SDK-51

MCS-51^{tm} SYSTEM DESIGN KIT

MONITOR LISTING MANUAL

Manual Order No: 121590-003

intel®

SDK-51

MCS-51^{tm} SYSTEM DESIGN KIT

MONITOR LISTING MANUAL

Manual Order No: 121590-003

REV.	REVISION HISTORY	PRINT DATE
-001	Original Issue	5/81
-002	Minor Monitor Upgrade	10/81

A455/1081/1.5K DD

```
 o   !    !
 !!-.T .-.!
 !! !! ! !!
        !_.
```

PREFACE

This manual contains the program listing of the SDK-51 system monitor. For details on the assembly and operation of the SDK-51 system design kit, refer to the following Intel publications.

SDK-51 MCS-51^{tm} System Design Kit Assembly Manual, manual order number 121589.

SDK-51 MCS-51^{tm} System Design Kit User's Guide, manual order number 121588.

```
ISIS-II MCS-51 MACRO ASSEMBLER X040
OBJECT MODULE PLACED IN :F3:SDKMON.HEX
ASSEMBLER INVOKED BY:  :F1:ASM51 :F1:SDKMON.SRC PRINT(:F2:SDKMON.LST) OBJECT(:F3:SDKMON.HEX) DATE(8,12,81) WORKFILES(:F3
                       :,:F3:) EP DB SB


LOC  OBJ            LINE     SOURCE

                       1     $nomacro
                       2     $XREF
                       3     $TITLE('SDK-51 MONITOR CODE   INTEL PROPRIETARY VERS. #1.03')
                       4     ;*****************************************************************
                       5     ;
                       6     ;
                       7     ;       SDK-51 MONITOR  INTEL PROPRIETARY
                       8     ;       THIS SOFTWARE IS COPYRIGHTED UNDER INTEL PART NUMBER 162787-004
                       9     ;
                      10     ;       VERSION 1.03            8-12-81;
                      11     ;
                      12     ;       N   N  00000  TTTTT  EEEEE    !!
                      13     ;       N N N  0   0    T    E        !!
                      14     ;       N N N  0   0    T    EEEE     !!
                      15     ;       N  NN  0   0    T    E        !!
                      16     ;       N  NN  0   0    T    E
                      17     ;       N   N  00000    T    EEEEE    !!
                      18     ;
                      19     ;
                      20     ;*****************************************************************
                      21     ;
                      22     ;
                      23     ;       COPYRIGHT (C) 1981 INTEL CORPORATION.;
                      24     ;       ALL RIGHTS RESERVED.
                      25     ;
                      26     ;       NO PART OF THIS PROGRAM OR PUBLICATION MAY BE REPRODUCED,
                      27     ;       TRANSMITTED, TRANSCRIBED, STORED IN A RETRIEVAL SYSTEM, OR
                      28     ;       TRANSLATED INTO ANY LANGUAGE OR COMPUTER LANGUAGE, IN ANY
                      29     ;       FORM OR BY ANY MEANS, ELECTRONIC, MECHANICAL, MAGNETIC,
                      30     ;       OPTICAL, CHEMICAL, MANUAL OR OTHERWISE, WITHOUT THE PRIOR
                      31     ;       WRITTEN PERMISSION OF INTEL CORPORATION, 3065 BOWERS AVENUE,
                      32     ;       SANTA CLARA, CALIFORNIA 95051.
                      33     ;
                      34     ;
                      35     ;
                      36     ;*****************************************************************
                      37 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                         ;******************************************************************
                         ;
                         ;               TABLE OF CONTENTS
                         ;
                         ;       PREFACE:  HOW TO USE THIS LISTING
                         ;
                         ;       This monitor and the assembler/disassembler are written
                         ;       in ASM51 code.  These listings may serve the user as
                         ;       both debug aids and as an example of how many of the unique
                         ;       ASM51 commands may be used in context.
                         ;
                         ;       In general, the organization on this monitor listing is as
                         ;       follows.  The POWER_ON routine is the 'cold start' location,
                         ;       that is, it does a hardware reset.  START is the main program
                         ;       which is the top of the idle loop.  It is also the 'warm start'
                         ;       location, that is it does software resets and initializations.
                         ;
                         ;       Upon receipt of a command from the user via the console, START
                         ;       determines which routine will handle each command and branches
                         ;       to it.  The command handler routines will always have a label
                         ;       with the suffix '_CMD'.
                         ;
                         ;       HEADER BLOCK INFORMATION:
                         ;
                         ;       At the beginning of each subroutine, on a new page, there will
                         ;       be a block containing the name of the routine.  The name may
                         ;       have an '(I)' or a '(U)' as a prefix.  The I indicates that
                         ;       the routine is internal only, the U indicates that the routine is
                         ;       only suitable for use by the user.
                         ;
                         ;       The abstract contains a brief description of what the function
                         ;       of that module is and highlights of any subtle cautions or user
                         ;       interface notes.  There will also be lists of inputs, outputs,
                         ;       error exits, variables modified and subroutines called.  The
                         ;       rules for these lists are strict.
                         ;
                         ;       Input lists contain only explicitly passed global or local variables.
                         ;       Information returned by any other procedure (i.e. passed parameters)
                         ;       that is called by the procedure whose block you are reading will not
                         ;       be included in the input list.
                         ;
                         ;       Output lists contain only variables altered by the procedure for
                         ;       the purpose of transmitting necessary information to another procedure.
                         ;
                         ;       The variables modified lists contains only local variables, registers
                         ;       or memory locations that are modified and not restored by the end on
                         ;       the routine.
                         ;
                         ;       The error exits will contain any error number that is locally
                         ;       generated.  There is the possibility that an error may be detected
                         ;       in a routine with no error exits noted if the error number was set
                         ;       in a previous routine and just 'falls through' because the error is
                         ;       still the same.
                         ;
                         ;       The subroutines called list will contain any other routine that is
```

```
LOC  OBJ            LINE     SOURCE

                         ;       directly called or jumped to by the procedure in question.
                         ;
                         ;       XREF:
                         ;
                         ;       At the back of the monitor listing and again at the back of the
                         ;       assembler/disassembler listing there is a table of cross references.
                         ;       Each variable name is listed in alphabetical order along with its
                         ;       type (that is in what type of memory does it reside, is it a label
                         ;       or a number), the address value it has and all of the line numbers
                         ;       where that variable name appears.  The line number with the '#'
                         ;       designation is the line where the variable is defined.
                         ;
                         ;
                         ;       CONTENTS:
                         ;
                         ;       This monitor listing contains one source file and five
                         ;       include files.  Each include file contains a number of functions,
                         ;       tables and subroutines which will each have their own header block
                         ;       and will begin on a new page.  The files are as follows:
                         ;
                         ;       SDKMON.SRC (SOURCE FILE)
                         ;
                         ;                       JUMP TABLE FOR USER ACCESSABLE ROUTINES
                         ;                       CONSTANTS
                         ;                       VARIABLES
                         ;                       FLAGS
                         ;                       TOKEN EQUATES
                         ;                       TOKEN TABLE
                         ;
                         ;                       POWER_ON
                         ;                       SIGN_ON
                         ;                       START
                         ;                       INIT_IO
                         ;                       (I)WAIT_FOR_USER
                         ;                       CHECK_EPROMS
                         ;
                         ;               COMMON.INC (INCLUDE FILE)
                         ;
                         ;                       CONSTANTS USED BY ALL MODULES
                         ;                       GLOBAL VARIABLES USED BY MORE THAT ONE MAIN MOD.
                         ;                       ARRAYS
                         ;                       VARIABLES
                         ;                       FLAGS
                         ;                       REGISTERS
                         ;                       JUMP TABLE ENTRY ADDRESSES FOR ALL MODULES
                         ;
                         ;               UTILIT.INC (INCLUDE FILE)
                         ;
                         ;                       (I)ERROR
                         ;                       (I)EOL_CHECK
                         ;                       INC_PNT/DEC_PNT/SWAP_POINTERS
                         ;                       SPACCO/(I)CO
                         ;                       ICI
                         ;                       ICSTS
                         ;                       (U)CSTS
```

```
LOC  OBJ            LINE     SOURCE

                     148     ;                      (U)CI
                     149     ;                      (I)UPI_CMD
                     150     ;                      UPI_OUT
                     151     ;                      UPI_IN
                     152     ;                      (I)CONTINUATION_LINE
                     153     ;                      (I)FETCH/(I)STORE
                     154     ;                      (I)NEWLINE
                     155     ;                      AZTEST/NMTEST/HXTEST/ALFNUM
                     156     ;                      LSSEQL
                     157     ;                      (I)GETNUM/(I)GETEOL/(I)GET_COMMA
                     158     ;                      ISIT_DISPLAY
                     159     ;                      (I)GET_PART
                     160     ;                      (I)SAVE_AND_DISPLAY
                     161     ;                      CONVHEX
                     162     ;                      (I)LSTWRD/(I)LSTBYT
                     163     ;                      PAINTER
                     164     ;                      GETCHR
                     165     ;                      (I)GETOKE
                     166     ;                      NUMBER
                     167     ;                      SYMBOL
                     168     ;                      STRING_SPACE
                     169     ;                      (I)PIRNT_STRING
                     170     ;                      (I)DISPLAY_TOKEN
                     171     ;                      ASCII_TO_HEX
                     172     ;                      ITIME
                     173     ;
                     174     ;              DISCHA.INC (INCLUDE FILE)
                     175     ;
                     176     ;                      DISPLAY
                     177     ;                      LODMEM
                     178     ;                      FILLMEM
                     179     ;                      DISMEM
                     180     ;                      BMOVE
                     181     ;                      MODBRK
                     182     ;                      ACC_MOD
                     183     ;                      KEYWORD_DISPLAY
                     184     ;
                     185     ;              XQT.INC (INCLUDE FILE)
                     186     ;
                     187     ;                      BREAK
                     188     ;                      UNBREAK
                     189     ;                      READ_PC/WRITE_PC
                     190     ;                      CHECK_FROM
                     191     ;                      BREAK_VECTOR
                     192     ;                      STEP_CMD
                     193     ;                      STEP51_RET
                     194     ;                      GO_CMD
                     195     ;
                     196     ;              MONFUN.INC (INCLUDE FILE)
                     197     ;
                     198     ;                      LIST_CMD
                     199     ;                      BAUD_CMD
                     200     ;                      TOP_CMD
                     201     ;                      CAUSE_CMD
                     202     ;                      SEND_BYTE
```

```
LOC  OBJ            LINE     SOURCE

                     203     ;                       HEXBIN
                     204     ;                       GET_TYPE
                     205     ;                       LOAD_HEX
                     206     ;                       STORE_HEX
                     207     ;                       LOAD_CMD
                     208     ;                       SAVE_CMD
                     209     ;                       DOWNLOAD_CMD
                     210     ;                       UPLOAD_CMD
                     211     ;
                     212     ;
                     213     ;*******************************************************************
                     214 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                          +1  $INCLUDE(:F1:COMMON.INC)
 E000             =1          BASE    EQU     0E000H
                  =1          ;***** CONSTANTS USED BY ALL MODULES *****
                  =1
  0001            =1          NUMBER_TOKE      EQU     01H             ;Constant (GETOKE,number token)
  0003            =1          BAR_TOKE         EQU     03H             ;Constant (GETOKE,slash (/) token)
  0006            =1          POUND_TOKE       EQU     06H     ;Constant
  0005            =1          PLUS_TOKE        EQU     05H
  0007            =1          EOL_TOKE         EQU     07H             ;Constant (GETOKE,end of line token)
  000A            =1          ATA_TOKE         EQU     0AH
  005E            =1          C_TOKE           EQU     05EH
  0080            =1          CBYTE_TOKE       EQU     080H
  00A1            =1          DPTR_TOKE        EQU     0A1H
  00D4            =1          ORG_TOKE         EQU     0D4H
  00A0            =1          PC_TOKE          EQU     0A0H
  0040            =1          REG              EQU     40H
  0010            =1          OFST             EQU     10H
  0018            =1          LINMAX           EQU     24
  0004            =1          TOKSIZ           EQU     4
  0080            =1          BLINK            EQU     80H             ;Set the blink bit in bytes to go to the UPI
  0000            =1          SELECT_CON       EQU     00H             ;Set up UPI for on-board console
                  =1
                  =1          ;***** GLOBAL VARIABLES USED BY MORE THAN ONE MAIN MODULE *****
----              =1                  DSEG
0024              =1                  ORG     24H
                  =1          ;***** ARRAYS *****
                  =1
0024              =1          LINBUF:                  DS      LINMAX          ;Input line buffer(24 chars)
003C              =1          STRGBF:                  DS      TOKSIZ          ;Buffer for string
0040              =1          WORKING_SPACE:           DS      3               ;Buffer for ASM/DASM
                  =1
                  =1          ;***** VARIABLES *****
                  =1
0043              =1          ERRNUM:                  DS      1
0044              =1          PNTHGH:                  DS      1
0045              =1          PNTLOW:                  DS      1
0046              =1          SELECT:                  DS      1
0047              =1          TEMP_LOW:                DS      1
0048              =1          TOKSTR:                  DS      1
0049              =1          VALHGH:                  DS      1
004A              =1          VALLOW:                  DS      1
004B              =1          ASM_PC_HIGH:             DS      1
004C              =1          ASM_PC_LOW:              DS      1
004D              =1          NUMBER_OF_BYTES:         DS      1
004E              =1          OUR_CODE_HIGH:           DS      1
004F              =1          OUR_CODE_LOW:            DS      1
0050              =1          CHARIN:                  DS      1
0051              =1          CHRCNT:                  DS      1
0052              =1          LINE_START:              DS      1
0053              =1          LINCNT:                  DS      1
0054              =1          LNLGTH:                  DS      1
0055              =1          STRGCT:                  DS      1
0056              =1          TEMP1:                   DS      1
0057              =1          PARTIT_LO_HIGH:          DS      1
0058              =1          PARTIT_LO_LOW:           DS      1
```

```
LOC  OBJ          LINE     SOURCE

0059         =1    270     PARTIT_HI_HIGH:           DS      1
005A         =1    271     PARTIT_HI_LOW:            DS      1
             =1    272
             =1    273     ;***** FLAGS *****
----         =1    274             BSEG
0000         =1    275             ORG     0
             =1    276
0000         =1    277     B_O_T:                    DBIT    1
0001         =1    278     LSTFLG:                   DBIT    1
----         =1    279             CSEG
             =1    280     ;***** REGISTERS *****
  REG        =1    281     POINT0          EQU       R0              ;Register (addr pointer)
  REG        =1    282     POINT1          EQU       R1              ;Register (addr pointer)
  REG        =1    283     PARAM1          EQU       R2              ;Register (parameter passing media #1)
  REG        =1    284     PARAM2          EQU       R3              ;REGISTER (Parameter passing media #2)
  REG        =1    285     PARAM3          EQU       R4              ;REGISTER (Parameter passing media #3)
  REG        =1    286     PARAM4          EQU       R5
  REG        =1    287     PARAM5          EQU       R6
  REG        =1    288     PARAM6          EQU       R7
  REG        =1    289     COUNT           EQU       R7
  REG        =1    290     CHECKSUM        EQU       R6
  REG        =1    291     TEMP            EQU       R5
             =1    292     ;***** END OF VARIABLE  EQUATES *****
             =1    293     ;*****************************************************************
             =1    294     ;                  JUMP TABLE ENTRY ADDRESSES FOR ALL MODULES
             =1    295     ;*****************************************************************
  E006       =1    296     CO                        EQU     6 + BASE
  E009       =1    297     CI                        EQU     9 + BASE
  E00C       =1    298     CSTS                      EQU     0CH + BASE
  E00F       =1    299     NEWLINE                   EQU     0FH + BASE
  E012       =1    300     TIME                      EQU     12H + BASE
  E015       =1    301     LSTBYT                    EQU     15H + BASE
  E018       =1    302     LSTWRD                    EQU     18H + BASE
  E01E       =1    303     PRINT_STRING              EQU     1EH + BASE
             =1    304
  E04A       =1    305     FETCH                     EQU     4AH + BASE
  E04D       =1    306     STORE                     EQU     4DH + BASE
  E050       =1    307     GETNUM                    EQU     50H + BASE
  E053       =1    308     GETEOL                    EQU     53H + BASE
  E056       =1    309     GETOKE                    EQU     56H + BASE
  E059       =1    310     DISPLAY_TOKEN             EQU     59H + BASE
  E05C       =1    311     SAVE_AND_DISPLAY          EQU     5CH + BASE
  E05F       =1    312     ERROR                     EQU     5FH + BASE
  E062       =1    313     WAIT_FOR_USER             EQU     62H + BASE
  E065       =1    314     GET_PART                  EQU     65H + BASE
  E068       =1    315     CONTINUATION_LINE         EQU     68H + BASE
  E06B       =1    316     GET_COMMA                 EQU     6BH + BASE
  E06E       =1    317     EOL_CHECK                 EQU     6EH + BASE
                   318 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                     319     ;*****************************************************************
E000                 320             ORG     BASE
                     321     ;
E000 02E274          322             JMP     POWER_ON        ; Initialize and start monitor.
