C+	6	LD C,A	; Output byte to C.
		C+	6	XOR A	; Set Z flag for put.
		C+	6	CALL U1DVR	; Out the UART we go.
		C+	6	JR __4351	
		C+	6	__4351a equ \$	
		C+	6	endif	
		C+	6	ENDMAC PUMP	
		C	7	GOSUB getchar	
		C+	7	scope	
006359	CD 7E 6F	C+	7	CALL getchar	
		C+	7	endmac GOSUB	
		C	8		
00635C	FE1B	C	9	cp ESC.asc	
		C	10	IF_EQ_RETURN	
		C+	10	scope	
00635E	C8	C+	10	ret z	
		C+	10	endmac IF_EQ_RETURN	
		C	11		
00635F	FE70	C	12	cp 'p'	
		C	13	IF_EQ_GOSUB ping_server	; Ping server.
		C+	13	scope	
006361	CC 26 5A	C+	13	call z,ping_server	
		C+	13	endmac IF_EQ_GOSUB	
006364	FE50	C	14	cp 'P'	
		C	15	IF_EQ_GOSUB ping_server	; testing if it replies.
		C+	15	scope	
006366	CC 26 5A	C+	15	call z,ping_server	
		C+	15	endmac IF_EQ_GOSUB	
		C	16		
006369	FE65	C	17	cp 'e'	
		C	18	IF_EQ_GOSUB echo_server	; Ping server.
		C+	18	scope	
00636B	CC BB 5B	C+	18	call z,echo_server	
		C+	18	endmac IF_EQ_GOSUB	
00636E	FE45	C	19	cp 'E'	
		C	20	IF_EQ_GOSUB echo_server	; testing if it replies.
		C+	20	scope	
006370	CC BB 5B	C+	20	call z,echo_server	
		C+	20	endmac IF_EQ_GOSUB	
		C	21		
006373	FE62	C	22	cp 'b'	
		C	23	IF_EQ_GOSUB bind_server	; Bind volume to server.
		C+	23	scope	



\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source
006375	CC 39 60	C+	23	call z,bind_server
		C+	23	endmac IF_EQ_GOSUB
006378	FE42	C	24	cp 'B'
		C	25	IF_EQ_GOSUB bind_server ; testing if it replies.
		C+	25	scope
00637A	CC 39 60	C+	25	call z,bind_server
		C+	25	endmac IF_EQ_GOSUB
		C	26	
00637D	FE71	C	27	cp 'q'
		C	28	IF_EQ_GOSUB query_server ; Query server & obtain parameters.
		C+	28	scope
00637F	CC 03 66	C+	28	call z,query_server
		C+	28	endmac IF_EQ_GOSUB
006382	FE51	C	29	cp 'Q'
		C	30	IF_EQ_GOSUB query_server ; Query server & obtain parameters.
		C+	30	scope
006384	CC 03 66	C+	30	call z,query_server
		C+	30	endmac IF_EQ_GOSUB
		C	31	
006387	FE73	C	32	cp 's'
		C	33	IF_EQ_GOSUB stat_server ; Command server to display stastics.
		C+	33	scope
006389	CC 1E 63	C+	33	call z,stat_server
		C+	33	endmac IF_EQ_GOSUB
00638C	FE53	C	34	cp 'S'
		C	35	IF_EQ_GOSUB stat_server ; Command server to display stastics.
		C+	35	scope
00638E	CC 1E 63	C+	35	call z,stat_server
		C+	35	endmac IF_EQ_GOSUB
		C	36	
006391	FEFF	C	37	cp 0ffh ; Did we go to a function & return back?
006393	20 A2	C	38	jr nz,net_utils
		C	39	
		C	40	
		C	41	PUMP DEVICE.console,utils_continue ; Ask operator to press enter to continue.
		C+	41	scope ; A tribute to Dave Shriner - Display Data Corp.
006395	21 BE 63	C+	41	LD HL,utils_continue ; Pump from this starting point until ETX.asc is found.
		C+	41	if DEVICE.console=0
006398	7E	C+	41	__4480: LD A,(HL) ; Get byte to send.
006399	23	C+	41	INC HL ; pointer=pointer+1
		C+	41	; CP EOT.asc ; End Of Transmission (block)?
		C+	41	; jr Z,\$0a ; Return if finished.
00639A	FE03	C+	41	CP ETX.asc ; End Of Text?
00639C	28 16	C+	41	jr Z,__4480a ; Return if finished.
00639E	FE0D	C+	41	CP CR.asc ; Was it a CR?
0063A0	CC AA 63	C+	41	call z,__4481 ; 1st add a LF to stream.
0063A3	4F	C+	41	LD C,A ; Output byte in C.
0063A4	AF	C+	41	XOR A ; Set Z flag for put.
0063A5	CD D9 10	C+	41	CALL U0DVR ; Out the UART we go.
0063A8	18 EE	C+	41	JR __4480
0063AA	3E0A	C+	41	__4481: ld a,LF.asc ; Get a LF.
0063AC	4F	C+	41	LD C,A ; Output byte in C.
0063AD	AF	C+	41	XOR A ; Set Z flag for put
0063AE	CD D9 10	C+	41	CALL U0DVR ; Out the UART we go.
0063B1	3E0D	C+	41	ld a,CR.asc ; Now put a CR back in stream.
0063B3	C9	C+	41	ret ; Return and print CR.
0063B4		C+	41	__4480a endif
		C+	41	if DEVICE.console=1
		C+	41	__4481: LD A,(HL) ; Get byte to send.
		C+	41	INC HL ; pointer=pointer+1

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl_net_utils.s
		C+	41	; CP	EOT.asc ; End Of Transmission (block)?
		C+	41	; jr	Z,\$1a ; Return if finished.
		C+	41	CP	ETX.asc ; End Of Text?
		C+	41	jr	Z,__4481a ; Return if finished.
		C+	41	LD	C,A ; Output byte to C.
		C+	41	XOR	A ; Set Z flag for put.
		C+	41	CALL	U1DVR ; Out the UART we go.
		C+	41	JR	__4481
		C+	41	__4481a	equ \$
		C+	41	endif	
		C+	41	ENDMAC	PUMP
0063B4		C	42	utils_getkey	GOSUB getchar
		C+	42	scope	
0063B4	CD 7E 6F	C+	42	CALL	getchar
		C+	42	endmac	GOSUB
0063B7	FE0D	C	43	cp	CR.asc
0063B9	CA 37 63	C	44	jp	z,net_utils
0063BC	18 F6	C	45	jr	utils_getkey
		C	46		
		C	47		
0063BE	0D204669 6E697368	C	48	utils_continue db	CR.asc," Finished, press enter to continue.",ETX.asc
0063C6	65642C20 70726573				
0063CE	7320656E 74657220				
0063D6	746F2063 6F6E7469				
0063DE	6E75652E 03				
		C	0	include "ipl_net_utilities.s"	; Menu containing network utilities.
		C	1		
		C	2	; TRS-OS Network Utilities.	
		C	3		
	000063E3	C	4	ipl_net_utils equ	\$ ; Screen pump starts here.
0063E3	0D	C	5	db	CR.asc
		C	6		
0063E4	20202020 20202020	C	7	DB	" T R S D O S N E T W O R K U T I L I T I E S",CR.asc
0063EC	20202020 20202020				
0063F4	54205220 53204420				
0063FC	4F205320 2020204E				
006404	20452054 2057204F				
00640C	2052204B 20202020				
006414	55205420 49204C20				
00641C	49205420 49204520				
006424	530D				
006426	20202020 20202020	C	8	db	" _____",CR.asc
00642E	20202020 20202020				
006436	5F5F5F5F 5F5F5F5F				
00643E	5F5F5F5F 5F5F5F5F				
006446	5F5F5F5F 5F5F5F5F				
00644E	5F5F5F5F 5F5F5F5F				
006456	5F5F5F5F 5F5F5F5F				
00645E	5F5F5F5F 5F5F5F5F				
006466	5F0D				
006468	0D	C	9	db	CR.asc
006469	20466F75 6E64206E	C	10	query_results\$ db	" Found network volume type: "
006471	6574776F 726B2076				
006479	6F6C756D 65207479				
006481	70653A20				
006485	20202020 20202020	C	11	query_srv_type db	" ,CR.asc ; Container to hold ascii volume.
00648D	0D				
	0000649C	C	12	query_volnam equ	+\$+14
00648E	20566F6C 756D6520	C	13	db	" Volume name: "
006496	6E616D65 3A20				

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\screens\ipl_net_utilities.s
00649C	2D2D2D2D 2D2D2D2D	C	14	db	"-----" ; Volume name.
0064A4	2020	C	15	db	" "
0064A6	2D2D2D2D 2D2D2D2D	C	16	db	"-----" ; Vol creation date.
0064AE	0D	C	17	db	CR.asc
0064AF	20546F74 616C2073	C	18	db	" Total space on volume: ",CR.asc
0064B7	70616365 206F6E20				
0064BF	766F6C75 6D653A20				
0064C7	20202020 20202020				
0064CF	0D				
	000064C7	C	19	query_space_fldequ	\$-9
		C	20		
0064D0	0D	C	21	db	CR.asc
		C	22		
	000064DA	C	23	query_dct3\$	equ \$+9
0064D1	20444354 202B3320	C	24	db	" DCT +3 ", " h",CR.asc
0064D9	20202068 0D				
	000064E7	C	25	query_dct4\$	equ \$+9
0064DE	20444354 202B3420	C	26	db	" DCT +4 ", " h",CR.asc
0064E6	20202068 0D				
	000064F4	C	27	query_dct5\$	equ \$+9
0064EB	20437572 2043796C	C	28	db	" Cur Cyl ", " h",CR.asc
0064F3	20202068 0D				
	00006501	C	29	query_dct6\$	equ \$+9
0064F8	204D6178 2043796C	C	30	db	" Max Cyl ", " h",CR.asc
006500	20202068 0D				
	0000650E	C	31	query_dct7\$	equ \$+9
006505	20444354 202B3720	C	32	db	" DCT +7 ", " h",CR.asc
00650D	20202068 0D				
	0000651B	C	33	query_dct8\$	equ \$+9
006512	20444354 202B3820	C	34	db	" DCT +8 ", " h",CR.asc
00651A	20202068 0D				
	00006528	C	35	query_dct9\$	equ \$+9
00651F	20444354 202B3920	C	36	db	" DCT +9 ", " h",CR.asc
006527	20202068 0D				
		C	37		
00652C	0D	C	38	db	CR.asc
00652D	203C423E 2042696E	C	39	DB	" <B> Bind volume to network server.",CR.asc
006535	6420766F 6C756D65				
00653D	20746F20 6E657477				
006545	6F726B20 73657276				
00654D	65722E0D				
006551	203C453E 20456368	C	40	DB	" <E> Echo packet between server & client. ",CR.asc
006559	6F207061 636B6574				
006561	20626574 7765656E				
006569	20736572 76657220				
006571	2620636C 69656E74				
006579	2E200D				
00657C	203C503E 2050696E	C	41	DB	" <P> Ping server & test for reply.",CR.asc
006584	67207365 72766572				
00658C	20262074 65737420				
006594	666F7220 7265706C				
00659C	792E0D				
00659F	203C513E 20517565	C	42	DB	" <Q> Query server & learn volume.",CR.asc
0065A7	72792073 65727665				
0065AF	72202620 6C656172				
0065B7	6E20766F 6C756D65				
0065BF	2E0D				
0065C1	203C533E 20446973	C	43	DB	" <S> Display stastics on server.",CR.asc
0065C9	706C6179 20737461				
0065D1	73746963 73206F6E				

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

```

PC      Object      I  Line      Source ..\SYS0_IPL\screens\ipl_net_utilities.s
0065D9 20736572 7665722E
0065E1 0D
0065E2 0D          C    44          DB      CR.asc
0065E3 20455343 20726574  C    45          DB      " ESC returns to previous menu.",CR.asc
0065EB 75726E73 20746F20
0065F3 70726576 696F7573
0065FB 206D656E 752E0D
          C    46
006602 03          C    47          db      ETX.asc                      ; Tell pump end of screen.
          C    48
          00000220  C    49  ipl_net_len  EQU  $-ipl_net_utils
          C    0      include "ipl_net_qserv.s"                      ; Query server to learn parameters.
          C    1      ; This section of code tests if server available.
          C    2
          C    3      ; After sending "@ping<CR>" this code tests for reply "@pong<CR>"
          C    4
          C    5      ; Only tests if server available, connected & responding.
          C    6
006603          C    7  query_server  GOSUB  scrolloff
          C+   7      scope
006603 CD ED 6E  C+   7      CALL  scrolloff
          C+   7      endmac      GOSUB
          C    8
          C    9      PUMP  DEVICE.console,query_screen$ ; Tell operator this is ping program.
          C+   9      scope                      ; A tribute to Dave Shriner - Display Data Corp.
006606 21 60 68  C+   9      LD      HL,query_screen$ ; Pump from this starting point until ETX.asc is found.
          C+   9      if      DEVICE.console=0
006609 7E          C+   9      __4510:  LD      A,(HL)          ; Get byte to send.
00660A 23          C+   9      INC      HL          ; pointer=pointer+1
          C+   9      ;      CP      EOT.asc          ; End Of Transmission (block)?
          C+   9      ;      jr      Z,$0a          ; Return if finished.
00660B FE03      C+   9      CP      ETX.asc          ; End Of Text?
00660D 28 16      C+   9      jr      Z,__4510a        ; Return if finished.
00660F FE0D      C+   9      CP      CR.asc          ; Was it a CR?
006611 CC 1B 66  C+   9      call   z,__4511          ; 1st add a LF to stream.
006614 4F          C+   9      LD      C,A          ; Output byte in C.
006615 AF          C+   9      XOR      A          ; Set Z flag for put.
006616 CD D9 10  C+   9      CALL  U0DVR          ; Out the UART we go.
006619 18 EE      C+   9      JR      __4510
00661B 3E0A      C+   9      __4511:  ld      a,LF.asc      ; Get a LF.
00661D 4F          C+   9      LD      C,A          ; Output byte in C.
00661E AF          C+   9      XOR      A          ; Set Z flag for put
00661F CD D9 10  C+   9      CALL  U0DVR          ; Out the UART we go.
006622 3E0D      C+   9      ld      a,CR.asc      ; Now put a CR back in stream.
006624 C9          C+   9      ret          ; Return and print CR.
006625          C+   9      __4510a  endif
          C+   9      if      DEVICE.console=1
          C+   9      __4511:  LD      A,(HL)          ; Get byte to send.
          C+   9      INC      HL          ; pointer=pointer+1
          C+   9      ;      CP      EOT.asc          ; End Of Transmission (block)?
          C+   9      ;      jr      Z,$1a          ; Return if finished.
          C+   9      CP      ETX.asc          ; End Of Text?
          C+   9      jr      Z,__4511a        ; Return if finished.
          C+   9      LD      C,A          ; Output byte to C.
          C+   9      XOR      A          ; Set Z flag for put.
          C+   9      CALL  U1DVR          ; Out the UART we go.
          C+   9      JR      __4511
          C+   9      __4511a  equ      $
          C+   9      endif
          C+   9      ENDMAC      PUMP

```

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source
		C	10	PUMP DEVICE.console,query_start\$ ; Report starting query to server.
		C+	10	scope ; A tribute to Dave Shriner - Display Data Corp.
006625	21 84 68	C+	10	LD HL,query_start\$ ; Pump from this starting point until ETX.asc is found.
		C+	10	if DEVICE.console=0
006628	7E	C+	10	__4520: LD A,(HL) ; Get byte to send.
006629	23	C+	10	INC HL ; pointer=pointer+1
		C+	10	; CP EOT.asc ; End Of Transmission (block)?
		C+	10	; jr Z,\$0a ; Return if finished.
00662A	FE03	C+	10	CP ETX.asc ; End Of Text?
00662C	28 16	C+	10	jr __4520a ; Return if finished.
00662E	FE0D	C+	10	CP CR.asc ; Was it a CR?
006630	CC 3A 66	C+	10	call z,__4521 ; 1st add a LF to stream.
006633	4F	C+	10	LD C,A ; Output byte in C.
006634	AF	C+	10	XOR A ; Set Z flag for put.
006635	CD D9 10	C+	10	CALL U0DVR ; Out the UART we go.
006638	18 EE	C+	10	JR __4520
00663A	3E0A	C+	10	__4521: ld a,LF.asc ; Get a LF.
00663C	4F	C+	10	LD C,A ; Output byte in C.
00663D	AF	C+	10	XOR A ; Set Z flag for put
00663E	CD D9 10	C+	10	CALL U0DVR ; Out the UART we go.
006641	3E0D	C+	10	ld a,CR.asc ; Now put a CR back in stream.
006643	C9	C+	10	ret ; Return and print CR.
006644		C+	10	__4520a endif
		C+	10	if DEVICE.console=1
		C+	10	__4521: LD A,(HL) ; Get byte to send.
		C+	10	INC HL ; pointer=pointer+1
		C+	10	; CP EOT.asc ; End Of Transmission (block)?
		C+	10	; jr Z,\$1a ; Return if finished.
		C+	10	CP ETX.asc ; End Of Text?
		C+	10	jr Z,__4521a ; Return if finished.
		C+	10	LD C,A ; Output byte to C.
		C+	10	XOR A ; Set Z flag for put.
		C+	10	CALL U1DVR ; Out the UART we go.
		C+	10	JR __4521
		C+	10	__4521a equ \$
		C+	10	endif
		C+	10	ENDMAC PUMP
		C	11	
		C	12	PUMP DEVICE.serialnet,query_command\$ ; Send query string to server @rhdr LF.asc.
		C+	12	scope ; A tribute to Dave Shriner - Display Data Corp.
006644	21 F9 68	C+	12	LD HL,query_command\$ ; Pump from this starting point until ETX.asc is found.
		C+	12	if DEVICE.serialnet=0
		C+	12	__4530: LD A,(HL) ; Get byte to send.
		C+	12	INC HL ; pointer=pointer+1
		C+	12	; CP EOT.asc ; End Of Transmission (block)?
		C+	12	; jr Z,\$0a ; Return if finished.
		C+	12	CP ETX.asc ; End Of Text?
		C+	12	jr Z,__4530a ; Return if finished.
		C+	12	CP CR.asc ; Was it a CR?
		C+	12	call z,__4531 ; 1st add a LF to stream.
		C+	12	LD C,A ; Output byte in C.
		C+	12	XOR A ; Set Z flag for put.
		C+	12	CALL U0DVR ; Out the UART we go.
		C+	12	JR __4530
		C+	12	__4531: ld a,LF.asc ; Get a LF.
		C+	12	LD C,A ; Output byte in C.
		C+	12	XOR A ; Set Z flag for put
		C+	12	CALL U0DVR ; Out the UART we go.
		C+	12	ld a,CR.asc ; Now put a CR back in stream.
		C+	12	ret ; Return and print CR.

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl_net_qserv.s
		C+	12	__4530a	endif
		C+	12		if DEVICE.serialnet=1
006647	7E	C+	12	__4531:	LD A,(HL) ; Get byte to send.
006648	23	C+	12		INC HL ; pointer=pointer+1
		C+	12	;	CP EOT.asc ; End Of Transmission (block)?
		C+	12	;	jr Z,\$1a ; Return if finished.
006649	FE03	C+	12		CP ETX.asc ; End Of Text?
00664B	28 07	C+	12		jr Z,__4531a ; Return if finished.
00664D	4F	C+	12		LD C,A ; Output byte to C.
00664E	AF	C+	12		XOR A ; Set Z flag for put.
00664F	CD F5 11	C+	12		CALL U1DVR ; Out the UART we go.
006652	18 F3	C+	12		JR __4531
	00006654	C+	12	__4531a	equ \$
		C+	12		endif
		C+	12		ENDMAC PUMP
		C	13		
		C	14		UART1.get.bytes 0,query_buffer,query_timeout ; Query server, get 256 bytes >buffer & vector
		C+	14		scope
006654	0600	C+	14		ld b,0 ; Input 5 bytes.
006656	21 00 69	C+	14		ld hl,query_buffer ; Get address of buffer.
006659		C+	14	__4541	UART1.get.byte query_timeout ; Get byte from UART1. If timeout goto no_pong.
		C+	14		scope
006659	3E01	C+	14		ld a,1
00665B	B7	C+	14		or a ; On timeout or error goto vector.
00665C	37	C+	14		scf
		C+	14		GOSUB U1DVR ; C flag set & NZ set means PUT.
		C+	14		scope
00665D	CD F5 11	C+	14		CALL U1DVR
		C+	14		endmac GOSUB
		C+	14		IF_NE_GOSUB query_timeout ; If nz we had a timeout or other error.
		C+	14		scope
006660	C4 0A 68	C+	14		call nz,query_timeout
		C+	14		endmac IF_NE_GOSUB
		C+	14		endmac UART1.get.byte
006663	71	C+	14		ld (hl),c ; Store char.
006664	23	C+	14		inc hl
		C+	14		LOOP __4541 ; Loop for count bytes.
		C+	14		scope
006665	10 F2	C+	14		djnz __4541
		C+	14		endmac LOOP
		C+	14		endmac UART1.get.bytes
		C	15		
		C	16		PUMP DEVICE.console,query_replied\$
		C+	16		scope ; A tribute to Dave Shriner - Display Data Corp.
006667	21 C4 68	C+	16		LD HL,query_replied\$ ; Pump from this starting point until ETX.asc is found.
		C+	16		if DEVICE.console=0
00666A	7E	C+	16	__4590:	LD A,(HL) ; Get byte to send.
00666B	23	C+	16		INC HL ; pointer=pointer+1
		C+	16	;	CP EOT.asc ; End Of Transmission (block)?
		C+	16	;	jr Z,\$0a ; Return if finished.
00666C	FE03	C+	16		CP ETX.asc ; End Of Text?
00666E	28 16	C+	16		jr Z,__4590a ; Return if finished.
006670	FE0D	C+	16		CP CR.asc ; Was it a CR?
006672	CC 7C 66	C+	16		call z,__4591 ; 1st add a LF to stream.
006675	4F	C+	16		LD C,A ; Output byte in C.
006676	AF	C+	16		XOR A ; Set Z flag for put.
006677	CD D9 10	C+	16		CALL U0DVR ; Out the UART we go.
00667A	18 EE	C+	16		JR __4590
00667C	3E0A	C+	16	__4591:	ld a,LF.asc ; Get a LF.
00667E	4F	C+	16		LD C,A ; Output byte in C.

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl_net_qserv.s
00667F	AF	C+	16	XOR A	; Set Z flag for put
006680	CD D9 10	C+	16	CALL U0DVR	; Out the UART we go.
006683	3E0D	C+	16	ld a,CR.asc	; Now put a CR back in stream.
006685	C9	C+	16	ret	; Return and print CR.
006686		C+	16	__4590a	endif
		C+	16	if	DEVICE.console=1
		C+	16	__4591:	LD A,(HL) ; Get byte to send.
		C+	16	INC HL	; pointer=pointer+1
		C+	16	CP EOT.asc	; End Of Transmission (block)?
		C+	16	; jr Z,\$1a	; Return if finished.
		C+	16	CP ETX.asc	; End Of Text?
		C+	16	jr Z,__4591a	; Return if finished.
		C+	16	LD C,A	; Output byte to C.
		C+	16	XOR A	; Set Z flag for put.
		C+	16	CALL U1DVR	; Out the UART we go.
		C+	16	JR __4591	
		C+	16	__4591a	equ \$
		C+	16	endif	
		C+	16	ENDMAC	PUMP
		C	17		
		C	18	MVC	query_buffer,query_srv_type,8 ; Put server type in this program menu.
		C+	18	scope	
006686	21 00 69	C+	18	ld hl,query_buffer	
006689	11 85 64	C+	18	ld de,query_srv_type	
00668C	010800	C+	18	ld bc,8	
00668F	EDB0	C+	18	LDIR	
		C+	18	ENDMAC	MVC
		C	19	MVC	query_buffer,nettyp\$,8 ; Put server type in main IPL menu.
		C+	19	scope	
006691	21 00 69	C+	19	ld hl,query_buffer	
006694	11 D4 72	C+	19	ld de,nettyp\$	
006697	010800	C+	19	ld bc,8	
00669A	EDB0	C+	19	LDIR	
		C+	19	ENDMAC	MVC
		C	20		
		C	21	MVC	query_buffer+11,DCT6\$+3,7 ; Move DiskDisk DCT to net drive 6 DCT.
		C+	21	scope	
00669C	21 0B 69	C+	21	ld hl,query_buffer+11	
00669F	11 AF 04	C+	21	ld de,DCT6\$+3	
0066A2	010700	C+	21	ld bc,7	
0066A5	EDB0	C+	21	LDIR	
		C+	21	ENDMAC	MVC
		C	22		
0066A7	3A AF 04	C	23	ld a,(DCT6\$+3)	; Make sure WP off.
0066AA	CBBF	C	24	RES 7,a	; Reset WP.
0066AC	32 AF 04	C	25	ld (DCT6\$+3),a	
		C	26		
0066AF	3A B0 04	C	27	ld a,(DCT6\$+4)	; Drive 1.
0066B2	E6F0	C	28	and 0F0h	; Strip out drive select bits.
0066B4	CBC7	C	29	set 0,a	; Make this drive #1.
0066B6	CBE7	C	30	set 4,a	; Tell world this is an alien controller.
0066B8	CBF7	C	31	set 6,a	; Capable of double density.
0066BA	32 B0 04	C	32	ld (DCT6\$+4),a	
0066BD	AF	C	33	xor a	; Update DCT, its way it is on real model 4.
0066BE	32 B1 04	C	34	ld (DCT6\$+5),a	; Cylinder=0.
		C	35		
		C	36	; Now read track=0 sector=0 of NETdrive.	
		C	37	; Recover directory cylinder pointer.	
		C	38		
		C	39	MVC	query_zeros\$,query_rd_sector+1,5 ; Make sure command lines are all 0's.

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source
		C+	39	scope
0066C1	21 0B 6A	C+	39	ld hl,query_zeros\$
0066C4	11 03 6A	C+	39	ld de,query_rd_sector+1
0066C7	010500	C+	39	ld bc,5
0066CA	EDB0	C+	39	LDIR
		C+	39	ENDMAC MVC
		C	40	
0066CC	110000	C	41	ld de,0 ; 1st sector.
		C	42	HEXDEC de,query_rd_sector+1 ; Convert 16 bit address to ascii in network command.
0066CF	D5E1	C+	42	ld hl,de
0066D1	11 03 6A	C+	42	ld de,query_rd_sector+1
		C+	42	SVC svc_hexdec
0066D4	3E61	C+	42	ld a,svc_hexdec
0066D6	EF	C+	42	restart28
		C+	42	ENDMAC SVC
		C+	42	ENDMAC HEXDEC
		C	43	
0066D7	21 03 6A	C	44	ld hl,query_rd_sector+1 ; Convert all spaces in string to 0's.
		C	45	GOSUB zeros_rpl_space1
		C+	45	scope
0066DA	CD 52 68	C+	45	CALL zeros_rpl_space1
		C+	45	endmac GOSUB
		C	46	
		C	47	
		C	48	PUMP DEVICE.serialnet,query_rd_sector ; Command server to send directory sector.
		C+	48	scope ; A tribute to Dave Shriner - Display Data Corp.
0066DD	21 02 6A	C+	48	LD HL,query_rd_sector ; Pump from this starting point until ETX.asc is found.
		C+	48	if DEVICE.serialnet=0
		C+	48	__4670: LD A,(HL) ; Get byte to send.
		C+	48	INC HL ; pointer=pointer+1
		C+	48	; CP EOT.asc ; End Of Transmission (block)?
		C+	48	; jr Z,\$0a ; Return if finished.
		C+	48	CP ETX.asc ; End Of Text?
		C+	48	jr Z,__4670a ; Return if finished.
		C+	48	CP CR.asc ; Was it a CR?
		C+	48	call z,__4671 ; 1st add a LF to stream.
		C+	48	LD C,A ; Output byte in C.
		C+	48	XOR A ; Set Z flag for put.
		C+	48	CALL U0DVR ; Out the UART we go.
		C+	48	JR __4670
		C+	48	__4671: ld a,LF.asc ; Get a LF.
		C+	48	LD C,A ; Output byte in C.
		C+	48	XOR A ; Set Z flag for put
		C+	48	CALL U0DVR ; Out the UART we go.
		C+	48	ld a,CR.asc ; Now put a CR back in stream.
		C+	48	ret ; Return and print CR.
		C+	48	__4670a endif
		C+	48	if DEVICE.serialnet=1
0066E0	7E	C+	48	__4671: LD A,(HL) ; Get byte to send.
0066E1	23	C+	48	INC HL ; pointer=pointer+1
		C+	48	; CP EOT.asc ; End Of Transmission (block)?
		C+	48	; jr Z,\$1a ; Return if finished.
0066E2	FE03	C+	48	CP ETX.asc ; End Of Text?
0066E4	28 07	C+	48	jr Z,__4671a ; Return if finished.
0066E6	4F	C+	48	LD C,A ; Output byte to C.
0066E7	AF	C+	48	XOR A ; Set Z flag for put.
0066E8	CD F5 11	C+	48	CALL U1DVR ; Out the UART we go.
0066EB	18 F3	C+	48	JR __4671
	000066ED	C+	48	__4671a equ \$
		C+	48	endif