                     323     ;
                     324     ;*****************************************************************
                     325     ;               JUMP TABLE FOR USER ACCESSABLE ROUTINES
                     326
                     327
E003 02EDC6          328     BREAK:  LJMP    IBREAK                  ;Do not access this vector except through
                     329                                             ;normal SDK system interrupts,
                     330                                             ;breaks and keyclosures
                     331
E006 02E5E8          332             LJMP    ICO
E009 02E619          333             LJMP    UCI
E00C 02E613          334             LJMP    UCSTS
E00F 02E717          335             LJMP    INEWLINE
E012 02EA45          336             LJMP    ITIME
E015 02E7F9          337             LJMP    ILSTBYT
E018 02E7F4          338             LJMP    ILSTWRD
E01B 02EA3C          339             LJMP    IASCII_TO_HEX
E01E 02E9FF          340             LJMP    IPRINT_STRING
E021 02E274          341             LJMP    POWER_ON                ;The rest of the jump table reserved
E024 02E274          342             LJMP    POWER_ON                ;for future expansion.
E027 02E274          343             LJMP    POWER_ON
E02A 02E274          344             LJMP    POWER_ON
E02D 02E274          345             LJMP    POWER_ON
                     346
E030                 347             ORG     BASE+30H
                     348
                     349
E030 20284329        350     COPYRIGHT:      DB      ' (C) 1981 INTEL CORP.
E034 20313938
E038 3120494E
E03C 54454C20
E040 434F5250
E044 2E20
E046 08              351     DATECODE:       DB      8H,12H,81H
E047 12
E048 81
E049 00              352     STORED_CHECK_SUM:       DB      0
E04A 02E66B          353             LJMP    IFETCH
E04D 02E672          354             LJMP    ISTORE
E050 02E769          355             LJMP    IGETNUM
E053 02E773          356             LJMP    IGETEOL
E056 02E8BC          357             LJMP    IGETOKE
E059 02EA12          358             LJMP    IDISPLAY_TOKEN
E05C 02E7DD          359             LJMP    ISAVE_AND_DISPLAY
E05F 02E3E4          360             LJMP    IERROR
E062 02E3B0          361             LJMP    IWAIT_FOR_USER
E065 02E7A2          362             LJMP    IGET_PART
E068 02E65D          363             LJMP    ICONTINUATION_LINE
E06B 02E77A          364             LJMP    IGET_COMMA
E06E 02E5BB          365             LJMP    IEOL_CHECK
                     366
```

```
LOC  OBJ            LINE     SOURCE

                     367     ;***** CONSTANTS *****
                     368
  0004               369     EQUAL_TOKE       EQU     4        ;Constant (GETOKE,EQUAL TOKEN)
  0002               370     COMMA_TOKE       EQU     02H      ;Constant(Comma token)
  0008               371     BACKSP           EQU     08H      ;Constant (GETCHR,LITERAL 'BACK SPACE')
  000D               372     CR               EQU     0DH      ;Constant (NEWLIN,LITERAL 'CARRAGE RETURN')
  000A               373     LF               EQU     0AH      ;Constant (NEWLIN,LITERAL 'LINE FEED')
  0009               374     HORIZONTAL_TAB   EQU     09H      ;Constant (TAB KEY)
  007F               375     RBOUT            EQU     7FH      ;Constant (GETCHR,LITERAL 'DELETE')
  001B               376     ESC              EQU     1BH      ;Constant (EXECUT,LISTER 'ESCAPE')
  0007               377     STACK            EQU     07H
  0004               378     RESET_CMD        EQU     04H      ;UPI reset command
  0008               379     CLR_BRK_LATCHES  EQU     08H
  0083               380     TOP_PORT         EQU     83H      ;UPI top port
  0003               381     GR_PORT          EQU     03H      ;UPI hardware GO register port
  0009               382     NO_BREAK         EQU     09H      ;Disables break logic
  0002               383     CASSETTE_READ    EQU     02H      ;UPI select cassette read mode
  0082               384     CASSETTE_WRITE   EQU     82H      ;UPI select cassette write mode
  0001               385     USART_MODE       EQU     01H      ;UPI serial port select for up/down load
  0001               386     SINGLE_BREAK     EQU     01H      ;Enables single step breaks.
  000D               387     DATA_BREAK       EQU     0DH      ;Enables data memory breaks
  000B               388     PROGRAM_BREAK    EQU     0BH      ;Enables program memory breaks
  A001               389     UPI_CONTROL      EQU     0A001H
  A000               390     UPI_DATA         EQU     0A000H
  B000               391     RAMOFF           EQU     0B000H   ;Constant (STORE,16-BIT INTERNAL RAM OFFSET)
  C000               392     BRKOFF           EQU     0C000H   ;Constant (STORE,16-BIT,BREAK RAM OFFSET)
  B800               393     RAMIO            EQU     0B800H   ;Constant (STORE,16-BIT INTERNAL RAM I/O OFFSET)
  0005               394     TIMER_HIGH       EQU     05H      ;Constant (ADDRESS OF 8155 TIMER HIGH BYTE)
  0040               395     CONTINUOUS_MODE  EQU     40H      ;Constant (COMMAND MODE FOR TIMER)
  00C0               396     START_16_TIMER   EQU     0C0H     ;Constant (COMMAND TO LOAD AND START TIMER)
  00FF               397     MAXLOW           EQU     0FFH     ;Constant
  001F               398     MAXHGH           EQU     01FH     ;Constant
  00F1               399     UPI_DATA_IMAGE   EQU     0F1H     ;Software version of UPI input data.
  00F2               400     SAVE_SEL         EQU     0F2H     ;Used to store the token during emulation.
  00F3               401     ADDR_SAVE_HIGH   EQU     0F3H     ;Saves display address during emulation.
  00F4               402     ADDR_SAVE_LOW    EQU     0F4H
  00F5               403     DELAY            EQU     0F5H     ;Stores multi-step delay count.
  00F6               404     GR               EQU     0F6H     ;GO register
  00F7               405     BAUD_HIGH        EQU     0F7H
  00F8               406     BAUD_LOW         EQU     0F8H     ;Stores baud rate information.
  00F9               407     TOP_STORE        EQU     0F9H     ;Stores the user TOP value
  00FA               408     MON_FLAGS        EQU     0FAH     ;Stores monitor flags
  00FB               409     BREAK_STATUS     EQU     0FBH     ;Used to store the step flag during emulation.
  00FC               410     BAUDKEY          EQU     0FCH     ;Stores coded baud info in one byte
  00FD               411     UPC              EQU     0FDH     ;Software copy of PC when not in execution.
  00FB               412     NOT_STEP         EQU     0FBH     ;Stored in BREAK_STATUS to indicate not stepping
  00FE               413     SINGLESTEP       EQU     0FEH     ;Stored in BREAK_STATUS to indicate single step
  00FF               414     MULTISTEP        EQU     0FFH     ;Stored to indicate multiple single steps.
                     415
                     416
                     417     ;**********  VARIABLES **********
                     418
----                 419             DSEG
005B                 420             ORG      (PARTIT_HI_LOW+1)
005B                 421     TOKSAV:          DS      1        ;DATA ADDR
```

```
LOC  OBJ           LINE     SOURCE

005C                422     DLYCNT:         DS      1       ;DATA ADDR
005D                423     COUNTR:         DS      1
005E                424     VPC_LOW:        DS      1
005F                425     VPC_HIGH:       DS      1
0060                426     CAUSE_IMAGE:    DS      1
0061                427     PCNTHI:         DS      1
0062                428     PCNTLO:         DS      1
0063                429     LENGTH_HIGH:    DS      1
0064                430     LENGTH_LOW:     DS      1
0065                431     TYPE:           DS      1
                    432
                    433     ;***** FLAGS *****
                    434
----                435             BSEG
0002                436             ORG     (LSTFLG+1)
0002                437     ANY_BR_FLAG:    DBIT    1
0003                438     FIRST_FLAG:     DBIT    1
0004                439     MAXNUM_FLAG:    DBIT    1
0005                440     BINARY_FLG:     DBIT    1
----                441             CSEG
                    442 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                     443     ;*******************************************************************
                     444     ;                        TOKEN EQUATES
                     445     ;*******************************************************************
                     446
  005F               447     ATDPTR_TOKE      EQU      15+REG+OFST
  0052               448     ATR0_TOKE        EQU      2+REG+OFST
  0053               449     ATR1_TOKE        EQU      3+REG+OFST
  0051               450     A_TOKE           EQU      051H
  005C               451     AB_TOKE          EQU      12+REG+OFST
  0088               452     ABR_TOKE         EQU      088H
  0012               453     ACALL_TOKE       EQU      2+OFST
  0098               454     ACC_TOKE         EQU      098H
  0024               455     ADD_TOKE         EQU      20+OFST
  0023               456     ADDC_TOKE        EQU      19+OFST
  0013               457     AJMP_TOKE        EQU      3+OFST
  0021               458     ANL_TOKE         EQU      17+OFST
  00B0               459     ASM_TOKE         EQU      0B0H
  009B               460     B_TOKE           EQU      09BH
  00D0               461     BAUD_TOKE        EQU      0D0H
  0089               462     BR_TOKE          EQU      089H
  00D2               463     CAUSE_TOKE       EQU      0D2H
  0019               464     CJNE_TOKE        EQU      9+OFST
  002A               465     CLR_TOKE         EQU      26+OFST
  002B               466     CPL_TOKE         EQU      27+OFST
  002C               467     DA_TOKE          EQU      28+OFST
  00B8               468     DASM_TOKE        EQU      0B8H
  00D3               469     DATA_TOKE        EQU      0D3H
  0082               470     DBYTE_TOKE       EQU      082H
  0035               471     DEC_TOKE         EQU      37+OFST
  0031               472     DIV_TOKE         EQU      33+OFST
  0025               473     DJNZ_TOKE        EQU      21+OFST
  00E0               474     DOWNLOAD_TOKE    EQU      0E0H
  0008               475     FOREVER_TOKE     EQU      008H
  0009               476     FROM_TOKE        EQU      009H
  00C2               477     GO_TOKE          EQU      0C2H
  0037               478     INC_TOKE         EQU      39+OFST
  0027               479     JB_TOKE          EQU      23+OFST
  0028               480     JBC_TOKE         EQU      24+OFST
  0018               481     JC_TOKE          EQU      8+OFST
  0032               482     JMP_TOKE         EQU      34+OFST
  0026               483     JNB_TOKE         EQU      22+OFST
  0017               484     JNC_TOKE         EQU      7+OFST
  0015               485     JNZ_TOKE         EQU      5+OFST
  0016               486     JZ_TOKE          EQU      6+OFST
  0010               487     LCALL_TOKE       EQU      0+OFST
  00D7               488     LIST_TOKE        EQU      0D7H
  0011               489     LJMP_TOKE        EQU      1+OFST
  00E2               490     LOAD_TOKE        EQU      0E2H
  00B9               491     MODE_TOKE        EQU      0B9H
  001F               492     MOV_TOKE         EQU      15+OFST
  001A               493     MOVC_TOKE        EQU      10+OFST
  001B               494     MOVX_TOKE        EQU      11+OFST
  0030               495     MUL_TOKE         EQU      32+OFST
  003B               496     NOP_TOKE         EQU      43+OFST
  000F               497     ON_TOKE          EQU      00FH
```

```
LOC  OBJ            LINE     SOURCE

  000B               498     OR_TOKE          EQU      0BH
  0022               499     ORL_TOKE         EQU      18+OFST
  002D               500     POP_TOKE         EQU      29+OFST
  00D5               501     PROGRAM_TOKE     EQU      0D5H
  0099               502     PSW_TOKE         EQU      099H
  002F               503     PUSH_TOKE        EQU      31+OFST
  0090               504     R0_TOKE          EQU      090H
  0091               505     R1_TOKE          EQU      091H
  0092               506     R2_TOKE          EQU      092H
  0093               507     R3_TOKE          EQU      093H
  0094               508     R4_TOKE          EQU      094H
  0095               509     R5_TOKE          EQU      095H
  0096               510     R6_TOKE          EQU      096H
  0097               511     R7_TOKE          EQU      097H
  0084               512     RBIT_TOKE        EQU      084H
  0000               513     RBS_TOKE         EQU      000
  0081               514     RBYTE_TOKE       EQU      081H
  000E               515     RESET_TOKE       EQU      00EH
  003A               516     RET_TOKE         EQU      42+OFST
  0039               517     RETI_TOKE        EQU      41+OFST
  0034               518     RL_TOKE          EQU      36+OFST
  0033               519     RLC_TOKE         EQU      35+OFST
  0038               520     RR_TOKE          EQU      40+OFST
  0036               521     RRC_TOKE         EQU      38+OFST
  00E3               522     SAVE_TOKE        EQU      0E3H
  0029               523     SETB_TOKE        EQU      25+OFST
  0014               524     SJMP_TOKE        EQU      4+OFST
  009A               525     SP_TOKE          EQU      09AH
  00C1               526     STEP_TOKE        EQU      0C1H
  001E               527     SUBB_TOKE        EQU      14+OFST
  002E               528     SWAP_TOKE        EQU      30+OFST
  000C               529     TILL_TOKE        EQU      00CH
  00A2               530     TM0_TOKE         EQU      0A2H
  00A3               531     TM1_TOKE         EQU      0A3H
  000D               532     TO_TOKE          EQU      00DH
  00D6               533     TOP_TOKE         EQU      0D6H
  00BA               534     TRANSFER_TOKE    EQU      0BAH
  00E1               535     UPLOAD_TOKE      EQU      0E1H
  00BB               536     VERIFY_TOKE      EQU      0BBH
  0086               537     XBYTE_TOKE       EQU      086H
  001D               538     XCH_TOKE         EQU      13+OFST
  001C               539     XCHD_TOKE        EQU      12+OFST
  0020               540     XRL_TOKE         EQU      16+OFST
                     541     ;
                     542     ;
                     543     ;***** TOKEN TABLE *****
                     544     ;                TOKTBL must match entry for entry with KEYTAB so that the ASCII
                     545     ;                for each token will match the token.
                     546     ;
                     547     TOKTBL:
E071 0A              548             DB       ATA_TOKE
E072 5F              549             DB       ATDPTR_TOKE
E073 52              550             DB       ATR0_TOKE
E074 53              551             DB       ATR1_TOKE
E075 51              552             DB       A_TOKE
```

```
LOC  OBJ            LINE     SOURCE

E076 5C              553             DB      AB_TOKE
E077 88              554             DB      ABR_TOKE
E078 12              555             DB      ACALL_TOKE
E079 98              556             DB      ACC_TOKE
E07A 24              557             DB      ADD_TOKE
E07B 23              558             DB      ADDC_TOKE
E07C 13              559             DB      AJMP_TOKE
E07D 21              560             DB      ANL_TOKE
E07E B0              561             DB      ASM_TOKE
E07F 9B              562             DB      B_TOKE
E080 D0              563             DB      BAUD_TOKE
E081 89              564             DB      BR_TOKE
E082 5E              565             DB      C_TOKE
E083 D2              566             DB      CAUSE_TOKE
E084 80              567             DB      CBYTE_TOKE
E085 19              568             DB      CJNE_TOKE
E086 2A              569             DB      CLR_TOKE
E087 2B              570             DB      CPL_TOKE
E088 B8              571             DB      DASM_TOKE
E089 2C              572             DB      DA_TOKE
E08A B8              573             DB      DASM_TOKE
E08B D3              574             DB      DATA_TOKE
E08C 82              575             DB      DBYTE_TOKE
E08D 35              576             DB      DEC_TOKE
E08E 31              577             DB      DIV_TOKE
E08F 25              578             DB      DJNZ_TOKE
E090 E0              579             DB      DOWNLOAD_TOKE
E091 A1              580             DB      DPTR_TOKE
E092 09              581             DB      FROM_TOKE
E093 08              582             DB      FOREVER_TOKE
E094 09              583             DB      FROM_TOKE
E095 C2              584             DB      GO_TOKE
E096 37              585             DB      INC_TOKE
E097 27              586             DB      JB_TOKE
E098 28              587             DB      JBC_TOKE
E099 18              588             DB      JC_TOKE
E09A 32              589             DB      JMP_TOKE
E09B 26              590             DB      JNB_TOKE
E09C 17              591             DB      JNC_TOKE
E09D 15              592             DB      JNZ_TOKE
E09E 16              593             DB      JZ_TOKE
E09F 10              594             DB      LCALL_TOKE
E0A0 D7              595             DB      LIST_TOKE
E0A1 11              596             DB      LJMP_TOKE
E0A2 E2              597             DB      LOAD_TOKE
E0A3 B9              598             DB      MODE_TOKE
E0A4 1F              599             DB      MOV_TOKE
E0A5 1A              600             DB      MOVC_TOKE
E0A6 1B              601             DB      MOVX_TOKE
E0A7 30              602             DB      MUL_TOKE
E0A8 3B              603             DB      NOP_TOKE
E0A9 0F              604             DB      ON_TOKE
E0AA 0B              605             DB      OR_TOKE
E0AB D4              606             DB      ORG_TOKE
E0AC 22              607             DB      ORL_TOKE
```

```
LOC  OBJ            LINE     SOURCE

E0AD A0              608            DB      PC_TOKE
E0AE 2D              609            DB      POP_TOKE
E0AF D5              610            DB      PROGRAM_TOKE
E0B0 99              611            DB      PSW_TOKE
E0B1 2F              612            DB      PUSH_TOKE
E0B2 90              613            DB      R0_TOKE
E0B3 91              614            DB      R1_TOKE
E0B4 92              615            DB      R2_TOKE
E0B5 93              616            DB      R3_TOKE
E0B6 94              617            DB      R4_TOKE
E0B7 95              618            DB      R5_TOKE
E0B8 96              619            DB      R6_TOKE
E0B9 97              620            DB      R7_TOKE
E0BA 84              621            DB      RBIT_TOKE
E0BB 00              622            DB      RBS_TOKE
E0BC 81              623            DB      RBYTE_TOKE
E0BD 0E              624            DB      RESET_TOKE
E0BE 3A              625            DB      RET_TOKE
E0BF 39              626            DB      RETI_TOKE
E0C0 34              627            DB      RL_TOKE
E0C1 33              628            DB      RLC_TOKE
E0C2 38              629            DB      RR_TOKE
E0C3 36              630            DB      RRC_TOKE
E0C4 E3              631            DB      SAVE_TOKE
E0C5 29              632            DB      SETB_TOKE
E0C6 14              633            DB      SJMP_TOKE
E0C7 9A              634            DB      SP_TOKE
E0C8 C1              635            DB      STEP_TOKE
E0C9 1E              636            DB      SUBB_TOKE
E0CA 2E              637            DB      SWAP_TOKE
E0CB 0C              638            DB      TILL_TOKE
E0CC 0C              639            DB      TILL_TOKE
E0CD A2              640            DB      TM0_TOKE
E0CE A3              641            DB      TM1_TOKE
E0CF 0D              642            DB      TO_TOKE
E0D0 D6              643            DB      TOP_TOKE
E0D1 BA              644            DB      TRANSFER_TOKE
E0D2 E1              645            DB      UPLOAD_TOKE
E0D3 BB              646            DB      VERIFY_TOKE
E0D4 86              647            DB      XBYTE_TOKE
E0D5 1D              648            DB      XCH_TOKE
E0D6 1C              649            DB      XCHD_TOKE
E0D7 20              650            DB      XRL_TOKE
                     651     ;***** END OF TOKTBL *****
                     652 +1  $EJECT
```

```
LOC  OBJ             LINE     SOURCE

                      653
                      654
                      655      ;***** KEY WORD TABLE *****
                      656
E0D8 40412020         657      KEYTAB: DB       '@A  '
E0DC 40445054         658              DB       '@DPT'
E0E0 40523020         659              DB       '@R0 '
E0E4 40523120         660              DB       '@R1 '
E0E8 41202020         661              DB       'A   '               ;5
E0EC 41422020         662              DB       'AB  '
E0F0 41425220         663              DB       'ABR '
E0F4 4143414C         664              DB       'ACAL'
E0F8 41434320         665              DB       'ACC '
E0FC 41444420         666              DB       'ADD '               ;10
E100 41444443         667              DB       'ADDC'
E104 414A4D50         668              DB       'AJMP'
E108 414E4C20         669              DB       'ANL '
E10C 41534D20         670              DB       'ASM '
E110 42202020         671              DB       'B   '               ;15
E114 42415544         672              DB       'BAUD'
E118 42522020         673              DB       'BR  '
E11C 43202020         674              DB       'C   '
E120 43415553         675              DB       'CAUS'
E124 43425954         676              DB       'CBYT'               ;20
E128 434A4E45         677              DB       'CJNE'
E12C 434C5220         678              DB       'CLR '
E130 43504C20         679              DB       'CPL '
E134 44202020         680              DB       'D   '
E138 44412020         681              DB       'DA  '               ;25
E13C 4441534D         682              DB       'DASM'
E140 44415441         683              DB       'DATA'
E144 44425954         684              DB       'DBYT'
E148 44454320         685              DB       'DEC '
E14C 44495620         686              DB       'DIV '               ;30
E150 444A4E5A         687              DB       'DJNZ'
E154 444F574E         688              DB       'DOWN'
E158 44505452         689              DB       'DPTR'
E15C 46202020         690              DB       'F   '
E160 464F5245         691              DB       'FORE'               ;35
E164 46524F4D         692              DB       'FROM'
E168 474F2020         693              DB       'GO  '
E16C 494E4320         694              DB       'INC '
E170 4A422020         695              DB       'JB  '
E174 4A424320         696              DB       'JBC '               ;40
E178 4A432020         697              DB       'JC  '
E17C 4A4D5020         698              DB       'JMP '
E180 4A4E4220         699              DB       'JNB '
E184 4A4E4320         700              DB       'JNC '
E188 4A4E5A20         701              DB       'JNZ '               ;45
E18C 4A5A2020         702              DB       'JZ  '
E190 4C43414C         703              DB       'LCAL'
E194 4C495354         704              DB       'LIST'
E198 4C4A4D50         705              DB       'LJMP'
E19C 4C4F4144         706              DB       'LOAD'               ;50
E1A0 4D4F4445         707              DB       'MODE'
```

```
LOC  OBJ            LINE     SOURCE

E1A4 4D4F5620        708               DB      'MOV '
E1A8 4D4F5643        709               DB      'MOVC'
E1AC 4D4F5658        710               DB      'MOVX'
E1B0 4D554C20        711               DB      'MUL '              ;55
E1B4 4E4F5020        712               DB      'NOP '
E1B8 4F4E2020        713               DB      'ON  '
E1BC 4F522020        714               DB      'OR  '
E1C0 4F524720        715               DB      'ORG '
E1C4 4F524C20        716               DB      'ORL '              ;60
E1C8 50432020        717               DB      'PC  '
E1CC 504F5020        718               DB      'POP '
E1D0 50524F47        719               DB      'PROG'
E1D4 50535720        720               DB      'PSW '
E1D8 50555348        721               DB      'PUSH'              ;65
E1DC 52302020        722               DB      'R0  '
E1E0 52312020        723               DB      'R1  '
E1E4 52322020        724               DB      'R2  '
E1E8 52332020        725               DB      'R3  '
E1EC 52342020        726               DB      'R4  '              ;70
E1F0 52352020        727               DB      'R5  '
E1F4 52362020        728               DB      'R6  '
E1F8 52372020        729               DB      'R7  '
E1FC 52424954        730               DB      'RBIT'
E200 52425320        731               DB      'RBS '              ;75
E204 52425954        732               DB      'RBYT'
E208 52455345        733               DB      'RESE'
E20C 52455420        734               DB      'RET '
E210 52455449        735               DB      'RETI'
E214 524C2020        736               DB      'RL  '              ;80
E218 524C4320        737               DB      'RLC '
E21C 52522020        738               DB      'RR  '
E220 52524320        739               DB      'RRC '
E224 53415645        740               DB      'SAVE'
E228 53455442        741               DB      'SETB'              ;85
E22C 534A4D50        742               DB      'SJMP'
E230 53502020        743               DB      'SP  '
E234 53544550        744               DB      'STEP'
E238 53554242        745               DB      'SUBB'
E23C 53574150        746               DB      'SWAP'              ;90
E240 54202020        747               DB      'T   '
E244 54494C4C        748               DB      'TILL'
E248 544D3020        749               DB      'TM0 '
E24C 544D3120        750               DB      'TM1 '
E250 544F2020        751               DB      'TO  '              ;95
E254 544F5020        752               DB      'TOP '
E258 5452414E        753               DB      'TRAN'
E25C 55504C4F        754               DB      'UPLO'
E260 56455249        755               DB      'VERI'
E264 58425954        756               DB      'XBYT'              ;100
E268 58434820        757               DB      'XCH '
E26C 58434844        758               DB      'XCHD'
E270 58524C20        759               DB      'XRL '
                     760      ;***** END OF KEYTAB *****
                     761 +1   $EJECT
```

```
LOC  OBJ            SOURCE

                    ;*****************************************************************
                    ;
                    ;    NAME: POWER_ON
                    ;
                    ;    ABSTRACT: This routine initializes the breakpoint RAM, I/O
                    ;       channels, output buffer flag, TOP value, break status, user
                    ;       DPTR, B register and user PC.  It sets the baud rate to 2400
                    ;       and the GO condition to forever.  At the end, it jumps to
                    ;       BREAK which sets up the user area and jumps to SIGN_ON
                    ;       since the step flag has been cleared.