\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl_net_qserv.s
		C+	48	ENDMAC	PUMP
		C	49	UART1.get.bytes	0,query_buffer,query_timeout ; Read directory sector from server, 256 bytes
		C+	49	scope	
0066ED	0600	C+	49	ld b,0	; Input 5 bytes.
0066EF	21 00 69	C+	49	ld hl,query_buffer	; Get address of buffer.
0066F2		C+	49	__4681 UART1.get.byte	query_timeout ; Get byte from UART1. If timeout goto no_pong.
		C+	49	scope	
0066F2	3E01	C+	49	ld a,1	
0066F4	B7	C+	49	or a	; On timeout or error goto vector.
0066F5	37	C+	49	scf	
		C+	49	GOSUB U1DVR	; C flag set & NZ set means PUT.
		C+	49	scope	
0066F6	CD F5 11	C+	49	CALL U1DVR	
		C+	49	endmac	GOSUB
		C+	49	IF_NE_GOSUB	query_timeout ; If nz we had a timeout or other error.
		C+	49	scope	
0066F9	C4 0A 68	C+	49	call nz,query_timeout	
		C+	49	endmac	IF_NE_GOSUB
		C+	49	endmac	UART1.get.byte
0066FC	71	C+	49	ld (hl),c	; Store char.
0066FD	23	C+	49	inc hl	
		C+	49	LOOP __4681	; Loop for count bytes.
		C+	49	scope	
0066FE	10 F2	C+	49	djnz __4681	
		C+	49	endmac	LOOP
		C+	49	endmac	UART1.get.bytes
		C	50		
006700	3A 02 69	C	51	ld a,(query_buffer+2)	; Get pointer to directory & place in DCT1
006703	32 B5 04	C	52	ld (DCT6\$+9),a	; directory pointer, pointing to true directory cylinder.
		C	53		
006706	3A AF 04	C	54	ld a,(DCT6\$+3)	; Convert to byte to hex ascii XX.
		C	55	GOSUB query_cvt_ascii	
		C+	55	scope	
006709	CD 2D 68	C+	55	CALL query_cvt_ascii	
		C+	55	endmac	GOSUB
00670C	78	C	56	ld a,b	
00670D	32 DA 64	C	57	ld (query_dct3\$),a	
006710	79	C	58	ld a,c	
006711	32 DB 64	C	59	ld (query_dct3\$+1),a	
		C	60		
006714	3A B0 04	C	61	ld a,(DCT6\$+4)	; Convert to byte to hex ascii XX.
		C	62	GOSUB query_cvt_ascii	
		C+	62	scope	
006717	CD 2D 68	C+	62	CALL query_cvt_ascii	
		C+	62	endmac	GOSUB
00671A	78	C	63	ld a,b	
00671B	32 E7 64	C	64	ld (query_dct4\$),a	
00671E	79	C	65	ld a,c	
00671F	32 E8 64	C	66	ld (query_dct4\$+1),a	
		C	67		
006722	3A B1 04	C	68	ld a,(DCT6\$+5)	; Convert to byte to hex ascii XX.
		C	69	GOSUB query_cvt_ascii	
		C+	69	scope	
006725	CD 2D 68	C+	69	CALL query_cvt_ascii	
		C+	69	endmac	GOSUB
006728	78	C	70	ld a,b	
006729	32 F4 64	C	71	ld (query_dct5\$),a	
00672C	79	C	72	ld a,c	
00672D	32 F5 64	C	73	ld (query_dct5\$+1),a	
		C	74		

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl_net_qserv.s
006730	3A B2 04	C	75	ld	a,(DCT6\$+6) ; Convert to byte to hex ascii XX.
		C	76	GOSUB	query_cvt_ascii
		C+	76	scope	
006733	CD 2D 68	C+	76	CALL	query_cvt_ascii
		C+	76	endmac	GOSUB
006736	78	C	77	ld	a,b
006737	32 01 65	C	78	ld	(query_dct6\$),a
00673A	79	C	79	ld	a,c
00673B	32 02 65	C	80	ld	(query_dct6\$+1),a
		C	81		
00673E	3A B3 04	C	82	ld	a,(DCT6\$+7) ; Convert to byte to hex ascii XX.
		C	83	GOSUB	query_cvt_ascii
		C+	83	scope	
006741	CD 2D 68	C+	83	CALL	query_cvt_ascii
		C+	83	endmac	GOSUB
006744	78	C	84	ld	a,b
006745	32 0E 65	C	85	ld	(query_dct7\$),a
006748	79	C	86	ld	a,c
006749	32 0F 65	C	87	ld	(query_dct7\$+1),a
		C	88		
00674C	3A B4 04	C	89	ld	a,(DCT6\$+8) ; Convert to byte to hex ascii XX.
		C	90	GOSUB	query_cvt_ascii
		C+	90	scope	
00674F	CD 2D 68	C+	90	CALL	query_cvt_ascii
		C+	90	endmac	GOSUB
006752	78	C	91	ld	a,b
006753	32 1B 65	C	92	ld	(query_dct8\$),a
006756	79	C	93	ld	a,c
006757	32 1C 65	C	94	ld	(query_dct8\$+1),a
		C	95		
00675A	3A B5 04	C	96	ld	a,(DCT6\$+9) ; Convert to byte to hex ascii XX.
		C	97	GOSUB	query_cvt_ascii
		C+	97	scope	
00675D	CD 2D 68	C+	97	CALL	query_cvt_ascii
		C+	97	endmac	GOSUB
006760	78	C	98	ld	a,b
006761	32 28 65	C	99	ld	(query_dct9\$),a
006764	79	C	100	ld	a,c
006765	32 29 65	C	101	ld	(query_dct9\$+1),a
		C	102		
		C	103		; <----->
		C	104		
		C	105		; PUMP DEVICE.console,query_buffer ; Tell operator this is ping program.
		C	106		
006768	3A B0 04	C	107	ld	a,(DCT6\$+4) ; Test if double sided drive?
00676B	57	C	108	ld	d,a
00676C	3A B3 04	C	109	ld	a,(DCT6\$+7) ; Get sectors per track.
00676F	E61F	C	110	AND	1Fh ; Strip out sectors per track only.
006771	3C	C	111	inc	a ; Adjust to actual sectors per track.
006772	CB6A	C	112	bit	5,d
006774	28 02	C	113	jr	z,query_not_dbl ; NO, skip doubling sectors per track.
006776	CB27	C	114	sla	a ; A = A * 2, double sided - double sectors per trk.
006778	57	C	115	query_not_dbl	ld d,a ; Store in D.
006779	3A B5 04	C	116	ld	a,(DCT6\$+9) ; Get directory cylinder.
00677C	5F	C	117	ld	e,a ; Dir cylinder.
00677D	ED5C	C	118	mlt	de ; sector = cylinder * sectors per track.
		C	119		
		C	120	HEXDEC	de,query_rd_sector+1 ; Convert 16 bit address to ascii in network command.
00677F	D5E1	C+	120	ld	hl,de
006781	11 03 6A	C+	120	ld	de,query_rd_sector+1

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl_net_qserv.s
		C+	120		SVC svc_hexdec
006784	3E61	C+	120		ld a,svc_hexdec
006786	EF	C+	120		restart28
		C+	120		ENDMAC SVC
		C+	120		ENDMAC HEXDEC
		C	121		
006787	21 03 6A	C	122		ld hl,query_rd_sector+1 ; Convert all spaces in string to 0's.
		C	123		GOSUB zeros_rpl_space1
		C+	123		scope
00678A	CD 52 68	C+	123		CALL zeros_rpl_space1
		C+	123		endmac GOSUB
		C	124		
		C	125		PUMP DEVICE.serialnet,query_rd_sector ; Command server to send directory sector.
		C+	125		scope ; A tribute to Dave Shriner - Display Data Corp.
00678D	21 02 6A	C+	125		LD HL,query_rd_sector ; Pump from this starting point until ETX.asc is found.
		C+	125		if DEVICE.serialnet=0
		C+	125	__4830:	LD A,(HL) ; Get byte to send.
		C+	125		INC HL ; pointer=pointer+1
		C+	125	;	CP EOT.asc ; End Of Transmission (block)?
		C+	125	;	jr Z,\$0a ; Return if finished.
		C+	125		CP ETX.asc ; End Of Text?
		C+	125		jr Z,__4830a ; Return if finished.
		C+	125		CP CR.asc ; Was it a CR?
		C+	125		call z,__4831 ; 1st add a LF to stream.
		C+	125		LD C,A ; Output byte in C.
		C+	125		XOR A ; Set Z flag for put.
		C+	125		CALL U0DVR ; Out the UART we go.
		C+	125		JR __4830
		C+	125	__4831:	ld a,LF.asc ; Get a LF.
		C+	125		LD C,A ; Output byte in C.
		C+	125		XOR A ; Set Z flag for put
		C+	125		CALL U0DVR ; Out the UART we go.
		C+	125		ld a,CR.asc ; Now put a CR back in stream.
		C+	125		ret ; Return and print CR.
		C+	125	__4830a	endif
		C+	125		if DEVICE.serialnet=1
006790	7E	C+	125	__4831:	LD A,(HL) ; Get byte to send.
006791	23	C+	125		INC HL ; pointer=pointer+1
		C+	125	;	CP EOT.asc ; End Of Transmission (block)?
		C+	125	;	jr Z,\$1a ; Return if finished.
		C+	125		CP ETX.asc ; End Of Text?
006792	FE03	C+	125		jr Z,__4831a ; Return if finished.
006794	28 07	C+	125		LD C,A ; Output byte to C.
006796	4F	C+	125		XOR A ; Set Z flag for put.
006797	AF	C+	125		CALL U1DVR ; Out the UART we go.
006798	CD F5 11	C+	125		JR __4831
00679B	18 F3	C+	125		equ \$
0000679D		C+	125	__4831a	endif
		C+	125		ENDMAC PUMP
		C	126		
		C	127		UART1.get.bytes 0,query_buffer,query_timeout ; Read directory sector from server, 256 by
		C+	127		scope
00679D	0600	C+	127		ld b,0 ; Input 5 bytes.
00679F	21 00 69	C+	127		ld hl,query_buffer ; Get address of buffer.
0067A2		C+	127	__4841	UART1.get.byte query_timeout ; Get byte from UART1. If timeout goto no_pong.
		C+	127		scope
0067A2	3E01	C+	127		ld a,1
0067A4	B7	C+	127		or a ; On timeout or error goto vector.
0067A5	37	C+	127		scf
		C+	127		GOSUB U1DVR ; C flag set & NZ set means PUT.

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl_net_qserv.s
		C+	127		scope
0067A6	CD F5 11	C+	127		CALL U1DVR
		C+	127		endmac GOSUB
		C+	127		IF_NE_GOSUB query_timeout ; If nz we had a timeout or other error.
		C+	127		scope
0067A9	C4 0A 68	C+	127		call nz,query_timeout
		C+	127		endmac IF_NE_GOSUB
		C+	127		endmac UART1.get.byte
0067AC	71	C+	127		ld (hl),c ; Store char.
0067AD	23	C+	127		inc hl
		C+	127		LOOP __4841 ; Loop for count bytes.
		C+	127		scope
0067AE	10 F2	C+	127		djnz __4841
		C+	127		endmac LOOP
		C+	127		endmac UART1.get.bytes
		C	128		
		C	129		MVC query_buffer+0D0h,netvol\$,8 ; Copy volume name to volname in main menu.
		C+	129		scope
0067B0	21 D0 69	C+	129		ld hl,query_buffer+0D0h
0067B3	11 B8 72	C+	129		ld de,netvol\$
0067B6	010800	C+	129		ld bc,8
0067B9	EDB0	C+	129		LDIR
		C+	129		ENDMAC MVC
		C	130		MVC query_buffer+0D8h,netvol\$+9,8 ; Copy volume creation date to main menu.
		C+	130		scope
0067BB	21 D8 69	C+	130		ld hl,query_buffer+0D8h
0067BE	11 C1 72	C+	130		ld de,netvol\$+9
0067C1	010800	C+	130		ld bc,8
0067C4	EDB0	C+	130		LDIR
		C+	130		ENDMAC MVC
		C	131		MVC query_buffer+0D0h,query_volnam,8 ; Copy volume name into this program.
		C+	131		scope
0067C6	21 D0 69	C+	131		ld hl,query_buffer+0D0h
0067C9	11 9C 64	C+	131		ld de,query_volnam
0067CC	010800	C+	131		ld bc,8
0067CF	EDB0	C+	131		LDIR
		C+	131		ENDMAC MVC
		C	132		MVC query_buffer+0D8h,query_volnam+10,8 ; Copy volume creation date.
		C+	132		scope
0067D1	21 D8 69	C+	132		ld hl,query_buffer+0D8h
0067D4	11 A6 64	C+	132		ld de,query_volnam+10
0067D7	010800	C+	132		ld bc,8
0067DA	EDB0	C+	132		LDIR
		C+	132		ENDMAC MVC
		C	133		
		C	134		; Stuff ASCII disk space in main menu.
		C	135		
0067DC	FD21 AC 04	C	136		ld iy,DCT6\$
0067E0	FD5606	C	137		ld d,(iy+6) ; Get max cylinders.
0067E3	14	C	138		inc d
0067E4	FD7E07	C	139		ld a,(iy+7) ; Get config, lower nibble contains sec per trk.
0067E7	E63F	C	140		and 3Fh ; Get lower nibble containing sectors per trk.
0067E9	3C	C	141		inc a
0067EA	FDCB046E	C	142		bit 5,(iy+4) ; Sides = 2?
0067EE	28 02	C	143		jr z,qserv_notdbl ; If so then we double sectors per track.
0067F0	CB27	C	144		sla a ; Sectors per track = sectors per track * 2.
0067F2	5F	C	145	qserv_notdbl	ld e,a ; Setup to multiply track * sectors per track.
0067F3	ED5C	C	146		MLT DE ; we then multiply (# CYL * SEC per CYL) 16b result in DE.
		C	147		
		C	148		; Following instruction changes to NOP if this is not a diskDISK image.

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl_net_qserv.s
		C	149		
0067F5	13	C	150	inc de	; Bump one sector higher to adjust for diskdisk header.
		C	151		
		C	152	; Convert DE to 24 bit by H=D, L=E, E=0	24 BIT--> E:HL.
		C	153	; Convert to packed BCD.	
		C	154	; Convert packed BCD to unpacked.	
		C	155	; Convert unpacked to ASCII.	
		C	156	; Trim leading 0's away.	
		C	157	; Stuff all these results in main menu.	
		C	158		
0067F6	01 DD 72	C	159	ld BC,netsiz\$	; Store decimal ASCII of 24 bit value in main memu.
0067F9	CD 41 70	C	160	call cvt_24_ascii	; Convert 24 bit value to ASCII & stuff in main menu.
		C	161		
		C	162	MVC netsiz\$,query_space_fld,8	; Place of copy of disk space in NETUTILS menu.
		C+	162	scope	
0067FC	21 DD 72	C+	162	ld hl,netsiz\$	
0067FF	11 C7 64	C+	162	ld de,query_space_fld	
006802	010800	C+	162	ld bc,8	
006805	EDB0	C+	162	LDIR	
		C+	162	ENDMAC MVC	
		C	163		
		C	164	GOTO query_exit	
		C+	164	scope	
006807	C3 2A 68	C+	164	jp query_exit	
		C+	164	endmac GOTO	
		C	165		
00680A	E1	C	166	query_timeout pop hl	; Discard timeout trap address.
		C	167	PUMP DEVICE.console,query_timeout\$	; Report to operator query timed out.
		C+	167	scope	; A tribute to Dave Shriner - Display Data Corp.
00680B	21 D8 68	C+	167	LD HL,query_timeout\$	; Pump from this starting point until ETX.asc is found.
		C+	167	if DEVICE.console=0	
00680E	7E	C+	167	__4950: LD A,(HL)	; Get byte to send.
00680F	23	C+	167	INC HL	; pointer=pointer+1
		C+	167	; CP EOT.asc	; End Of Transmission (block)?
		C+	167	; jr Z,\$0a	; Return if finished.
006810	FE03	C+	167	CP ETX.asc	; End Of Text?
006812	28 16	C+	167	jr Z,__4950a	; Return if finished.
006814	FE0D	C+	167	CP CR.asc	; Was it a CR?
006816	CC 20 68	C+	167	call z,__4951	; 1st add a LF to stream.
006819	4F	C+	167	LD C,A	; Output byte in C.
00681A	AF	C+	167	XOR A	; Set Z flag for put.
00681B	CD D9 10	C+	167	CALL U0DVR	; Out the UART we go.
00681E	18 EE	C+	167	JR __4950	
006820	3E0A	C+	167	__4951: ld a,LF.asc	; Get a LF.
006822	4F	C+	167	LD C,A	; Output byte in C.
006823	AF	C+	167	XOR A	; Set Z flag for put
006824	CD D9 10	C+	167	CALL U0DVR	; Out the UART we go.
006827	3E0D	C+	167	ld a,CR.asc	; Now put a CR back in stream.
006829	C9	C+	167	ret	; Return and print CR.
00682A		C+	167	__4950a endif	
		C+	167	if DEVICE.console=1	
		C+	167	__4951: LD A,(HL)	; Get byte to send.
		C+	167	INC HL	; pointer=pointer+1
		C+	167	; CP EOT.asc	; End Of Transmission (block)?
		C+	167	; jr Z,\$1a	; Return if finished.
		C+	167	CP ETX.asc	; End Of Text?
		C+	167	jr Z,__4951a	; Return if finished.
		C+	167	LD C,A	; Output byte to C.
		C+	167	XOR A	; Set Z flag for put.
		C+	167	CALL U1DVR	; Out the UART we go.

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl_net_qserv.s
		C+	167	JR	__4951
		C+	167	__4951a	equ \$
		C+	167	endif	
		C+	167	ENDMAC	PUMP
00682A	3EFF	C	168	query_exit	LD a,0ffh
00682C		C	169	query_exit1	RETURN
		C+	169	scope	
00682C	C9	C+	169	ret	; Return from call.
		C+	169	endmac	RETURN
		C	170		
00682D	4F	C	171	query_cvt_asciild	c,a
00682E	E6F0	C	172	and	0f0h ; Strip out upper 4 bits.
006830	CB1F	C	173	rr	a
006832	CB1F	C	174	rr	a
006834	CB1F	C	175	rr	a
006836	CB1F	C	176	rr	a
006838	E60F	C	177	and	0fh
00683A	C630	C	178	add	'0'
00683C	FE3A	C	179	cp	':' ; Is it >9
		C	180	IF_GT_GOTO	query_no_more2
		C+	180	scope	
00683E	DA 43 68	C+	180	jp	c,query_no_more2
		C+	180	endmac	IF_GT_GOTO
006841	C607	C	181	add	7 ; Adjust for A-F
006843	47	C	182	query_no_more2	ld b,a
006844	79	C	183	ld	a,c
006845	E60F	C	184	and	0fh
006847	C630	C	185	add	'0'
006849	FE3A	C	186	cp	':' ; Is it >9
		C	187	IF_GT_GOTO	query_no_more1
		C+	187	scope	
00684B	DA 50 68	C+	187	jp	c,query_no_more1
		C+	187	endmac	IF_GT_GOTO
00684E	C607	C	188	add	7 ; Adjust for A-F
006850	4F	C	189	query_no_more1	ld c,a
		C	190	RETURN	
		C+	190	scope	
006851	C9	C+	190	ret	; Return from call.
		C+	190	endmac	RETURN
		C	191		
006852	1630	C	192	zeros_rpl_space1	ld d,'0' ; 0 to replace space.
006854	3E20	C	193	ld	a,' ' ; Test for space.
006856	0605	C	194	ld	b,5 ; Test 5 char string.
006858	BE	C	195	qry_rpl_spaces1	cp (hl)
006859	20 01	C	196	jr	nz,qry_by11 ; Replace all spaces with 0's.
00685B	72	C	197	ld	(hl),d
00685C	23	C	198	qry_by11	inc hl
		C	199	LOOP	qry_rpl_spaces1
		C+	199	scope	
00685D	10 F9	C+	199	djnz	qry_rpl_spaces1
		C+	199	endmac	LOOP
		C	200	RETURN	
		C+	200	scope	
00685F	C9	C+	200	ret	; Return from call.
		C+	200	endmac	RETURN
		C	201		
006860	20517565 72792073	C	202	query_screen\$	DB " Query server for network volume.",CR.asc,CR.asc,ETX.asc
006868	65727665 7220666F				
006870	72206E65 74776F72				
006878	6B20766F 6C756D65				

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl_net_qserv.s
006880	2E0D0D03				
		C	203		
006884	2053656E 64696E67	C	204	query_start\$ db	" Sending server query command.",CR.asc
00688C	20736572 76657220				
006894	71756572 7920636F				
00689C	6D6D616E 642E0D				
0068A3	20576169 74696E67	C	205	db	" Waiting for server to reply...",CR.asc,ETX.asc
0068AB	20666F72 20736572				
0068B3	76657220 746F2072				
0068BB	65706C79 2E2E2E0D				
0068C3	03				
		C	206		
0068C4	20536572 76657220	C	207	query_replied\$ db	" Server replied...",CR.asc,ETX.asc
0068CC	7265706C 6965642E				
0068D4	2E2E0D03				
0068D8	20536572 76657220	C	208	query_timeout\$ db	" Server did not reply to query.",CR.asc,ETX.asc
0068E0	64696420 6E6F7420				
0068E8	7265706C 7920746F				
0068F0	20717565 72792E0D				
0068F8	03				
		C	209		
0068F9	40726864 720A03	C	210	query_command\$ db	"@rhdr",LF.asc,ETX.asc ; Get header for volume.
		C	211		
006900		C	212	query_buffer ds	256 ; Buffer to hold query i/o.
006A00	0D03	C	213	db	CR.asc,ETX.asc
		C	214		
006A02	3C303030 30300A0D	C	215	query_rd_sector db	"<00000",LF.asc,CR.asc,ETX.asc
006A0A	03				
		C	216		
006A0B	30303030 30	C	217	query_zeros\$ db	"00000"
		C	218		
		C	219		
		C	220	;query_8space db	" "
006A10	03	C	221	db	ETX.asc ; End of screen.
		C	0	include	"ipl-SLICE.s" ; Manage SLICE tables.
	00006A11	C	1	editslice equ	\$
		C	2	GOSUB	scrolloff ; Start with a fresh clean screen.
		C+	2	scope	
006A11	CD ED 6E	C+	2	CALL	scrolloff
		C+	2	endmac	GOSUB
		C	3	PUMP	DEVICE.console,slicemenu ; Pump out message to a fresh clear screen.
		C+	3	scope	; A tribute to Dave Shriner - Display Data Corp.
006A14	21 39 6A	C+	3	LD	HL,slicemenu ; Pump from this starting point until ETX.asc is found.
		C+	3	if	DEVICE.console=0
006A17	7E	C+	3	LD	A,(HL) ; Get byte to send.
006A18	23	C+	3	INC	HL ; pointer=pointer+1
		C+	3	CP	EOT.asc ; End Of Transmission (block)?
		C+	3	jr	Z,\$0a ; Return if finished.
		C+	3	CP	ETX.asc ; End Of Text?
006A19	FE03	C+	3	jr	Z,___5030a ; Return if finished.
006A1B	28 16	C+	3	CP	CR.asc ; Was it a CR?
006A1D	FE0D	C+	3	call	z,___5031 ; 1st add a LF to stream.
006A1F	CC 29 6A	C+	3	LD	C,A ; Output byte in C.
006A22	4F	C+	3	XOR	A ; Set Z flag for put.
006A23	AF	C+	3	CALL	U0DVR ; Out the UART we go.
006A24	CD D9 10	C+	3	JR	___5030
006A27	18 EE	C+	3	LD	a,LF.asc ; Get a LF.
006A29	3E0A	C+	3	LD	C,A ; Output byte in C.
006A2B	4F	C+	3	XOR	A ; Set Z flag for put
006A2C	AF	C+	3	CALL	U0DVR ; Out the UART we go.
006A2D	CD D9 10	C+	3		

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-SLICE.s
006A30	3E0D	C+	3	ld	a,CR.asc ; Now put a CR back in stream.
006A32	C9	C+	3	ret	; Return and print CR.
006A33		C+	3	__5030a	endif
		C+	3	if	DEVICE.console=1
		C+	3	__5031:	LD A,(HL) ; Get byte to send.
		C+	3	INC	HL ; pointer=pointer+1
		C+	3	;	CP EOT.asc ; End Of Transmission (block)?
		C+	3	;	jr Z,\$1a ; Return if finished.
		C+	3		CP ETX.asc ; End Of Text?
		C+	3	jr	Z,__5031a ; Return if finished.
		C+	3	LD	C,A ; Output byte to C.
		C+	3	XOR	A ; Set Z flag for put.
		C+	3	CALL	U1DVR ; Out the UART we go.
		C+	3	JR	__5031
		C+	3	__5031a	equ \$
		C+	3		endif
		C+	3	ENDMAC	PUMP
		C	4	GOSUB	getchar ; Pause until key hit.
		C+	4	scope	
006A33	CD 7E 6F	C+	4	CALL	getchar
		C+	4	endmac	GOSUB
		C	5	GOTO	main_ipl_menu ; Back to main menu.
		C+	5	scope	
006A36	C3 00 48	C+	5	jp	main_ipl_menu
		C+	5	endmac	GOTO
		C	6		
	00006C1A	C	7	sliceitem1	equ sl1+11
	00006C4E	C	8	sliceitem2	equ sl2+11
	00006C82	C	9	sliceitem3	equ sl3+11
	00006CB6	C	10	sliceitem4	equ sl4+11
	00006CEA	C	11	sliceitem5	equ sl5+11
	00006D1E	C	12	sliceitem6	equ sl6+11
	00006D52	C	13	sliceitem7	equ sl7+11
	00006D86	C	14	sliceitem8	equ sl8+11
	00006DBA	C	15	sliceitem9	equ sl9+11
		C	16		
		C	17		
006A39	0D	C	18	slicemenu db	CR.asc
006A3A	0D	C	19	db	CR.asc
006A3B	2B2D2D2D 2D2D2D2D	C	20	DB	"-----+",CR.asc
006A43	2D2D2D2D 2D2D2D2D				
006A4B	2D2D2D2D 2D2D2D2D				
006A53	2D2D2D2D 2D2D2D2D				
006A5B	2D2D2D2D 2D2D2D2D				
006A63	2D2D2D2D 2D2D2D2D				
006A6B	2D2D2B0D				
006A6F	7C202020 20202020	C	21	DB	"  S L I C E  ",CR.asc
006A77	20202020 20202020				
006A7F	20202020 20205320				
006A87	4C204920 43204520				
006A8F	20202020 20202020				
006A97	20202020 20202020				
006A9F	20207C0D				
006AA3	7C202020 20202020	C	22	DB	"  T A B L E S  ",CR.asc
006AAB	20202020 20202020				
006AB3	20202020 20542041				
006ABB	2042204C 20452053				
006AC3	20202020 20202020				
006ACB	20202020 20202020				
006AD3	20207C0D				



```
< SYS0 IPL - DOS-initialization code & welcome message >
```

PC	Object		I	Line	Source	..\SYS0_IPL\ipl-SLICE.s								
006AD7	7C202020	20202020	C	23		db	"	-	-	-	-	-	-	" ,CR.asc
006ADF	20202020	20202020												
006AE7	20202020	2D202D20												
006AEF	2D202D20	2D202D20												
006AF7	2D202020	20202020												
006AFF	20202020	20202020												
006B07	20207C0D													
006B0B	7C202020	20202020	C	24		db	"							" ,CR.asc
006B13	20202020	20202020												
006B1B	20202020	20202020												
006B23	20202020	20202020												
006B2B	20202020	20202020												
006B33	20202020	20202020												
006B3B	20207C0D													
006B3F	7C20536C	69636520	C	25		db	"	Slice	Name	Start	Length			" ,CR.asc
006B47	20202020	4E616D65												
006B4F	20202020	20202053												
006B57	74617274	20202020												
006B5F	204C656E	67746820												
006B67	20202020	20202020												
006B6F	20207C0D													
006B73	7C205F5F	5F5F5F5F	C	26		db	"							" ,CR.asc
006B7B	5F5F5F5F	5F5F5F5F												
006B83	5F5F5F5F	5F5F5F5F												
006B8B	5F5F5F5F	5F5F5F5F												
006B93	5F5F5F5F	5F5F5F5F												
006B9B	5F5F5F5F	5F5F5F5F												
006BA3	5F207C0D													
006BA7	7C202020	20202020	C	27		db	"							" ,CR.asc
006BAF	20202020	20202020												
006BB7	20202020	20202020												
006BBF	20202020	20202020												
006BC7	20202020	20202020												
006BCF	20202020	20202020												
006BD7	20207C0D													
006BDB	7C20303A	20536C69	C	28	sl0	db	"	0: Slice 0 system		0001:0000h	0000:0100h			" ,CR.asc
006BE3	63652030	20737973												
006BEB	74656D20	20303030												
006BF3	313A3030	30306820												
006BFB	30303030	3A303130												
006C03	30682020	20202020												
006C0B	20207C0D													
006C0F	7C20313A	20536C69	C	29	sl1	db	"	1: Slice 1 user		0001:0100h	0000:0100h			" ,CR.asc
006C17	63652031	20757365												
006C1F	72202020	2030303												

```
< SYS0 IPL - DOS-initialization code & welcome message >
```

PC	Object		I	Line	Source ..\SYS0_IPL\ipl-SLICE.s				
006C97	30303030	3A303030							
006C9F	30682020	20202020							
006CA7	20207C0D								
006CAB	7C20343A	20202020	C	32	sl4	DB	"  4:	0000:0000h 0000:0000h	" ,CR.asc
006CB3	20202020	20202020							
006CBB	20202020	20303030							
006CC3	303A3030	30306820							
006CCB	30303030	3A303030							
006CD3	30682020	20202020							
006CDB	20207C0D								
006CDF	7C20353A	20202020	C	33	sl5	DB	"  5:	0000:0000h 0000:0000h	" ,CR.asc
006CE7	20202020	20202020							
006CEF	20202020	20303030							
006CF7	303A3030	30306820							
006CFF	30303030	3A303030							
006D07	30682020	20202020							
006D0F	20207C0D								
006D13	7C20363A	20202020	C	34	sl6	DB	"  6:	0000:0000h 0000:0000h	" ,CR.asc
006D1B	20202020	20202020							
006D23	20202020	20303030							
006D2B	303A3030	30306820							
006D33	30303030	3A303030							
006D3B	30682020	20202020							
006D43	20207C0D								
006D47	7C20373A	20202020	C	35	sl7	DB	"  7:	0000:0000h 0000:0000h	" ,CR.asc
006D4F	20202020	20202020							
006D57	20202020	20303030							
006D5F	303A3030	30306820							
006D67	30303030	3A303030							
006D6F	30682020	20202020							
006D77	20207C0D								
006D7B	7C20383A	20202020	C	36	sl8	DB	"  8:	0000:0000h 0000:0000h	" ,CR.asc
006D83	20202020	20202020							
006D8B	20202020	20303030							
006D93	303A3030	30306820							
006D9B	30303030	3A303030							
006DA3	30682020	20202020							
006DAB	20207C0D								
006DAF	7C20393A	20202020	C	37	sl9	DB	"  9:	0000:0000h 0000:0000h	" ,CR.asc
006DB7	20202020	20202020							
006DBF	20202020	20303030							
006DC7	303A3030	30306820							
006DCF	30303030	3A303030							
006DD7	30682020	20202020							
006DDF	20207C0D								
006DE3	2B2D2D2D	2D2D2D2D	C	38		DB	" +-----+ "		,CR.asc
006DEB	2D2D2D2D	2D2D2D2D							
006DF3	2D2D2D2D	2D2D2D2D							
006DFB	2D2D2D2D	2D2D2D2D							
006E03	2D2D2D2D	2D2D2D2D							
006E0B	2D2D2D2D	2D2D2D2D							
006E13	2D2D2B0D								
006E17	03		C	39		db			ETX.asc
			C	40					
	000								

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-subroutines.s
		C	2	;-----< S U B R O U T I N E S >-----	
		C	3		
		C	4		
006E30	3A 79 00	C	5	cpu_dly	ld a, (PFLAG\$) ; Which processor we running on?
006E33	CB2F	C	6	sra	a
006E35	FE01	C	7	cp	01h ; Is this a 91?
		C	8	IF_EQ_BRANCH	cpu_dly_91
		C+	8	scope	
006E37	28 57	C+	8	jz	z, cpu_dly_91
		C+	8	endmac	IF_EQ_BRANCH
		C	9		
006E39		C	10	cpu_dly_92	FOR 1,19,1 ; Half the delay for a f92, its slower.
		C+	10	SCOPE	
006E39	ED4B 45 6E	C+	10	ld	bc, (__5071)
006E3D	ED43 43 6E	C+	10	ld	(__5074), bc
006E41	18 08	C+	10	JR	__5075
006E43	0000	C+	10	__5074	DW 0 ; I hold count when running (+8).
006E45	0100	C+	10	__5071	DW 1 ; Start count from xx (+10).
006E47	1300	C+	10	__5072	DW 19 ; Stop when equal or greater than STOP (+12).
006E49	0100	C+	10	__5073	DW 1 ; Step by (+14).
	00006E4B	C+	10	__5075	equ \$
		C+	10	ENDMAC	FOR
006E4B		C	11	cpu_dly_92a	FOR 1,30000,1
		C+	11	SCOPE	
006E4B	ED4B 57 6E	C+	11	ld	bc, (__5081)
006E4F	ED43 55 6E	C+	11	ld	(__5084), bc
006E53	18 08	C+	11	JR	__5085
006E55	0000	C+	11	__5084	DW 0 ; I hold count when running (+8).
006E57	0100	C+	11	__5081	DW 1 ; Start count from xx (+10).
006E59	3075	C+	11	__5082	DW 30000 ; Stop when equal or greater than STOP (+12).
006E5B	0100	C+	11	__5083	DW 1 ; Step by (+14).
	00006E5D	C+	11	__5085	equ \$
		C+	11	ENDMAC	FOR
		C	12	NEXT	cpu_dly_92a
		C+	12	scope	
006E5D	2A 55 6E	C+	12	ld	hl, (cpu_dly_92a+10) ; Counter.
006E60	ED4B 5B 6E	C+	12	ld	bc, (cpu_dly_92a+16) ; STEP
006E64	09	C+	12	add	hl, bc
006E65	38 0F	C+	12	JR	C, __5090 ; Overflow, exceeds limit of counter, were done.
006E67	E5	C+	12	PUSH	HL
006E68	ED4B 59 6E	C+	12	LD	BC, (cpu_dly_92a+14) ; Get limit.
006E6C	A7	C+	12	and	A ; Clear carry flag.
006E6D	ED42	C+	12	SBC	HL, BC ; count-limit.
006E6F	C1	C+	12	POP	bc
006E70	CA 4F 6E	C+	12	JP	z, cpu_dly_92a+4 ; limit=counter? Go around again.
006E73	FA 4F 6E	C+	12	JP	m, cpu_dly_92a+4 ; limit>counter? Yes, go around again.
	00006E76	C+	12	__5090:	equ \$
		C+	12	endmac	NEXT
		C	13	NEXT	cpu_dly_92
		C+	13	scope	
006E76	2A 43 6E	C+	13	ld	hl, (cpu_dly_92+10) ; Counter.
006E79	ED4B 49 6E	C+	13	ld	bc, (cpu_dly_92+16) ; STEP
006E7D	09	C+	13	add	hl, bc
006E7E	38 0F	C+	13	JR	C, __5100 ; Overflow, exceeds limit of counter, were done.
006E80	E5	C+	13	PUSH	HL
006E81	ED4B 47 6E	C+	13	LD	BC, (cpu_dly_92+14) ; Get limit.
006E85	A7	C+	13	and	A ; Clear carry flag.
006E86	ED42	C+	13	SBC	HL, BC ; count-limit.
006E88	C1	C+	13	POP	bc
006E89	CA 3D 6E	C+	13	JP	z, cpu_dly_92+4 ; limit=counter? Go around again.