                    ;
                    ;    INPUTS: None
                    ;
                    ;    OUTPUTS: LSTFLG, GR, UPI_DATA_IMAGE, BAUDKEY, BAUD_HIGH, BAUD_LOW,
                    ;       ERRNUM, TOP_STORE, MON_FLAG, BREAK_STATUS, CAUSE_IMAGE, ASM_PC_LOW,
                    ;       ASM_PC_HIGH, DPTR, B, 2 stack locations, CHRCNT, LNLGTH, CHARIN,
                    ;       MAXNUM_FLG
                    ;
                    ;    VARIABLES MODIFIED: SP, LSTFLG, DPTR, A, PARAM1, DPL, ERRNUM,
                    ;       ASM_PC_HIGH, ASM_PC_LOW, CAUSE_IMAGE, DPH, B
                    ;
                    ;    ERROR EXITS: None
                    ;
                    ;    SUBROUTINES ACCESSED DIRECTLY: CHECK_EPROMS, INIT_IO, UPI_CMD,
                    ;       UPI_IN, UPI_OUT, SET_BAUD, BREAK
                    ;
                    ;*****************************************************************
                    POWER_ON:
E274 12ECE1                 CALL    CLRBRK                          ;Clear breakpoint RAM and
                                                                    ;remove Monitor from over-
                                                                    ;laying user Config. Memory
E277 758107                 MOV     SP,#07H
E27A C201                   CLR     LSTFLG
E27C 12E3BA                 CALL    CHECK_EPROMS                    ;Verify integrity of Monitor code.
E27F 12E386                 CALL    INIT_IO
E282 90A000                 MOV     DPTR,#UPI_DATA
E285 E0                     MOVX    A,@DPTR                         ;Initialize the IO channel and
E286 90B0F6                 MOV     DPTR,#(RAMOFF+GR)               ;copy break enable image
E289 7409                   MOV     A,#NO_BREAK                     ;into hardware
E28B F0                     MOVX    @DPTR,A
                                                                    ;Sets GO FOREVER as the power up
                                                                    ;Default condition
                                                                    ;Clear the users output buffer flag.
E28C 90B0F1                 MOV     DPTR,#(RAMOFF+UPI_DATA_IMAGE)
E28F E4                     CLR     A
E290 F0                     MOVX    @DPTR,A
E291 7A83                   MOV     PARAM1,#TOP_PORT                ;Initialize TOP port.
E293 12E625                 CALL    UPI_CMD
E296 12E64C                 CALL    UPI_IN                          ;Ignore current port value.
E299 7A00                   MOV     PARAM1,#00H                     ;Reselect the console.
E29B 12E638                 CALL    UPI_OUT
E29E 7582FC                 MOV     DPL,#BAUDKEY                    ;Set up the initial baud rate
E2A1 7404                   MOV     A,#04H
E2A3 F0                     MOVX    @DPTR,A
E2A4 12F229                 CALL    SET_BAUD                        ;for 2400.
```

```
LOC  OBJ            LINE     SOURCE

E2A7 90B0F7          817              MOV     DPTR,#(RAMOFF+BAUD_HIGH)
E2AA 7424            818              MOV     A,#24H
E2AC F0              819              MOVX    @DPTR,A
E2AD E4              820              CLR     A
E2AE F543            821              MOV     ERRNUM,A                  ;Firmware checksum error
E2B0 A3              822              INC     DPTR                      ;Point to BAUD_LOW
E2B1 F0              823              MOVX    @DPTR,A
E2B2 A3              824              INC     DPTR                      ;Point to TOP_STORE and zero.
E2B3 F0              825              MOVX    @DPTR,A
E2B4 A3              826              INC     DPTR                      ;Point to MON_FLAGS and zero
E2B5 F0              827              MOVX    @DPTR,A
E2B6 A3              828              INC     DPTR                      ;Point to BREAK_STATUS
E2B7 F0              829              MOVX    @DPTR,A                   ;Set it to the power on flag
E2B8 F54B            830              MOV     ASM_PC_HIGH,A             ;Zero out the asm PC
E2BA F54C            831              MOV     ASM_PC_LOW,A
E2BC F560            832              MOV     CAUSE_IMAGE,A
E2BE F582            833              MOV     DPL,A
E2C0 F583            834              MOV     DPH,A                     ;Clear DPTR and B so that
E2C2 F5D0            835              MOV     PSW,A                     ;break will report them correctly
E2C4 F5F0            836              MOV     B,A
E2C6 C0E0            837              PUSH    ACC                       ;Simulate the user PC in the stack
E2C8 C0E0            838              PUSH    ACC
E2CA 0103            839              JMP     BREAK
                     840 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                     841     ;*****************************************************************
                     842     ;
                     843     ;    NAME: SIGN_ON
                     844     ;
                     845     ;    ABSTRACT: Puts sign on message on the display and waits for
                     846     ;       a character to be input.
                     847     ;
                     848     ;    INPUTS: None
                     849     ;
                     850     ;    OUTPUTS: None
                     851     ;
                     852     ;    VARIABLES MODIFIED: PARAM1, PARAM2
                     853     ;
                     854     ;    ERROR EXITS: None
                     855     ;
                     856     ;    SUBROUTINES ACCESSED DIRECTLY: IPRINT_STRING, IWAIT_FOR_USER
                     857     ;
                     858     ;*****************************************************************
                     859     SIGN_ON:
E2CC 7AE3            860             MOV     PARAM1,#HIGH(SIGN_ON_MSG)
E2CE 7B6B            861             MOV     PARAM2,#LOW(SIGN_ON_MSG)
E2D0 12E9FF          862             CALL    IPRINT_STRING
E2D3 12E3B0          863             CALL    IWAIT_FOR_USER
                     864 +1  $EJECT
```

```
LOC  OBJ            SOURCE

                    ;*****************************************************************
                    ;
                    ;    NAME: START
                    ;
                    ;    ABSTRACT: This routine initializes the stack and gets tokens
                    ;       until an EOL is encountered.  It then decodes the first token and
                    ;       branches to appropriate command routine.
                    ;
                    ;    INPUTS: None
                    ;
                    ;    OUTPUTS: LINE_START, SP, TOKSTR
                    ;
                    ;    VARIABLES MODIFIED: PARAM1, PARAM2, DPTR, A, SP, B,
                    ;
                    ;    ERROR EXITS: 02H (INVALID COMMAND)
                    ;
                    ;    SUBROUTINE  ACCESSED DIRECTLY: IGETOKE, INIT_IO, IERROR
                    ;
                    ;*****************************************************************
E2D6 758107         START:  MOV     SP,#STACK
E2D9 755200                 MOV     LINE_START,#00H              ;Default beginning of line
E2DC 12E386                 CALL    INIT_IO                      ;Reset UPI
E2DF 12E8BC                 CALL    IGETOKE
E2E2 B40702                 CJNE    A,#EOL_TOKE,DECODE_CALL      ;If EOL, branch to cmd routine
E2E5 80EF                   JMP     START
                    DECODE_CALL:
E2E7 90E30E                 MOV     DPTR,#CMDTAB
E2EA 7A1F                   MOV     PARAM1,#((SIGN_ON_MSG-CMDTAB)/3);Length of command table
E2EC 12E2F1                 CALL    DECODE
E2EF 80E5                   JMP     START
E2F1 E4             DECODE: CLR     A
E2F2 93                     MOVC    A,@A+DPTR
E2F3 B5480D                 CJNE    A,TOKSTR,NEXT_ENTRY          ;Check next entry if no match
E2F6 E4                     CLR     A
E2F7 A3                     INC     DPTR
E2F8 93                     MOVC    A,@A+DPTR                    ;Get high byte of cmd addr
E2F9 F5F0                   MOV     B,A
E2FB E4                     CLR     A
E2FC A3                     INC     DPTR
E2FD 93                     MOVC    A,@A+DPTR                    ;Get low byte of cmd addr
E2FE C0E0                   PUSH    ACC
E300 C0F0                   PUSH    B
E302 22                     RET                                  ;'Return' to cmd addr
                    NEXT_ENTRY:
E303 A3                     INC     DPTR
E304 A3                     INC     DPTR
E305 A3                     INC     DPTR                         ;Skip over 3 byte entries
E306 DAE9                   DJNZ    PARAM1,DECODE                ;Check for end of table
E308 754302                 MOV     ERRNUM,#02H                  ;Invalid command
E30B 02E3E4                 JMP     IERROR
                    CMDTAB:
E30E 88                     DB      ABR_TOKE
E30F EBC7                   DW      BR_CMD
E311 98                     DB      ACC_TOKE
E312 ED2A                   DW      ACC_CMD
```

```
LOC  OBJ            LINE     SOURCE

E314 B0              920             DB      ASM_TOKE
E315 F581            921             DW      ASMBASE                           ;Assemble command.
E317 9B              922             DB      B_TOKE
E318 ED3C            923             DW      B_CMD
E31A D0              924             DB      BAUD_TOKE
E31B F1FD            925             DW      BAUD_CMD
E31D 89              926             DB      BR_TOKE
E31E EBC7            927             DW      BR_CMD
E320 D2              928             DB      CAUSE_TOKE
E321 F2B8            929             DW      CAUSE_CMD
E323 80              930             DB      CBYTE_TOKE
E324 EA5B            931             DW      MEMORY_CMD
E326 B8              932             DB      DASM_TOKE
E327 F584            933             DW      (ASMBASE + 3)                     ;Disassemble command.
E329 82              934             DB      DBYTE_TOKE
E32A EA5B            935             DW      MEMORY_CMD
E32C E0              936             DB      DOWNLOAD_TOKE
E32D F4F9            937             DW      DOWNLOAD_CMD
E32F A1              938             DB      DPTR_TOKE
E330 ED7D            939             DW      DPTR_CMD
E332 C2              940             DB      GO_TOKE
E333 F10F            941             DW      GO_CMD
E335 D7              942             DB      LIST_TOKE
E336 F1CD            943             DW      LIST_CMD
E338 E2              944             DB      LOAD_TOKE
E339 F44D            945             DW      LOAD_CMD
E33B B9              946             DB      MODE_TOKE
E33C F571            947             DW      MODE_CMD
E33E A0              948             DB      PC_TOKE
E33F ED5F            949             DW      PC_CMD
E341 D5              950             DB      PROGRAM_TOKE
E342 F56C            951             DW      PROGRAM_CMD
E344 99              952             DB      PSW_TOKE
E345 ED30            953             DW      PSW_CMD
E347 84              954             DB      RBIT_TOKE
E348 EA5B            955             DW      MEMORY_CMD
E34A 81              956             DB      RBYTE_TOKE
E34B EA5B            957             DW      MEMORY_CMD
E34D E3              958             DB      SAVE_TOKE
E34E F4B7            959             DW      SAVE_CMD
E350 9A              960             DB      SP_TOKE
E351 ED36            961             DW      SP_CMD
E353 C1              962             DB      STEP_TOKE
E354 EFDC            963             DW      STEP_CMD
E356 A2              964             DB      TM0_TOKE
E357 ED86            965             DW      TM0_CMD
E359 A3              966             DB      TM1_TOKE
E35A ED8F            967             DW      TM1_CMD
E35C D6              968             DB      TOP_TOKE
E35D F278            969             DW      TOP_CMD
E35F BA              970             DB      TRANSFER_TOKE
E360 F567            971             DW      TRANSFER_CMD
E362 E1              972             DB      UPLOAD_TOKE
E363 F50F            973             DW      UPLOAD_CMD
E365 BB              974             DB      VERIFY_TOKE
```

```
LOC  OBJ            LINE     SOURCE

E366 F562            975             DW      VERIFY_CMD
E368 86              976             DB      XBYTE_TOKE
E369 EA5B            977             DW      MEMORY_CMD
                     978     ;*******************************************************************
                     979     ;
                     980     SIGN_ON_MSG:
E36B 1A              981             DB      26,CR,LF,('SDK-51 MONITOR VER. 1.03')
E36C 0D
E36D 0A
E36E 53444B2D
E372 3531204D
E376 4F4E4954
E37A 4F522056
E37E 45522E20
E382 312E3033
                     982 +1  $EJECT
```

```
LOC  OBJ            SOURCE

                    ;***************************************************************
                    ;
                    ;    NAME: INIT_IO
                    ;
                    ;    ABSTRACT: This routine initialized the UPI hardware ports
                    ;       and resets the line buffer.
                    ;
                    ;    INPUTS: None
                    ;
                    ;    OUTPUTS: CHRCNT, LNLGTH, CHARIN, MAXNUM_FLAG
                    ;
                    ;    VARIABLES MODIFIED: A, CHRCNT, CHARIN, PARAM1, PARAM2,
                    ;       LNLGTH, PSW
                    ;
                    ;    ERROR EXITS: None
                    ;
                    ;    SUBROUTINES ACCESSED DIRECTLY: UPI_CMD, ITIME
                    ;
                    ;***************************************************************
E386 C204           INIT_IO:CLR      MAXNUM_FLAG
E388 E4                     CLR      A
E389 F551                   MOV      CHRCNT,A
E38B F554                   MOV      LNLGTH,A
E38D 755020                 MOV      CHARIN,#' '
E390 7A04                   MOV      PARAM1,#RESET_CMD
E392 12E625                 CALL     UPI_CMD
E395 7A03                   MOV      PARAM1,#GR_PORT
E397 12E625                 CALL     UPI_CMD
E39A 7A08                   MOV      PARAM1,#CLR_BRK_LATCHES
E39C 12E638                 CALL     UPI_OUT
E39F 7A09                   MOV      PARAM1,#NO_BREAK
E3A1 12E638                 CALL     UPI_OUT
E3A4 7A00                   MOV      PARAM1,#SELECT_CON
E3A6 12E625                 CALL     UPI_CMD
E3A9 7A00                   MOV      PARAM1,#00H
E3AB 7B70                   MOV      PARAM2,#70H              ;Delay approx. one UPI display scan (11.2ms)
E3AD 02EA45                 JMP      ITIME                    ;so the display won't flicker on reset.
                 +1 $EJECT
```

```
LOC  OBJ            SOURCE

                    ;******************************************************************
                    ;
                    ;    NAME: (I)WAIT_FOR_USER
                    ;
                    ;    ABSTRACT: Clears keyboard closures, waits for next keyboard
                    ;       entry and then returns.  The entry causing the return is NOT
                    ;       read, therefore, the UPI will not overwrite it until it is
                    ;       read by some other procedure.