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-subroutines.s
006E8C	FA 3D 6E	C+	13	JP	m,cpu_dly_92+4 ; limit>counter? Yes, go around again.
	00006E8F	C+	13	__5100:	equ \$
		C+	13	endmac	NEXT
		C	14	RETURN	
		C+	14	scope	
006E8F	C9	C+	14	ret	; Return from call.
		C+	14	endmac	RETURN
		C	15		
006E90		C	16	cpu_dly_91	FOR 1,37,1
		C+	16	SCOPE	
006E90	ED4B 9C 6E	C+	16	ld	bc,(__5121)
006E94	ED43 9A 6E	C+	16	ld	(__5124),bc
006E98	18 08	C+	16	JR	__5125
006E9A	0000	C+	16	__5124	DW 0 ; I hold count when running (+8).
006E9C	0100	C+	16	__5121	DW 1 ; Start count from xx (+10).
006E9E	2500	C+	16	__5122	DW 37 ; Stop when equal or greater than STOP (+12).
006EA0	0100	C+	16	__5123	DW 1 ; Step by (+14).
	00006EA2	C+	16	__5125	equ \$
		C+	16	ENDMAC	FOR
006EA2		C	17	cpu_dly_91a	FOR 1,30000,1
		C+	17	SCOPE	
006EA2	ED4B AE 6E	C+	17	ld	bc,(__5131)
006EA6	ED43 AC 6E	C+	17	ld	(__5134),bc
006EAA	18 08	C+	17	JR	__5135
006EAC	0000	C+	17	__5134	DW 0 ; I hold count when running (+8).
006EAE	0100	C+	17	__5131	DW 1 ; Start count from xx (+10).
006EB0	3075	C+	17	__5132	DW 30000 ; Stop when equal or greater than STOP (+12).
006EB2	0100	C+	17	__5133	DW 1 ; Step by (+14).
	00006EB4	C+	17	__5135	equ \$
		C+	17	ENDMAC	FOR
		C	18	NEXT	cpu_dly_91a
		C+	18	scope	
006EB4	2A AC 6E	C+	18	ld	hl,(cpu_dly_91a+10) ; Counter.
006EB7	ED4B B2 6E	C+	18	ld	bc,(cpu_dly_91a+16) ; STEP
006EBB	09	C+	18	add	hl,bc
006EBC	38 0F	C+	18	JR	C,__5140 ; Overflow, exceeds limit of counter, were done.
006EBE	E5	C+	18	PUSH	HL
006EBF	ED4B B0 6E	C+	18	LD	BC,(cpu_dly_91a+14) ; Get limit.
006EC3	A7	C+	18	and	A ; Clear carry flag.
006EC4	ED42	C+	18	SBC	HL,BC ; count-limit.
006EC6	C1	C+	18	POP	bc
006EC7	CA A6 6E	C+	18	JP	z,cpu_dly_91a+4 ; limit=counter? Go around again.
006ECA	FA A6 6E	C+	18	JP	m,cpu_dly_91a+4 ; limit>counter? Yes, go around again.
	00006ECD	C+	18	__5140:	equ \$
		C+	18	endmac	NEXT
		C	19		
006ECD	ED38C5	C	20	IN0	A,(UART0_LSR) ; Get status of data waiting?
006ED0	E601	C	21	AND	UART_DR ; Char waiting?
		C	22	IF_NE_RETURN	
		C+	22	scope	
006ED2	C0	C+	22	ret	nz
		C+	22	endmac	IF_NE_RETURN
		C	23	NEXT	cpu_dly_91
		C+	23	scope	
006ED3	2A 9A 6E	C+	23	ld	hl,(cpu_dly_91+10) ; Counter.
006ED6	ED4B A0 6E	C+	23	ld	bc,(cpu_dly_91+16) ; STEP
006EDA	09	C+	23	add	hl,bc
006EDB	38 0F	C+	23	JR	C,__5160 ; Overflow, exceeds limit of counter, were done.
006EDD	E5	C+	23	PUSH	HL
006EDE	ED4B 9E 6E	C+	23	LD	BC,(cpu_dly_91+14) ; Get limit.

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-subroutines.s
006EE2	A7	C+	23	and	A ; Clear carry flag.
006EE3	ED42	C+	23	SBC	HL,BC ; count-limit.
006EE5	C1	C+	23	POP	bc
006EE6	CA 94 6E	C+	23	JP	z,cpu_dly_91+4 ; limit=counter? Go around again.
006EE9	FA 94 6E	C+	23	JP	m,cpu_dly_91+4 ; limit>counter? Yes, go around again.
	00006EEC	C+	23	__5160:	equ \$
		C+	23		endmac NEXT
		C	24		RETURN
		C+	24		scope
006EEC	C9	C+	24	ret	; Return from call.
		C+	24		endmac RETURN
		C	25		
006EED		C	26	scrolloff	FOR 1,25,1
		C+	26		SCOPE
006EED	ED4B F9 6E	C+	26	ld	bc,(__5181)
006EF1	ED43 F7 6E	C+	26	ld	(__5184),bc
006EF5	18 08	C+	26	JR	__5185
006EF7	0000	C+	26	__5184	DW 0 ; I hold count when running (+8).
006EF9	0100	C+	26	__5181	DW 1 ; Start count from xx (+10).
006EFB	1900	C+	26	__5182	DW 25 ; Stop when equal or greater than STOP (+12).
006EFD	0100	C+	26	__5183	DW 1 ; Step by (+14).
	00006EFF	C+	26	__5185	equ \$
		C+	26		ENDMAC FOR
		C	27	PUMP	DEVICE.console,crlf
		C+	27		scope ; A tribute to Dave Shriner - Display Data Corp.
006EFF	21 FA 86	C+	27	LD	HL,crlf ; Pump from this starting point until ETX.asc is found.
		C+	27		if DEVICE.console=0
006F02	7E	C+	27	__5190:	LD A,(HL) ; Get byte to send.
006F03	23	C+	27	INC	HL ; pointer=pointer+1
		C+	27	; CP	EOT.asc ; End Of Transmission (block)?
		C+	27	; jr	Z,\$0a ; Return if finished.
006F04	FE03	C+	27	CP	ETX.asc ; End Of Text?
006F06	28 16	C+	27	jr	Z,__5190a ; Return if finished.
006F08	FE0D	C+	27	CP	CR.asc ; Was it a CR?
006F0A	CC 14 6F	C+	27	call	z,__5191 ; 1st add a LF to stream.
006F0D	4F	C+	27	LD	C,A ; Output byte in C.
006F0E	AF	C+	27	XOR	A ; Set Z flag for put.
006F0F	CD D9 10	C+	27	CALL	U0DVR ; Out the UART we go.
006F12	18 EE	C+	27	JR	__5190
006F14	3E0A	C+	27	__5191:	ld a,LF.asc ; Get a LF.
006F16	4F	C+	27	LD	C,A ; Output byte in C.
006F17	AF	C+	27	XOR	A ; Set Z flag for put
006F18	CD D9 10	C+	27	CALL	U0DVR ; Out the UART we go.
006F1B	3E0D	C+	27	ld	a,CR.asc ; Now put a CR back in stream.
006F1D	C9	C+	27	ret	; Return and print CR.
006F1E		C+	27	__5190a	endif
		C+	27		if DEVICE.console=1
		C+	27	__5191:	LD A,(HL) ; Get byte to send.
		C+	27	INC	HL ; pointer=pointer+1
		C+	27	; CP	EOT.asc ; End Of Transmission (block)?
		C+	27	; jr	Z,\$1a ; Return if finished.
		C+	27	CP	ETX.asc ; End Of Text?
		C+	27	jr	Z,__5191a ; Return if finished.
		C+	27	LD	C,A ; Output byte to C.
		C+	27	XOR	A ; Set Z flag for put.
		C+	27	CALL	U1DVR ; Out the UART we go.
		C+	27	JR	__5191
		C+	27	__5191a	equ \$
		C+	27		endif
		C+	27		ENDMAC PUMP

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-subroutines.s
		C	28		NEXT scrolloff
		C+	28		scope
006F1E	2A F7 6E	C+	28	ld	hl,(scrolloff+10) ; Counter.
006F21	ED4B FD 6E	C+	28	ld	bc,(scrolloff+16) ; STEP
006F25	09	C+	28	add	hl,bc
006F26	38 0F	C+	28	JR	C, __5200 ; Overflow, exceeds limit of counter, were done.
006F28	E5	C+	28	PUSH	HL
006F29	ED4B FB 6E	C+	28	LD	BC,(scrolloff+14) ; Get limit.
006F2D	A7	C+	28	and	A ; Clear carry flag.
006F2E	ED42	C+	28	SBC	HL,BC ; count-limit.
006F30	C1	C+	28	POP	bc
006F31	CA F1 6E	C+	28	JP	z,scrolloff+4 ; limit=counter? Go around again.
006F34	FA F1 6E	C+	28	JP	m,scrolloff+4 ; limit>counter? Yes, go around again.
	00006F37	C+	28	__5200:	equ \$
		C+	28		endmac NEXT
		C	29		RETURN
		C+	29		scope
006F37	C9	C+	29	ret	; Return from call.
		C+	29		endmac RETURN
		C	30		
006F38		C	31	GET.yn	GOSUB getchar ; Get users one character response.
		C+	31		scope
006F38	CD 7E 6F	C+	31		CALL getchar
		C+	31		endmac GOSUB
		C	32		
006F3B	FE7F	C	33	cp	DEL.asc ; User hit RUBOUT (shift del/delete)?
		C	34		IF_EQ_RETURN ; If yes return this key if yes.
		C+	34		scope
006F3D	C8	C+	34	ret	z
		C+	34		endmac IF_EQ_RETURN
006F3E	FE1B	C	35	cp	ESC.asc ; User hit escape key?
		C	36		IF_EQ_RETURN ; If yes return this key if yes.
		C+	36		scope
006F40	C8	C+	36	ret	z
		C+	36		endmac IF_EQ_RETURN
006F41	FE01	C	37	cp	SOH.asc ; Test if user hit ctrl^a or HOME?
		C	38		IF_EQ_RETURN ; If yes return this key if yes.
		C+	38		scope
006F43	C8	C+	38	ret	z
		C+	38		endmac IF_EQ_RETURN
		C	39		
006F44	FE59	C	40	cp	'Y' ; Y reply?
		C	41		IF_EQ_BRANCH cngYy ; Change to lower case.
		C+	41		scope
006F46	28 0E	C+	41	jr	z,cngYy
		C+	41		endmac IF_EQ_BRANCH
006F48	FE4E	C	42	cp	'N' ; N reply?
		C	43		IF_EQ_BRANCH cngNn ; Change to lower case.
		C+	43		scope
006F4A	28 0E	C+	43	jr	z,cngNn
		C+	43		endmac IF_EQ_BRANCH
006F4C	FE79	C	44	cp	'y' ; y reply?
		C	45		IF_EQ_BRANCH clnup
		C+	45		scope
006F4E	28 0C	C+	45	jr	z,clnup
		C+	45		endmac IF_EQ_BRANCH
006F50	FE6E	C	46	cp	'n' ; n reply?
		C	47		IF_EQ_BRANCH clnup
		C+	47		scope
006F52	28 08	C+	47	jr	z,clnup

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-subroutines.s
		C+	47	endmac	IF_EQ_BRANCH
		C	48	BRANCH	GET.yn ; Not a yes or no so get another key.
		C+	48	scope	
006F54	18 E2	C+	48	jr	GET.yn
		C+	48	endmac	BRANCH
006F56	3E79	C	49	cngYy ld	a,'y' ; Change to lower case.
		C	50	BRANCH	clnup
		C+	50	scope	
006F58	18 02	C+	50	jr	clnup
		C+	50	endmac	BRANCH
		C	51		
006F5A	3E6E	C	52	cngNn ld	a,'n' ; Change to lower case.
006F5C	F5	C	53	clnup PUSH	AF
		C	54	PUMP	DEVICE.console,crlf+1
		C+	54	scope	; A tribute to Dave Shriner - Display Data Corp.
006F5D	21 FB 86	C+	54	LD	HL,crlf+1 ; Pump from this starting point until ETX.asc is found.
		C+	54	if	DEVICE.console=0
006F60	7E	C+	54	__5320: LD	A,(HL) ; Get byte to send.
006F61	23	C+	54	INC	HL ; pointer=pointer+1
		C+	54	; CP	EOT.asc ; End Of Transmission (block)?
		C+	54	; jr	Z,\$0a ; Return if finished.
006F62	FE03	C+	54	CP	ETX.asc ; End Of Text?
006F64	28 16	C+	54	jr	Z,__5320a ; Return if finished.
006F66	FE0D	C+	54	CP	CR.asc ; Was it a CR?
006F68	CC 72 6F	C+	54	call	z,__5321 ; 1st add a LF to stream.
006F6B	4F	C+	54	LD	C,A ; Output byte in C.
006F6C	AF	C+	54	XOR	A ; Set Z flag for put.
006F6D	CD D9 10	C+	54	CALL	U0DVR ; Out the UART we go.
006F70	18 EE	C+	54	JR	__5320
006F72	3E0A	C+	54	__5321: ld	a,LF.asc ; Get a LF.
006F74	4F	C+	54	LD	C,A ; Output byte in C.
006F75	AF	C+	54	XOR	A ; Set Z flag for put
006F76	CD D9 10	C+	54	CALL	U0DVR ; Out the UART we go.
006F79	3E0D	C+	54	ld	a,CR.asc ; Now put a CR back in stream.
006F7B	C9	C+	54	ret	; Return and print CR.
006F7C		C+	54	__5320a endif	
		C+	54	if	DEVICE.console=1
		C+	54	__5321: LD	A,(HL) ; Get byte to send.
		C+	54	INC	HL ; pointer=pointer+1
		C+	54	; CP	EOT.asc ; End Of Transmission (block)?
		C+	54	; jr	Z,\$1a ; Return if finished.
		C+	54	CP	ETX.asc ; End Of Text?
		C+	54	jr	Z,__5321a ; Return if finished.
		C+	54	LD	C,A ; Output byte to C.
		C+	54	XOR	A ; Set Z flag for put.
		C+	54	CALL	U1DVR ; Out the UART we go.
		C+	54	JR	__5321
		C+	54	__5321a equ	\$
		C+	54	endif	
		C+	54	ENDMAC	PUMP
006F7C	F1	C	55	POP	af
		C	56	RETURN	
		C+	56	scope	
006F7D	C9	C+	56	ret	; Return from call.
		C+	56	endmac	RETURN
		C	57		
006F7E	ED38C5	C	58	getchar IN0	A,(UART0_LSR) ; Get status of data waiting?
006F81	E601	C	59	AND	UART_DR ; Char waiting?
		C	60	IF_EQ_BRANCH	getchar ; Make NZ, NO char waiting.
		C+	60	scope	

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-subroutines.s
006F83	28 F9	C+	60		jr z, getchar
		C+	60		endmac IF_EQ_BRANCH
006F85	ED38C0	C	61		IN0 A, (UART0_RBR) ; Get byte from UART0.
006F88	BF	C	62		cp a
		C	63		RETURN
		C+	63		scope
006F89	C9	C+	63		ret ; Return from call.
		C+	63		endmac RETURN
		C	64		
006F8A	0600	C	65	delay	ld b, 0 ; Delay a bit.
006F8C	110000	C	66		ld de, 0h
006F8F	1B	C	67	dly_lp	dec de
006F90	7A	C	68		ld a, d
006F91	B3	C	69		or e
006F92	20 FB	C	70		jr nz, dly_lp
006F94	10 F6	C	71		djnz delay+2
		C	72		RETURN
		C+	72		scope
006F96	C9	C+	72		ret ; Return from call.
		C+	72		endmac RETURN
		C	73		
006F97		C	74	mounterr0PUMP	DEVICE.console, mounterrmsg0
		C+	74		scope ; A tribute to Dave Shriner - Display Data Corp.
006F97	21 4A 7B	C+	74		LD HL, mounterrmsg0 ; Pump from this starting point until ETX.asc is found.
		C+	74		if DEVICE.console=0
006F9A	7E	C+	74	__5370:	LD A, (HL) ; Get byte to send.
006F9B	23	C+	74		INC HL ; pointer=pointer+1
		C+	74		; CP EOT.asc ; End Of Transmission (block)?
		C+	74		; jr Z, \$0a ; Return if finished.
006F9C	FE03	C+	74		CP ETX.asc ; End Of Text?
006F9E	28 16	C+	74		jr Z, __5370a ; Return if finished.
006FA0	FE0D	C+	74		CP CR.asc ; Was it a CR?
006FA2	CC AC 6F	C+	74		call z, __5371 ; 1st add a LF to stream.
006FA5	4F	C+	74		LD C, A ; Output byte in C.
006FA6	AF	C+	74		XOR A ; Set Z flag for put.
006FA7	CD D9 10	C+	74		CALL U0DVR ; Out the UART we go.
006FAA	18 EE	C+	74		JR __5370
006FAC	3E0A	C+	74	__5371:	ld a, LF.asc ; Get a LF.
006FAE	4F	C+	74		LD C, A ; Output byte in C.
006FAF	AF	C+	74		XOR A ; Set Z flag for put
006FB0	CD D9 10	C+	74		CALL U0DVR ; Out the UART we go.
006FB3	3E0D	C+	74		ld a, CR.asc ; Now put a CR back in stream.
006FB5	C9	C+	74		ret ; Return and print CR.
006FB6		C+	74	__5370a	endif
		C+	74		if DEVICE.console=1
		C+	74	__5371:	LD A, (HL) ; Get byte to send.
		C+	74		INC HL ; pointer=pointer+1
		C+	74		; CP EOT.asc ; End Of Transmission (block)?
		C+	74		; jr Z, \$1a ; Return if finished.
		C+	74		CP ETX.asc ; End Of Text?
		C+	74		jr Z, __5371a ; Return if finished.
		C+	74		LD C, A ; Output byte to C.
		C+	74		XOR A ; Set Z flag for put.
		C+	74		CALL U1DVR ; Out the UART we go.
		C+	74		JR __5371
		C+	74	__5371a	equ \$
		C+	74		endif
		C+	74		ENDMAC PUMP
006FB6	18 FE	C	75		jr \$ ; If no root volume we cannot boot.
006FB8	C9	C	76	mounterr1	ret ;PUMP DEVICE.console, mounterrmsg1



\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-subroutines.s
006FB9	C9	C	77	ret	; Problems with data vol, go back and try continue.
006FBA		C	78	nodirectort1	PUMP DEVICE.console,nodirdrv0msg1
		C+	78	scope	; A tribute to Dave Shriner - Display Data Corp.
006FBA	21 40 80	C+	78	LD HL,nodirdrv0msg1	; Pump from this starting point until ETX.asc is found.
		C+	78	if DEVICE.console=0	
006FBD	7E	C+	78	LD A,(HL)	; Get byte to send.
006FBE	23	C+	78	INC HL	; pointer=pointer+1
		C+	78	CP EOT.asc	; End Of Transmission (block)?
		C+	78	jr Z,\$0a	; Return if finished.
006FBF	FE03	C+	78	CP ETX.asc	; End Of Text?
006FC1	28 16	C+	78	jr Z,__5380a	; Return if finished.
006FC3	FE0D	C+	78	CP CR.asc	; Was it a CR?
006FC5	CC CF 6F	C+	78	call z,__5381	; 1st add a LF to stream.
006FC8	4F	C+	78	LD C,A	; Output byte in C.
006FC9	AF	C+	78	XOR A	; Set Z flag for put.
006FCA	CD D9 10	C+	78	CALL U0DVR	; Out the UART we go.
006FCD	18 EE	C+	78	JR __5380	
006FCF	3E0A	C+	78	LD a,LF.asc	; Get a LF.
006FD1	4F	C+	78	LD C,A	; Output byte in C.
006FD2	AF	C+	78	XOR A	; Set Z flag for put
006FD3	CD D9 10	C+	78	CALL U0DVR	; Out the UART we go.
006FD6	3E0D	C+	78	ld a,CR.asc	; Now put a CR back in stream.
006FD8	C9	C+	78	ret	; Return and print CR.
006FD9		C+	78	__5380a	endif
		C+	78	if DEVICE.console=1	
		C+	78	LD A,(HL)	; Get byte to send.
		C+	78	INC HL	; pointer=pointer+1
		C+	78	CP EOT.asc	; End Of Transmission (block)?
		C+	78	jr Z,\$1a	; Return if finished.
		C+	78	CP ETX.asc	; End Of Text?
		C+	78	jr Z,__5381a	; Return if finished.
		C+	78	LD C,A	; Output byte to C.
		C+	78	XOR A	; Set Z flag for put.
		C+	78	CALL U1DVR	; Out the UART we go.
		C+	78	JR __5381	
		C+	78	__5381a	equ \$
		C+	78	endif	
		C+	78	ENDMAC	PUMP
006FD9	18 FE	C	79	jr \$	
006FDB		C	80	cant_ipl_data	PUMP DEVICE.console,nobootdatamsg
		C+	80	scope	; A tribute to Dave Shriner - Display Data Corp.
006FDB	21 75 7C	C+	80	LD HL,nobootdatamsg	; Pump from this starting point until ETX.asc is found.
		C+	80	if DEVICE.console=0	
006FDE	7E	C+	80	LD A,(HL)	; Get byte to send.
006FDF	23	C+	80	INC HL	; pointer=pointer+1
		C+	80	CP EOT.asc	; End Of Transmission (block)?
		C+	80	jr Z,\$0a	; Return if finished.
006FE0	FE03	C+	80	CP ETX.asc	; End Of Text?
006FE2	28 16	C+	80	jr Z,__5390a	; Return if finished.
006FE4	FE0D	C+	80	CP CR.asc	; Was it a CR?
006FE6	CC F0 6F	C+	80	call z,__5391	; 1st add a LF to stream.
006FE9	4F	C+	80	LD C,A	; Output byte in C.
006FEA	AF	C+	80	XOR A	; Set Z flag for put.
006FEB	CD D9 10	C+	80	CALL U0DVR	; Out the UART we go.
006FEE	18 EE	C+	80	JR __5390	
006FF0	3E0A	C+	80	LD a,LF.asc	; Get a LF.
006FF2	4F	C+	80	LD C,A	; Output byte in C.
006FF3	AF	C+	80	XOR A	; Set Z flag for put
006FF4	CD D9 10	C+	80	CALL U0DVR	; Out the UART we go.
006FF7	3E0D	C+	80	ld a,CR.asc	; Now put a CR back in stream.

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-subroutines.s
006FF9	C9	C+	80	ret	; Return and print CR.
006FFA		C+	80	__5390a	endif
		C+	80		if DEVICE.console=1
		C+	80	__5391:	LD A,(HL) ; Get byte to send.
		C+	80		INC HL ; pointer=pointer+1
		C+	80	;	CP EOT.asc ; End Of Transmission (block)?
		C+	80	;	jr Z,\$1a ; Return if finished.
		C+	80		CP ETX.asc ; End Of Text?
		C+	80		jr Z,__5391a ; Return if finished.
		C+	80		LD C,A ; Output byte to C.
		C+	80		XOR A ; Set Z flag for put.
		C+	80		CALL U1DVR ; Out the UART we go.
		C+	80		JR __5391
		C+	80	__5391a	equ \$
		C+	80		endif
		C+	80		ENDMAC PUMP
006FFA	18 FE	C	81		jr \$
006FFC		C	82	not56fmt	PUMP DEVICE.console,not56fmtmsg
		C+	82		scope ; A tribute to Dave Shriner - Display Data Corp.
006FFC	21 A0 7D	C+	82		LD HL,not56fmtmsg ; Pump from this starting point until ETX.asc is found.
		C+	82		if DEVICE.console=0
006FFF	7E	C+	82	__5400:	LD A,(HL) ; Get byte to send.
007000	23	C+	82		INC HL ; pointer=pointer+1
		C+	82	;	CP EOT.asc ; End Of Transmission (block)?
		C+	82	;	jr Z,\$0a ; Return if finished.
		C+	82		CP ETX.asc ; End Of Text?
007001	FE03	C+	82		jr Z,__5400a ; Return if finished.
007003	28 16	C+	82		CP CR.asc ; Was it a CR?
007005	FE0D	C+	82		call z,__5401 ; 1st add a LF to stream.
007007	CC 11 70	C+	82		LD C,A ; Output byte in C.
00700A	4F	C+	82		XOR A ; Set Z flag for put.
00700B	AF	C+	82		CALL U0DVR ; Out the UART we go.
00700C	CD D9 10	C+	82		JR __5400
00700F	18 EE	C+	82		__5401: ld a,LF.asc ; Get a LF.
007011	3E0A	C+	82		LD C,A ; Output byte in C.
007013	4F	C+	82		XOR A ; Set Z flag for put
007014	AF	C+	82		CALL U0DVR ; Out the UART we go.
007015	CD D9 10	C+	82		ld a,CR.asc ; Now put a CR back in stream.
007018	3E0D	C+	82		ret ; Return and print CR.
00701A	C9	C+	82		__5400a
00701B		C+	82		endif
		C+	82		if DEVICE.console=1
		C+	82	__5401:	LD A,(HL) ; Get byte to send.
		C+	82		INC HL ; pointer=pointer+1
		C+	82	;	CP EOT.asc ; End Of Transmission (block)?
		C+	82	;	jr Z,\$1a ; Return if finished.
		C+	82		CP ETX.asc ; End Of Text?
		C+	82		jr Z,__5401a ; Return if finished.
		C+	82		LD C,A ; Output byte to C.
		C+	82		XOR A ; Set Z flag for put.
		C+	82		CALL U1DVR ; Out the UART we go.
		C+	82		JR __5401
		C+	82	__5401a	equ \$
		C+	82		endif
		C+	82		ENDMAC PUMP
00701B	C9	C	83		ret
		C	84		
00701C		C	85	notimp	PUMP DEVICE.console,NOTIMPMENU ; Pump.
		C+	85		scope ; A tribute to Dave Shriner - Display Data Corp.
00701C	21 1B 7A	C+	85		LD HL,NOTIMPMENU ; Pump from this starting point until ETX.asc is found.
		C+	85		if DEVICE.console=0