                    ;
                    ;    INPUTS: None
                    ;
                    ;    OUTPUTS: None
                    ;
                    ;    VARIABLES MODIFIED: DPTR, PARAM1, PARAM2
                    ;
                    ;    ERROR EXITS: None
                    ;
                    ;    SUBROUTINES ACCESSED DIRECTLY: ITIME, ICSTS
                    ;
                    ;******************************************************************
                    IWAIT_FOR_USER:
E3B0 90A000                 MOV     DPTR,#UPI_DATA
E3B3 E0                     MOVX    A,@DPTR                 ;Clear any keyboard closures
                    IWAIT_FOR_USER_1:
E3B4 12E602                 CALL    ICSTS
E3B7 50FB                   JNC     IWAIT_FOR_USER_1
E3B9 22                     RET
              +1    $EJECT
```

```
LOC  OBJ            SOURCE

                    ;*****************************************************************
                    ;
                    ;    NAME: CHECK_EPROMS
                    ;
                    ;    ABSTRACT: This routine calculates the checksum for both
                    ;       EPROMS.  If not ok, print an error message and lock up
                    ;       forever.
                    ;
                    ;    INPUTS: None
                    ;
                    ;    OUTPUTS: None
                    ;
                    ;    VARIABLES MODIFIED: DPTR, CHECK_SUM, PARAM1, PARAM2, A
                    ;
                    ;    ERROR EXITS: None
                    ;
                    ;    SUBROUTINES ACCESSED DIRECTLY: IPRINT_STRING, ILSTBYT, SPACCO
                    ;
                    ;*****************************************************************
                    CHECK_EPROMS:
E3BA 90E000                 MOV     DPTR,#BASE                      ;Load dptr with beginning address
E3BD 7E00                   MOV     CHECKSUM,#00H                   ;Clear scratch pad
                    CHECK_LOOP:
E3BF E4                     CLR     A
E3C0 93                     MOVC    A,@A+DPTR                       ;Get code byte
E3C1 2E                     ADD     A,CHECKSUM                      ;Accumulate a running total
E3C2 FE                     MOV     CHECKSUM,A                      ;Save it
E3C3 A3                     INC     DPTR                            ;Point to next byte
E3C4 E583                   MOV     A,DPH                           ;If address has not wrapped around,
E3C6 70F7                   JNZ     CHECK_LOOP                      ;continue adding
E3C8 EE                     MOV     A,CHECKSUM                      ;else, check tally
E3C9 6018                   JZ      CHECK_OUT_OK                    ;If everthing adds up, return
E3CB 7AE4                   MOV     PARAM1,#HIGH(ERROR_MSG)
E3CD 7B26                   MOV     PARAM2,#LOW(ERROR_MSG)
E3CF 12E9FF                 CALL    IPRINT_STRING
E3D2 7A00                   MOV     PARAM1,#00H                     ;Firmware checksum error
E3D4 12E7F9                 CALL    ILSTBYT
E3D7 12E5E6                 CALL    SPACCO
E3DA 7AE4                   MOV     PARAM1,#HIGH(ERROR_TABLE)
E3DC 7B2D                   MOV     PARAM2,#LOW(ERROR_TABLE)
E3DE 12E9FF                 CALL    IPRINT_STRING
E3E1 80FE                   JMP     $                               ;and hang up here
                    CHECK_OUT_OK:
E3E3 22                     RET

                +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                       +1  $INCLUDE(:F1:UTILIT.INC)
                  =1       ;*****************************************************************
                  =1       ;
                  =1       ;    NAME: (I)ERROR
                  =1       ;
                  =1       ;    ABSTRACT: This routine handles all error messages for the SDK-51
                  =1       ;       except error 0.  These are not intended to be a standard format
                  =1       ;       for any other SDK product. After printing an error message, it
                  =1       ;       waits for any console entry and then starts fresh from START.
                  =1       ;       To find the routine which generates a particular error number,
                  =1       ;       check the cross reference listing (XREF) at the back of this
                  =1       ;       document for all uses of the variable name ERRNUM.
                  =1       ;
                  =1       ;    INPUTS: ERRNUM, LSTFLG
                  =1       ;
                  =1       ;    OUTPUTS: None
                  =1       ;
                  =1       ;    VARIABLES MODIFIED: PARAM1, PARAM2, C, A, TEMP1
                  =1       ;
                  =1       ;    ERROR EXITS: None
                  =1       ;
                  =1       ;    SUBROUTINES ACCESSED DIRECTLY: ITIME, INIT_IO, UPI_CMD,
                  =1       ;       IPRINT_STRING, ILSTBYT, SPACCO, IWAIT_FOR_USER
                  =1       ;
                  =1       ;
                  =1       ;*****************************************************************
E3E4 7A07         =1       IERROR: MOV     PARAM1,#07H
E3E6 7B00         =1               MOV     PARAM2,#00H
E3E8 12EA45       =1               CALL    ITIME                       ;Wait for the completion of any
E3EB 7186         =1               CALL    INIT_IO                     ;list activity before emptying usart
E3ED A201         =1               MOV     C,LSTFLG                    ;about 180ms
E3EF E4           =1               CLR     A
E3F0 92E6         =1               MOV     ACC.6,C
E3F2 FA           =1               MOV     PARAM1,A
E3F3 12E625       =1               CALL    UPI_CMD                     ;Select console with list status
E3F6 7AE4         =1               MOV     PARAM1,#HIGH(ERROR_MSG)
E3F8 7B26         =1               MOV     PARAM2,#LOW(ERROR_MSG)
E3FA 12E9FF       =1               CALL    IPRINT_STRING
E3FD AA43         =1               MOV     PARAM1,ERRNUM
E3FF 12E7F9       =1               CALL    ILSTBYT
E402 12E5E6       =1               CALL    SPACCO
E405 755600       =1               MOV     TEMP1,#00                   ;Table search counter
E408 90E42D       =1               MOV     DPTR,#ERROR_TABLE           ;Table entry
                  =1       ERROR_TEST:
E40B E543         =1               MOV     A,ERRNUM
E40D B5560B       =1               CJNE    A,TEMP1,ERROR_BEGIN         ;Is it this entry?
E410 AA83         =1               MOV     PARAM1,DPH
E412 AB82         =1               MOV     PARAM2,DPL
E414 12E9FF       =1               CALL    IPRINT_STRING
E417 71B0         =1               CALL    IWAIT_FOR_USER
E419 41D6         =1               JMP     START                       ;Yes, print message
                  =1       ERROR_BEGIN:
E41B E4           =1               CLR     A
E41C 93           =1               MOVC    A,@A+DPTR                   ;No, get num of letters to skip
                  =1       ERROR_LOOP:
```

```
LOC  OBJ            LINE     SOURCE

E41D A3        =1  1150              INC     DPTR
E41E D5E0FC    =1  1151              DJNZ    ACC,ERROR_LOOP
E421 A3        =1  1152              INC     DPTR                        ;Adjust addr of next table entry
E422 0556      =1  1153              INC     TEMP1                       ;Adjust table search counter
E424 80E5      =1  1154              JMP     ERROR_TEST
               =1  1155     ERROR_MSG:
E426 06        =1  1156              DB      6,CR,LF,('ERR=')
E427 0D
E428 0A
E429 4552523D
               =1  1157     ERROR_TABLE:
E42D 0A        =1  1158              DB      10,('PROM CKSUM')           ;Error #00
E42E 50524F4D
E432 20434B53
E436 554D
E438 0C        =1  1159              DB      12,('INVALID WORD')         ;       01
E439 494E5641
E43D 4C494420
E441 574F5244
E445 0F        =1  1160              DB      15,('INVALID COMMAND')      ;       02
E446 494E5641
E44A 4C494420
E44E 434F4D4D
E452 414E44
E455 0A        =1  1161              DB      10,('NUMBER REQ')           ;       03
E456 4E554D42
E45A 45522052
E45E 4551
E460 0A        =1  1162              DB      10,('RETURN REQ')           ;       04
E461 52455455
E465 524E2052
E469 4551
E46B 11        =1  1163              DB      17,('EQUAL OR RTRN REQ')    ;       05
E46C 45515541
E470 4C204F52
E474 20525452
E478 4E205245
E47C 51
E47D 09        =1  1164              DB      09,('COMMA REQ')            ;       06
E47E 434F4D4D
E482 41205245
E486 51
E487 0D        =1  1165              DB      13,('PARTITION ADR')        ;       07
E488 50415254
E48C 4954494F
E490 4E204144
E494 52
E495 0F        =1  1166              DB      15,('RESET OR ON REQ')      ;       08
E496 52455345
E49A 54204F52
E49E 204F4E20
E4A2 524551
E4A5 0F        =1  1167              DB      15,('DECIMAL NUM REQ')      ;       09
E4A6 44454349
E4AA 4D414C20
```

```
LOC  OBJ            LINE     SOURCE

E4AE 4E554D20
E4B2 524551
E4B5 10        =1  1168             DB      16,('ILLEGAL BAUD VAL')        ;       0A
E4B6 494C4C45
E4BA 47414C20
E4BE 42415544
E4C2 2056414C
E4C6 10        =1  1169             DB      16,('BRK ENABL SYNTAX')        ;       0B
E4C7 42524B20
E4CB 454E4142
E4CF 4C205359
E4D3 4E544158
E4D7 10        =1  1170             DB      16,('NUM OR RESET REQ')        ;       0C
E4D8 4E554D20
E4DC 4F522052
E4E0 45534554
E4E4 20524551
E4E8 0B        =1  1171             DB      11,('TOP ) 7FFFH')             ;       0D
E4E9 544F5020
E4ED 29203746
E4F1 464648
E4F4 0C        =1  1172             DB      12,('DISPLAY ONLY')            ;       0E
E4F5 44495350
E4F9 4C415920
E4FD 4F4E4C59
E501 10        =1  1173             DB      16,('UNDEFINED OPCODE')        ;       0F
E502 554E4445
E506 46494E45
E50A 44204F50
E50E 434F4445
E512 0F        =1  1174             DB      15,('ASSEMBLY SYNTAX')         ;       10
E513 41535345
E517 4D424C59
E51B 2053594E
E51F 544158
E522 10        =1  1175             DB      16,('ADR OUT OF RANGE')        ;       11
E523 41445220
E527 4F555420
E52B 4F462052
E52F 414E4745
E533 10        =1  1176             DB      16,('ADR OUT OF RANGE')        ;       12
E534 41445220
E538 4F555420
E53C 4F462052
E540 414E4745
E544 0F        =1  1177             DB      15,('ASM PC ) 0FFFFH')         ;       13
E545 41534D20
E549 50432029
E54D 20304646
E551 464648
E554 0D        =1  1178             DB      13,('FILE RD OR WR')           ;       14
E555 46494C45
E559 20524420
E55D 4F522057
E561 52
```

```
LOC  OBJ             LINE     SOURCE

E562 0C          =1  1179             DB      12,('MEMORY WRITE')            ;       15
E563 4D454D4F
E567 52592057
E56B 52495445
E56F 10          =1  1180             DB      16,('EX ACROSS ADR 03')        ;       16
E570 45582041
E574 43524F53
E578 53204144
E57C 52203033
E580 10          =1  1181             DB      16,('NO RAM AT ADR 03')        ;       17
E581 4E4F2052
E585 414D2041
E589 54204144
E58D 52203033
E591 0E          =1  1182             DB      14,('CBYTE TYPE REQ')          ;       18
E592 43425954
E596 45205459
E59A 50452052
E59E 4551
E5A0 0B          =1  1183             DB      11,('CHANGE ONLY')             ;       19
E5A1 4348414E
E5A5 4745204F
E5A9 4E4C59
E5AC 0E          =1  1184             DB      14,('CBY OR NUM REQ')          ;       1A
E5AD 43425920
E5B1 4F52204E
E5B5 554D2052
E5B9 4551
                 =1  1185 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1          ;*****************************************************************
                =1          ;
                =1          ;    NAME: (I)EOL_CHECK
                =1          ;
                =1          ;    ABSTRACT: This routine will check for a carraige return and error
                =1          ;       if one is not found.  It returns to calling routine if one is.
                =1          ;
                =1          ;    INPUTS: A (byte to be checked)
                =1          ;
                =1          ;    OUTPUTS: None
                =1          ;
                =1          ;    VARIABLES MODIFIED: ERRNUM
                =1          ;
                =1          ;    ERROR EXITS: 04H (CARRAIGE RETURN EXPECTED)
                =1          ;
                =1          ;    SUBROUTINES ACCESSED DIRECTLY: IERROR
                =1          ;
                =1          ;
                =1          ;*****************************************************************
                =1          IEOL_CHECK:
E5BB B40701     =1                  CJNE    A,#EOL_TOKE,EOL_ERROR
E5BE 22         =1                  RET
                =1          EOL_ERROR:
E5BF 754304     =1                  MOV     ERRNUM,#04H                     ;Carriage return expected
E5C2 61E4       =1                  JMP     IERROR
                =1       +1 $EJECT
```

```
LOC  OBJ            LINE     SOURCE
                =1          ;*****************************************************************
                =1          ;
                =1          ;    NAME: INC_PNT/ DEC_PNT/ SWAP_POINTERS
                =1          ;
                =1          ;    ABSTRACT: These are general purpose 16 bit arithmetic
                =1          ;       routines which will increment, decrement or swap pointers.
                =1          ;
                =1          ;    INPUTS: PNTLOW, PNTHGH, PCNTLO, PCNTHI
                =1          ;
                =1          ;    OUTPUTS: PNTLOW, PNTHGH, PNCTLO, PCNTHI
                =1          ;
                =1          ;    VARIABLES MODIFIED: A, PNTLOW, PNTHGH, PCNTLO, PCNTHI
                =1          ;
                =1          ;    ERROR EXITS: None
                =1          ;
                =1          ;    SUBROUTINES ACCESSED DIRECTLY: None
                =1          ;
                =1          ;
                =1          ;
                =1          ;*****************************************************************
E5C4 0545       =1          INC_PNT:INC     PNTLOW
E5C6 E545       =1                  MOV     A,PNTLOW
E5C8 7002       =1                  JNZ     INC_HIGH
E5CA 0544       =1                  INC     PNTHGH
                =1          INC_HIGH:
E5CC 22         =1                  RET
                =1          ;*****************************************************************
E5CD 1545       =1          DEC_PNT:DEC     PNTLOW
E5CF E545       =1                  MOV     A,PNTLOW
E5D1 F4         =1                  CPL     A
E5D2 7002       =1                  JNZ     DEC_HIGH
E5D4 1544       =1                  DEC     PNTHGH
                =1          DEC_HIGH:
 5D6 22         =1                  RET
                =1          ;*****************************************************************
                =1          SWAP_POINTERS:
E5D7 E545       =1                  MOV     A,PNTLOW
E5D9 856245     =1                  MOV     PNTLOW,PCNTLO
E5DC F562       =1                  MOV     PCNTLO,A
E5DE E544       =1                  MOV     A,PNTHGH
E5E0 856144     =1                  MOV     PNTHGH,PCNTHI
E5E3 F561       =1                  MOV     PCNTHI,A
E5E5 22         =1                  RET
                =1        +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

               =1   1256     ;*****************************************************************
               =1   1257     ;
               =1   1258     ;    NAME: SPACCO/ (I)CO
               =1   1259     ;
               =1   1260     ;    ABSTRACT: Outputs a space to the system console, falls through
               =1   1261     ;       to ICO then returns.