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-subroutines.s
00701F	7E	C+	85	__5410:	LD A,(HL) ; Get byte to send.
007020	23	C+	85		INC HL ; pointer=pointer+1
		C+	85		; CP EOT.asc ; End Of Transmission (block)?
		C+	85		; jr Z,\$0a ; Return if finished.
007021	FE03	C+	85		CP ETX.asc ; End Of Text?
007023	28 16	C+	85		jr Z,__5410a ; Return if finished.
007025	FE0D	C+	85		CP CR.asc ; Was it a CR?
007027	CC 31 70	C+	85		call z,__5411 ; 1st add a LF to stream.
00702A	4F	C+	85		LD C,A ; Output byte in C.
00702B	AF	C+	85		XOR A ; Set Z flag for put.
00702C	CD D9 10	C+	85		CALL U0DVR ; Out the UART we go.
00702F	18 EE	C+	85		JR __5410
007031	3E0A	C+	85	__5411:	ld a,LF.asc ; Get a LF.
007033	4F	C+	85		LD C,A ; Output byte in C.
007034	AF	C+	85		XOR A ; Set Z flag for put
007035	CD D9 10	C+	85		CALL U0DVR ; Out the UART we go.
007038	3E0D	C+	85		ld a,CR.asc ; Now put a CR back in stream.
00703A	C9	C+	85		ret ; Return and print CR.
00703B		C+	85	__5410a	endif
		C+	85		if DEVICE.console=1
		C+	85	__5411:	LD A,(HL) ; Get byte to send.
		C+	85		INC HL ; pointer=pointer+1
		C+	85		; CP EOT.asc ; End Of Transmission (block)?
		C+	85		; jr Z,\$1a ; Return if finished.
		C+	85		CP ETX.asc ; End Of Text?
		C+	85		jr Z,__5411a ; Return if finished.
		C+	85		LD C,A ; Output byte to C.
		C+	85		XOR A ; Set Z flag for put.
		C+	85		CALL U1DVR ; Out the UART we go.
		C+	85		JR __5411
		C+	85	__5411a	equ \$
		C+	85		endif
		C+	85		ENDMAC PUMP
		C	86		GOSUB getchar
		C+	86		scope
00703B	CD 7E 6F	C+	86		CALL getchar
		C+	86		endmac GOSUB
		C	87		GOTO main_ipl_menu
		C+	87		scope
00703E	C3 00 48	C+	87		jp main_ipl_menu
		C+	87		endmac GOTO
		C	88		
		C	89		
		C	90		; Store ASCII representation of 24 bit value in main memu.
		C	91		; Convert to packed BCD.
		C	92		; Convert packed BCD to unpacked.
		C	93		; Convert unpacked to ASCII.
		C	94		; Trim leading 0's away.
		C	95		; Stuff all these results in main menu.
		C	96		
		C	97		
007041	2E00	C	98	cvt_24_ascii	ld l,0 ; Pack DE into 24 bit
007043	63	C	99		ld h,e ; register combo.
007044	5A	C	100		ld e,d ; E:HL.
		C	101		
007045	C5	C	102		PUSH BC ; Save pointer to menu buffer.
007046	CD 6A 70	C	103		call Alwin ; convert to packed BCD in DE:HL.
007049	C1	C	104		pop bc ; recover pointer to bootsiz\$
		C	105		
00704A	E5	C	106		push hl ; Needed for conversion

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-subroutines.s
00704B	E5	C	107	push	hl ; routines & stuffing menu.
00704C	D5	C	108	push	de
		C	109		
		C	110		
00704D	C5E1	C	111	ld	hl,bc ; Address to store ASCII result in string.
		C	112		
00704F	7A	C	113	ld	a,d ; Convert 2 packed BCD to ASCII digits and store at (HL).
007050	CD 9B 70	C	114	call	bin24ascii
		C	115		
007053	D1	C	116	pop	de ; Recover digits.
007054	7B	C	117	ld	a,e ; Get 3rd & 4th digit.
007055	CD 9B 70	C	118	call	bin24ascii
		C	119		
007058	D1	C	120	pop	de ; Recover size.
007059	7A	C	121	ld	a,d ; Convert 2 packed BCD to ASCII digits and store at (HL).
00705A	CD 9B 70	C	122	call	bin24ascii
		C	123		
00705D	D1	C	124	pop	de ; Recover digits.
00705E	7B	C	125	ld	a,e ; Get 3rd & 4th digit.
00705F	CD 9B 70	C	126	call	bin24ascii
		C	127		
007062	C5E1	C	128	ld	hl,bc ; Scan this buffer.
007064	0608	C	129	ld	b,8 ; Scan 8 chars.
007066	CD B1 70	C	130	call	trim_zeros ; Trim off leading zeros.
		C	131		
007069	C9	C	132	ret	
		C	133		
		C	134		; Routine for converting a 24-bit binary number to decimal
		C	135		; In: E:HL = 24-bit binary number (0-16777215)
		C	136		; Out: DE:HL = 8 digit decimal form (packed BCD)
		C	137		; Changes: AF, BC, DE, HL & IX
		C	138		
		C	139		
		C	140		; Straight from my time-tested toolbox Smile If you want bigger numbers: let me know.
		C	141		; I also have a ; routine that can do up to 64 bit numbers (20 digits decimal) and produces ready-f
		C	142		; but it's quite a bit longer. Probably did it once just to see how easy it was to expand, or to ha
		C	143		; something that can take anything you throw at it...
		C	144		; by Alwin Henseler
		C	145		
00706A	4B	C	146	Alwin:	LD C,E
00706B	E5	C	147		PUSH HL
00706C	DDE1	C	148		POP IX ; input value in C:IX
00706E	210100	C	149		LD HL,1
007071	54	C	150		LD D,H
007072	5C	C	151		LD E,H ; start value corresponding with 1st 1-bit
007073	0618	C	152		LD B,24 ; bitnr. being processed + 1
		C	153		
007075	DD29	C	154	FIND1:	ADD IX,IX
007077	CB11	C	155		RL C ; shift bit 23-0 from C:IX into carry
007079	38 1D	C	156		JR C,NEXTBIT
00707B	10 F8	C	157		DJNZ FIND1 ; find highest 1-bit
		C	158		
		C	159		; All bits 0:
00707D	CB85	C	160		RES 0,L ; least significant bit not 1 after all ..
00707F	C9	C	161		RET
		C	162		
007080	7D	C	163	DBLLOOP:	LD A,L
007081	87	C	164		ADD A,A
007082	27	C	165		DAA
007083	6F	C	166		LD L,A

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

```

PC      Object      I  Line      Source  ..\SYS0_IPL\ipl-subroutines.s
007084 7C           C   167          LD A,H
007085 8F           C   168          ADC A,A
007086 27           C   169          DAA
007087 67           C   170          LD H,A
007088 7B           C   171          LD A,E
007089 8F           C   172          ADC A,A
00708A 27           C   173          DAA
00708B 5F           C   174          LD E,A
00708C 7A           C   175          LD A,D
00708D 8F           C   176          ADC A,A
00708E 27           C   177          DAA
00708F 57           C   178          LD D,A          ; double the value found so far
007090 DD29         C   179          ADD IX,IX
007092 CB11         C   180          RL C          ; shift next bit from C:IX into carry
007094 30 02        C   181          JR NC,NEXTBIT  ; bit = 0 -> don't add 1 to the number
007096 CBC5         C   182          SET 0,L        ; bit = 1 -> add 1 to the number
007098 10 E6        C   183  NEXTBIT:      DJNZ DBLL00P
00709A C9           C   184          RET
C   185
C   186  ;----- snip -----
C   187
C   188
00709B F5           C   189  bin24ascii  push  af          ; Save binary copy for later.
00709C E6F0         C   190          and  F0h          ; Strip all but upper 4 bits.
00709E CB3F         C   191          srl  a          ; Shift right.
0070A0 CB3F         C   192          srl  a
0070A2 CB3F         C   193          srl  a
0070A4 CB3F         C   194          srl  a
0070A6 CD AC 70     C   195          call cvtbinasc
C   196
0070A9 F1           C   197          pop  af          ; Recover original values.
0070AA E60F         C   198          and  0Fh          ; Strip out upper 4 bits.
C   199
C   200
0070AC C630         C   201  cvtbinascadd '0'          ; Convert nibble to ASCII.
0070AE 77           C   202          ld   (hl),a        ; Store it in buffer.
0070AF 23           C   203          inc  hl          ; Bump pointer up by 1.
0070B0 C9           C   204          ret
C   205
C   206  ; routine to trim leading zeros from a string.
C   207  ; B = char to check.
C   208
0070B1 3E30         C   209  trim_zeros  ld    a,'0'          ; ASCII zero.
0070B3 BE           C   210          cp   (hl)          ; HL points to buffer under test.
0070B4 C0           C   211          ret  nz
0070B5 3620         C   212          ld   (hl),SP.asc ; SPACE Stuff it.
0070B7 23           C   213          inc  hl
0070B8 10 F7        C   214          djnz trim_zeros
0070BA 2B           C   215          dec  hl
0070BB 3630         C   216          ld   (hl),'0'
0070BD C9           C   217          ret
C   0
C   1  include "main_menu_screen.s"          ; Get main menu of TRS-OS
C   2  ; System startup & initialization screen.
C   3
0070BE 20202020 20202020 C   3  BOOTMENU db  "          www.TRSDOS.com",CR.asc
0070C6 20202020 20202020
0070CE 20202020 20202020
0070D6 20202020 20202020
0070DE 7777772E 54525344
0070E6 4F532E63 6F6D0D

```

\*\*\*\*\* TRSDOS \*\*\*\*\*  
< SYS0 IPL - DOS-initialization code & welcome message >

PC	Object		I	Line	Source ..\SYS0_IPL\screens\main_menu_screen.s	
0070ED	20202020	20202020	C	4	db "	www.DANIELPAULMARTIN.com",CR.asc
0070F5	20202020	20202020				
0070FD	20202020	20202020				
007105	20202077	77772E44				
00710D	414E4945	4C504155				
007115	4C4D4152	54494E2E				
00711D	636F6D0D					
			C	5		
			C	6		
007121	0D		C	7	db CR.asc	
007122	20446174	653A2020	C	8	BERTSAYS:DB " Date: Time: Kernel: .1"	
00712A	20202020	20202020				
007132	54696D65	3A202020				
00713A	20202020	20202020				
007142	4B65726E	656C3A20				
00714A	20202020	20202020				
007152	20202020	20202E31				
	00007149		C	9	paknam\$ equ \$-17	
00715A	0D		C	10	DB CR.asc	
			C	11		
00715B	2049504C	20747970	C	12	db " IPL type = "	
007163	65203D20					
007167	20202020	2020	C	13	ipl_from db " "	
			C	14		
	00007179		C	15	bootinitmsg equ cpu_typ_msg-15	;IPL-BIOS routine will display message from bootintmsg to B
			C	16		
00716D	20506C61	74666F72	C	17	db " Platform = "	
007175	6D203D20					
007179	20202020	20202020	C	18	vendor\$ db " "	
007181	20					
			C	19		
007182	20435055	203D	C	20	db " CPU ="	
	00007188		C	21	cpu_typ_msg equ \$	
007188	20202020	20202020	C	22	DB " ",CR.asc	
007190	0D					
			C	23		
	00000018		C	24	bootinitmsglen equ \$-bootinitmsg	
			C	25		
007191	20425245	414B2065	C	26	db " BREAK enabled="	
007199	6E61626C	65643D				
	000071A0		C	27	brkflg\$ equ \$	
0071A0	20		C	28	DB " "	
			C	29		
0071A1	20535953	47454E3D	C	30	db " SYSGEN="	
	000071A9		C	31	SYSGN\$ equ \$	
0071A9	20202020	20202020	C	32	db " ",CR.asc	
0071B1	20202020	200D				
			C	33		
0071B7	20417574	6F3D	C	34	DB " Auto="	
	000071BD		C	35	autoflg\$ EQU \$	
0071BD	20 20 20 20 20 20		C	36	blkb 73,20h	
0071C3	20 20 20 20 20 20					
0071C9	20 20 20 20 20 20					
0071CF	20 20 20 20 20 20					
0071D5	20 20 20 20 20 20					
0071DB	20 20 20 20 20 20					
0071E1	20 20 20 20 20 20					
0071E7	20 20 20 20 20 20					
0071ED	20 20 20 20 20 20					
0071F3	20 20 20 20 20 20					

\*\*\*\*\* TRSDOS \*\*\*\*\*  
< SYS0 IPL - DOS-initialization code & welcome message >

PC	Object	I	Line	Source	.. \SYS0_IPL\screens\main_menu_screen.s		
0071F9	20 20 20 20 20 20						
0071FF	20 20 20 20 20 20						
007205	20						
007206	0D	C	37	DB	CR.asc		
		C	38				
	00007217	C	39	server_stat	equ \$+16		
	00007214	C	40	serv_vol_num	equ \$+13		
007207	20536572 76657220	C	41	DB	" Server vol # : disconnected "		
00720F	766F6C20 23203A20						
007217	64697363 6F6E6E65						
00721F	63746564 20						
		C	42	;server_type	equ \$		
007224	20202020 20202020	C	43	db	" ,CR.asc		
00722C	0D						
		C	44				
	00007238	C	45	bootvol\$ EQU	\$+11		
	00007254	C	46	boottyp\$ EQU	\$+39		
	0000725D	C	47	bootsiz\$ equ	\$+48		
00722D	20566F6C 756D6520	C	48	DB	" Volume 0:	Type:	Bytes.",CR.asc
007235	303A2020 20202020						
00723D	20202020 20202020						
007245	20202020 20202020						
00724D	20547970 653A2020						
007255	20202020 20202020						
00725D	20202020 20202020						
007265	20427974 65732E0D						
		C	49				
	00007278	C	50	datavol\$ EQU	\$+11		
	00007294	C	51	datatyp\$ equ	\$+39		
	0000729D	C	52	datasiz\$ equ	\$+48		
00726D	20566F6C 756D6520	C	53	DB	" Volume 1:	Type:	Bytes.",CR.asc
007275	313A2020 20202020						
00727D	20202020 20202020						
007285	20202020 20202020						
00728D	20547970 653A2020						
007295	20202020 20202020						
00729D	20202020 20202020						
0072A5	20427974 65732E0D						
		C	54				
	000072B8	C	55	netvol\$ EQU	\$+11		
	000072D4	C	56	nettyp\$ equ	\$+39		
	000072DD	C	57	netsiz\$ equ	\$+48		
0072AD	204E6574 566F6C20	C	58	DB	" NetVol 6:	Type:	Bytes.",CR.asc
0072B5	363A2020 20202020						
0072BD	20202020 20202020						
0072C5	20202020 20202020						
0072CD	20547970 653A2020						
0072D5	20202020 20202020						
0072DD	20202020 20202020						
0072E5	20427974 65732E0D						
		C	59				
0072ED	0D	C	60	db	CR.asc		
		C	61				
0072EE	20302E20 3C535441	C	62	DB	" 0. <START> Start TRSDOS normally using default terminal .",CR.asc		
0072F6	52543E20 20205374						
0072FE	61727420 54525344						
007306	4F53206E 6F726D61						
00730E	6C6C7920 7573696E						
007316	67206465 6661756C						
00731E	74207465 726D696E						

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\screens\main_menu_screen.s
007326	616C2020 2E0D 00007329	C	63	default_tequ	\$-3
		C	64		
00732C	00	C	65	db	00h ;***** Test byte to test network for null detection.
		C	66		
00732D	20312E20 3C555044	C	67	db	" 1. <UPDATE> Run system updates.",CR.asc
007335	4154453E 20205275				
00733D	6E207379 7374656D				
007345	20757064 61746573				
00734D	2E0D				
00734F	20322E20 3C534146	C	68	db	" 2. <SAFE> Prompt SYSGEN AUTO DEBUG etc.",CR.asc
007357	453E2020 20205072				
00735F	6F6D7074 20535953				
007367	47454E20 4155544F				
00736F	20444542 55472065				
007377	74632E0D				
00737B	20332E20 3C244249	C	69	db	" 3. <\$BIT> Binary Instrumentation Telemetry mode.",CR.asc
007383	543E2020 20204269				
00738B	6E617279 20496E73				
007393	7472756D 656E7461				
00739B	74696F6E 2054656C				
0073A3	656D6574 7279206D				
0073AB	6F64652E 0D				
0073B0	20342E20 3C434C4F	C	70	db	" 4. <CLOCK> Manage real-time clock settings.",CR.asc
0073B8	434B3E20 20204D61				
0073C0	6E616765 20726561				
0073C8	6C2D7469 6D652063				
0073D0	6C6F636B 20736574				
0073D8	74696E67 732E0D				
0073DF	20352E20 3C534C49	C	71	db	" 5. <SLICE> Manage SLICE maps.",CR.asc
0073E7	43453E20 20204D61				
0073EF	6E616765 20534C49				
0073F7	4345206D 6170732E				
0073FF	0D				
007400	0D	C	72	db	CR.asc
007401	20442E20 3C444546	C	73	db	" D. <DEFAULT> Change default startup for this session.",CR.asc
007409	41554C54 3E204368				
007411	616E6765 20646566				
007419	61756C74 20737461				
007421	72747570 20666F72				
007429	20746869 73207365				
007431	7373696F 6E2E0D				
007438	204E2E20 3C4E4554	C	74	db	" N. <NET> Network utilities & tests.",CR.asc
007440	3E202020 20204E65				
007448	74776F72 6B207574				
007450	696C6974 69657320				
007458	26207465 7374732E				
007460	0D				
007461	20582E20 3C455849	C	75	db	" X. <EXIT> Exit TRS-OS.",CR.asc
007469	543E2020 20204578				
007471	69742054 52532D4F				
007479	532E0D				
		C	76		
00747C	03	C	77	db	ETX.asc ; Tell pump this is end of screen.
		C	0	include	"lsdos_splash_screen.s" ; Original splash screen of LS-DOS 6.3.1
		C	1		
00747D	0C	C	2	DB	FF.asc ; Clear screen on terminal.
00747E	1B5B66	C	3	DB	ESC.asc,"[f" ; Cursor to home position.
007481	0A0A	C	4	lsdos631\$db	LF.asc,LF.asc
007483	20202020 20202020	C	5	DB	' LS-DOS 06.03.01 - Copyright 1986/90 MISOSYS, Inc.',CR.asc ;<631>



\*\*\*\*\* TRSDOS \*\*\*\*\*  
< SYS0 IPL - DOS-initialization code & welcome message >

PC	Object	I	Line	Source	..\\SYS0_IPL\\screens\\lsdos_splash_screen.s
00748B	20202020 204C532D				
007493	444F5320 30362E30				
00749B	332E3031 202D2043				
0074A3	6F707972 69676874				
0074AB	20313938 362F3930				
0074B3	204D4953 4F535953				
0074BB	2C20496E 632E0D				
0074C2	20202020 20202020	C	6	DB	' All Rights Reserved.',CR.asc
0074CA	20202020 20202020				
0074D2	20202020 20202020				
0074DA	20202020 20416C6C				
0074E2	20526967 68747320				
0074EA	52657365 72766564				
0074F2	2E0D				
0074F4	20202020 20202020	C	7	DB	' LS-DOS63 Level-1L',CR.asc
0074FC	20202020 20202020				
007504	20202020 20202020				
00750C	20202020 20204C53				
007514	2D444F53 36332020				
00751C	4C657665 6C2D314C				
007524	0D				
007525	0A0A	C	8	db	LF.asc,LF.asc
007527	20202020 20202020	C	9	DB	' *****',CR.asc
00752F	20202020 20202020				
007537	20202020 20202020				
00753F	20202020 202A2A2A				
007547	2A2A2A2A 2A2A2A2A				
00754F	2A2A2A2A 2A2A2A2A				
007557	0D				
007558	20202020 20202020	C	10	DB	' * *',CR.asc
007560	20202020 20202020				
007568	20202020 20202020				
007570	20202020 202A2020				
007578	20202020 20202020				
007580	20202020 2020202A				
007588	0D				
007589	20202020 20202020	C	11	DB	' * W E L C O M E *',CR.asc
007591	20202020 20202020				
007599	20202020 20202020				
0075A1	20202020 202A2020				
0075A9	57204520 4C204320				
0075B1	4F204D20 4520202A				
0075B9	0D				
0075BA	20202020 20202020	C	12	DB	' * *',CR.asc
0075C2	20202020 20202020				
0075CA	20202020 20202020				
0075D2	20202020 202A2020				
0075DA	20202020 20202020				
0075E2	20202020 2020202A				
0075EA	0D				
0075EB	20202020 20202020	C	13	DB	' * t o *',CR.asc
0075F3	20202020 20202020				
0075FB	20202020 20202020				
007603	20202020 202A2020				
00760B	20202020 2074206F				
007613	20202020 2020202A				
00761B	0D				
00761C	20202020 20202020	C	14	DB	' * *',CR.asc
007624	20202020 20202020				
00762C	20202020 20202020				

\*\*\*\*\* TRSDOS \*\*\*\*\*  
< SYS0 IPL - DOS-initialization code & welcome message >

PC	Object	I	Line	Source	.. \SYS0_IPL\screens\lsdos_splash_screen.s
007634	20202020 202A2020				
00763C	20202020 20202020				
007644	20202020 2020202A				
00764C	0D				
00764D	20202020 20202020	C	15	DB	' * www.TRSDOS.com *',CR.asc
007655	20202020 20202020				
00765D	20202020 20202020				
007665	20202020 202A2020				
00766D	7777772E 54525344				
007675	4F532E63 6F6D202A				
00767D	0D				
00767E	20202020 20202020	C	16	DB	' * *',CR.asc
007686	20202020 20202020				
00768E	20202020 20202020				
007696	20202020 202A2020				
00769E	20202020 20202020				
0076A6	20202020 2020202A				
0076AE	0D				
0076AF	20202020 20202020	C	17	DB	' *****',CR.asc
0076B7	20202020 20202020				
0076BF	20202020 20202020				
0076C7	20202020 202A2A2A				
0076CF	2A2A2A2A 2A2A2A2A				
0076D7	2A2A2A2A 2A2A2A2A				
0076DF	0D				
0076E0	0A0A0A	C	18	db	LF.asc,LF.asc,LF.asc
		C	19		
	00007705	C	20	lsdosdate equ	+\$+34
0076E3	20202020 20202020	C	21	db	' ,CR.asc
0076EB	20202020 20202020				
0076F3	20202020 20202020				
0076FB	20202020 20202020				
007703	20202020 20202020				
00770B	20202020 2020200D				
	00007735	C	22	lsdostime equ	+\$+34
007713	20202020 20202020	C	23	db	' ,CR.asc
00771B	20202020 20202020				
007723	20202020 20202020				
00772B	20202020 20202020				
007733	20202020 20202020				
00773B	20202020 2020200D				
007743	20202020 20202020	C	24	sysgmsg db	' ,ETX.asc
00774B	20202020 03				
007750	CF02	C	25	splash\$_len dw	\$_lsdos631\$
		C	0	include	"welcome_screen.s" ; TRS-OS welcome screen with copyright noti
		C	1		
		C	2		; TRS-OS welcome screen with copyright messages.
		C	3		
	00007752	C	4	welcomemsg equ	\$ ; Screen pump starts here.
007752	0D	C	5	db	CR.asc
		C	6	; db	LF.asc
		C	7	; db	LF.asc
		C	8		
007753	20205720 45204C20	C	9	DB	" W E L C O M E to T R S D O S 7 ",CR.asc
00775B	43204F20 4D204520				
007763	20746F20 20542052				
00776B	20532044 204F2053				
007773	20203720 200D				
007779	5F5F5F5F 5F5F5F5F	C	10	db	" _____",CR.asc
007781	5F5F5F5F 5F5F5F5F				

\*\*\*\*\* TRSDOS \*\*\*\*\*  
< SYS0 IPL - DOS-initialization code & welcome message >

PC	Object	I	Line	Source	..\SYS0_IPL\screens\welcome_screen.s
007789	5F5F5F5F 5F5F5F5F				
007791	5F5F5F5F 5F5F5F5F				
007799	5F5F5F5F 5F0D				
00779F	0D	C	11	db	CR.asc
0077A0	20436F70 79726967	C	12	DB	" Copyright 2009 - 2024 Daniel Paul Martin. All rights reserved.",CR.asc
0077A8	68742032 30303920				
0077B0	2D203230 32342020				
0077B8	44616E69 656C2050				
0077C0	61756C20 4D617274				
0077C8	696E2E20 416C6C20				
0077D0	72696768 74732072				
0077D8	65736572 7665642E				
0077E0	0D				
0077E1	20507265 6C6F6164	C	13	DB	" Preloaded LSDOS 6.3.1 Copyright Roy Soltoff. All rights reserved. ",CR.asc
0077E9	6564204C 53444F53				
0077F1	20362E33 2E312043				
0077F9	6F707972 69676874				
007801	20526F79 20536F6C				
007809	746F6666 2E20416C				
007811	6C207269 67687473				
007819	20726573 65727665				
007821	642E200D				
007825	20424153 434F4D20	C	14	DB	" BASCOM Copyright 1984 Microsoft Inc. All rights reserved.",CR.asc
00782D	436F7079 72696768				
007835	74203139 3834204D				
00783D	6963726F 736F6674				
007845	20496E63 2E20416C				
00784D	6C207269 67687473				
007855	20726573 65727665				
00785D	642E0D				
007860	20467265 6520746F	C	15	DB	" Free to distribute & use this program provided above copyright ",CR.asc
007868	20646973 74726962				
007870	75746520 26207573				
007878	65207468 69732070				
007880	726F6772 616D2070				
007888	726F7669 64656420				
007890	61626F76 6520636F				
007898	70797269 67687420				
0078A0	0D				
0078A1	20697320 70726573	C	16	db	" is preserved & displayed. All rights reserved.",CR.asc
0078A9	65727665 64202620				
0078B1	64697370 6C617965				
0078B9	642E2041 6C6C2072				
0078C1	69676874 73207265				
0078C9	73657276 65642E0D				
		C	17		
0078D1	03	C	18	db	ETX.asc ; Tell pump end of screen.
		C	19		
	00000180	C	20	RDYMSGLENEQU	\$-welcomemsg
		C	0		include "initial_message_screen.s" ; Contains initialization messages.
		C	1		
0078D2	20426572 74207361	C	2	initialmsg	DB " Bert says, it's kinda nice in here.",CR.asc
0078DA	79732C20 69742773				
0078E2	206B696E 6461206E				
0078EA	69636520 696E2068				
0078F2	6572652E 0D				
0078F7	20536861 776E2073	C	3	DB	" Shawn says, waste of time.",CR.asc
0078FF	6179732C 20776173				
007907	7465206F 66207469				
00790F	6D652E0D				

\*\*\*\*\* TRSDOS \*\*\*\*\*  
< SYS0 IPL - DOS-initialization code & welcome message >

PC	Object		I	Line	Source ..\SYS0_IPL\screens\initial_message_screen.s	
007913	20486F6D	65722073	C	4	DB " Homer said it rocks!",CR.asc,ETX.asc	; Tell pump this is end of screen.
00791B	61696420	69742072				
007923	6F636B73	210D03				
			C	5		
00792A	20746578	74206669	C	6	splashiplmode db " text filled @ipl	",C
007932	6C6C6564	20406970				
00793A	6C202020	20202020				
007942	20202020	20202020				
00794A	20202020	20202020				
007952	20202020	20202020				
00795A	20202020	20202020				
007962	20202020	20202020				
00796A	20202020	20202020				
007972	2020200D					
			C	7		
007976	20537973	74656D20	C	8	DB " System stack set.",CR.asc	
00797E	73746163	6B207365				
007986	742E0D					
007989	20537973	74656D20	C	9	DB " System clock set.",CR.asc	
007991	636C6F63	6B207365				
007999	742E0D					
00799C	204D656D	6F727920	C	10	DB " Memory management initialized.",CR.asc	
0079A4	6D616E61	67656D65				
0079AC	6E742069	6E697469				
0079B4	616C697A	65642E0D				
0079BC	204D6163	68696E65	C	11	DB " Machine hardware flag set.",CR.asc	
0079C4	20686172	64776172				
0079CC	6520666C	61672073				
0079D4	65742E0D					
0079D8	20435055	20696E69	C	12	DB " CPU initializization completed.",CR.asc	
0079E0	7469616C	697A697A				
0079E8	6174696F	6E20636F				
0079F0	6D706C65	7465642E				
0079F8	0D					
0079F9	2042494F	53207265	C	13	DB " BIOS ready. ",CR.asc	
007A01	6164792E	200D				
007A07	20506C65	61736520	C	14	DB " Please wait.....",CR.asc	
007A0F	77616974	2E2E2E2E				
007A17	2E2E0D					
007A1A	03		C	15	db ETX.asc	; Tell pump this is end of screen.
			C	16		
			C	0	include "ipl_error_screen.s"	; Error messages possible during IPL.
			C	1		
007A1B	0D0A		C	2	NOTIMPMENU db CR.asc,LF.asc	
007A1D	3E2D2D2D	2D2D2D2D	C	3	DB ">-----<",CR.asc	
007A25	2D2D2D2D	2D2D2D2D				
007A2D	2D2D2D2D	2D2D2D2D				
007A35	2D2D2D2D	2D2D2D2D				
007A3D	2D2D2D2D	2D2D2D2D				
007A45	2D2D2D2D	2D2D2D2D				
007A4D	2D2D2D2D	2D2D2D2D				
007A55	2D2D2D2D	2D2D2D2D				
007A5D	2D2D2D2D	2D2D2D2D				
007A65	2D3C0D					
007A68	7C202020	20202020	C	4	DB "  S E L E C T I O N  ",CR.asc	
007A70	20202020	20202020				
007A78	20202020	20202020				
007A80	20202020	20205320				
007A88	45204C20	45204320				
007A90	54204920	4F204E20				

\*\*\*\*\* TRSDOS \*\*\*\*\*  
< SYS0 IPL - DOS-initialization code & welcome message >

PC	Object	I	Line	Source	..\\SYS0_IPL\\screens\\ipl_error_screen.s
007A98	20202020 20202020				
007AA0	20202020 20202020				
007AA8	20202020 20202020				
007AB0	207C0D				
007AB3	7C202020 20202020	C	5	DB	"  only available in registered version.  ",CR.asc
007ABB	20202020 20202020				
007AC3	20202020 206F6E6C				
007ACB	79206176 61696C61				
007AD3	626C6520 696E2072				
007ADB	65676973 74657265				
007AE3	64207665 7273696F				
007AEB	6E2E2020 20202020				
007AF3	20202020 20202020				
007AFB	207C0D				
007AFE	3E2D2D2D 2D2D2D2D	C	6	DB	">-----<",CR.asc
007B06	2D2D2D2D 2D2D2D2D				
007B0E	2D2D2D2D 2D2D2D2D				
007B16	2D2D2D2D 2D2D2D2D				
007B1E	2D2D2D2D 2D2D2D2D				
007B26	2D2D2D2D 2D2D2D2D				
007B2E	2D2D2D2D 2D2D2D2D				
007B36	2D2D2D2D 2D2D2D2D				
007B3E	2D2D2D2D 2D2D2D2D				
007B46	2D3C0D				
007B49	03	C	7	DB	ETX.asc
		C	8		
007B4A	0D0A	C	9	mounterrmsg0	db CR.asc,LF.asc
007B4C	3E2D2D2D 2D2D2D2D	C	10	DB	">-----<",CR.asc
007B54	2D2D2D2D 2D2D2D2D				
007B5C	2D2D2D2D 2D2D2D2D				
007B64	2D2D2D2D 2D2D2D2D				
007B6C	2D2D2D2D 2D2D2D2D				
007B74	2D2D2D2D 2D2D2D2D				
007B7C	2D2D2D2D 2D2D2D2D				
007B84	2D2D2D2D 2D2D2D2D				
007B8C	2D2D2D2D 2D2D2D2D				
007B94	3C0D				
007B96	7C202020 20202020	C	11	DB	"  E R R O R  ",CR.asc
007B9E	20202020 20202020				
007BA6	20202020 20202020				
007BAE	20202020 20202020				
007BB6	20452052 2052204F				
007BBE	20522020 20202020				
007BC6	20202020 20202020				
007BCE	20202020 20202020				
007BD6	20202020 20202020				
007BDE	7C0D				
007BE0	7C202020 20202020	C	12	DB	"  mounting BOOT volume.  ",CR.asc
007BE8	20202020 20202020				
007BF0	20202020 20202020				
007BF8	2020206D 6F756E74				
007C00	696E6720 424F4F54				
007C08	20766F6C 756D652E				
007C10	20202020 20202020				
007C18	20202020 20202020				
007C20	20202020 20202020				
007C28	7C0D				
007C2A	3E2D2D2D 2D2D2D2D	C	13	DB	">-----<",CR.asc
007C32	2D2D2D2D 2D2D2D2D				
007C3A	2D2D2D2D 2D2D2D2D				

\*\*\*\*\* TRSDOS \*\*\*\*\*

< SYS0 IPL - DOS-initialization code & welcome message >

PC	Object	I	Line	Source	..\\SYS0_IPL\\screens\\ipl_error_screen.s
007C42	2D2D2D2D 2D2D2D2D				
007C4A	2D2D2D2D 2D2D2D2D				
007C52	2D2D2D2D 2D2D2D2D				
007C5A	2D2D2D2D 2D2D2D2D				
007C62	2D2D2D2D 2D2D2D2D				
007C6A	2D2D2D2D 2D2D2D2D				
007C72	3C0D				
007C74	03	C	14	DB	ETX.asc
		C	15		
007C75	0D0A	C	16	nobootdatamsg	db CR.asc,LF.asc
007C77	3E2D2D2D 2D2D2D2D	C	17	DB	">-----<",CR.asc
007C7F	2D2D2D2D 2D2D2D2D				
007C87	2D2D2D2D 2D2D2D2D				
007C8F	2D2D2D2D 2D2D2D2D				
007C97	2D2D2D2D 2D2D2D2D				
007C9F	2D2D2D2D 2D2D2D2D				
007CA7	2D2D2D2D 2D2D2D2D				
007CAF	2D2D2D2D 2D2D2D2D				
007CB7	2D2D2D2D 2D2D2D2D				
007CBF	3C0D				
007CC1	7C202020 20202020	C	18	DB	"  E R R O R  ",CR.asc
007CC9	20202020 20202020				
007CD1	20202020 20202020				
007CD9	20202020 20202020				
007CE1	20452052 2052204F				
007CE9	20522020 20202020				
007CF1	20202020 20202020				
007CF9	20202020 20202020				
007D01	20202020 20202020				
007D09	7C0D				
007D0B	7C202020 20202020	C	19	DB	"  cannot boot data diskette.  ",CR.asc
007D13	20202020 20202020				
007D1B	20202020 20202020				
007D23	2063616E 6E6F7420				
007D2B	626F6F74 20646174				
007D33	61206469 736B6574				
007D3B	74652E20 20202020				
007D43	20202020 20202020				
007D4B	20202020 20202020				
007D53	7C0D				
007D55	3E2D2D2D 2D2D2D2D	C	20	DB	">-----<",CR.asc
007D5D	2D2D2D2D 2D2D2D2D				
007D65	2D2D2D2D 2D2D2D2D				
007D6D	2D2D2D2D 2D2D2D2D				
007D75	2D2D2D2D 2D2D2D2D				
007D7D	2D2D2D2D 2D2D2D2D				
007D85	2D2D2D2D 2D2D2D2D				
007D8D	2D2D2D2D 2D2D2D2D				
007D95	2D2D2D2D 2D2D2D2D				
007D9D	3C0D				
007D9F	03	C	21	DB	ETX.asc
		C	22		
007DA0	0D0A	C	23	not56fmtmsg	db CR.asc,LF.asc
007DA2	3E2D2D2D 2D2D2D2D	C	24	DB	">-----<",CR.asc
007DAA	2D2D2D2D 2D2D2D2D				
007DB2	2D2D2D2D 2D2D2D2D				
007DBA	2D2D2D2D 2D2D2D2D				
007DC2	2D2D2D2D 2D2D2D2D				
007DCA	2D2D2D2D 2D2D2D2D				
007DD2	2D2D2D2D 2D2D2D2D				

\*\*\*\*\* TRSDOS \*\*\*\*\*  
< SYS0 IPL - DOS-initialization code & welcome message >

PC	Object	I	Line	Source	..\SYS0_IPL\screens\ipl_error_screen.s
007DDA	2D2D2D2D 2D2D2D2D				
007DE2	2D2D2D2D 2D2D2D2D				
007DEA	3C0D				
007DEC	7C202020 20202020	C	25	DB	" E R R O R ",CR.asc
007DF4	20202020 20202020				
007DFC	20202020 20202020				
007E04	20202020 20202020				
007E0C	20452052 2052204F				
007E14	20522020 20202020				
007E1C	20202020 20202020				
007E24	20202020 20202020				
007E2C	20202020 20202020				
007E34	7C0D				
007E36	7C202020 20202020	C	26	DB	" data diskette ",CR.asc
007E3E	20202020 20202020				
007E46	20202020 20202020				
007E4E	20202020 20202064				
007E56	61746120 6469736B				
007E5E	65747465 20202020				
007E66	20202020 20202020				
007E6E	20202020 20202020				
007E76	20202020 20202020				
007E7E	7C0D				
007E80	7C202020 20202020	C	27	DB	" not dos 5 or 6 format. ",CR.asc
007E88	20202020 20202020				
007E90	20202020 20202020				
007E98	2020206E 6F742064				
007EA0	6F732035 206F7220				
007EA8	3620666F 726D6174				
007EB0	2E202020 20202020				
007EB8	20202020 20202020				
007EC0	20202020 20202020				
007EC8	7C0D				
007ECA	3E2D2D2D 2D2D2D2D	C	28	DB	">-----<",CR.asc
007ED2	2D2D2D2D 2D2D2D2D				
007EDA	2D2D2D2D 2D2D2D2D				
007EE2	2D2D2D2D 2D2D2D2D				
007EEA	2D2D2D2D 2D2D2D2D				
007EF2	2D2D2D2D 2D2D2D2D				
007EFA	2D2D2D2D 2D2D2D2D				
007F02	2D2D2D2D 2D2D2D2D				
007F0A	2D2D2D2D 2D2D2D2D				
007F12	3C0D				
007F14	03	C	29	DB	ETX.asc
		C	30		
007F15	0D0A	C	31	mounterrmsg1	db CR.asc,LF.asc
007F17	3E2D2D2D 2D2D2D2D	C	32	DB	">-----<",CR.asc
007F1F	2D2D2D2D 2D2D2D2D				
007F27	2D2D2D2D 2D2D2D2D				
007F2F	2D2D2D2D 2D2D2D2D				
007F37	2D2D2D2D 2D2D2D2D				
007F3F	2D2D2D2D 2D2D2D2D				
007F47	2D2D2D2D 2D2D2D2D				
007F4F	2D2D2D2D 2D2D2D2D				
007F57	2D2D2D2D 2D2D2D2D				
007F5F	3C0D				
007F61	7C202020 20202020	C	33	DB	" E R R O R ",CR.asc
007F69	20202020 20202020				
007F71	20202020 20202020				
007F79	20202020 20202020				

\*\*\*\*\* TRSDOS \*\*\*\*\*  
< SYS0 IPL - DOS-initialization code & welcome message >

PC	Object	I	Line	Source	..\\SYS0_IPL\\screens\\ipl_error_screen.s
007F81	20452052 2052204F				
007F89	20522020 20202020				
007F91	20202020 20202020				
007F99	20202020 20202020				
007FA1	20202020 20202020				
007FA9	7C0D				
007FAB	7C202020 20202020	C	34	DB	"  mounting data volume.  ",CR.asc
007FB3	20202020 20202020				
007FBB	20202020 20202020				
007FC3	2020206D 6F756E74				
007FCB	696E6720 64617461				
007FD3	20766F6C 756D652E				
007FDB	20202020 20202020				
007FE3	20202020 20202020				
007FEB	20202020 20202020				
007FF3	7C0D				
007FF5	3E2D2D2D 2D2D2D2D	C	35	DB	">-----<",CR.asc
007FFD	2D2D2D2D 2D2D2D2D				
008005	2D2D2D2D 2D2D2D2D				
00800D	2D2D2D2D 2D2D2D2D				
008015	2D2D2D2D 2D2D2D2D				
00801D	2D2D2D2D 2D2D2D2D				
008025	2D2D2D2D 2D2D2D2D				
00802D	2D2D2D2D 2D2D2D2D				
008035	2D2D2D2D 2D2D2D2D				
00803D	3C0D				
00803F	03	C	36	DB	ETX.asc
		C	37		
008040	0D0A	C	38	nodirdrv0msg1 db	CR.asc,LF.asc
008042	3E2D2D2D 2D2D2D2D	C	39	DB	">-----<",CR.asc
00804A	2D2D2D2D 2D2D2D2D				
008052	2D2D2D2D 2D2D2D2D				
00805A	2D2D2D2D 2D2D2D2D				
008062	2D2D2D2D 2D2D2D2D				
00806A	2D2D2D2D 2D2D2D2D				
008072	2D2D2D2D 2D2D2D2D				
00807A	2D2D2D2D 2D2D2D2D				
008082	2D2D2D2D 2D2D2D2D				
00808A	3C0D				
00808C	7C202020 20202020	C	40	DB	"  E R R O R  ",CR.asc
008094	20202020 20202020				
00809C	20202020 20202020				
0080A4	20202020 20202020				
0080AC	20452052 2052204F				
0080B4	20522020 20202020				
0080BC	20202020 20202020				
0080C4	20202020 20202020				
0080CC	20202020 20202020				
0080D4	7C0D				
0080D6	7C202020 20202020	C	41	DB	"  data volume no directory.  ",CR.asc
0080DE	20202020 20202020				
0080E6	20202020 20202020				
0080EE	64617461 20766F6C				
0080F6	756D6520 6E6F2064				
0080FE	69726563 746F7279				
008106	2E202020 20202020				
00810E	20202020 20202020				
008116	20202020 20202020				
00811E	7C0D				
008120	3E2D2D2D 2D2D2D2D	C	42	DB	">-----<",CR.asc



\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\screens\ipl_error_screen.s
008128	2D2D2D2D 2D2D2D2D				
008130	2D2D2D2D 2D2D2D2D				
008138	2D2D2D2D 2D2D2D2D				
008140	2D2D2D2D 2D2D2D2D				
008148	2D2D2D2D 2D2D2D2D				
008150	2D2D2D2D 2D2D2D2D				
008158	2D2D2D2D 2D2D2D2D				
008160	2D2D2D2D 2D2D2D2D				
008168	3C0D				
00816A	03	C	43	DB ETX.asc	
		C	0	include "ipl_sessions.s"	; Include menu for session connection selec
		C	1		
		C	2	; Use which terminal for your session connection?	
		C	3		
00816B	0D	C	4	whichterminal\$ db CR.asc	
00816C	0A	C	5	db LF.asc	
00816D	20202020 20202020	C	6	DB " S E S S I O N C O N N E C T I O N",CR.asc	
008175	53204520 53205320				
00817D	49204F20 4E202043				
008185	204F204E 204E2045				
00818D	20432054 2049204F				
008195	204E0D				
008198	20202020 20202020	C	7	db " _____",CR.asc,LF.asc	
0081A0	5F5F5F5F 5F5F5F5F				
0081A8	5F5F5F5F 5F5F5F5F				
0081B0	5F5F5F5F 5F5F5F5F				
0081B8	5F5F5F5F 5F5F5F5F				
0081C0	5F5F0D0A				
		C	8		
0081C4	2020302E 20446F20	C	9	db ' 0. Do not load a terminal session.',CR.asc	
0081CC	6E6F7420 6C6F6164				
0081D4	20612074 65726D69				
0081DC	6E616C20 73657373				
0081E4	696F6E2E 0D				
0081E9	2020312E 20492061	C	10	db ' 1. I am connecting with grandpas teletype (TTY).',CR.asc	
0081F1	6D20636F 6E6E6563				
0081F9	74696E67 20776974				
008201	68206772 616E6470				
008209	61732074 656C6574				
008211	79706520 28545459				
008219	292E0D				
00821C	2020322E 20557365	C	11	db ' 2. Use ADDS Regent 25 standard (ADDS25).',CR.asc	
008224	20414444 53205265				
00822C	67656E74 20323520				
008234	7374616E 64617264				
00823C	20284144 44533235				
008244	292E0D				
008247	2020332E 20557365	C	12	db ' 3. Use real model 4 or emulator (TERM6).',CR.asc	
00824F	20726561 6C206D6F				
008257	64656C20 34206F72				
00825F	20656D75 6C61746F				
008267	72202854 45524D36				
00826F	292E0D				
008272	2020342E 20565431	C	13	db ' 4. VT100 real time screen updating.',CR.asc	
00827A	30302072 65616C20				
008282	74696D65 20736372				
00828A	65656E20 75706461				
008292	74696E67 2E0D				
008298	2020352E 20244249	C	14	db ' 5. \$BIT Binary Instrumentation Telemetry.',CR.asc	
0082A0	54204269 6E617279				

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl_sessions.s
0082A8	20496E73 7472756D				
0082B0	656E7461 74696F6E				
0082B8	2054656C 656D6574				
0082C0	72792E0D				
0082C4	0D	C	15	db	CR.asc
0082C5	20455343 206F7220	C	16	db	" ESC or enter connection method: "
0082CD	656E7465 7220636F				
0082D5	6E6E6563 74696F6E				
0082DD	206D6574 686F643A				
0082E5	20				
0082E6	03	C	17	db	ETX.asc ; Tell pump macro this is end of screen.
		C	18		
0082E7		C	19	set_terminal	GOSUB scrolloff
		C+	19	scope	
0082E7	CD ED 6E	C+	19	CALL	scrolloff
		C+	19	endmac	GOSUB
		C	20	PUMP	DEVICE.console,whichterminal\$
		C+	20	scope	; A tribute to Dave Shriner - Display Data Corp.
0082EA	21 6B 81	C+	20	LD	HL,whichterminal\$ ; Pump from this starting point until ETX.asc is found.
		C+	20	if	DEVICE.console=0
0082ED	7E	C+	20	LD	A,(HL) ; Get byte to send.
0082EE	23	C+	20	INC	HL ; pointer=pointer+1
		C+	20	CP	EOT.asc ; End Of Transmission (block)?
		C+	20	jr	Z,\$0a ; Return if finished.
0082EF	FE03	C+	20	CP	ETX.asc ; End Of Text?
0082F1	28 16	C+	20	jr	Z,__5450a ; Return if finished.
0082F3	FE0D	C+	20	CP	CR.asc ; Was it a CR?
0082F5	CC FF 82	C+	20	call	z,__5451 ; 1st add a LF to stream.
0082F8	4F	C+	20	LD	C,A ; Output byte in C.
0082F9	AF	C+	20	XOR	A ; Set Z flag for put.
0082FA	CD D9 10	C+	20	CALL	U0DVR ; Out the UART we go.
0082FD	18 EE	C+	20	JR	__5450
0082FF	3E0A	C+	20	LD	a,LF.asc ; Get a LF.
008301	4F	C+	20	LD	C,A ; Output byte in C.
008302	AF	C+	20	XOR	A ; Set Z flag for put
008303	CD D9 10	C+	20	CALL	U0DVR ; Out the UART we go.
008306	3E0D	C+	20	ld	a,CR.asc ; Now put a CR back in stream.
008308	C9	C+	20	ret	; Return and print CR.
008309		C+	20	endif	
		C+	20	if	DEVICE.console=1
		C+	20	LD	A,(HL) ; Get byte to send.
		C+	20	INC	HL ; pointer=pointer+1
		C+	20	CP	EOT.asc ; End Of Transmission (block)?
		C+	20	jr	Z,\$1a ; Return if finished.
		C+	20	CP	ETX.asc ; End Of Text?
		C+	20	jr	Z,__5451a ; Return if finished.
		C+	20	LD	C,A ; Output byte to C.
		C+	20	XOR	A ; Set Z flag for put.
		C+	20	CALL	U1DVR ; Out the UART we go.
		C+	20	JR	__5451
		C+	20	__5451a	equ \$
		C+	20	endif	
		C+	20	ENDMAC	PUMP
008309		C	21	getterm	GOSUB getchar
		C+	21	scope	
008309	CD 7E 6F	C+	21	CALL	getchar
		C+	21	endmac	GOSUB
00830C	FE1B	C	22	cp	ESC.asc ; Did they hit escape?
		C	23	IF_EQ_GOTO	main_ipl_menu
		C+	23	scope	

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source
00830E	CA 00 48	C+	23	jp z,main_ipl_menu
		C+	23	endmac IF_EQ_GOTO
		C	24	
008311	FE30	C	25	term_setup cp '0' ; 0 - NO terminal.
		C	26	IF_EQ_GOTO defaultterm
		C+	26	scope
008313	CA 2D 83	C+	26	jp z,defaultterm
		C+	26	endmac IF_EQ_GOTO
008316	FE31	C	27	cp '1' ; 1 - Dumb TTY configuration?
		C	28	IF_EQ_BRANCH ttyterm
		C+	28	scope
008318	28 24	C+	28	jr z,ttyterm
		C+	28	endmac IF_EQ_BRANCH
00831A	FE32	C	29	cp '2' ; 2 - ADDS 25 configuration?
		C	30	IF_EQ_BRANCH addstern
		C+	30	scope
00831C	28 31	C+	30	jr z,addstern
		C+	30	endmac IF_EQ_BRANCH
00831E	FE33	C	31	cp '3' ; 3 - Term6 special model 4 connection.
		C	32	IF_EQ_BRANCH term6v
		C+	32	scope
008320	28 3E	C+	32	jr z,term6v
		C+	32	endmac IF_EQ_BRANCH
008322	FE34	C	33	cp '4' ; 4 - Load default SYSGEN.
		C	34	IF_EQ_BRANCH realtim
		C+	34	scope
008324	28 4B	C+	34	jr z,realtim
		C+	34	endmac IF_EQ_BRANCH
008326	FE35	C	35	cp '5' ; 5 - Startup in \$BIT mode.
		C	36	IF_EQ_GOTO dollar_bit
		C+	36	scope
008328	CA 86 83	C+	36	jp z,dollar_bit
		C+	36	endmac IF_EQ_GOTO
		C	37	
		C	38	BRANCH getterm
		C+	38	scope
00832B	18 DC	C+	38	jr getterm
		C+	38	endmac BRANCH
		C	39	
00832D		C	40	defaultterm MVC defltterm\$,session_cmd\$,15
		C+	40	scope
00832D	21 D7 83	C+	40	ld hl,defltterm\$
008330	11 F7 83	C+	40	ld de,session_cmd\$
008333	010F00	C+	40	ld bc,15
008336	EDB0	C+	40	LDIR
		C+	40	ENDMAC MVC
008338	3E00	C	41	ld a,0
00833A	320300	C	42	ld (quiet_default_flg),a
00833D	C9	C	43	ret
00833E		C	44	ttyterm MVC dmbtty\$,session_cmd\$,15 ; Move dumtty into SYSGEN file name.
		C+	44	scope
00833E	21 A7 83	C+	44	ld hl,dmbtty\$
008341	11 F7 83	C+	44	ld de,session_cmd\$
008344	010F00	C+	44	ld bc,15
008347	EDB0	C+	44	LDIR
		C+	44	ENDMAC MVC
008349	3E01	C	45	ld a,1
00834B	320300	C	46	ld (quiet_default_flg),a
00834E	C9	C	47	ret
00834F		C	48	addstern MVC adds25\$,session_cmd\$,15 ; Move ADDS 25 SYSGEN file name.

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl_sessions.s
		C+	48		scope
00834F	21 97 83	C+	48		ld hl,adds25\$
008352	11 F7 83	C+	48		ld de,session_cmd\$
008355	010F00	C+	48		ld bc,15
008358	EDB0	C+	48		LDIR
		C+	48		ENDMAC MVC
00835A	3E02	C	49		ld a,2
00835C	320300	C	50		(quiet_default_flg),a
00835F	C9	C	51		ret
008360		C	52	term6v	MVC term6v\$,session_cmd\$,15 ; Move special TERM6 file name.
		C+	52		scope
008360	21 B7 83	C+	52		ld hl,term6v\$
008363	11 F7 83	C+	52		ld de,session_cmd\$
008366	010F00	C+	52		ld bc,15
008369	EDB0	C+	52		LDIR
		C+	52		ENDMAC MVC
00836B	3E03	C	53		ld a,3
00836D	320300	C	54		(quiet_default_flg),a
008370	C9	C	55		ret
008371	3E04	C	56	realtim	ld a,4
008373	320300	C	57		(quiet_default_flg),a
		C	58		MVC realtim\$,session_cmd\$,15
		C+	58		scope
008376	21 E7 83	C+	58		ld hl,realtim\$
008379	11 F7 83	C+	58		ld de,session_cmd\$
00837C	010F00	C+	58		ld bc,15
00837F	EDB0	C+	58		LDIR
		C+	58		ENDMAC MVC
008381	AF	C	59		xor a ; A=0 or NOP.
008382	32 85 F7	C	60		ld (pump_switch_value),a ; Enable pump switch, this updates display in real time.
008385	C9	C	61		ret
008386		C	62	dollar_bit	
008386	3E05	C	63		ld a,5
008388	320300	C	64		(quiet_default_flg),a
		C	65		MVC bit_boot\$,session_cmd\$,15 ; Move special TERM6 file name.
		C+	65		scope
00838B	21 C7 83	C+	65		ld hl,bit_boot\$
00838E	11 F7 83	C+	65		ld de,session_cmd\$
008391	010F00	C+	65		ld bc,15
008394	EDB0	C+	65		LDIR
		C+	65		ENDMAC MVC
008396	C9	C	66		ret
		C	67		
008397	646F2048 4F535432	C	68	adds25\$	db "do HOST25 ",CR.asc
00839F	35202020 2020200D				
0083A7	646F2048 4F535454	C	69	dmbtty\$	db "do HOSTTTY ",CR.asc
0083AF	54592020 2020200D				
0083B7	646F2048 4F535436	C	70	term6v\$	db "do HOST6V ",CR.asc
0083BF	56202020 2020200D				
0083C7	2E52756E 6E696E67	C	71	bit_boot\$db	".Running \$BIT ",CR.asc
0083CF	20244249 5420200D				
0083D7	20202020 20202020	C	72	defltterm\$	db " ",CR.asc
0083DF	20202020 2020200D				
0083E7	636C7320 20202020	C	73	realtim\$ db	"cls ",CR.asc
0083EF	20202020 2020200D				
0083F7	20202020 20202020	C	74	session_cmd\$ db	" ",CR.asc
0083FF	20202020 2020200D				
		C	0		include "config_menu_screen.s" ; Menu to enter CONFIG to load.
		C	1		
		C	2		; Load CONFIG menu.

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\screens\config_menu_screen.s
		C	3		
008407	0D0A	C	4	whichsysmsg\$	db CR.asc,LF.asc
008409	53595347 454E2073	C	5	db	"SYSGEN selection.",CR.asc
008411	656C6563 74696F6E				
008419	2E0D				
00841B	0D	C	6	db	CR.asc
00841C	20456E74 65722063	C	7	db	" Enter custom name of saved SYSGEN.",CR.asc
008424	7573746F 6D206E61				
00842C	6D65206F 66207361				
008434	76656420 53595347				
00843C	454E2E0D				
008440	0D	C	8	db	CR.asc
008441	203E2043 7573746F	C	9	db	" > Custom file name must be a valid TRSDOS filename syntax",CR.asc
008449	6D206669 6C65206E				
008451	616D6520 6D757374				
008459	20626520 61207661				
008461	6C696420 54525344				
008469	4F532066 696C656E				
008471	616D6520 73796E74				
008479	61780D				
00847C	203E2045 78616374	C	10	db	" > Exactly 6 characters long",CR.asc
008484	6C792036 20636861				
00848C	72616374 65727320				
008494	6C6F6E67 0D				
008499	203E2055 73696E67	C	11	db	" > Using all upper case.",CR.asc
0084A1	20616C6C 20757070				
0084A9	65722063 6173652E				
0084B1	0D				
0084B2	203E2054 5253444F	C	12	DB	" > TRSDOS will add extension /SYS.",CR.asc,CR.asc
0084BA	53207769 6C6C2061				
0084C2	64642065 7874656E				
0084CA	73696F6E 202F5359				
0084D2	532E0D0D				
0084D6	203E2045 78616D70	C	13	db	" > Example, TEST01.",CR.asc
0084DE	6C652C20 54455354				
0084E6	30312E0D				
0084EA	0D	C	14	db	CR.asc
0084EB	206F7220 454E5445	C	15	db	" or ENTER to use default filename CONFIG/SYS.",CR.asc
0084F3	5220746F 20757365				
0084FB	20646566 61756C74				
008503	2066696C 656E616D				
00850B	6520434F 4E464947				
008513	2F535953 2E0D				
008519	0D	C	16	db	CR.asc
00851A	03	C	17	db	ETX.asc
		C	18		; Tell pump this is end of screen.
00851B	456E7465 72206375	C	19	pick_config\$	db "Enter custom SYSGEN filename: ",ETX.asc
008523	73746F6D 20535953				
00852B	47454E20 66696C65				
008533	6E616D65 3A2003				
		C	20		
00853A	4C6F6164 20535953	C	21	ldSYSGN\$	db 'Load SYSGEN? (y/n)?',ETX.asc
008542	47454E3F 2028792F				
00854A	6E293F03				
		C	22		
00854E	2A2A2053 59534745	C	23	CONFIG\$	DB "*** SYSGEN ***",ETX.asc ; Config DSP.
008556	4E202A2A 03				
00855B	73797367 656E2065	C	24	sysgener	db "sysgen error",ETX.asc
008563	72726F72 03				
		C	25		

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\screens\config_menu_screen.s
008568	434F4E46 4947	C	26	SYSGN0\$	db "CONFIG"
		C	27		
00856E	4C6F6164 696E672D	C	28	lding\$	db "Loading-> SYSGEN/SYS",ETX.asc
008576	3E205359 5347454E				
00857E	2F535953 03				
008583	3C2D206C 6F616420	C	29	ldingdn\$	db "<- load finished.",CR.asc,ETX.asc
00858B	66696E69 73686564				
008593	2E0D03				
		C	0		include "ipl-strings.s" ; IPL messages, screens, texts.
008596	5741524D	C	1	from_cpm	db "WARM"
00859A	434F4C44	C	2	from_por	db "COLD"
		C	3		
00859E	655A3830 463931	C	4	cpu_f91_msg	db "eZ80F91"
0085A5	655A3830 463932	C	5	cpu_f92_msg	DB "eZ80F92"
		C	6		
0085AC	43697263 6C652D4D	C	7	vendor_cirm\$	db "Circle-M"
0085B4	4D616964 656E5F31	C	8	vendor_dinno\$	db "Maiden_1"
0085BC	41474F4E 20202020	C	9	vendor_agon\$	db "AGON "
0085C4	5A494C4F 47646576	C	10	vendor_z\$db	"ZILOGdev"
0085CC	556E6465 66696E65	C	11	vendor_undef\$	db "Undefine"
		C	12		
		C	13		
0085D4	53746172 74206261	C	14	stbgrnd\$	db 'Start background processing (y/n)?',ETX.asc
0085DC	636B6772 6F756E64				
0085E4	2070726F 63657373				
0085EC	696E6720 28792F6E				
0085F4	293F03				
0085F7	4261636B 67726F75	C	15	bkgrd_start\$	db 'Background multitasking -> ',ETX.asc
0085FF	6E64206D 756C7469				
008607	7461736B 696E6720				
00860F	2D3E2003				
008613	6E6F7720 72756E6E	C	16	bkgrd_run\$	db 'now running.',CR.asc,ETX.asc
00861B	696E672E 0D03				
008621	4E4F5420 72756E6E	C	17	bkgrd_fail\$	db 'NOT running.',CR.asc,ETX.asc
008629	696E672E 0D03				
		C	18		
00862F	45786563 7574696E	C	19	exinitial\$	db 'Executing ICNFG initialization chain, ',ETX.asc
008637	67204943 4E464720				
00863F	696E6974 69616C69				
008647	7A617469 6F6E2063				
00864F	6861696E 2C2003				
008656	66696E69 73686564	C	20	initialdne\$	db 'finished.',CR.asc,ETX.asc
00865E	2E0D03				
		C	21		
008661	3C4E4F20 4155544F	C	22	no_auto\$	db "<NO AUTO COMMAND>"
008669	20434F4D 4D414E44				
008671	3E				
008672	52756E20 4155544F	C	23	runauto\$	db 'Run AUTO command (y/n)?',ETX.asc
00867A	20636F6D 6D616E64				
008682	2028792F 6E293F03				
		C	24		
00868A	49504C20 636F6D70	C	25	ipldone\$	db "IPL complete, starting LS-DOS.",CR.asc,ETX.asc
008692	6C657465 2C207374				
00869A	61727469 6E67204C				
0086A2	532D444F 532E0D03				
		C	26		
0086AA	4469736B 4449534B	C	27	string1\$	db "DiskDISK"
0086B2	494D4720 20202020	C	28	jv1\$	db "IMG "
0086BA	4E6F2064 61746120	C	29	novolavail	db "No data volume "
0086C2	766F6C75 6D652020				

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\ipl-strings.s
0086CA	20				
0086CB	6E6F6E65 20202020	C	30	nodatatyp db	"none "
		C	31		
0086D3	303030	C	32	XXX DB	"000"
0086D6	20202020 0D0A	C	33	PARTYR DB	"", CR.asc, LF.asc
0086DC	44617465 20202020	C	34	DATEPR DB	"Date ", CR.asc
0086E4	20202020 2020200D				
0086EC	54696D65 20202020	C	35	TIMEPR DB	"Time ", CR.asc
0086F4	20202020 200D				
0086FA	0A0D03	C	36	crlf db	LF.asc, CR.asc, ETX.asc
		C	37		
0086FD	00	C	38	initialflg db	0
0086FE	00	C	39	score db	0
0086FF	6E0D	C	40	safeipl\$ db	"n", CR.asc
008701	1B5B4A03	C	41	vt100_clear\$ db	ESC.asc, "[J", ETX.asc ; Clear video screen.
008705	1B5B6603	C	42	vt100_home\$ DB	ESC.asc, "[f", ETX.asc ; Cursor to home position.
		C	43		
008709	0000	C	44	quiet_default dw	0 ; Default value for IPL in quiet mode.
		C	45		
00870B	00 00 00 00 00 00	C	46	slicetable blkb	27,0
008711	00 00 00 00 00 00				
008717	00 00 00 00 00 00				
00871D	00 00 00 00 00 00				
008723	00 00 00				
		C	0	include "ipl_buffers.s"	; Buffers used during IPL of system.
		C	1		
		C	2	; TRS-OS buffers for IPL.	
		C	3		
008726		C	4	slicebuf ds	256
008826		C	5	volbuf ds	256
008926		C	6	savedct ds	80
008976	20 20 20 20 20 20	C	7	spaceout\$blkb	79,20h
00897C	20 20 20 20 20 20				
008982	20 20 20 20 20 20				
008988	20 20 20 20 20 20				
00898E	20 20 20 20 20 20				
008994	20 20 20 20 20 20				
00899A	20 20 20 20 20 20				
0089A0	20 20 20 20 20 20				
0089A6	20 20 20 20 20 20				
0089AC	20 20 20 20 20 20				
0089B2	20 20 20 20 20 20				
0089B8	20 20 20 20 20 20				
0089BE	20 20 20 20 20 20				
0089C4	20				
		B	47		
		B	48	; At runtime one of two vector tables (f91/f92) is installed in HIGH\$ memory.	
		B	49		
		C	0	include "f91_interrupt_vectors.s"	; f91 Interrupt vector tables.
	000089C5	C	1	f91INTERRUPT.table EQU	\$
		C	2		
0089C5	86F70000	C	3	priority0 dl	EMAC.RX.handler
0089C9	86F70000	C	4	priority1 dl	EMAC.TX..handler
0089CD	86F70000	C	5	priority2 dl	EMAC.SYS.handler
0089D1	86F70000	C	6	priority3 dl	PLL.handler
0089D5	86F70000	C	7	priority4 dl	FLASH.handler
0089D9	86F70000	C	8	priority5 dl	TIMER0.handler
0089DD	00F70000	C	9	priority6 dl	f91_TIMER1.handler
0089E1	86F70000	C	10	priority7 dl	TIMER2.handler
0089E5	86F70000	C	11	priority8 dl	TIMER3.handler