               =1   1262     ;
               =1   1263     ;    INPUTS: PARAM1 (ASCII character to be printed)
               =1   1264     ;
               =1   1265     ;    OUTPUTS: None
               =1   1266     ;
               =1   1267     ;    VARIABLES MODIFIED: PARAM1
               =1   1268     ;
               =1   1269     ;    ERROR EXITS: None
               =1   1270     ;
               =1   1271     ;    SUBROUTINES ACCESSED DIRECTLY: UPI_OUT
               =1   1272     ;
               =1   1273     ;
               =1   1274     ;*****************************************************************
E5E6 7A20      =1   1275     SPACCO: MOV     PARAM1,#' '
E5E8 02E638    =1   1276     ICO:    JMP     UPI_OUT
               =1   1277 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

               =1   1278     ;*****************************************************************
               =1   1279     ;
               =1   1280     ;    NAME: ICI
               =1   1281     ;
               =1   1282     ;    ABSTRACT: Inputs an ASCII character from the system console, clears
               =1   1283     ;       the parity bit and converts to upper case.  If there is no
               =1   1284     ;       user abort, it returns to caller.
               =1   1285     ;
               =1   1286     ;    INPUTS: None
               =1   1287     ;
               =1   1288     ;    OUTPUTS: A
               =1   1289     ;
               =1   1290     ;    VARIABLES MODIFIED: A
               =1   1291     ;
               =1   1292     ;    ERROR EXITS: None
               =1   1293     ;
               =1   1294     ;    SUBROUTINES ACCESSED DIRECTLY:  IUPI_IN
               =1   1295     ;
               =1   1296     ;*****************************************************************
               =1   1297
E5EB 12E64C    =1   1298     ICI:    CALL    UPI_IN
E5EE C2E7      =1   1299             CLR     ACC.7                  ;Clear parity bit
E5F0 B46100    =1   1300             CJNE    A,#'a',UPI_INA
               =1   1301     UPI_INA:
E5F3 4007      =1   1302             JC      UPI_INR
E5F5 B47B00    =1   1303             CJNE    A,#('z'+1),UPI_INB
               =1   1304     UPI_INB:
E5F8 5002      =1   1305             JNC     UPI_INR
E5FA C2E5      =1   1306             CLR     ACC.5                  ;Convert to upper case
               =1   1307     UPI_INR:
E5FC B41B02    =1   1308             CJNE    A,#ESC,UPI_INE         ;Abort if its an ESC key
E5FF 41D6      =1   1309             JMP     START
E601 22        =1   1310     UPI_INE:        RET                    ;And return to the caller.
               =1   1311 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

               =1  1312     ;*****************************************************************
               =1  1313     ;
               =1  1314     ;    NAME: ICSTS
               =1  1315     ;
               =1  1316     ;    ABSTRACT: Returns carry=1 if there is a character waiting from
               =1  1317     ;       the system console.  If no character is ready, carry will be
               =1  1318     ;       cleared. CAUTION: this is not available for use except to the
               =1  1319     ;       monitor itself.  See UCSTS for a general purpose version of
               =1  1320     ;       this routine.
               =1  1321     ;
               =1  1322     ;    INPUTS: None
               =1  1323     ;
               =1  1324     ;    OUTPUTS: Carry bit (C)
               =1  1325     ;
               =1  1326     ;    VARIABLES MODIFIED: DPTR, A, C, 2 locations of the stack
               =1  1327     ;
               =1  1328     ;    ERROR EXITS: None
               =1  1329     ;
               =1  1330     ;    SUBROUTINES ACCESSED DIRECTLY: None
               =1  1331     ;
               =1  1332     ;
               =1  1333     ;*****************************************************************
E602 C082      =1  1334     ICSTS:  PUSH    DPL
E604 C083      =1  1335             PUSH    DPH
E606 90A001    =1  1336             MOV     DPTR,#UPI_CONTROL
E609 E0        =1  1337     CSTS_1: MOVX    A,@DPTR
E60A 20E2FC    =1  1338             JB      ACC.2,CSTS_1           ;Wait for status to be valid
E60D 13        =1  1339             RRC     A                      ;Rotate UPI OBF into CARRY
E60E D083      =1  1340             POP     DPH
E610 D082      =1  1341             POP     DPL
E612 22        =1  1342             RET
               =1  1343 +1  $EJECT
```

```
LOC  OBJ                    SOURCE
                   =1       ;*******************************************************************
                   =1       ;
                   =1       ;    NAME: (U)CSTS
                   =1       ;
                   =1       ;    ABSTRACT: This routine gets the console status bit from bit 7
                   =1       ;       of the accumulator into carry.  Carry = 1 if a character
                   =1       ;       is present.
                   =1       ;
                   =1       ;       Users writing application programs should use
                   =1       ;       this routine instead of ICSTS.  This reflects the buffered
                   =1       ;       version of the console port.
                   =1       ;
                   =1       ;    INPUTS: None
                   =1       ;
                   =1       ;    OUTPUTS: Carry bit (C)
                   =1       ;
                   =1       ;    VARIABLES MODIFIED: DPTR, A
                   =1       ;
                   =1       ;    ERROR EXITS: None
                   =1       ;
                   =1       ;    SUBROUTINES ACCESSED DIRECTLY: None
                   =1       ;
                   =1       ;
                   =1       ;*******************************************************************
                   =1
E613 90B0F1        =1       UCSTS:  MOV     DPTR,#(RAMOFF+UPI_DATA_IMAGE)
E616 E0            =1               MOVX    A,@DPTR
E617 33            =1               RLC     A
E618 22            =1               RET
                   =1    +1 $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                    =1       ;*****************************************************************
                    =1       ;
                    =1       ;    NAME: (U)CI
                    =1       ;
                    =1       ;    ABSTRACT: This routine waits for the console status bit to
                    =1       ;       indicate that a character is ready (C=1), inputs it from
                    =1       ;       the console, clears the status bit and returns.
                    =1       ;
                    =1       ;       Users writing application programs should use
                    =1       ;       this routine instead of ICSTS.  This reflects the buffered
                    =1       ;       version of the console port.
                    =1       ;
                    =1       ;    INPUTS: None
                    =1       ;
                    =1       ;    OUTPUTS: UPI_DATA_IMAGE
                    =1       ;
                    =1       ;    VARIABLES MODIFIED: DPTR, A
                    =1       ;
                    =1       ;    ERROR EXITS: None
                    =1       ;
                    =1       ;    SUBROUTINES ACCESSED DIRECTLY: UCSTS
                    =1       ;
                    =1       ;
                    =1       ;*****************************************************************
E619 D113           =1       UCI:    CALL    UCSTS
E61B 50FC           =1               JNC     UCI
E61D 90B0F1         =1               MOV     DPTR,#(RAMOFF+UPI_DATA_IMAGE)
E620 E0             =1               MOVX    A,@DPTR
E621 C2E7           =1               CLR     ACC.7
E623 F0             =1               MOVX    @DPTR,A
E624 22             =1               RET
                    =1    +1 $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                 =1       ;*****************************************************************
                 =1       ;
                 =1       ;    NAME: (I)UPI_CMD
                 =1       ;
                 =1       ;    ABSTRACT: Waits till the UPI is ready and then outputs a
                 =1       ;       command to it.
                 =1       ;
                 =1       ;    INPUTS: PARAM1=byte to be sent to UPI command port
                 =1       ;
                 =1       ;    OUTPUTS: None
                 =1       ;
                 =1       ;    VARIABLES MODIFIED: A, 2 locations in the stack
                 =1       ;
                 =1       ;    ERROR EXITS: None
                 =1       ;
                 =1       ;    SUBROUTINES ACCESSED DIRECTLY: None
                 =1       ;
                 =1       ;
                 =1       ;*****************************************************************
                 =1       UPI_CMD:
E625 C082        =1               PUSH    DPL                       ;Save DPTR in the stack.
E627 C083        =1               PUSH    DPH
E629 90A001      =1               MOV     DPTR,#UPI_CONTROL         ;Point to UPI control channel
                 =1       UPI_C_1:
E62C E0          =1               MOVX    A,@DPTR                   ;And wait for valid status.
E62D 5416        =1               ANL     A,#16H
E62F 70FB        =1               JNZ     UPI_C_1
E631 EA          =1               MOV     A,PARAM1                  ;Then send out the command.
E632 F0          =1               MOVX    @DPTR,A
E633 D083        =1               POP     DPH                       ;Restore DPTR
E635 D082        =1               POP     DPL
E637 22          =1               RET                               ;Return to caller.
                 =1       +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1  1439     ;*****************************************************************
                =1  1440     ;
                =1  1441     ;    NAME: UPI_OUT
                =1  1442     ;
                =1  1443     ;    ABSTRACT: Waits until the UPI is ready and then outputs data to it.
                =1  1444     ;
                =1  1445     ;    INPUTS: PARAM1 = data to be sent to UPI
                =1  1446     ;
                =1  1447     ;    OUTPUTS: None
                =1  1448     ;
                =1  1449     ;    VARIABLES MODIFIED: A, 2 locations on the stack
                =1  1450     ;
                =1  1451     ;    ERROR EXITS: None
                =1  1452     ;
                =1  1453     ;    SUBROUTINES ACCESSED DIRECTLY: None
                =1  1454     ;
                =1  1455     ;
                =1  1456     ;*****************************************************************
E638 C082       =1  1457     UPI_OUT:PUSH    DPL                     ;Save DPTR in the stack.
E63A C083       =1  1458             PUSH    DPH
E63C 90A001     =1  1459             MOV     DPTR,#UPI_CONTROL       ;Point to UPI control channel
E63F E0         =1  1460     UPI_O_1:MOVX    A,@DPTR                 ;and wait for valid status.
E640 5416       =1  1461             ANL     A,#16H
E642 70FB       =1  1462             JNZ     UPI_O_1
E644 A3         =1  1463             INC     DPTR                    ;Point to data port
E645 EA         =1  1464             MOV     A,PARAM1
E646 F0         =1  1465             MOVX    @DPTR,A
E647 D083       =1  1466             POP     DPH                     ;Restore DPTR
E649 D082       =1  1467             POP     DPL
E64B 22         =1  1468             RET                             ;Return to caller.
                =1  1469 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1  1470     ;*****************************************************************
                =1  1471     ;
                =1  1472     ;     NAME: UPI_IN
                =1  1473     ;
                =1  1474     ;     ABSTRACT: Waits for a character from the UPI and returns it to
                =1  1475     ;        the caller in the accumulator.
                =1  1476     ;
                =1  1477     ;     INPUTS: None
                =1  1478     ;
                =1  1479     ;     OUTPUTS: A
                =1  1480     ;
                =1  1481     ;     VARIABLES MODIFIED: A, 2 locations of the stack
                =1  1482     ;
                =1  1483     ;     ERROR EXITS: None
                =1  1484     ;
                =1  1485     ;     SUBROUTINES ACCESSED DIRECTLY: ICSTS
                =1  1486     ;
                =1  1487     ;
                =1  1488     ;*****************************************************************
E64C D102       =1  1489     UPI_IN: CALL    ICSTS
E64E 50FC       =1  1490             JNC     UPI_IN                   ;Wait for character
E650 C082       =1  1491             PUSH    DPL
E652 C083       =1  1492             PUSH    DPH
E654 90A000     =1  1493             MOV     DPTR,#UPI_DATA           ;Point to UPI data port
E657 E0         =1  1494             MOVX    A,@DPTR                  ;Get byte
E658 D083       =1  1495             POP     DPH                      ;Restore DPTR
E65A D082       =1  1496             POP     DPL
E65C 22         =1  1497             RET                              ;and return to the caller
                =1  1498 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

               =1  1499      ;*******************************************************************
               =1  1500      ;
               =1  1501      ;     NAME: (I)CONTINUATION_LINE
               =1  1502      ;
               =1  1503      ;     ABSTRACT: This routine looks for LIST=ON.  If there is no user
               =1  1504      ;        abort, it gets a character and returns.  If LIST=RESET,
               =1  1505      ;        it outputs a blinking comma to the display, discards the
               =1  1506      ;        character, waits for the user to hit any key and returns.
               =1  1507      ;
               =1  1508      ;     INPUTS: LSTFLG
               =1  1509      ;
               =1  1510      ;     OUTPUTS: None
               =1  1511      ;
               =1  1512      ;     VARIABLES MODIFIED: PARAM1
               =1  1513      ;
               =1  1514      ;     ERROR EXITS: None
               =1  1515      ;
               =1  1516      ;     SUBROUTINES ACCESSED DIRECTLY: ICO, ICI, ICSTS
               =1  1517      ;
               =1  1518      ;
               =1  1519      ;*******************************************************************
               =1  1520      ICONTINUATION_LINE:
E65D 200106    =1  1521              JB      LSTFLG,DONT_WAIT
E660 7AAC      =1  1522              MOV     PARAM1,#(',''+BLINK)
E662 B1E8      =1  1523              CALL    ICO
               =1  1524      CHECK_ESC:
E664 8085      =1  1525              JMP     ICI
               =1  1526      DONT_WAIT:
E666 D102      =1  1527              CALL    ICSTS
E668 40FA      =1  1528              JC      CHECK_ESC
E66A 22        =1  1529              RET
               =1  1530 +1   $EJECT
```

```
LOC  OBJ           LINE     SOURCE

                   =1      ;*****************************************************************
                   =1      ;
                   =1      ;    NAME: (I)FETCH/(I)STORE
                   =1      ;
                   =1      ;    ABSTRACT:
                   =1      ;       This routine reads or writes one byte of data.  SELECT indicates
                   =1      ;       the type of memory operation to be performed.  The following
                   =1      ;       table lists the values of SELECT:
                   =1      ;               0H) CBYTE - Program memory
                   =1      ;               1H) RBYTE - Register memory
                   =1      ;               2H) DBYTE - Internal data memory
                   =1      ;               3H) Not used
                   =1      ;               4H) RBIT - Bit memory
                   =1      ;               5H) Not used
                   =1      ;               6H) XBYTE - External data memory
                   =1      ;       PNTLOW holds lower 8 bits of address
                   =1      ;       PNTHGH Holds upper 8 bits of address and must be
                   =1      ;               zeroed out if not used
                   =1      ;       PARAM1 holds value to be stored, is only used by STORE
                   =1      ;       A      holds the result of the fetch
                   =1      ;       CBYTE does a read after write to verify byte value written,
                   =1      ;               (i.e. detects writes to ROM).