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYSRES\interrupts\f91_interrupt_vectors.s
0089E9	86F70000	C	12	priority9	dl STRAY.handler ; Unused vector.
0089ED	86F70000	C	13	priority10	dl STRAY.handler ; Unused vector.
0089F1	86F70000	C	14	priority11	dl RTC.handler
0089F5	86F70000	C	15	priority12	dl UART0.handler
0089F9	86F70000	C	16	priority13	dl UART1.handler
0089FD	86F70000	C	17	priority14	dl I2C.handler
008A01	86F70000	C	18	priority15	dl SPI.handler
008A05	86F70000	C	19	priority16	dl PORT.A.BIT0.handler
008A09	86F70000	C	20	priority17	dl PORT.A.BIT1.handler
008A0D	86F70000	C	21	priority18	dl PORT.A.BIT2.handler
008A11	86F70000	C	22	priority19	dl PORT.A.BIT3.handler
008A15	86F70000	C	23	priority20	dl PORT.A.BIT4.handler
008A19	86F70000	C	24	priority21	dl PORT.A.BIT5.handler
008A1D	86F70000	C	25	priority22	dl PORT.A.BIT6.handler
008A21	86F70000	C	26	priority23	dl PORT.A.BIT7.handler
008A25	86F70000	C	27	priority24	dl PORT.B.BIT0.handler
008A29	86F70000	C	28	priority25	dl PORT.B.BIT1.handler
008A2D	86F70000	C	29	priority26	dl PORT.B.BIT2.handler
008A31	86F70000	C	30	priority27	dl PORT.B.BIT3.handler
008A35	86F70000	C	31	priority28	dl PORT.B.BIT4.handler
008A39	86F70000	C	32	priority29	dl PORT.B.BIT5.handler
008A3D	86F70000	C	33	priority30	dl PORT.B.BIT6.handler
008A41	86F70000	C	34	priority31	dl PORT.B.BIT7.handler
008A45	86F70000	C	35	priority32	dl PORT.C.BIT0.handler
008A49	86F70000	C	36	priority33	dl PORT.C.BIT1.handler
008A4D	86F70000	C	37	priority34	dl PORT.C.BIT2.handler
008A51	86F70000	C	38	priority35	dl PORT.C.BIT3.handler
008A55	86F70000	C	39	priority36	dl PORT.C.BIT4.handler
008A59	86F70000	C	40	priority37	dl PORT.C.BIT5.handler
008A5D	86F70000	C	41	priority38	dl PORT.C.BIT6.handler
008A61	86F70000	C	42	priority39	dl PORT.C.BIT7.handler
008A65	86F70000	C	43	priority40	dl PORT.D.BIT0.handler
008A69	86F70000	C	44	priority41	dl PORT.D.BIT1.handler
008A6D	86F70000	C	45	priority42	dl PORT.D.BIT2.handler
008A71	86F70000	C	46	priority43	dl PORT.D.BIT3.handler
008A75	86F70000	C	47	priority44	dl PORT.D.BIT4.handler
008A79	86F70000	C	48	priority45	dl PORT.D.BIT5.handler
008A7D	86F70000	C	49	priority46	dl PORT.D.BIT6.handler
008A81	86F70000	C	50	priority47	dl PORT.D.BIT7.handler
		C	0	include "f92_interrupt_vectors.s"	; f92 Interrupt vector tables.
		C	1		
	00008A85	C	2	f92INTERRUPT.table	EQU \$
		C	3		
008A85	86F7	C	4	f92_priority0	dw STRAY.handler
008A87	86F7	C	5	f92_priority1	dw STRAY.handler
008A89	86F7	C	6	f92_priority2	dw STRAY.handler
008A8B	86F7	C	7	f92_priority3	dw STRAY.handler
008A8D	86F7	C	8	f92_priority4	dw FLASH.handler
008A8F	86F7	C	9	f92_priority5	dw TIMER0.handler
008A91	00F7	C	10	f92_priority6	dw f92_TIMER1.handler
008A93	86F7	C	11	f92_priority7	dw TIMER2.handler
008A95	86F7	C	12	f92_priority8	dw TIMER3.handler
008A97	86F7	C	13	f92_priority9	dw TIMER4.handler ; Unused vector.
008A99	86F7	C	14	f92_priority10	dw TIMER5.handler ; Unused vector.
008A9B	86F7	C	15	f92_priority11	dw RTC.handler
008A9D	86F7	C	16	f92_priority12	dw UART0.handler
008A9F	86F7	C	17	f92_priority13	dw UART1.handler
008AA1	86F7	C	18	f92_priority14	dw I2C.handler
008AA3	86F7	C	19	f92_priority15	dw SPI.handler
008AA5	86F7	C	20	f92_priority16	dw PORT.A.BIT0.handler



\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYSRES\interrupts\f92_interrupt_vectors.s
008AA7	86F7	C	21	f92_priority17	dw PORT.A.BIT1.handler
008AA9	86F7	C	22	f92_priority18	dw PORT.A.BIT2.handler
008AAB	86F7	C	23	f92_priority19	dw PORT.A.BIT3.handler
008AAD	86F7	C	24	f92_priority20	dw PORT.A.BIT4.handler
008AAF	86F7	C	25	f92_priority21	dw PORT.A.BIT5.handler
008AB1	86F7	C	26	f92_priority22	dw PORT.A.BIT6.handler
008AB3	86F7	C	27	f92_priority23	dw PORT.A.BIT7.handler
008AB5	86F7	C	28	f92_priority24	dw PORT.B.BIT0.handler
008AB7	86F7	C	29	f92_priority25	dw PORT.B.BIT1.handler
008AB9	86F7	C	30	f92_priority26	dw PORT.B.BIT2.handler
008ABB	86F7	C	31	f92_priority27	dw PORT.B.BIT3.handler
008ABD	86F7	C	32	f92_priority28	dw PORT.B.BIT4.handler
008ABF	86F7	C	33	f92_priority29	dw PORT.B.BIT5.handler
008AC1	86F7	C	34	f92_priority30	dw PORT.B.BIT6.handler
008AC3	86F7	C	35	f92_priority31	dw PORT.B.BIT7.handler
008AC5	86F7	C	36	f92_priority32	dw PORT.C.BIT0.handler
008AC7	86F7	C	37	f92_priority33	dw PORT.C.BIT1.handler
008AC9	86F7	C	38	f92_priority34	dw PORT.C.BIT2.handler
008ACB	86F7	C	39	f92_priority35	dw PORT.C.BIT3.handler
008ACD	86F7	C	40	f92_priority36	dw PORT.C.BIT4.handler
008ACF	86F7	C	41	f92_priority37	dw PORT.C.BIT5.handler
008AD1	86F7	C	42	f92_priority38	dw PORT.C.BIT6.handler
008AD3	86F7	C	43	f92_priority39	dw PORT.C.BIT7.handler
008AD5	86F7	C	44	f92_priority40	dw PORT.D.BIT0.handler
008AD7	86F7	C	45	f92_priority41	dw PORT.D.BIT1.handler
008AD9	86F7	C	46	f92_priority42	dw PORT.D.BIT2.handler
008ADB	86F7	C	47	f92_priority43	dw PORT.D.BIT3.handler
008ADD	86F7	C	48	f92_priority44	dw PORT.D.BIT4.handler
008ADF	86F7	C	49	f92_priority45	dw PORT.D.BIT5.handler
008AE1	86F7	C	50	f92_priority46	dw PORT.D.BIT6.handler
008AE3	86F7	C	51	f92_priority47	dw PORT.D.BIT7.handler
		B	52		
		B	53		
		B	54	;----- Test Code.-----	
		B	55		
008AE5		B	56	test1;	stat_server_x
008AE5		B	57	test2;	bind_server_x
008AE5		B	58	test3;	query_server_x
008AE5		B	59	test4;	echo echo_on_exit,echo_on_error
008AE5		B	60	echo_on_exit	put.DSP.LINE echo_done\$
008AE5	21 EE 8A	B+	60	ld hl,echo_done\$	; Get the buffer address. Any valid ld hl,
		B+	60	SVC svc_dsply	; Ask OS to display up to 1st
008AE8	3E0A	B+	60	ld a,svc_dsply	
008AEA	EF	B+	60	restart28	
		B+	60	ENDMAC SVC	
		B+	60	ENDMAC	put.DSP.LINE ; 0Dh or 04h (CR or ETX).
008AEB	3E0D	B	61	ld a,CR.asc	
		B	62	RETURN	
		B+	62	scope	
008AED	C9	B+	62	ret	; Return from call.
		B+	62	endmac	RETURN
008AEE	20454348 4F20636F	B	63	echo_done\$ db	" ECHO command done.",CR.asc
008AF6	6D6D616E 6420646F				
008AFE	6E652E0D				
008B02		B	64	echo_on_error	error.ABORT a ; Report error, there is no return.
		B+	64	scope	
008B02	4F	B+	64	ld c,a	
		B+	64	SVC svc_error	
008B03	3E1A	B+	64	ld a,svc_error	
008B05	EF	B+	64	restart28	

\*\*\*\*\* TRSDOS \*\*\*\*\*

&lt; SYS0 IPL - DOS-initialization code &amp; welcome message &gt;

PC	Object	I	Line	Source	.. \SYS0_IPL\IPL_Code.s
		B+	64	ENDMAC	SVC
		B+	64	endmac	error.ABORT
008B06		B	65	test5;	ping ping_on_exit, ping_on_error
008B06		B	66	ping_on_error	error.ABORT a ; Report error, there is no return.
		B+	66	scope	
008B06	4F	B+	66	ld	c, a
		B+	66	SVC	svc_error
008B07	3E1A	B+	66	ld	a, svc_error
008B09	EF	B+	66	restart28	
		B+	66	ENDMAC	SVC
		B+	66	endmac	error.ABORT
008B0A		B	67	ping_on_exit	put.DSP.LINE ping_done\$ ; Display message line.
008B0A	21 13 8B	B+	67	ld	hl, ping_done\$ ; Get the buffer address. Any valid ld hl,
		B+	67	SVC	svc_dsply ; Ask OS to display up to 1st
008B0D	3E0A	B+	67	ld	a, svc_dsply
008B0F	EF	B+	67	restart28	
		B+	67	ENDMAC	SVC
		B+	67	ENDMAC	put.DSP.LINE ; 0Dh or 04h (CR or ETX).
008B10	3E0D	B	68	ld	a, CR.asc
		B	69	RETURN	
		B+	69	scope	
008B12	C9	B+	69	ret	; Return from call.
		B+	69	endmac	RETURN
008B13	2050494E 4720636F	B	70	ping_done\$	db " PING command done.", CR.asc
008B1B	6D6D616E 6420646F				
008B23	6E652E0D				
		B	71		
		B	72		include "stat_server.s"
		B	73		include "bind_server.s"
		B	74		include "query_server.s"
		B	75		include "macro_ECHO.s"
		B	76		include "macro_PING.s"
		B	77		
		B	78		;----- END of TEST CODE -----
		B	79		
		A	151		
008B27	11 E0 00	A	152	LD	DE, CFGFCB\$ ; Set up to load config.
		A	153	GOSUB	zLOAD ; Go to load config.
		A+	153	scope	
008B2A	CD 38 1B	A+	153	CALL	zLOAD
		A+	153	endmac	GOSUB
		A	154		
		A	155	GOTO	zICNFG ; Here we start initilization chain.
		A+	155	scope	
008B2D	C3 86 00	A+	155	jp	zICNFG
		A+	155	endmac	GOTO
		A	156	SUBTITLE	"< SYSRES HIGH memory region. >"
		A	157		
		A	158		; HIGH\$ memory contains video RAM, interrupt handler, reboot code etc.
		A	159		
		A	160	ORG	INTERRUPT.handler-0bh
00F5F5	18 FE	A	161	JR	\$
00F5F7	FFFF	A	162	DW	ENDRAM\$
00F5F9	06545253 2D4F53	A	163	DB	6, "TRS-OS"
		B	0	include	"interrupt_handler.s" ; Container for code handling interrupts.
		B	1	org	INTERRUPT.handler ; Beginning of handler + vector table.
00F600		B	2	ds	256 ; Vector table 192 bytes for f91 or 96 bytes for f92 will be relocated
		B	3		; this block of memory on IPL.
		B	4	org	INTERRUPT.handler+256 ; Start of actual interrupt handler code.
		B	5		

\*\*\*\*\* TRSDOS \*\*\*\*\*  
 < SYSRES HIGH memory region. >

PC	Object	I	Line	Source	Comment
	0000F700	B	6	f91_TIMER1.handler	equ \$
	0000F700	B	7	f92_TIMER1.handler	equ \$
		B	8		
00F700	F5	B	9	push af	; Save register pairs used in background tasker.
00F701	E5	B	10	push hl	
00F702	C5	B	11	push bc	
00F703	D5	B	12	push de	
		B	13		
		B	14	INC_32 int_tmr1_cnt	; Beat 32 bit heart counter (located below).
		B+	14	scope	
00F704	21 80 F7	B+	14	ld hl,int_tmr1_cnt	; 32 bit interrupt counter.
00F707	34	B+	14	inc (hl)	; INC LSB.
00F708	20 0A	B+	14	jr nz,end_inc_32	; Did not roll over to zero?
00F70A	23	B+	14	inc hl	; Bump to next digit.
00F70B	34	B+	14	inc (hl)	; Increase by one since we had carry from last digit.
00F70C	20 06	B+	14	jr nz,end_inc_32	; Did this digit carry?
00F70E	23	B+	14	inc hl	; Bump pointer to next digit.
00F70F	34	B+	14	inc (hl)	; Increase counter by 1.
00F710	20 02	B+	14	jr nz,end_inc_32	; Test if that increase caused carry to next column.
00F712	23	B+	14	inc hl	; Bump ponter to next column.
00F713	34	B+	14	inc (hl)	; INC MSB of 32 bit 4 byte counter if needed.
	0000F714	B+	14	end_inc_32 EQU \$	
		B+	14	ENDMAC INC_32	; End of macro.
		B	15		
		B	16		; At this point if Z flag is set then we have a winner winner of our best chicken liver dinner.
		B	17		; Livers wrapped in bacon & fried crisp served with salad, mashed potatoes, mixed veggies and a rol
		B	18		; If Z flag set it means system has been up long enough that interrupts @ 60 times per second has o
		B	19		; It would take years of uptime to roll this counter over.
		B	20		
		B	21		; IF Z = this 32 bit counter has now rolled over to 0000 0000h.
		B	22		
00F714	3A 2B 00	B	23	wp_check ld a,(TIMSL\$)	; Here we will check if vol 0 should be write protected. If yes
00F717	FEFF	B	24	cp 0FFh	; System fast or slow? Fast=55, slow=FF
		B	25	IF_EQ_BRANCH pump_check	; If drive does not need WP, branch.
		B+	25	scope	
00F719	28 05	B+	25	jr z,pump_check	
		B+	25	endmac IF_EQ_BRANCH	
00F71B	21 73 04	B	26	ld hl,DCT0\$+3	; Address of drive 0 DCT WP byte.
00F71E	CBFE	B	27	set 7,(hl)	; Turn ON WP.
		B	28		
00F720	CD 40 F7	B	29	pump_check call pump_switch	; Dudes this is it, check switch & video pump counter, see
		B	30		
00F723	3A 79 00	B	31	ld a,(PFLAG\$)	; Determine which processor we are on (F91 or F92)?
00F726	CB2F	B	32	sra a	; Unpack.
00F728	FE01	B	33	cp 01h	; PFLAG\$ was set to 91h or 92h on IPL depending on processor.
00F72A	28 05	B	34	jr z,f91tmr1ack	; Were running on f91. Handle it like a 91.
		B	35		
00F72C	ED3883	B	36	in0 a,(f92_TMR1_CTL)	; OK, so were a 92. Handle it like were a 92, read to reset int
00F72F	18 03	B	37	jr call_rst38	; Skip past eZ80f91 code to reset interrupts.
		B	38		
00F731	ED3867	B	39	f91tmr1ack in0 a,(TMR1_IIR)	; eZ80f91 read to reset.
		B	40		
00F734	21 3C 00	B	41	call_rst38 ld hl,INTIM\$	; Tell TRSDOS wakeup, PRT1 fired with 60hz interrupt using Inte
00F737	CBEE	B	42	set 5,(hl)	
		B	43		
00F739	D1	B	44	pop de	
00F73A	C1	B	45	pop bc	
00F73B	E1	B	46	pop hl	
00F73C	F1	B	47	pop af	; Restore registers.
		B	48		; GOTO TRSDOS interrupt subsystem & execute.

\*\*\*\*\* TRSDOS \*\*\*\*\*  
 < SYSRES HIGH memory region. >

PC	Object	I	Line	Source	..\\SYSRES\\interrupts\\interrupt_handler.s
		B	49		GOTO RST38z ; Then back to what we were doing before interrupted.
		B+	49	scope	
00F73D	C3 FF 1B	B+	49	jp	RST38z
		B+	49	endmac	GOTO
		B	50		; There is no return.
		B	51		
00F740		B	52	pump_switch	RETURN ; This changes to NOP then pump video enabled, if RET then pump
		B+	52	scope	
00F740	C9	B+	52	ret	; Return from call.
		B+	52	endmac	RETURN
00F741	3A 84 F7	B	53	ld	a,(pumpcnt) ; Get pump counter. If video RAM has been accessed, then inc counte
00F744	3D	B	54	dec	a ; Counter=counter-1.
00F745	32 84 F7	B	55	ld	(pumpcnt),a ; Store it back.
00F748	C0	B	56	ret	nz ; Counter did NOT reach 0 thus do NOT pump video to console.
		B	57		
00F749	3EC9	B	58	ld	a,0c9h ; Get RET instruction, turn pump routine OFF.
00F74B	32 40 F7	B	59	ld	(pump_switch),a ; Make this pump then disable pump until ENADIS_DO_RAM routine
		B	60		
00F74E	3E03	B	61	ld	a,video_refresh ; Get value & reset counter.
00F750	32 84 F7	B	62	ld	(pumpcnt),a ; Reset counter to countdown value.
		B	63		
00F753	21F7F7	B	64	ld	hl,zspace.VIDEO\$-9 ; Point to video RAM.
00F756	118207	B	65	ld	de,CRTSIZE+2 ; Size of video screen in dec (1920).
		B	66		
00F759	ED38C6	B	67	canisend1	IN0 a,(UART0_MSR) ; Get CTS status & test if CTS is ready..
00F75C	CB67	B	68	BIT	4,A ; If other end sets CTS true, they are ready to receive.I just
00F75E	28 F9	B	69	JR	Z,canisend1 ; If not ready wait. Come on I am waiting in CANISND loop.
		B	70		
00F760	ED38C5	B	71	IN0	A,(UART0_LSR) ; Find out if UART ready to transmit.
00F763	2F	B	72	cpl	
00F764	E620	B	73	AND	UART_THRE ; Determine if our own UART is ready to send a character.
00F766	20 F1	B	74	jr	nz,canisend1 ; Loop until we can send.
		B	75		
		B	76		
00F768	3A 6D 00	B	77	ld	a,(ZDFLAG\$) ; Bit 7 contains graphic bit.
00F76B	CB7F	B	78	BIT	7,a ; Test graphic bit. If on pass graphic unchanged.
00F76D	7E	B	79	ld	a,(hl) ; Get char to display.
		B	80		
		B	81	IF_NE_BRANCH	pass_grap
		B+	81	scope	
00F76E	20 06	B+	81	jr	nz,pass_grap
		B+	81	endmac	IF_NE_BRANCH
		B	82		
00F770	CB7F	B	83	BIT	7,A ; Test if this char >=\$80h.
		B	84	IF_EQ_BRANCH	non_grap ; <\$80h then branch.
		B+	84	scope	
00F772	28 02	B+	84	jr	z,non_grap
		B+	84	endmac	IF_EQ_BRANCH
		B	85		
00F774	3E20	B	86	ld	a,' ' ; Change char >=\$80h to ascii space.
00F776		B	87	pass_grap	
00F776	ED39C0	B	88	non_grap	OUT0 (UART0_THR),a ; Out the UART our character goes.....wheeee!!!!
		B	89		
00F779	23	B	90	inc	hl ; Bump pointer to next char.
00F77A	1B	B	91	dec	de ; Counter - 1.
		B	92		
00F77B	7A	B	93	ld	a,d ; Test if counter rolled over to 0.
00F77C	B3	B	94	or	e ; If both registers 0 then flag will be Z.
		B	95	IF_EQ_RETURN	; Thank you we are finished pumping!
		B+	95	scope	

\*\*\*\*\* TRSDOS \*\*\*\*\*  
 < SYSRES HIGH memory region. >

PC	Object	I	Line	Source	.. \SYSRES\interrupts\interrupt_handler.s
00F77D	C8	B+	95	ret	z
		B+	95	endmac	IF_EQ_RETURN
00F77E	18 D9	B	96	jr	canisend1 ; Loop if more chars to dump.
		B	97		
00F780	00	B	98	int_tmr1_cnt	db 0 ; Heartbeat counter holding 1/60th counts of uptime.
00F781	00	B	99		db 0
00F782	00	B	100		db 0
00F783	00	B	101		db 0
		B	102		
00F784	14	B	103	pumpcnt	db pump_retard ; Counter to pump video RAM to terminal. Pump every x countdown
00F785	C9	B	104	pump_switch_value	db 0C9h ; Start off with RET or C9h, this prevents pump from pu
		B	105		
		B	106		
00F786		B	107	EMAC.RX.handler	
00F786		B	108	EMAC.TX..handler	
00F786		B	109	EMAC.SYS.handler	
00F786		B	110	PLL.handler	
00F786		B	111	FLASH.handler	
00F786		B	112	TIMER0.handler	
		B	113	;TIMER1.handler	
00F786		B	114	TIMER2.handler	
00F786		B	115	TIMER3.handler	
00F786		B	116	TIMER4.handler	
00F786		B	117	TIMER5.handler	
00F786		B	118	RTC.handler	
00F786		B	119	UART0.handler	
00F786		B	120	UART1.handler	
00F786		B	121	I2C.handler	
00F786		B	122	SPI.handler	
00F786		B	123	PORT.A.BIT0.handler	
00F786		B	124	PORT.A.BIT1.handler	
00F786		B	125	PORT.A.BIT2.handler	
00F786		B	126	PORT.A.BIT3.handler	
00F786		B	127	PORT.A.BIT4.handler	
00F786		B	128	PORT.A.BIT5.handler	
00F786		B	129	PORT.A.BIT6.handler	
00F786		B	130	PORT.A.BIT7.handler	
00F786		B	131	PORT.B.BIT0.handler	
00F786		B	132	PORT.B.BIT1.handler	
00F786		B	133	PORT.B.BIT2.handler	
00F786		B	134	PORT.B.BIT3.handler	
00F786		B	135	PORT.B.BIT4.handler	
00F786		B	136	PORT.B.BIT5.handler	
00F786		B	137	PORT.B.BIT6.handler	
00F786		B	138	PORT.B.BIT7.handler	
00F786		B	139	PORT.C.BIT0.handler	
00F786		B	140	PORT.C.BIT1.handler	
00F786		B	141	PORT.C.BIT2.handler	
00F786		B	142	PORT.C.BIT3.handler	
00F786		B	143	PORT.C.BIT4.handler	
00F786		B	144	PORT.C.BIT5.handler	
00F786		B	145	PORT.C.BIT6.handler	
00F786		B	146	PORT.C.BIT7.handler	
00F786		B	147	PORT.D.BIT0.handler	
00F786		B	148	PORT.D.BIT1.handler	
00F786		B	149	PORT.D.BIT2.handler	
00F786		B	150	PORT.D.BIT3.handler	
00F786		B	151	PORT.D.BIT4.handler	
00F786		B	152	PORT.D.BIT5.handler	
00F786		B	153	PORT.D.BIT6.handler	

\*\*\*\*\* TRSDOS \*\*\*\*\*  
 < SYSRES HIGH memory region. >

PC	Object	I	Line	Source	.. \SYSRES\interrupts\interrupt_handler.s
00F786		B	154	PORT.D.BIT7.handler	
00F786		B	155	f92_EMAC.RX.handler	
00F786		B	156	f92_EMAC.TX..handler	
00F786		B	157	f92_EMAC.SYS.handler	
00F786		B	158	f92_PLL.handler	
00F786		B	159	f92_FLASH.handler	
00F786		B	160	f92_TIMER0.handler	
00F786		B	161	f92_TIMER2.handler	
00F786		B	162	f92_TIMER3.handler	
00F786		B	163	f92_TIMER4.handler	
00F786		B	164	f92_TIMER5.handler	
00F786		B	165	f92_RTC.handler	
00F786		B	166	f92_UART0.handler	
00F786		B	167	f92_UART1.handler	
00F786		B	168	f92_I2C.handler	
00F786		B	169	f92_SPI.handler	
00F786		B	170	f92_PORT.A.BIT0.handler	
00F786		B	171	f92_PORT.A.BIT1.handler	
00F786		B	172	f92_PORT.A.BIT2.handler	
00F786		B	173	f92_PORT.A.BIT3.handler	
00F786		B	174	f92_PORT.A.BIT4.handler	
00F786		B	175	f92_PORT.A.BIT5.handler	
00F786		B	176	f92_PORT.A.BIT6.handler	
00F786		B	177	f92_PORT.A.BIT7.handler	
00F786		B	178	f92_PORT.B.BIT0.handler	
00F786		B	179	f92_PORT.B.BIT1.handler	
00F786		B	180	f92_PORT.B.BIT2.handler	
00F786		B	181	f92_PORT.B.BIT3.handler	
00F786		B	182	f92_PORT.B.BIT4.handler	
00F786		B	183	f92_PORT.B.BIT5.handler	
00F786		B	184	f92_PORT.B.BIT6.handler	
00F786		B	185	f92_PORT.B.BIT7.handler	
00F786		B	186	f92_PORT.C.BIT0.handler	
00F786		B	187	f92_PORT.C.BIT1.handler	
00F786		B	188	f92_PORT.C.BIT2.handler	
00F786		B	189	f92_PORT.C.BIT3.handler	
00F786		B	190	f92_PORT.C.BIT4.handler	
00F786		B	191	f92_PORT.C.BIT5.handler	
00F786		B	192	f92_PORT.C.BIT6.handler	
00F786		B	193	f92_PORT.C.BIT7.handler	
00F786		B	194	f92_PORT.D.BIT0.handler	
00F786		B	195	f92_PORT.D.BIT1.handler	
00F786		B	196	f92_PORT.D.BIT2.handler	
00F786		B	197	f92_PORT.D.BIT3.handler	
00F786		B	198	f92_PORT.D.BIT4.handler	
00F786		B	199	f92_PORT.D.BIT5.handler	
00F786		B	200	f92_PORT.D.BIT6.handler	
00F786		B	201	f92_PORT.D.BIT7.handler	
		B	202		
00F786	3EFF	B	203	STRAY.handler	ld a,0ffh
00F788	32 80 F7	B	204		ld (int_tmr1_cnt),a
00F78B	18 FE	B	205		jr \$
		A	165		
00F78D	5BC300F0 FF	A	166	reboot	JP.Lil 0FFF000h ; Code to boot back to host resides @FFF00.
		A	167		
		B	0	include "memory_video.s"	; Container for 1920 bytes video RAM.
		B	1		
		B	2	org zspace.VIDEO\$-9	; Video control codes resides here.
00F7F7	1B5B66	B	3	db ESC.asc,"[f"	; Cursor to home position.
00F7FA	1B5B3F32 356C	B	4	db ESC.asc,"[?25l"	

\*\*\*\*\* TRSDOS \*\*\*\*\*  
< SYSRES HIGH memory region. >

PC	Object		I	Line	Source ..\SYSRES\memory\memory_video.s	
			B	5		
			B	6	org	zspace.VIDEO\$ ; Video RAM resides here.
00F800	436F7079	72696768	B	7	db	"Copyright 2024 Daniel Paul Martin, All rights reserved."
00F808	74203230	32342044				
00F810	616E6965	6C205061				
00F818	756C204D	61727469				
00F820	6E2C2041	6C6C2072				
00F828	69676874	73207265				
00F830	73657276	65642E				
00F837	436F7079	72696768	B	8	db	"Copyright 2024 Daniel Paul Martin, All rights reserved."
00F83F	74203230	32342044				
00F847	616E6965	6C205061				
00F84F	756C204D	61727469				
00F857	6E2C2041	6C6C2072				
00F85F	69676874	73207265				
00F867	73657276	65642E				
00F86E	436F7079	72696768	B	9	db	"Copyright 2024 Daniel Paul Martin, All rights reserved."
00F876	74203230	32342044				
00F87E	616E6965	6C205061				
00F886	756C204D	61727469				
00F88E	6E2C2041	6C6C2072				
00F896	69676874	73207265				
00F89E	73657276	65642E				
00F8A5	436F7079	72696768	B	10	db	"Copyright 2024 Daniel Paul Martin, All rights reserved."
00F8AD	74203230	32342044				
00F8B5	616E6965	6C205061				
00F8BD	756C204D	61727469				
00F8C5	6E2C2041	6C6C2072				
00F8CD	69676874	73207265				
00F8D5	73657276	65642E				
00F8DC	436F7079	72696768	B	11	db	"Copyright 2024 Daniel Paul Martin, All rights reserved."
00F8E4	74203230	32342044				
00F8EC	616E6965	6C205061				
00F8F4	756C204D	61727469				
00F8FC	6E2C2041	6C6C2072				
00F904	69676874	73207265				
00F90C	73657276	65642E				
00F913	436F7079	72696768	B	12	db	"Copyright 2024 Daniel Paul Martin, All rights reserved."
00F91B	74203230	32342044				
00F923	616E6965	6C205061				
00F92B	756C204D	61727469				
00F933	6E2C2041	6C6C2072				
00F93B	69676874	73207265				
00F943	73657276	65642E				
00F94A	436F7079	72696768	B	13	db	"Copyright 2024 Daniel Paul Martin, All rights reserved."
00F952	74203230	32342044				
00F95A	616E6965	6C205061				
00F962	756C204D	61727469				
00F96A	6E2C2041	6C6C2072				
00F972	69676874	73207265				
00F97A	73657276	65642E				
00F981	436F7079	72696768	B	14	db	"Copyright 2024 Daniel Paul Martin, All rights reserved."
00F989	74203230	32342044				
00F991	616E6965	6C205061				
00F999	756C204D	61727469				
00F9A1	6E2C2041	6C6C2072				
00F9A9	69676874	73207265				
00F9B1	73657276	65642E				
00F9B8	436F7079	72696768	B	15	db	"Copyright 2024 Daniel Paul Martin, All rights reserved."
00F9C0	74203230	32342044				

\*\*\*\*\* TRSDOS \*\*\*\*\*  
< SYSRES HIGH memory region. >

PC	Object	I	Line	Source	..\SYSRES\memory\memory_video.s
00F9C8	616E6965 6C205061				
00F9D0	756C204D 61727469				
00F9D8	6E2C2041 6C6C2072				
00F9E0	69676874 73207265				
00F9E8	73657276 65642E				
00F9EF	436F7079 72696768	B	16	db	"Copyright 2024 Daniel Paul Martin, All rights reserved."
00F9F7	74203230 32342044				
00F9FF	616E6965 6C205061				
00FA07	756C204D 61727469				
00FA0F	6E2C2041 6C6C2072				
00FA17	69676874 73207265				
00FA1F	73657276 65642E				
00FA26	436F7079 72696768	B	17	db	"Copyright 2024 Daniel Paul Martin, All rights reserved."
00FA2E	74203230 32342044				
00FA36	616E6965 6C205061				
00FA3E	756C204D 61727469				
00FA46	6E2C2041 6C6C2072				
00FA4E	69676874 73207265				
00FA56	73657276 65642E				
00FA5D	436F7079 72696768	B	18	db	"Copyright 2024 Daniel Paul Martin, All rights reserved."
00FA65	74203230 32342044				
00FA6D	616E6965 6C205061				
00FA75	756C204D 61727469				
00FA7D	6E2C2041 6C6C2072				
00FA85	69676874 73207265				
00FA8D	73657276 65642E				
00FA94	436F7079 72696768	B	19	db	"Copyright 2024 Daniel Paul Martin, All rights reserved."
00FA9C	74203230 32342044				
00FAA4	616E6965 6C205061				
00FAAC	756C204D 61727469				
00FAB4	6E2C2041 6C6C2072				
00FABC	69676874 73207265				
00FAC4	73657276 65642E				
00FACB	436F7079 72696768	B	20	db	"Copyright 2024 Daniel Paul Martin, All rights reserved."
00FAD3	74203230 32342044				
00FADB	616E6965 6C205061				
00FAE3	756C204D 61727469				
00FAEB	6E2C2041 6C6C2072				
00FAF3	69676874 73207265				
00FAFB	73657276 65642E				
00FB02	436F7079 72696768	B	21	db	"Copyright 2024 Daniel Paul Martin, All rights reserved."
00FB0A	74203230 32342044				
00FB12	616E6965 6C205061				
00FB1A	756C204D 61727469				
00FB22	6E2C2041 6C6C2072				
00FB2A	69676874 73207265				
00FB32	73657276 65642E				
00FB39	436F7079 72696768	B	22	db	"Copyright 2024 Daniel Paul Martin, All rights reserved."
00FB41	74203230 32342044				
00FB49	616E6965 6C205061				
00FB51	756C204D 61727469				
00FB59	6E2C2041 6C6C2072				
00FB61	69676874 73207265				
00FB69	73657276 65642E				
00FB70	436F7079 72696768	B	23	db	"Copyright 2024 Daniel Paul Martin, All rights reserved."
00FB78	74203230 32342044				
00FB80	616E6965 6C205061				
00FB88	756C204D 61727469				
00FB90	6E2C2041 6C6C2072				
00FB98	69676874 73207265				



\*\*\*\*\* TRSDOS \*\*\*\*\*  
< SYSRES HIGH memory region. >

PC	Object	I	Line	Source	..\SYSRES\memory\memory_video.s
00FBA0	73657276 65642E				
00FBA7	436F7079 72696768	B	24	db	"Copyright 2024 Daniel Paul Martin, All rights reserved."
00FBAF	74203230 32342044				
00FBB7	616E6965 6C205061				
00FBBF	756C204D 61727469				
00FBC7	6E2C2041 6C6C2072				
00FBCF	69676874 73207265				
00FBD7	73657276 65642E				
00FBDE	436F7079 72696768	B	25	db	"Copyright 2024 Daniel Paul Martin, All rights reserved."
00FBE6	74203230 32342044				
00FBEE	616E6965 6C205061				
00FBF6	756C204D 61727469				
00FBFE	6E2C2041 6C6C2072				
00FC06	69676874 73207265				
00FC0E	73657276 65642E				
00FC15	436F7079 72696768	B	26	db	"Copyright 2024 Daniel Paul Martin, All rights reserved."
00FC1D	74203230 32342044				
00FC25	616E6965 6C205061				
00FC2D	756C204D 61727469				
00FC35	6E2C2041 6C6C2072				
00FC3D	69676874 73207265				
00FC45	73657276 65642E				
00FC4C	436F7079 72696768	B	27	db	"Copyright 2024 Daniel Paul Martin, All rights reserved."
00FC54	74203230 32342044				
00FC5C	616E6965 6C205061				
00FC64	756C204D 61727469				
00FC6C	6E2C2041 6C6C2072				
00FC74	69676874 73207265				
00FC7C	73657276 65642E				
00FC83	436F7079 72696768	B	28	db	"Copyright 2024 Daniel Paul Martin, All rights reserved."
00FC8B	74203230 32342044				
00FC93	616E6965 6C205061				
00FC9B	756C204D 61727469				
00FCA3	6E2C2041 6C6C2072				
00FCAB	69676874 73207265				
00FCB3	73657276 65642E				
00FCBA	436F7079 72696768	B	29	db	"Copyright 2024 Daniel Paul Martin, All rights reserved."
00FCC2	74203230 32342044				
00FCCA	616E6965 6C205061				
00FCD2	756C204D 61727469				
00FCDA	6E2C2041 6C6C2072				
00FCE2	69676874 73207265				
00FCEA	73657276 65642E				
00FCF1	436F7079 72696768	B	30	db	"Copyright 2024 Daniel Paul Martin, All rights reserved."
00FCF9	74203230 32342044				
00FD01	616E6965 6C205061				
00FD09	756C204D 61727469				
00FD11	6E2C2041 6C6C2072				
00FD19	69676874 73207265				
00FD21	73657276 65642E				
00FD28	436F7079 72696768	B	31	db	"Copyright 2024 Daniel Paul Martin, All rights reserved."
00FD30	74203230 32342044				
00FD38	616E6965 6C205061				
00FD40	756C204D 61727469				
00FD48	6E2C2041 6C6C2072				
00FD50	69676874 73207265				
00FD58	73657276 65642E				
00FD5F	436F7079 72696768	B	32	db	"Copyright 2024 Daniel Paul Martin, All rights reserved."
00FD67	74203230 32342044				
00FD6F	616E6965 6C205061				

\*\*\*\*\* TRSDOS \*\*\*\*\*  
< SYSRES HIGH memory region. >

PC	Object	I	Line	Source	..\SYSRES\memory\memory_video.s
00FD77	756C204D 61727469				
00FD7F	6E2C2041 6C6C2072				
00FD87	69676874 73207265				
00FD8F	73657276 65642E				
00FD96	436F7079 72696768	B	33	db	"Copyright 2024 Daniel Paul Martin, All rights reserved."
00FD9E	74203230 32342044				
00FDA6	616E6965 6C205061				
00FDAE	756C204D 61727469				
00FDB6	6E2C2041 6C6C2072				
00FDBE	69676874 73207265				
00FDC6	73657276 65642E				
00FDCD	436F7079 72696768	B	34	db	"Copyright 2024 Daniel Paul Martin, All rights reserved."
00FDD5	74203230 32342044				
00FDDD	616E6965 6C205061				
00FDE5	756C204D 61727469				
00FDED	6E2C2041 6C6C2072				
00FDF5	69676874 73207265				
00FDFD	73657276 65642E				
00FE04	436F7079 72696768	B	35	db	"Copyright 2024 Daniel Paul Martin, All rights reserved."
00FE0C	74203230 32342044				
00FE14	616E6965 6C205061				
00FE1C	756C204D 61727469				
00FE24	6E2C2041 6C6C2072				
00FE2C	69676874 73207265				
00FE34	73657276 65642E				
00FE3B	436F7079 72696768	B	36	db	"Copyright 2024 Daniel Paul Martin, All rights reserved."
00FE43	74203230 32342044				
00FE4B	616E6965 6C205061				
00FE53	756C204D 61727469				
00FE5B	6E2C2041 6C6C2072				
00FE63	69676874 73207265				
00FE6B	73657276 65642E				
00FE72	436F7079 72696768	B	37	db	"Copyright 2024 Daniel Paul Martin, All rights reserved."
00FE7A	74203230 32342044				
00FE82	616E6965 6C205061				
00FE8A	756C204D 61727469				
00FE92	6E2C2041 6C6C2072				
00FE9A	69676874 73207265				
00FEA2	73657276 65642E				
00FEA9	436F7079 72696768	B	38	db	"Copyright 2024 Daniel Paul Martin, All rights reserved."
00FEB1	74203230 32342044				
00FEB9	616E6965 6C205061				
00FEC1	756C204D 61727469				
00FEC9	6E2C2041 6C6C2072				
00FED1	69676874 73207265				
00FED9	73657276 65642E				
00FEE0	436F7079 72696768	B	39	db	"Copyright 2024 Daniel Paul Martin, All rights reserved."
00FEE8	74203230 32342044				
00FEF0	616E6965 6C205061				
00FEF8	756C204D 61727469				
00FF00	6E2C2041 6C6C2072				
00FF08	69676874 73207265				
00FF10	73657276 65642E				
00FF17	436F7079 72696768	B	40	db	"Copyright 2024 Daniel Paul Martin, All rights reserved."
00FF1F	74203230 32342044				
00FF27	616E6965 6C205061				
00FF2F	756C204D 61727469				
00FF37	6E2C2041 6C6C2072				
00FF3F	69676874 73207265				
00FF47	73657276 65642E				

\*\*\*\*\* TRSDOS \*\*\*\*\*  
< SYSRES HIGH memory region. >

PC	Object	I	Line	Source	..\\SYSRES\\memory\\memory_video.s
00FF4E	436F7079 72696768	B	41	db	"Copyright 2024 Daniel Paul Martin, All rights reserved."
00FF56	74203230 32342044				
00FF5E	616E6965 6C205061				
00FF66	756C204D 61727469				
00FF6E	6E2C2041 6C6C2072				
00FF76	69676874 73207265				
00FF7E	73657276 65642E				
00FF85	436F7079 72696768	B	42	db	"Copyright 2024 Daniel Paul Martin, All rights reserved."
00FF8D	74203230 32342044				
00FF95	616E6965 6C205061				
00FF9D	756C204D 61727469				
00FFA5	6E2C2041 6C6C2072				
00FFAD	69676874 73207265				
00FFB5	73657276 65642E				
00FFBC	436F7079 72696768	B	43	db	"Copyright 2024 Daniel Paul Martin, All rights reserved."
00FFC4	74203230 32342044				
00FFCC	616E6965 6C205061				
00FFD4	756C204D 61727469				
00FFDC	6E2C2041 6C6C2072				
00FFE4	69676874 73207265				
00FFEC	73657276 65642E				
		A	169		
		A	170	; Memory >64K starting @010000h	
		A	171		
		B	0	include "memory_slice0.s"	; Container for slice 0 data.
		B	1		
		B	2	org 010000h	; Slice 0, systems slice start in RAM.
010000	000101	B	3	db 00, 01, 01	; Pointer to slice 1.
010003		B	4	ds 100	
010067	2020202A 2A2A2A2A	B	5	db " *****	Copyright 2024 Daniel Paul Martin, All rights reserved. ***** "
01006F	2A2A2020 20436F70				
010077	79726967 68742032				
01007F	30323420 44616E69				
010087	656C2050 61756C20				
01008F	4D617274 696E2C20				
010097	416C6C20 72696768				
01009F	74732072 65736572				
0100A7	7665642E 2020202A				
0100AF	2A2A2A2A 2A202020				
		B	0	include "memory_slice1.s"	; Container for slice 1 data.
		B	1		
		B	2	org 010100h	; Slice 1 start in RAM.
		B	3		
010100	00010000	B	4	db 00h, 01h, 00h, 00h	; Offset to volume 0 (drive 0).
010104	00000000	B	5	db 00h, 00h, 00h, 00h	; Offset to volume 1 (drive 1).
010108	00000000	B	6	db 00h, 00h, 00h, 00h	; Offset to volume 2 (drive 2).
01010C	00000000	B	7	db 00h, 00h, 00h, 00h	; Offset to volume 3 (drive 3).
010110	00000000	B	8	db 00h, 00h, 00h, 00h	; Offset to volume 4 (drive 4).
010114	00000000	B	9	db 00h, 00h, 00h, 00h	; Offset to volume 5 (drive 5).
010118	00000000	B	10	db 00h, 00h, 00h, 00h	; Offset to volume 6 (drive 6).
01011C	00000000	B	11	db 00h, 00h, 00h, 00h	; Offset to volume 7 (drive 7).
010120	00000000	B	12	db 00h, 00h, 00h, 00h	; Blank volume offsets.
010124	00000000	B	13	db 00h, 00h, 00h, 00h	; Blank volume offsets.
010128	00000000	B	14	db 00h, 00h, 00h, 00h	; Blank volume offsets.
01012C	00000000	B	15	db 00h, 00h, 00h, 00h	; Blank volume offsets.
010130	00000000	B	16	db 00h, 00h, 00h, 00h	; Blank volume offsets.
010134	00000000	B	17	db 00h, 00h, 00h, 00h	; Blank volume offsets.
010138	00000000	B	18	db 00h, 00h, 00h, 00h	; Blank volume offsets.
01013C	00000000	B	19	db 00h, 00h, 00h, 00h	; Blank volume offsets.
		B	20		

\*\*\*\*\* TRSDOS \*\*\*\*\*  
 < SYSRES HIGH memory region. >

PC	Object	I	Line	Source	.. \SYSRES\memory\memory_slice1.s
010140	00000000	B	21	db	00h, 00h, 00h, 00h ; Blank volume offsets.
010144	00000000	B	22	db	00h, 00h, 00h, 00h ; Blank volume offsets.
010148	00000000	B	23	db	00h, 00h, 00h, 00h ; Blank volume offsets.
01014C	00000000	B	24	db	00h, 00h, 00h, 00h ; Blank volume offsets.
010150	00000000	B	25	db	00h, 00h, 00h, 00h ; Blank volume offsets.
010154	00000000	B	26	db	00h, 00h, 00h, 00h ; Blank volume offsets.
010158	00000000	B	27	db	00h, 00h, 00h, 00h ; Blank volume offsets.
01015C	00000000	B	28	db	00h, 00h, 00h, 00h ; Blank volume offsets.
010160	00000000	B	29	db	00h, 00h, 00h, 00h ; Blank volume offsets.
010164	00000000	B	30	db	00h, 00h, 00h, 00h ; Blank volume offsets.
010168	00000000	B	31	db	00h, 00h, 00h, 00h ; Blank volume offsets.
01016C	00000000	B	32	db	00h, 00h, 00h, 00h ; Blank volume offsets.
010170	00000000	B	33	db	00h, 00h, 00h, 00h ; Blank volume offsets.
010174	00000000	B	34	db	00h, 00h, 00h, 00h ; Blank volume offsets.
010178	00000000	B	35	db	00h, 00h, 00h, 00h ; Blank volume offsets.
01017C	00000000	B	36	db	00h, 00h, 00h, 00h ; Blank volume offsets.
		B	37		
010180	00000000	B	38	db	00h, 00h, 00h, 00h ; Blank volume offsets.
010184	00000000	B	39	db	00h, 00h, 00h, 00h ; Blank volume offsets.
010188	00000000	B	40	db	00h, 00h, 00h, 00h ; Blank volume offsets.
01018C	00000000	B	41	db	00h, 00h, 00h, 00h ; Blank volume offsets.
010190	00000000	B	42	db	00h, 00h, 00h, 00h ; Blank volume offsets.
010194	00000000	B	43	db	00h, 00h, 00h, 00h ; Blank volume offsets.
010198	00000000	B	44	db	00h, 00h, 00h, 00h ; Blank volume offsets.
01019C	00000000	B	45	db	00h, 00h, 00h, 00h ; Blank volume offsets.
0101A0	00000000	B	46	db	00h, 00h, 00h, 00h ; Blank volume offsets.
0101A4	00000000	B	47	db	00h, 00h, 00h, 00h ; Blank volume offsets.
0101A8	00000000	B	48	db	00h, 00h, 00h, 00h ; Blank volume offsets.
0101AC	00000000	B	49	db	00h, 00h, 00h, 00h ; Blank volume offsets.
0101B0	00000000	B	50	db	00h, 00h, 00h, 00h ; Blank volume offsets.
0101B4	00000000	B	51	db	00h, 00h, 00h, 00h ; Blank volume offsets.
0101B8	00000000	B	52	db	00h, 00h, 00h, 00h ; Blank volume offsets.
0101BC	00000000	B	53	db	00h, 00h, 00h, 00h ; Blank volume offsets.
		B	54		
0101C0	00000000	B	55	db	00h, 00h, 00h, 00h ; Blank volume offsets.
0101C4	00000000	B	56	db	00h, 00h, 00h, 00h ; Blank volume offsets.
0101C8	00000000	B	57	db	00h, 00h, 00h, 00h ; Blank volume offsets.
0101CC	00000000	B	58	db	00h, 00h, 00h, 00h ; Blank volume offsets.
0101D0	00000000	B	59	db	00h, 00h, 00h, 00h ; Blank volume offsets.
0101D4	00000000	B	60	db	00h, 00h, 00h, 00h ; Blank volume offsets.
0101D8	00000000	B	61	db	00h, 00h, 00h, 00h ; Blank volume offsets.
0101DC	00000000	B	62	db	00h, 00h, 00h, 00h ; Blank volume offsets.
0101E0	00000000	B	63	db	00h, 00h, 00h, 00h ; Blank volume offsets.
0101E4	00000000	B	64	db	00h, 00h, 00h, 00h ; Blank volume offsets.
0101E8	00000000	B	65	db	00h, 00h, 00h, 00h ; Blank volume offsets.
0101EC	00000000	B	66	db	00h, 00h, 00h, 00h ; Blank volume offsets.
0101F0	00000000	B	67	db	00h, 00h, 00h, 00h ; Blank volume offsets.
0101F4	00000000	B	68	db	00h, 00h, 00h, 00h ; Blank volume offsets.
0101F8	00000000	B	69	db	00h, 00h, 00h, 00h ; Blank volume offsets.
0101FC	00000000	B	70	db	00h, 00h, 00h, 00h ; Blank volume offsets.
		A	174		
		B	0	include	'equates-check.s'
		B	1	; Hardware vectors.	
		B	2		
		B	3	if zRST00	!= 0000H
		B	4	warning	'zRST00 Module location error.'
		B	5	endif	
		B	6		
		B	7	if zRST08	!= 0008H
		B	8	warning	'zRST08 Module location error.'

\*\*\*\*\* TRSDOS \*\*\*\*\*  
 < SYSRES HIGH memory region. >

PC	Object	I	Line	Source ..\..\TRSDOS_GLOBAL\equates\equates-check.s
		B	9	endif
		B	10	
		B	11	if zRST10 != 0010H
		B	12	warning 'zRST10 Module location error.'
		B	13	endif
		B	14	
		B	15	if zRST18 != 0018H
		B	16	warning 'zRST18 Module location error.'
		B	17	endif
		B	18	
		B	19	if zRST20 != 0020H
		B	20	warning 'zRST20 Module location error.'
		B	21	endif
		B	22	
		B	23	if zRST28 != 0028H
		B	24	warning 'zRST28 Module location error.'
		B	25	endif
		B	26	
		B	27	if zRST30 != 0030H
		B	28	warning 'zRST30 Module location error.'
		B	29	endif
		B	30	
		B	31	if zRST38 != 0038H
		B	32	warning 'zRST38 Module location error.'
		B	33	endif
		B	34	
		B	35	if zNMI != 0066H
		B	36	warning 'zNMI Module location error.'
		B	37	endif
		B	38	
		B	39	; Memory Routines.
		B	40	
		B	41	if BAR\$ != 0201H
		B	42	warning 'BAR\$ Module location error.'
		B	43	endif
		B	44	
		B	45	if BUR\$ != 0200H
		B	46	warning 'BUR\$ Module location error.'
		B	47	endif
		B	48	
		B	49	if zHIGH\$ != 040EH
		B	50	warning 'zHIGH\$ Module location error.'
		B	51	endif
		B	52	
		B	53	if zLOW\$ != 001EH
		B	54	warning 'zLOW\$ Module location error.'
		B	55	endif
		B	56	
		B	57	; Video modules.
		B	58	
		B	59	if ENADIS_DO_RAM != 0817H
		B	60	warning 'ENADIS_DO_RAM Module location error.'
		B	61	endif
		B	62	
		B	63	if DIS_DO_RAM != 0846H
		B	64	warning 'DIS_DO_RAM Module location error.'
		B	65	endif
		B	66	
		B	67	if DODATA\$ != 0B94H
		B	68	warning 'DODATA\$ Module location error.'

\*\*\*\*\* TRSDOS \*\*\*\*\*  
 < SYSRES HIGH memory region. >

PC	Object	I	Line	Source ..\..\TRSDOS_GLOBAL\equates\equates-check.s
		B	69	endif
		B	70	
		B	71	if DO_CONTROL != 0C44H
		B	72	warning 'DO_CONTROL Module location error.'
		B	73	endif
		B	74	
		B	75	if DO_DSPCHAR != 0CB8H
		B	76	warning 'DO_DSPCHAR Module location error.'
		B	77	endif
		B	78	
		B	79	if DO_INVERT_DIS != 0C8CH
		B	80	warning 'DO_INVERT_DIS Module location error.'
		B	81	endif
		B	82	
		B	83	if DO_INVERT_ENA != 0C89H
		B	84	warning 'DO_INVERT_ENA Module location error.'
		B	85	endif
		B	86	
		B	87	if DO_INVERT_OFF != 0C9BH
		B	88	warning 'DO_INVERT_OFF Module location error.'
		B	89	endif
		B	90	
		B	91	if DO_MASK != 0000H
		B	92	warning 'DO_MASK Module location error.'
		B	93	endif
		B	94	
		B	95	if DO_RET != 0BCBH
		B	96	warning 'DO_RET Module location error.'
		B	97	endif
		B	98	
		B	99	if DO_RET1 != 0BCCH
		B	100	warning 'DO_RET1 Module location error.'
		B	101	endif
		B	102	
		B	103	if DO_SCROLL != 0CCEH
		B	104	warning 'DO_SCROLL Module location error.'
		B	105	endif
		B	106	
		B	107	if DO_TABS != 0BEAH
		B	108	warning 'DO_TABS Module location error.'
		B	109	endif
		B	110	
		B	111	
		B	112	
		B	113	
		B	114	; Test system flag table.
		B	115	
		B	116	
		B	117	if FLGTAB\$ != 006AH
		B	118	warning 'FLGTAB\$ Module location error.'
		B	119	endif
		B	120	
		B	121	
		B	122	if AFLAG\$ != 006AH
		B	123	warning 'zAFLAG\$ Module location error.'
		B	124	endif
		B	125	
		B	126	if BFLAG\$ != 006BH
		B	127	warning 'zBFLAG\$ Module location error.'
		B	128	endif

\*\*\*\*\* TRSDOS \*\*\*\*\*  
 < SYSRES HIGH memory region. >

PC	Object	I	Line	Source ..\..\TRSDOS_GLOBAL\equates\equates-check.s
		B	129	
		B	130	if CFLAG\$ != 006CH
		B	131	warning 'zCFLAG\$ Module location error.'
		B	132	endif
		B	133	
		B	134	if DFLAG\$ != 006DH
		B	135	warning 'zDFLAG\$ Module location error.'
		B	136	endif
		B	137	
		B	138	if EFLAG\$ != 006EH
		B	139	warning 'zEFLAG\$ Module location error.'
		B	140	endif
		B	141	
		B	142	if GFLAG\$ != 0070H
		B	143	warning 'zGFLAG\$ Module location error.'
		B	144	endif
		B	145	
		B	146	if HFLAG\$ != 0071H
		B	147	warning 'zHFLAG\$ Module location error.'
		B	148	endif
		B	149	
		B	150	if IFLAG\$ != 0072H
		B	151	warning 'zIFLAG\$ Module location error.'
		B	152	endif
		B	153	
		B	154	if JFLAG\$ != 0073H
		B	155	warning 'zJFLAG\$ Module location error.'
		B	156	endif
		B	157	
		B	158	if KFLAG\$ != 0074H
		B	159	warning 'zKFLAG\$ Module location error.'
		B	160	endif
		B	161	
		B	162	if LFLAG\$ != 0075H
		B	163	warning 'zLFLAG\$ Module location error.'
		B	164	endif
		B	165	
		B	166	if NFLAG\$ != 0077H
		B	167	warning 'zNFLAG\$ Module location error.'
		B	168	endif
		B	169	
		B	170	if PFLAG\$ != 0079H
		B	171	warning 'zPFLAG\$ Module location error.'
		B	172	endif
		B	173	
		B	174	if QFLAG\$ != 007AH
		B	175	warning 'zQFLAG\$ Module location error.'
		B	176	endif
		B	177	
		B	178	if RFLAG\$ != 007BH
		B	179	warning 'zRFLAG\$ Module location error.'
		B	180	endif
		B	181	
		B	182	if SFLAG\$ != 007CH
		B	183	warning 'zSFLAG\$ Module location error.'
		B	184	endif
		B	185	
		B	186	if TFLAG\$ != 007DH
		B	187	warning 'zTFLAG\$ Module location error.'
		B	188	endif

\*\*\*\*\* TRSDOS \*\*\*\*\*  
 < SYSRES HIGH memory region. >

PC	Object	I	Line	Source ..\..\TRSDOS_GLOBAL\equates\equates-check.s
		B	189	
		B	190	if UFLAG\$ != 007EH
		B	191	warning 'zUFLAG\$ Module location error.'
		B	192	endif
		B	193	
		B	194	if VFLAG\$ != 007FH
		B	195	warning 'zVFLAG\$ Module location error.'
		B	196	endif
		B	197	
		B	198	if XFLAG\$ != 0081H
		B	199	warning 'zXFLAG\$ Module location error.'
		B	200	endif
		B	201	
		B	202	if YFLAG\$ != 0082H
		B	203	warning 'zYFLAG\$ Module location error.'
		B	204	endif
		B	205	
		B	206	; Function calls.
		B	207	
		B	208	if z\$SYS != z\$SYS
		B	209	warning 'z\$SYS Module location error.'
		B	210	endif
		B	211	
		B	212	if zABORT != 1B08H
		B	213	warning 'zABORT Module location error.'
		B	214	endif
		B	215	
		B	216	if zADTSK != 1CDAH
		B	217	warning 'zADTSK Module location error.'
		B	218	endif
		B	219	
		B	220	if zBANK != 0877H
		B	221	warning 'zBANK Module location error.'
		B	222	endif
		B	223	
		B	224	if zBKSP != 14ADH
		B	225	warning 'zBKSP Module location error.'
		B	226	endif
		B	227	
		B	228	if zBREAK != 196FH
		B	229	warning 'zBREAK Module location error.'
		B	230	endif
		B	231	
		B	232	if zspace.BYTEIO\$ != 1300H
		B	233	warning 'BYTEIO\$ Module location error.'
		B	234	endif
		B	235	
		B	236	if zCHNIO != 0689H
		B	237	warning 'zCHNIO Module location error.'
		B	238	endif
		B	239	
		B	240	if zCKBRKC != 0553H
		B	241	warning 'zCKBRKC Module location error.'
		B	242	endif
		B	243	
		B	244	if zCKDRV != 1993H
		B	245	warning 'zCKDRV Module location error.'
		B	246	endif
		B	247	
		B	248	if zCKEOF != 158FH



\*\*\*\*\* TRSDOS \*\*\*\*\*  
 < SYSRES HIGH memory region. >

PC	Object	I	Line	Source	...	...	...	...	...
		B	249	warning	'zCKEOF	Module location error.'			
		B	250	endif					
		B	251						
		B	252	if zCKTSK	!= 1CF5H				
		B	253	warning	'zCKTSK	Module location error.'			
		B	254	endif					
		B	255						
		B	256	if zCLOSE	!= 1999H				
		B	257	warning	'zCLOSE	Module location error.'			
		B	258	endif					
		B	259						
		B	260	if zCLS	!= 0545H				
		B	261	warning	'zCLS	Module location error.'			
		B	262	endif					
		B	263						
		B	264	if zCMNDI	!= 197EH				
		B	265	warning	'zCMNDI	Module location error.'			
		B	266	endif					
		B	267						
		B	268	if zCMNDR	!= 197BH				
		B	269	warning	'zCMNDR	Module location error.'			
		B	270	endif					
		B	271						
		B	272	if zCTL	!= 0623H				
		B	273	warning	'zCTL	Module location error.'			
		B	274	endif					
		B	275						
		B	276	if DATEz	!= 07A8H				
		B	277	warning	'zDATE	Module location error.'			
		B	278	endif					
		B	279						
		B	280	if zDBGHK	!= 199FH				
		B	281	warning	'zDBGHK	Module location error.'			
		B	282	endif					
		B	283						
		B	284	if zDCINIT	!= 19C0H				
		B	285	warning	'zDCINIT	Module location error.'			
		B	286	endif					
		B	287						
		B	288	if zDCRES	!= 19C4H				
		B	289	warning	'zDCRES	Module location error.'			
		B	290	endif					
		B	291						
		B	292	if zDCSTAT	!= 19B5H				
		B	293	warning	'zDCSTAT	Module location error.'			
		B	294	endif					
		B	295						
		B	296	if zDCTBYT	!= 1A2BH				
		B	297	warning	'zDCTBYT	Module location error.'			
		B	298	endif					
		B	299						
		B	300	if zDEBUG	!= 19A0H				
		B	301	warning	'zDEBUG	Module location error.'			
		B	302	endif					
		B	303						
		B	304	if zDECHEX	!= 03E1H				
		B	305	warning	'zDECHEX	Module location error.'			
		B	306	endif					
		B	307						
		B	308	if zDIRCYL	!= 18F7H				