                   =1      ;
                   =1      ;    INPUTS: SELECT, PARAM1, PNTLO
                   =1      ;
                   =1      ;    OUTPUTS: A, contents of memory being addressed
                   =1      ;
                   =1      ;    VARIABLES MODIFIED: A, PSW, DPTR, ERRNUM, TEMP1, B, C
                   =1      ;
                   =1      ;    ERROR EXITS: 12H (ADDRESS OUT OF RANGE )
                   =1      ;                 15H (READ AFTER WRITE ERROR)
                   =1      ;
                   =1      ;    SUBROUTINES ACCESSED DIRECTLY: IERROR
                   =1      ;
                   =1      ;
                   =1      ;*****************************************************************
E66B E546          =1      IFETCH: MOV     A,SELECT                        ;Data value passed from calling routine
E66D C2D5          =1              CLR     F0                              ;Zero = read memory
E66F 02E676        =1              JMP     MEMORY
E672 E546          =1      ISTORE: MOV     A,SELECT
E674 D2D5          =1              SETB    F0                              ;One = write memory
E676 854483        =1      MEMORY: MOV     DPH, PNTHGH
E679 854582        =1              MOV     DPL, PNTLOW                     ;Put addr in data pointer
E67C B40012        =1              CJNE    A,#(CBYTE_TOKE AND 07H),XBYTE
E67F 30D50A        =1              JNB     F0,C_READ                       ;Jump if not CBYTE
E682 EA            =1              MOV     A,PARAM1
E683 F0            =1              MOVX    @DPTR,A                         ;Program memory write
E684 E4            =1              CLR     A
E685 93            =1              MOVC    A,@A+DPTR                       ;Program memory read after write
E686 6A            =1              XRL     A,PARAM1
E687 7041          =1              JNZ     FETERR                          ;Verify error if read doesn't match write
E689 02E69D        =1              JMP     FETEND
E68C E4            =1      C_READ: CLR     A
E68D 93            =1              MOVC    A,@A+DPTR                       ;Program memory read
E68E 02E69D        =1              JMP     FETEND
```

```
LOC  OBJ          LINE      SOURCE

E691 B4060C   =1   1586     XBYTE:  CJNE    A,#(XBYTE_TOKE AND 07H),RBYTE    ;Check if external RAM was selected
E694 20D504   =1   1587             JB      F0,XWRITE                        ;Jump to STORE if flag is set
E697 E0       =1   1588     XREAD:  MOVX    A,@DPTR                          ;Load EXT RAM into ACC
E698 02E69D   =1   1589             JMP     FETEND
E69B EA       =1   1590     XWRITE: MOV     A,PARAM1                         ;Load ACC with data to be output
E69C F0       =1   1591     X_WRT:  MOVX    @DPTR,A                          ;Output ACC to EXT RAM
E69D C2D5     =1   1592     FETEND: CLR     F0                               ;Clear flag
E69F 22       =1   1593             RET
E6A0 90B000   =1   1594     RBYTE:  MOV     DPTR,#RAMOFF                     ;Load DPTR with base addr of 8155 RAM
E6A3 754312   =1   1595             MOV     ERRNUM,#12H                      ;Address out of range
E6A6 E544     =1   1596             MOV     A,PNTHGH
E6A8 7023     =1   1597             JNZ     ERR                              ;Error if address is not 00XXH
E6AA E546     =1   1598             MOV     A,SELECT
E6AC B4010C   =1   1599             CJNE    A,#(RBYTE_TOKE AND 07H),DBYTE    ;Jump if not RBYTE
E6AF E545     =1   1600             MOV     A,PNTLOW
E6B1 30E719   =1   1601             JNB     ACC.7,ERR                        ;Error if address is between 0 and 7FH
E6B4 F582     =1   1602             MOV     DPL,A
E6B6 20D5E2   =1   1603             JB      F0,XWRITE                        ;Jump to STORE if flag is set
E6B9 80DC     =1   1604             JMP     XREAD                            ;Exit from FETCH
E6BB B40211   =1   1605     DBYTE:  CJNE    A,#(DBYTE_TOKE AND 07H),RBIT     ;Jump if RBIT is selected
E6BE E545     =1   1606             MOV     A,PNTLOW                         ;Load ACC with low pointer
E6C0 20E70A   =1   1607             JB      ACC.7,ERR                        ;Error if addr if between 80H and FFH
E6C3 F582     =1   1608             MOV     DPL,A                            ;Load DPL with new low point value
E6C5 20D5D3   =1   1609             JB      F0,XWRITE                        ;Jump to STORE if flag is set
E6C8 80CD     =1   1610             JMP     XREAD                            ;Exit from FETCH
E6CA 754315   =1   1611     FETERR: MOV     ERRNUM,#15H                      ;Read after write did not match.
E6CD 61E4     =1   1612     ERR:    JMP     IERROR                           ;Exit from FETCH/STORE
E6CF B404CB   =1   1613     RBIT:   CJNE    A,#(RBIT_TOKE AND 07H),FETEND    ;Check if selector is for direct bit
E6D2 E545     =1   1614             MOV     A,PNTLOW                         ;Load ACC with pointer
E6D4 54F8     =1   1615             ANL     A,#0F8H                          ;Mask off lower 3 bits
E6D6 20E705   =1   1616             JB      ACC.7,SPEFUN                     ;Jump to register bit array if over 7FH
E6D9 13       =1   1617             RRC     A
E6DA 03       =1   1618             RR      A
E6DB 03       =1   1619             RR      A                                ;Rotate ACC to obtain correct addr
E6DC 2420     =1   1620             ADD     A,#20H                           ;Add offset of internal bit registers
E6DE 2582     =1   1621     SPEFUN: ADD     A,DPL                            ;Add offset to pointer
E6E0 F582     =1   1622             MOV     DPL,A                            ;Load DPL with new addr
E6E2 20D513   =1   1623             JB      F0,BITSTR                        ;Jump to STORE if flag is set
E6E5 E0       =1   1624             MOVX    A,@DPTR                          ;Move INT RAM simulator byte into ACC
E6E6 854556   =1   1625             MOV     TEMP1,PNTLOW                     ;Move pointer into TEMP1
E6E9 535607   =1   1626             ANL     TEMP1,#07H                       ;Mask lower 3 bits
E6EC 0556     =1   1627             INC     TEMP1                            ;For DJNZ
E6EE D55604   =1   1628     BITLOP: DJNZ    TEMP1,BITROT                     ;Loop until PARAM1=0
E6F1 5401     =1   1629             ANL     A,#1                             ;Mask lowest bit
E6F3 80A8     =1   1630             JMP     FETEND                           ;Exit from FETCH
E6F5 03       =1   1631     BITROT: RR      A                                ;Rotate until PARAM1=0
E6F6 80F6     =1   1632             JMP     BITLOP
E6F8 854556   =1   1633     BITSTR: MOV     TEMP1,PNTLOW                     ;Load TEMP1 with pointer
E6FB 535607   =1   1634             ANL     TEMP1,#07H                       ;Mask lower 3 bits
E6FE 0556     =1   1635             INC     TEMP1                            ;For DJNZ
E700 E0       =1   1636             MOVX    A,@DPTR                          ;Load ACC with data in RAM simulator
E701 13       =1   1637     RHTROT: RRC     A
E702 D556FC   =1   1638             DJNZ    TEMP1,RHTROT                     ;Rotate until pointer reaches zero
E705 8AF0     =1   1639             MOV     B,PARAM1                         ;Move data to be output into B reg
E707 A2F0     =1   1640             MOV     C,B.0                            ;Move into carry data to be output
```

```
LOC  OBJ            LINE      SOURCE

E709 854556      =1  1641                MOV     TEMP1,PNTLOW                      ;Load TEMP1 with pointer
E70C 535607      =1  1642                ANL     TEMP1,#07H                        ;Mask lower 3 bits for counter
E70F 0556        =1  1643                INC     TEMP1
E711 33          =1  1644      LFTROT:   RLC     A
E712 D556FC      =1  1645                DJNZ    TEMP1,LFTROT                      ;Rotate left until TEMP1 reaches zero
E715 8085        =1  1646                JMP     X_WRT
                 =1  1647 +1   $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1           ;*****************************************************************
                =1           ;
                =1           ;    NAME: (I)NEWLINE
                =1           ;
                =1           ;    ABSTRACT: Outputs a CR/LF to the console device.
                =1           ;
                =1           ;    INPUTS: None
                =1           ;
                =1           ;    OUTPUTS: None
                =1           ;
                =1           ;    VARIABLES MODIFIED: PARAM1
                =1           ;
                =1           ;    ERROR EXITS: None
                =1           ;
                =1           ;    SUBROUTINES ACCESSED DIRECTLY: ICO
                =1           ;
                =1           ;
                =1           ;*****************************************************************
                =1           INEWLINE:
E717 7A0D       =1                   MOV     PARAM1,#CR              ;Output a CR
E719 B1E8       =1                   CALL    ICO
E71B 7A0A       =1                   MOV     PARAM1,#LF              ;Output a LF
E71D B1E8       =1                   CALL    ICO
E71F 22         =1                   RET
                =1         +1 $EJECT
```

```
LOC  OBJ                 SOURCE

                   =1    ;*****************************************************************
                   =1    ;
                   =1    ;    NAME: AZTEST / NMTEST / HXTEST / ALFNUM
                   =1    ;
                   =1    ;    ABSTRACT: AZTEST will check to see if the input character is
                   =1    ;       an ASCII value between @ and Z.  Carry is set if it is.
                   =1    ;       NMTEST will check to see if the character was an ASCII number
                   =1    ;       between 0 and 9 and set carry if true.  HXTEST will look for the
                   =1    ;       ASCII representation of a hex value 0-9 and A-F and will set carry
                   =1    ;       if true.  ALFNUM will test for character to be alpha or numeric
                   =1    ;       and set carry if true.
                   =1    ;
                   =1    ;    INPUTS: PARAM1 (byte to be checked)
                   =1    ;
                   =1    ;    OUTPUTS: Carry bit (C)
                   =1    ;
                   =1    ;    VARIABLES MODIFIED: A, C
                   =1    ;
                   =1    ;    ERROR EXITS: None
                   =1    ;
                   =1    ;    SUBROUTINES ACCESSED DIRECTLY: None
                   =1    ;
                   =1    ;
                   =1    ;*****************************************************************
E720 EA            =1    AZTEST: MOV     A,PARAM1          ;Move char to be tested into ACC
E721 B44002        =1            CJNE    A,#'@',ZTEST      ;Carry will reset if char is <= '@'
E724 8005          =1            SJMP    CARSET            ;Set carry if equal to '@'
E726 4003          =1    ZTEST:  JC      CARSET            ;Reset carry if char is <= '@'
E728 B45A01        =1            CJNE    A,#'Z',AZEND      ;Carry will set if char is <= 'Z'
E72B B3            =1    CARSET: CPL     C                 ;Set carry if equal to 'Z'
E72C 22            =1    AZEND:  RET                       ;Exit from AZTEST
                   =1    ;*****************************************************************
E72D EA            =1    NMTEST:MOV      A,PARAM1          ;Move char into ACC
E72E C3            =1            CLR     C
E72F 9430          =1            SUBB    A,#('0')          ;See if char is <= ASCII '0'
E731 B3            =1            CPL     C
E732 5002          =1            JNC     NUMEND            ;Carry left 0 if false
E734 9409          =1            SUBB    A,#('9'-'0')      ;See if char is > ASCII '9'
E736 22            =1    NUMEND: RET                       ;Exit from NMTEST
                   =1    ;*****************************************************************
E737 F12D          =1    HXTEST: CALL NMTEST               ;See if char is between '0' and '9'
                   =1                                      ;Extra level of subroutine added
E739 4008          =1            JC      HEXEND            ;Jump if char between '0' and '9'
E73B EA            =1            MOV     A,PARAM1          ;Move char into ACC
E73C 9441          =1            SUBB    A,#'A'            ;See if char is > 'A'
E73E B3            =1            CPL     C
E73F 5002          =1            JNC     HEXEND            ;Carry left 0 if false
E741 9405          =1            SUBB    A,#('F'-'A')      ;See if char is less than 'F'
E743 22            =1    HEXEND: RET                       ;Exit from HXTEST
                   =1    ;*****************************************************************
E744 F120          =1    ALFNUM: CALL    AZTEST            ;See if char is between '@' and 'Z'
                   =1                                      ;Add extra level of subroutine
E746 4002          =1            JC      ANEND             ;Carry set if true
E748 F12D          =1            CALL    NMTEST            ;See if char is between '0' and '9'
                   =1                                      ;Added extra level of subroutine
```

```
LOC  OBJ            LINE      SOURCE

E74A 22          =1  1728      ANEND:  RET                     ;Exit from ALFNUM
                 =1  1729 +1   $EJECT
```

```
LOC  OBJ            LINE     SOURCE

               =1   1730     ;*************************************************************
               =1   1731     ;
               =1   1732     ;    NAME: LSSEQL
               =1   1733     ;
               =1   1734     ;    ABSTRACT: This is a 16-bit 'less than' or 'equal' check.  The
               =1   1735     ;       carry bit is set to indicate true. If MAXNUM_FLAGS is
               =1   1736     ;       true, no check is made.