\*\*\*\*\* TRSDOS \*\*\*\*\*  
 < SYSRES HIGH memory region. >

PC	Object	I	Line	Source	...	...	...	...	...
		B	309	warning		'zDIRCYL	Module location error.'		
		B	310	endif					
		B	311						
		B	312	if zDIRRD	!=	18BBH			
		B	313	warning		'zDIRRD	Module location error.'		
		B	314	endif					
		B	315						
		B	316	if zDIRWR	!=	1803H			
		B	317	warning		'zDIRWR	Module location error.'		
		B	318	endif					
		B	319						
		B	320	if zDIV16		!= 06E3H			
		B	321	warning		'zDIV16	Module location error.'		
		B	322	endif					
		B	323						
		B	324	if zDIV8		!= 1927H			
		B	325	warning		'zDIV8	Module location error.'		
		B	326	endif					
		B	327						
		B	328	if zDODIR		!= 19AFH			
		B	329	warning		'zDODIR	Module location error.'		
		B	330	endif					
		B	331						
		B	332	if zDOKEY	!=	19A9H			
		B	333	warning		'zDOKEY	Module location error.'		
		B	334	endif					
		B	335						
		B	336	if zDSP		!= 0642H			
		B	337	warning		'zDSP	Module location error.'		
		B	338	endif					
		B	339						
		B	340	if zDSPLY	!=	052DH			
		B	341	warning		'zDSPLY	Module location error.'		
		B	342	endif					
		B	343						
		B	344	if zERROR	!=	1B0FH			
		B	345	warning		'zERROR	Module location error.'		
		B	346	endif					
		B	347						
		B	348	if zEXIT		!= 1B0BH			
		B	349	warning		'zEXIT	Module location error.'		
		B	350	endif					
		B	351						
		B	352	if zFEXT		!= 1984H			
		B	353	warning		'Module location error.'			
		B	354	endif					
		B	355						
		B	356	if zFLAGS		!= 196AH			
		B	357	warning		'Module location error.'			
		B	358	endif					
		B	359						
		B	360	if zFNAME	!=	199CH			
		B	361	warning		'Module location error.'			
		B	362	endif					
		B	363						
		B	364	if zFSPEC		!= 1981H			
		B	365	warning		'Module location error.'			
		B	366	endif					
		B	367						
		B	368	if zGATRD	!=	1874H			

\*\*\*\*\* TRSDOS \*\*\*\*\*  
 < SYSRES HIGH memory region. >

PC	Object	I	Line	Source	...	...	...	...	...
		B	369	warning					'Module location error.'
		B	370	endif					
		B	371						
		B	372	if zGATWR	!=	1875H			
		B	373	warning					'Module location error.'
		B	374	endif					
		B	375						
		B	376	if zGET		!=	0638H		
		B	377	warning					'Module location error.'
		B	378	endif					
		B	379						
		B	380	if zGTDCB	!=	1990H			
		B	381	warning					'Module location error.'
		B	382	endif					
		B	383						
		B	384	if zGTDCB		!=	1A1EH		
		B	385	warning					'Module location error.'
		B	386	endif					
		B	387						
		B	388	if zGTMOD	!=	19B2H			
		B	389	warning					'Module location error.'
		B	390	endif					
		B	391						
		B	392	if zHDFMT	!=	19E4H			
		B	393	warning					'Module location error.'
		B	394	endif					
		B	395						
		B	396	if zHEX16		!=	07BDH		
		B	397	warning					'zHEX16 Module location error.'
		B	398	endif					
		B	399						
		B	400	if zHEX8		!=	07C2H		
		B	401	warning					'zHEX8 Module location error.'
		B	402	endif					
		B	403						
		B	404	if zHEXDEC	!=	06F6H			
		B	405	warning					'Module location error.'
		B	406	endif					
		B	407						
		B	408	if zHIGH		!=	1948H		
		B	409	warning					'zHIGH\$ Module location error.'
		B	410	endif					
		B	411						
		B	412	if zHITRD		!=	1897H		
		B	413	warning					'Module location error.'
		B	414	endif					
		B	415						
		B	416	if zHITWR		!=	1898H		
		B	417	warning					'Module location error.'
		B	418	endif					
		B	419						
		B	420	if zICNFG		!=	0086H		
		B	421	warning					'Module location error.'
		B	422	endif					
		B	423						
		B	424	if zINIT		!=	198DH		
		B	425	warning					'Module location error.'
		B	426	endif					
		B	427						
		B	428	if zINTL		!=	0000H		

\*\*\*\*\* TRSDOS \*\*\*\*\*  
 < SYSRES HIGH memory region. >

PC	Object	I	Line	Source	...	TRSDOS_GLOBAL\equates\equates-check.s
		B	429	warning		'Module location error.'
		B	430	endif		
		B	431			
		B	432	if zJCL	!=	0630H
		B	433	warning		'Module location error.'
		B	434	endif		
		B	435			
		B	436	if zKBD	!=	0635H
		B	437	warning		'Module location error.'
		B	438	endif		
		B	439			
		B	440	if zKEY	!=	0628H
		B	441	warning		'Module location error.'
		B	442	endif		
		B	443			
		B	444	if zKEYIN	!=	0585H
		B	445	warning		'Module location error.'
		B	446	endif		
		B	447			
		B	448	if zKITSK	!=	0089H
		B	449	warning		'Module location error.'
		B	450	endif		
		B	451			
		B	452	if zKLTSK	!=	1CD0H
		B	453	warning		'zKLTSK Module location error.'
		B	454	endif		
		B	455			
		B	456	if zLOAD	!=	1B38H
		B	457	warning		'zLOAD Module location error.'
		B	458	endif		
		B	459			
		B	460	if zLOC	!=	14DAH
		B	461	warning		'Module location error.'
		B	462	endif		
		B	463			
		B	464	if zLOF	!=	1505H
		B	465	warning		'Module location error.'
		B	466	endif		
		B	467			
		B	468	if zLOGGER	!=	0503H
		B	469	warning		'Module location error.'
		B	470	endif		
		B	471			
		B	472	if zLOGOT	!=	0500H
		B	473	warning		'Module location error.'
		B	474	endif		
		B	475			
		B	476	if zMSG	!=	0530H
		B	477	warning		'Module location error.'
		B	478	endif		
		B	479			
		B	480	if zMUL16	!=	06C9H
		B	481	warning		'Module location error.'
		B	482	endif		
		B	483			
		B	484	if zMUL8	!=	190AH
		B	485	warning		'Module location error.'
		B	486	endif		
		B	487			
		B	488	if zNMI	!=	0066H

\*\*\*\*\* TRSDOS \*\*\*\*\*  
 < SYSRES HIGH memory region. >

PC	Object	I	Line	Source	...	TRSDOS_GLOBAL\equates\equates-check.s
		B	489	warning		'Module location error.'
		B	490	endif		
		B	491			
		B	492	if zOPEN	!=	198AH
		B	493	warning		'Module location error.'
		B	494	endif		
		B	495			
		B	496	if zPARAM	!=	1987H
		B	497	warning		'zPARAM Module location error.'
		B	498	endif		
		B	499			
		B	500	if zPAUSE	!=	0382H
		B	501	warning		'zPAUSE Module location error.'
		B	502	endif		
		B	503			
		B	504	if zPEOF	!=	14C9H
		B	505	warning		'Module location error.'
		B	506	endif		
		B	507			
		B	508	if zPOSN	!=	145BH
		B	509	warning		'zPOSN Module location error.'
		B	510	endif		
		B	511			
		B	512	if zPRINT	!=	0528H
		B	513	warning		'Module location error.'
		B	514	endif		
		B	515			
		B	516	if zPRT	!=	063DH
		B	517	warning		'Module location error.'
		B	518	endif		
		B	519			
		B	520	if zVDPRT	!=	0935H
		B	521	warning		'zPRTSCR Module location error.'
		B	522	endif		
		B	523			
		B	524	if zPUT	!=	0645H
		B	525	warning		'Module location error.'
		B	526	endif		
		B	527			
		B	528	if zRAMDIR	!=	19ACH
		B	529	warning		'Module location error.'
		B	530	endif		
		B	531			
		B	532	if zRDHDR	!=	19D8H
		B	533	warning		'Module location error.'
		B	534	endif		
		B	535			
		B	536	if zRDSEC	!=	19F4H
		B	537	warning		'Module location error.'
		B	538	endif		
		B	539			
		B	540	if zRDSSC	!=	18D8H
		B	541	warning		'Module location error.'
		B	542	endif		
		B	543			
		B	544	if zRDTRK	!=	19E0H
		B	545	warning		'Module location error.'
		B	546	endif		
		B	547			
		B	548	if zREAD	!=	1513H

\*\*\*\*\* TRSDOS \*\*\*\*\*  
 < SYSRES HIGH memory region. >

PC	Object	I	Line	Source	...	...	...	...	...
		B	549	warning					'Module location error.'
		B	550	endif					
		B	551						
		B	552	if zREMOVE !=	19A6H				
		B	553	warning					'Module location error.'
		B	554	endif					
		B	555						
		B	556	if zRENAME !=	1996H				
		B	557	warning					'Module location error.'
		B	558	endif					
		B	559						
		B	560	if zREW	!=	14C2H			
		B	561	warning					'Module location error.'
		B	562	endif					
		B	563						
		B	564	if zRMTSK !=	1CD7H				
		B	565	warning					'zRMTSK Module location error.'
		B	566	endif					
		B	567						
		B	568	if zRPTSK !=	1CEBH				
		B	569	warning					'zRPTSK Module location error.'
		B	570	endif					
		B	571						
		B	572	if zRREAD !=	149AH				
		B	573	warning					'zRREAD Module location error.'
		B	574	endif					
		B	575						
		B	576	if zRSLCT !=	19D4H				
		B	577	warning					'Module location error.'
		B	578	endif					
		B	579						
		B	580	if zRST00 !=	0000H				
		B	581	warning					'Module location error.'
		B	582	endif					
		B	583						
		B	584	if zRST08 !=	0008H				
		B	585	warning					'Module location error.'
		B	586	endif					
		B	587						
		B	588	if zRST10 !=	0010H				
		B	589	warning					'Module location error.'
		B	590	endif					
		B	591						
		B	592	if zRST18 !=	0018H				
		B	593	warning					'Module location error.'
		B	594	endif					
		B	595						
		B	596	if zRST20 !=	0020H				
		B	597	warning					'Module location error.'
		B	598	endif					
		B	599						
		B	600	if zRST28 !=	0028H				
		B	601	warning					'Module location error.'
		B	602	endif					
		B	603						
		B	604	if zRST30 !=	0030H				
		B	605	warning					'Module location error.'
		B	606	endif					
		B	607						
		B	608	if zRST38 !=	0038H				

\*\*\*\*\* TRSDOS \*\*\*\*\*  
 < SYSRES HIGH memory region. >

PC	Object	I	Line	Source	..\\..\\TRSDOS_GLOBAL\\equates\\equates-check.s
		B	609	warning	'Module location error.'
		B	610	endif	
		B	611		
		B	612	if zRSTOR !=	19C8H
		B	613	warning	'Module location error.'
		B	614	endif	
		B	615		
		B	616	if zRSTREG !=	0680H
		B	617	warning	'Module location error.'
		B	618	endif	
		B	619		
		B	620	if zRUN !=	1B1DH
		B	621	warning	'zRUN Module location error.'
		B	622	endif	
		B	623		
		B	624	if zRWRTIT !=	13ADH
		B	625	warning	'Module location error.'
		B	626	endif	
		B	627		
		B	628	if zSEEK !=	19D0H
		B	629	warning	'Module location error.'
		B	630	endif	
		B	631		
		B	632	if zSKIP !=	1457H
		B	633	warning	'zSKIP Module location error.'
		B	634	endif	
		B	635		
		B	636	if zSLCT !=	19BCH
		B	637	warning	'Module location error.'
		B	638	endif	
		B	639		
		B	640	if zSOUND !=	0392H
		B	641	warning	'zSOUND Module location error.'
		B	642	endif	
		B	643		
		B	644	if zSTEPI !=	19CCH
		B	645	warning	'Module location error.'
		B	646	endif	
		B	647		
		B	648	if zTIME !=	078DH
		B	649	warning	'zTIME Module location error.'
		B	650	endif	
		B	651		
		B	652	if zUSA !=	TRUE
		B	653	warning	'zUSA Module location error.'
		B	654	endif	
		B	655		
		B	656	if zVDCTL !=	0B99H
		B	657	warning	'Module location error.'
		B	658	endif	
		B	659		
		B	660	if zVDCTL3 !=	0D38H
		B	661	warning	'zVDCTL3 Module location error.'
		B	662	endif	
		B	663		
		B	664	if zVER !=	1560H
		B	665	warning	'Module location error.'
		B	666	endif	
		B	667		
		B	668	if zVRSEC !=	19DCH

\*\*\*\*\* TRSDOS \*\*\*\*\*  
 < SYSRES HIGH memory region. >

PC	Object	I	Line	Source	...
		B	669	warning	'Module location error.'
		B	670	endif	
		B	671		
		B	672	if zWEOF !=	1430H
		B	673	warning	'zWEOF Module location error.'
		B	674	endif	
		B	675		
		B	676	if zWHERE !=	1979H
		B	677	warning	'Module location error.'
		B	678	endif	
		B	679		
		B	680	if zWRITE !=	1531H
		B	681	warning	'Module location error.'
		B	682	endif	
		B	683		
		B	684	if zWRSEC !=	19E8H
		B	685	warning	'Module location error.'
		B	686	endif	
		B	687		
		B	688	if zWRSSC !=	19ECH
		B	689	warning	'Module location error.'
		B	690	endif	
		B	691		
		B	692	if zWRTRK !=	19F0H
		B	693	warning	'Module location error.'
		B	694	endif	
		B	695		
		B	696	if z_VDCTL !=	0D42H
		B	697	warning	'z_VDCTL Module location error.'
		B	698	endif	
		B	699		
		B	700	if ADDR_2_ROWCOL !=	0DF1H
		B	701	warning	'ADDR_2_ROWCOL Module location error.'
		B	702	endif	
		B	703		
		B	704	; File control blocks.	
		B	705		
		B	706	if CFGFCB\$ !=	00E0H
		B	707	warning	'CFGFCB\$ Module location error.'
		B	708	endif	
		B	709		
		B	710	if CFCB\$ !=	00E0H
		B	711	warning	'CFCB\$ Module location error.'
		B	712	endif	
		B	713		
		B	714	if JFCB\$ !=	00C0H
		B	715	warning	'JFCB\$ Module location error.'
		B	716	endif	
		B	717		
		B	718	if SFCB\$ !=	008CH
		B	719	warning	'SFCB\$ Module location error.'
		B	720	endif	
		B	721		
		B	722		
		B	723	; Device control blocks.	
		B	724		
		B	725		
		B	726	if DODCB\$ !=	0210H
		B	727	warning	'DODCB\$ Module location error.'
		B	728	endif	



\*\*\*\*\* TRSDOS \*\*\*\*\*  
 < SYSRES HIGH memory region. >

PC	Object	I	Line	Source	...
		B	729		Source ..\..\TRSDOS_GLOBAL\equates\equates-check.s
		B	730	if JCLCB\$	!= 0203H
		B	731	warning	'JCLCB\$ Module location error.'
		B	732	endif	
		B	733		
		B	734	if JLDCB\$	!= 0230H
		B	735	warning	'JLDCB\$ Module location error.'
		B	736	endif	
		B	737		
		B	738	if JDCB\$ !=	0024H
		B	739	warning	'JDCB\$ Module location error.'
		B	740	endif	
		B	741		
		B	742	if KIDCB\$	!= 0208H
		B	743	warning	'KIDCB\$ Module location error.'
		B	744	endif	
		B	745		
		B	746	if U0DCB\$	!= 0238H ; Uart 0 DCB no resides here.
		B	747	warning	'S1DCB\$ Module location error.'
		B	748	endif	
		B	749		
		B	750	if SIDCB\$	!= 0220H
		B	751	warning	'SIDCB\$ Module location error.'
		B	752	endif	
		B	753		
		B	754	if SODCB\$	!= 0228H
		B	755	warning	'SODCB\$ Module location error.'
		B	756	endif	
		B	757		
		B	758	if BRKVEC\$	!= 1C88H
		B	759	warning	'BRKVEC\$ Module location error.'
		B	760	endif	
		B	761		
		B	762		
		B	763		
		B	764	if CASHK\$	!= 0A7BH
		B	765	warning	'CASHK\$ Module location error.'
		B	766	endif	
		B	767		
		B	768		
		B	769	if CASHK\$	!= 0A7BH
		B	770	warning	'CASHK\$ Module location error.'
		B	771	endif	
		B	772		
		B	773	if CKOPENz	!= 1568H
		B	774	warning	'CKOPEN@ Module location error.'
		B	775	endif	
		B	776		
		B	777	if CRTBGN\$	!= 0F800H
		B	778	warning	'ZUFLAG\$ Module location error.'
		B	779	endif	
		B	780		
		B	781	if CYL_GRN	!= 16AEH
		B	782	warning	'CYL_GRN Module location error.'
		B	783	endif	
		B	784		
		B	785	if DATE\$ !=	0033H
		B	786	warning	'DATE\$ Module location error.'
		B	787	endif	
		B	788		

\*\*\*\*\* TRSDOS \*\*\*\*\*  
 < SYSRES HIGH memory region. >

PC	Object	I	Line	Source	...
		B	789	if DAYTBL\$	!= 04C7H
		B	790	warning	'DAYTBL\$ Module location error.'
		B	791	endif	
		B	792		
		B	793	if DBGSV\$	!= 00A0H
		B	794	warning	'DBGSV\$ Module location error.'
		B	795	endif	
		B	796		
		B	797	if DCBKL\$	!= 0031H
		B	798	warning	'DCBKL\$ Module location error.'
		B	799	endif	
		B	800		
		B	801	if DCT\$	!= 0470H
		B	802	warning	'DCT\$ Module location error.'
		B	803	endif	
		B	804		
		B	805	if DIRBUF\$	!= 2300H
		B	806	warning	'DIRBUF\$ Module location error.'
		B	807	endif	
		B	808		
		B	809	if DSKTYP\$	!= 04C0H
		B	810	warning	'DSKTYP\$ Module location error.'
		B	811	endif	
		B	812		
		B	813	if DTPMT\$	!= 04C2H
		B	814	warning	'DTPMT\$ Module location error.'
		B	815	endif	
		B	816		
		B	817	; DVRREMD\$ has changed bc we have added more drivers.	
		B	818		
		B	819	; if DVREND\$	!= 1047H
		B	820	; warning	'DVREND\$ Module location error.'
		B	821	; endif	
		B	822		
		B	823	if DVRHI\$	!= 0206H
		B	824	warning	'DVRHI\$ Module location error.'
		B	825	endif	
		B	826		
		B	827	if EXTDBG\$	!= 19A4H
		B	828	warning	'EXTDBG\$ Module location error.'
		B	829	endif	
		B	830		
		B	831	if FDDINT\$	!= 000EH
		B	832	warning	'FDDINT\$ Module location error.'
		B	833	endif	
		B	834		
		B	835	if FEMSK\$	!= 006FH
		B	836	warning	'FEMSK\$ Module location error.'
		B	837	endif	
		B	838		
		B	839	if HKRES\$	!= 1A6CH
		B	840	warning	'HKRES\$ Module location error.'
		B	841	endif	
		B	842		
		B	843	if INBUF\$	!= 0420H
		B	844	warning	'INBUF\$ Module location error.'
		B	845	endif	
		B	846		
		B	847	if INTIM\$	!= 003CH
		B	848	warning	'INTIM\$ Module location error.'

\*\*\*\*\* TRSDOS \*\*\*\*\*  
 < SYSRES HIGH memory region. >

PC	Object	I	Line	Source ..\..\TRSDOS_GLOBAL\equates\equates-check.s
		B	849	endif
		B	850	
		B	851	if INTMSK\$ != 003DH
		B	852	warning 'INTMSK\$ Module location error.'
		B	853	endif
		B	854	
		B	855	if INTVC\$ != 003EH
		B	856	warning 'INTVC\$ Module location error.'
		B	857	endif
		B	858	
		B	859	
		B	860	if JRET\$ != 0026H
		B	861	warning 'JRET\$ Module location error.'
		B	862	endif
		B	863	
		B	864	if KIDATA\$ != 08FCH
		B	865	warning 'KIDATA\$ Module location error.'
		B	866	endif
		B	867	
		B	868	if LBANK\$ != 0202H
		B	869	warning 'LBANK\$ Module location error.'
		B	870	endif
		B	871	
		B	872	if LDRV\$ != 0023H
		B	873	LNKFCB@ != 1566H
		B	874	warning 'LBANK\$ Module location error.'
		B	875	endif
		B	876	
		B	877	if LSVCS\$ != 000DH
		B	878	warning 'LSVC\$ Module location error.'
		B	879	endif
		B	880	
		B	881	if COREMAX\$ != 2400H
		B	882	warning 'MAXCOR\$ Module location error.'
		B	883	endif
		B	884	
		B	885	if MAXDAY\$ != 0401H
		B	886	warning 'MAXDAY\$ Module location error.'
		B	887	endif
		B	888	
		B	889	if COREMIN\$ != 3000H
		B	890	warning 'MINCOR\$\$ Module location error.'
		B	891	endif
		B	892	
		B	893	if MONTBL\$ != 04DCH
		B	894	warning 'MONTBL\$ Module location error.'
		B	895	endif
		B	896	
		B	897	; if NHIT4@ != 181FH
		B	898	; warning 'NHIT4@ Module location error.'
		B	899	; endif
		B	900	
		B	901	if OPREG_SV_AREA != 086EH
		B	902	warning 'OPREG_SV_AREA Module location error.'
		B	903	endif
		B	904	
		B	905	if OPREG_SV_PTR != 0835H
		B	906	; warning 'OPREG_SV_PTR\$ Module location error.'
		B	907	endif
		B	908	

\*\*\*\*\* TRSDOS \*\*\*\*\*  
 < SYSRES HIGH memory region. >

PC	Object	I	Line	Source	...
		B	909	if ORARETz	!= 1503H
		B	910	warning	'ORARET@ Module location error.'
		B	911	endif	
		B	912		
		B	913	if OSRLS\$	!= 003BH
		B	914	warning	'OSRLS\$ Module location error.'
		B	915	endif	
		B	916		
		B	917	if OSVER\$	!= 0085H
		B	918	warning	'OSVER\$ Module location error.'
		B	919	endif	
		B	920		
		B	921	if OVRLY\$	!= 0069H
		B	922	warning	'OVRLY\$ Module location error.'
		B	923	endif	
		B	924		
		B	925	if PAKNAM\$	!= 0410H
		B	926	warning	'PAKNAM\$ Module location error.'
		B	927	endif	
		B	928		
		B	929	if zPAUSE	!= 0382H
		B	930	warning	'PAUSE@ Module location error.'
		B	931	endif	
		B	932		
		B	933	if PCSAVE\$	!= 07AFH
		B	934	warning	'PCSAVE\$ Module location error.'
		B	935	endif	
		B	936		
		B	937	if PDRV\$ !=	001BH
		B	938	warning	'PDRV\$ Module location error.'
		B	939	endif	
		B	940		
		B	941	if PHIGH\$	!= 001CH
		B	942	warning	'PHIGH\$ Module location error.'
		B	943	endif	
		B	944		
		B	945	if PRDCB\$	!= 0218H
		B	946	warning	'PRDCB\$ Module location error.'
		B	947	endif	
		B	948		
		B	949	if PUTazDE	!= 0DCDH
		B	950	warning	'PUTA@DE Module location error.'
		B	951	endif	
		B	952		
		B	953	if PUT_z !=	0DCAH
		B	954	warning	'PUT_@ Module location error.'
		B	955	endif	
		B	956		
		B	957	if PUT_z_ROWCOL	!= 0DC6H
		B	958	warning	'PUT_z_ROWCOL Module location error.'
		B	959	endif	
		B	960		
		B	961	if ROWCOL_2_ADDR	!= 0DD0H
		B	962	warning	'ROWCOL_2_ADDR Module location error.'
		B	963	endif	
		B	964		
		B	965	; if RST38z	!= 1BFFH
		B	966	; warning	'RST38@ Module location error.'
		B	967	; endif	
		B	968		

\*\*\*\*\* TRSDOS \*\*\*\*\*  
 < SYSRES HIGH memory region. >

PC	Object	I	Line	Source	...	...	...	...
		B	969	if RSTOR\$	!=	04C4H		
		B	970	warning		'RSTOR\$	Module location error.'	
		B	971	endif				
		B	972					
		B	973	if RWRIz	!=	13A2H		
		B	974	warning		'RWRI@	Module location error.'	
		B	975	endif				
		B	976					
		B	977	if SBUFF\$	!=	1D00H		
		B	978	warning		'SBUFF\$	Module location error.'	
		B	979	endif				
		B	980					
		B	981	if SETzEXEC	!=	1A79H		
		B	982	warning		'SET@EXEC	Module location error.'	
		B	983	endif				
		B	984					
		B	985	if SET_SCROLL	!=	0CF3H		
		B	986	warning		'SET_SCROLL	Module location error.'	
		B	987	endif				
		B	988					
		B	989	; if SPACE4\$	!=	2180H		
		B	990	; warning		'SPACE4\$	Module location error.'	
		B	991	; endif				
		B	992					
		B	993	if STACK\$	!=	0380H		
		B	994	warning		'STACK\$	Module location error.'	
		B	995	endif				
		B	996					
		B	997	if START\$	!=	0000H		
		B	998	warning		'START\$	Module location error.'	
		B	999	endif				
		B	1000					
		B	1001	if SVCRET\$	!=	000BH		
		B	1002	warning		'SVCRET\$	Module location error.'	
		B	1003	endif				
		B	1004					
		B	1005	if SVCTAB\$	!=	0100H		
		B	1006	warning		'SVCTAB\$	Module location error.'	
		B	1007	endif				
		B	1008					
		B	1009	if SYSERR\$	!=	1B13H		
		B	1010	warning		'SYSERR\$	Module location error.'	
		B	1011	endif				
		B	1012					
		B	1013	if TCB\$ !=	004EH			
		B	1014	warning		'TCB\$	Module location error.'	
		B	1015	endif				
		B	1016					
		B	1017	if TIME\$ !=	002DH			
		B	1018	warning		'TIME\$	Module location error.'	
		B	1019	endif				
		B	1020					
		B	1021	if TIMER\$	!=	002CH		
		B	1022	warning		'TIMER\$	Module location error.'	
		B	1023	endif				
		B	1024					
		B	1025	if TIMSL\$	!=	002BH		
		B	1026	warning		'TIMSL\$	Module location error.'	
		B	1027	endif				
		B	1028					

\*\*\*\*\* TRSDOS \*\*\*\*\*  
 < SYSRES HIGH memory region. >

PC	Object	I	Line	Source	...
		B	1029	; if TIMTSK\$	!= 0713H
		B	1030	; warning	'TIMTSK\$ Module location error.'
		B	1031	; endif	
		B	1032		
		B	1033	if TMPMT\$	!= 04C3H
		B	1034	warning	'TMPMT\$ Module location error.'
		B	1035	endif	
		B	1036		
		B	1037	if TRACE_INT	!= 07b1H
		B	1038	warning	'TRACE_INT Module location error.'
		B	1039	endif	
		B	1040		
		B	1041	; if TSLPX@	!= 0E29H
		B	1042	; warning	'TSLPX@ Module location error.'
		B	1043	; endif	
		B	1044		
		B	1045	if TYPHK\$	!= 0A8FH
		B	1046	warning	'TYPHK\$ Module location error.'
		B	1047	endif	
		B	1048		
		B	1049	if TYPTSK\$	!= 0B26H
		B	1050	; warning	'TYPTSK\$ Module location error.'
		B	1051	endif	
		B	1052		
		B	1053	if WRINT\$	!= 0080H
		B	1054	warning	'WRINT\$ Module location error.'
		B	1055	endif	
		B	1056		
		B	1057	if USTOR\$	!= 0013H
		B	1058	warning	'USTOR\$ Module location error.'
		B	1059	endif	
		B	1060		
		B	1061	if ZERO\$ !=	0401H
		B	1062	warning	'ZERO\$ Module location error.'
		B	1063	endif	
		B	1064		
		B	1065	if ZEROAz	!= 13A0H
		B	1066	warning	'ZEROA@ Module location error.'
		B	1067	endif	
		B	1068		
		B	1069		
		B	1070	; if zDTHDLR	!= 1420H
		B	1071	; warning	'zDTHDLR Module location error.'
		B	1072	; endif	
		B	1073		
		B	1074	; if zIPL	!= 1BF2H
		B	1075	; warning	'zIPL Module location error.'
		B	1076	; endif	
		B	1077		
		B	1078	; if zOPREG	!= 0084H
		B	1079	; warning	'zOPREG Module location error.'
		B	1080	; endif	
		B	1081		
		B	1082	; if zPCERV	!= 0000H
		B	1083	; warning	'Module location error.'
		B	1084	; endif	
		B	1085		
		B	1086	; if zRSTNMI	!= 0FE9H
		B	1087	; warning	'zRSTNMI Module location error.'
		B	1088	; endif	

\*\*\*\*\* TRSDOS \*\*\*\*\*  
 < SYSRES HIGH memory region. >

PC	Object	I	Line	Source	...
		B	1089		
		B	1090		
		B	1091	; if zHERTZ\$	!= 0750H
		B	1092	; warning	'zHERTZ\$ Module location error.'
		B	1093	; endif	
		B	1094		
		B	1095	; if BOOTST\$	!= 439DH
		B	1096	; warning	'BOOTST\$ Module location error.'
		B	1097	; endif	
		B	1098		
		B	1099	; if zAUTO\$	!= 2019H
		B	1100	; warning	'AUTO? Module location error.'
		B	1101	; endif	
		B	1102		
		B	1103	if BREAKz	!= 1C60H
		B	1104	warning	'BREAK? Module location error.'
		B	1105	endif	
		B	1106		
		B	1107	; if CONFIG\$	!= 2067H
		B	1108	; warning	'CONFIG\$ Module location error.'
		B	1109	; endif	
		B	1110		
		B	1111	if DzFBYT8	!= 1A26H
		B	1112	warning	'D@FBYT8 Module location error.'
		B	1113	endif	
		B	1114		
		B	1115	if GET_z_ROWCOL	!= 0DAEH
		B	1116	warning	'GET_@_ROWCOL Module location error.'
		B	1117	endif	
		B	1118		
		B	1119	if KCKz	!= 07D6H
		B	1120	warning	'KCK@ Module location error.'
		B	1121	endif	
		B	1122		
		B	1123	if DCTBYT8z	!= 1A29H
		B	1124	warning	'DCTBYT8@ Module location error.'
		B	1125	endif	
		B	1126		
		B	1127	if DCTFLDz	!= 1A34H
		B	1128	warning	'DCTFLD@ Module location error.'
		B	1129	endif	
		A	176		
		A	177	END	trs_os

Errors: 0  
 Warnings: 0  
 Lines Assembled: 18396