               =1   1737     ;
               =1   1738     ;    INPUTS: PARAM1 (high byte to be compared to)
               =1   1739     ;            PARAM2 (low byte to be compared to)
               =1   1740     ;            PARAM3 (high byte to be compared)
               =1   1741     ;            PARAM4 (low byte to be compared)
               =1   1742     ;
               =1   1743     ;    OUTPUTS: Carry bit (C)
               =1   1744     ;
               =1   1745     ;    VARIABLES MODIFIED: C, MAXNUM_FLAG, PARAM1
               =1   1746     ;
               =1   1747     ;    ERROR EXITS: None
               =1   1748     ;
               =1   1749     ;    SUBROUTINES ACCESSED DIRECTLY: None
               =1   1750     ;
               =1   1751     ;
               =1   1752     ;*************************************************************
E74B 200417    =1   1753     LSSEQL: JB      MAXNUM_FLAG,LAB1B
E74E BCFF05    =1   1754             CJNE    PARAM3,#0FFH,START_COMPARE
E751 BDFF02    =1   1755             CJNE    PARAM4,#0FFH,START_COMPARE
E754 D204      =1   1756             SETB    MAXNUM_FLAG
               =1   1757     START_COMPARE:
E756 C3        =1   1758             CLR     C
E757 EB        =1   1759             MOV     A,PARAM2           ;Move byte to be compared to into ACC
E758 9D        =1   1760             SUBB    A,PARAM4           ;Subtract byte to be compared
E759 5006      =1   1761             JNC     LAB1
E75B 1A        =1   1762             DEC     PARAM1             ;Decrement upper byte if lower byte was smaller
E75C EA        =1   1763             MOV     A,PARAM1
E75D F4        =1   1764             CPL     A
E75E C3        =1   1765             CLR     C
E75F 6003      =1   1766             JZ      LAB1A              ;Error if PARAM1 decremented to FF
E761 EA        =1   1767     LAB1:   MOV     A,PARAM1           ;Move upper byte to be compared to into ACC
E762 9C        =1   1768             SUBB    A,PARAM3           ;Subtract upper byte to be compared
E763 B3        =1   1769             CPL     C                  ;Set C if <= is true
E764 22        =1   1770     LAB1A:  RET                        ;Exit from LSSEQL
E765 C204      =1   1771     LAB1B:  CLR     MAXNUM_FLAG
E767 C3        =1   1772             CLR     C
E768 22        =1   1773             RET
               =1   1774 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE
               =1  1775     ;*****************************************************************
               =1  1776     ;
               =1  1777     ;     NAME: (I)GETNUM / (I)GETEOL / (I)GET_COMMA
               =1  1778     ;
               =1  1779     ;     ABSTRACT: These routines are general purpose token checks.
               =1  1780     ;        IGETNUM will get a token and error if it is not
               =1  1781     ;        a number token, it will return if it is.  IGETEOL will
               =1  1782     ;        look for an end-of-line token and error if it is not
               =1  1783     ;        found, it will return if it is.  IGET_COMMA will look for
               =1  1784     ;        a comma token and will error if one is not found and return
               =1  1785     ;        if it is.
               =1  1786     ;
               =1  1787     ;     INPUTS: None
               =1  1788     ;
               =1  1789     ;     OUTPUTS: None
               =1  1790     ;
               =1  1791     ;     VARIABLES MODIFIED: ERRNUM
               =1  1792     ;
               =1  1793     ;     ERROR EXITS: 03H (NUMBER EXPECTED)
               =1  1794     ;                  06H (COMMA REQUIRED)
               =1  1795     ;
               =1  1796     ;     SUBROUTINES ACCESSED DIRECTLY: IERROR, IGETOKE
               =1  1797     ;
               =1  1798     ;
               =1  1799     ;*****************************************************************
E769 12E8BC    =1  1800     IGETNUM:CALL    IGETOKE
E76C 754303    =1  1801             MOV     ERRNUM,#03H             ;Number expected
E76F B40106    =1  1802             CJNE    A,#NUMBER_TOKE,UTILIT_ERROR
E772 22        =1  1803             RET
               =1  1804     ;*****************************************************************
E773 12E8BC    =1  1805     IGETEOL:CALL    IGETOKE
E776 A1BB      =1  1806             JMP     IEOL_CHECK              ;Check for end of line token
               =1  1807     UTILIT_ERROR:
E778 61E4      =1  1808             JMP     IERROR
               =1  1809     ;*****************************************************************
               =1  1810     IGET_COMMA:
E77A 12E8BC    =1  1811             CALL    IGETOKE
E77D 754306    =1  1812             MOV     ERRNUM,#06H             ;Comma required
E780 B402F5    =1  1813             CJNE    A,#COMMA_TOKE,UTILIT_ERROR
E783 22        =1  1814             RET
               =1  1815 +1  $EJECT
```

```
LOC  OBJ            LINE      SOURCE

                    =1        ;*****************************************************************
                    =1        ;
                    =1        ;    NAME: ISIT_DISPLAY
                    =1        ;
                    =1        ;    ABSTRACT: This routine checks for an equal or an EOL token,
                    =1        ;       sends the command token to the display with an = sign and
                    =1        ;       sets carry if and equal sign is found.  Carry is cleared
                    =1        ;       if an EOL is found..  The value is filled in by another routine.
                    =1        ;
                    =1        ;    INPUTS: TOKSTR
                    =1        ;
                    =1        ;    OUTPUTS: Carry bit (C)
                    =1        ;
                    =1        ;    VARIABLES MODIFIED: C, TOKSAV, PARAM1
                    =1        ;
                    =1        ;    ERROR EXITS: 05H (EQUAL OR RETURN EXPECTED)
                    =1        ;
                    =1        ;    SUBROUTINES ACCESSED DIRECTLY: IGETOKE, INEWLINE, ICO, IERROR
                    =1        ;
                    =1        ;
                    =1        ;*****************************************************************
                    =1        ISIT_DISPLAY:
E784 C3             =1                CLR     C
E785 85485B         =1                MOV     TOKSAV,TOKSTR
E788 12E8BC         =1                CALL    IGETOKE
E78B B4070D         =1                CJNE    A,#EOL_TOKE,CHANGE_CHECK
E78E F117           =1                CALL    INEWLINE
E790 AA5B           =1                MOV     PARAM1,TOKSAV
E792 12EA12         =1                CALL    IDISPLAY_TOKEN
E795 7A3D           =1                MOV     PARAM1,#'='
E797 B1E8           =1                CALL    ICO
E799 D3             =1                SETB    C
E79A 22             =1                RET
                    =1        CHANGE_CHECK:
E79B 754305         =1                MOV     ERRNUM,#05H            ;Equal or return expected
E79E B404D7         =1                CJNE    A,#EQUAL_TOKE,UTILIT_ERROR
E7A1 22             =1                RET
                    =1     +1 $EJECT
```

```
LOC  OBJ            LINE     SOURCE

               =1   1854     ;*****************************************************************
               =1   1855     ;
               =1   1856     ;    NAME: (I)GET_PART
               =1   1857     ;
               =1   1858     ;    ABSTRACT: This routine checks a token which is expected to be
               =1   1859     ;       a number, sets up the partition addresses and looks for
               =1   1860     ;       the upper partition limits from the user.  Carry will be set
               =1   1861     ;       if there is a partition or if there is an error condition.
               =1   1862     ;       The partition range, or length, will also be calculated.
               =1   1863     ;
               =1   1864     ;    INPUTS: TOKSTR, VALLOW, VALHGH
               =1   1865     ;
               =1   1866     ;    OUTPUTS: Carry bit (C)
               =1   1867     ;
               =1   1868     ;    VARIABLES MODIFIED: A, ERRNUM, PARTIT_HI_LOW, PARTIT_HI_HIGH,
               =1   1869     ;       PARTIT_LO_LOW, PARTIT_LO_HIGH, C, LENGTH_LOW, LENGTH_HIGH
               =1   1870     ;
               =1   1871     ;    ERROR EXITS: 07H (PARTITION ERROR, LOW ADDR > HIGH ADDR)
               =1   1872     ;
               =1   1873     ;    SUBROUTINES ACCESSED DIRECTLY: IGETOKE, IGETNUM, IERROR
               =1   1874     ;
               =1   1875     ;
               =1   1876     ;*****************************************************************
               =1   1877     IGET_PART:
E7A2 E548      =1   1878             MOV     A,TOKSTR
E7A4 754303    =1   1879             MOV     ERRNUM,#03H            ;Number expected
E7A7 B401CE    =1   1880             CJNE    A,#NUMBER_TOKE,UTILIT_ERROR     ;Set EA and SA to the value of the number.
E7AA 854A5A    =1   1881             MOV     PARTIT_HI_LOW,VALLOW
E7AD 854959    =1   1882             MOV     PARTIT_HI_HIGH,VALHGH
E7B0 854A58    =1   1883             MOV     PARTIT_LO_LOW,VALLOW
E7B3 854957    =1   1884             MOV     PARTIT_LO_HIGH,VALHGH
E7B6 12E8BC    =1   1885             CALL    IGETOKE                ;Get the next token.
E7B9 B40D1F    =1   1886             CJNE    A,#TO_TOKE,PARTITION_E  ;else set EA to the ending address of
E7BC F169      =1   1887             CALL    IGETNUM                ;the partition
E7BE 854A5A    =1   1888             MOV     PARTIT_HI_LOW,VALLOW
E7C1 854959    =1   1889             MOV     PARTIT_HI_HIGH,VALHGH
E7C4 C3        =1   1890             CLR     C
E7C5 E55A      =1   1891             MOV     A,PARTIT_HI_LOW
E7C7 9558      =1   1892             SUBB    A,PARTIT_LO_LOW
E7C9 F564      =1   1893             MOV     LENGTH_LOW,A
E7CB E559      =1   1894             MOV     A,PARTIT_HI_HIGH
E7CD 9557      =1   1895             SUBB    A,PARTIT_LO_HIGH
E7CF F563      =1   1896             MOV     LENGTH_HIGH,A
E7D1 754307    =1   1897             MOV     ERRNUM,#07H            ;Partition error, low adr > high adr
E7D4 40A2      =1   1898             JC      UTILIT_ERROR
E7D6 12E8BC    =1   1899             CALL    IGETOKE                ;and then read in the next token.
E7D9 D3        =1   1900             SETB    C
E7DA 22        =1   1901             RET
               =1   1902     PARTITION_E:
E7DB C3        =1   1903             CLR     C
E7DC 22        =1   1904             RET
               =1   1905 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                     =1  1906     ;*****************************************************************
                     =1  1907     ;
                     =1  1908     ;    NAME: (I)SAVE_AND_DISPLAY
                     =1  1909     ;
                     =1  1910     ;    ABSTRACT: This routine will convert a hex byte into two ASCII
                     =1  1911     ;       characters for display the next time PAINTER is called.
                     =1  1912     ;       POINTO must be set before calling this routine to the character
                     =1  1913     ;       position desired on the screen (ie LINBUF or LINBUF+n).  LNLGTH
                     =1  1914     ;       and CHRCNT are not adjusted by this routine.
                     =1  1915     ;
                     =1  1916     ;    INPUTS: POINTO (the location in the line buffer desired), PARAM1
                     =1  1917     ;       (the character to be displayed)
                     =1  1918     ;
                     =1  1919     ;    OUTPUTS: POINTO, 1 location in the line buffer
                     =1  1920     ;
                     =1  1921     ;    VARIABLES MODIFIED: POINTO, A, 1 location in the line buffer
                     =1  1922     ;
                     =1  1923     ;    ERROR EXITS: None
                     =1  1924     ;
                     =1  1925     ;    SUBROUTINES ACCESSED DIRECTLY: CONVHEX
                     =1  1926     ;
                     =1  1927     ;
                     =1  1928     ;*****************************************************************
                     =1  1929     ISAVE_AND_DISPLAY:
E7DD EA              =1  1930             MOV     A,PARAM1
E7DE C4              =1  1931             SWAP    A
E7DF 12E7EB          =1  1932             CALL    CONVHEX
E7E2 F6              =1  1933             MOV     @POINTO,A              ;ASCII of high byte in acc.
E7E3 08              =1  1934             INC     POINTO
E7E4 EA              =1  1935             MOV     A,PARAM1
E7E5 12E7EB          =1  1936             CALL    CONVHEX
E7E8 F6              =1  1937             MOV     @POINTO,A              ;ASCII of low byte in acc.
E7E9 08              =1  1938             INC     POINTO
E7EA 22              =1  1939             RET
                     =1  1940 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1        ;*****************************************************************
                =1        ;
                =1        ;    NAME: CONVHEX
                =1        ;
                =1        ;    ABSTRACT: Converts 4 bits to a hex character.
                =1        ;
                =1        ;    INPUTS: A (byte to be converted)
                =1        ;
                =1        ;    OUTPUTS: A
                =1        ;
                =1        ;    VARIABLES MODIFIED: A
                =1        ;
                =1        ;    ERROR EXITS: None
                =1        ;
                =1        ;    SUBROUTINES ACCESSED DIRECTLY: None
                =1        ;
                =1        ;
                =1        ;*****************************************************************
                =1        CONVHEX:
E7EB 540F       =1                ANL     A,#0FH                  ;ASCII No. 90-99, aux.C=0
E7ED 2490       =1                ADD     A,#90H                  ;9A-9F aux. C=1
E7EF D4         =1                DA      A
E7F0 3440       =1                ADDC    A,#40H
E7F2 D4         =1                DA      A
E7F3 22         =1                RET
                =1     +1 $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                 =1  1967     ;*****************************************************************
                 =1  1968     ;
                 =1  1969     ;    NAME: (I)LSTWRD/ (I)LSTBYT
                 =1  1970     ;
                 =1  1971     ;    ABSTRACT: Outputs a word or a byte to the system console.
                 =1  1972     ;
                 =1  1973     ;    INPUTS: PARAM2 (low byte of a word), PARAM1 (high byte of a word
                 =1  1974     ;       or the single byte in a byte display)
                 =1  1975     ;
                 =1  1976     ;    OUTPUTS: None
                 =1  1977     ;
                 =1  1978     ;    VARIABLES MODIFIED: A, PARAM1, PARAM3
                 =1  1979     ;
                 =1  1980     ;    ERROR EXITS: None
                 =1  1981     ;
                 =1  1982     ;    SUBROUTINES ACCESSED DIRECTLY: CONVHEX, ICO
                 =1  1983     ;
                 =1  1984     ;
                 =1  1985     ;*****************************************************************
E7F4 12E7F9      =1  1986     ILSTWRD:CALL    ILSTBYT
E7F7 EB          =1  1987             MOV     A,PARAM2
E7F8 FA          =1  1988             MOV     PARAM1,A
                 =1  1989     ;*****************************************************************
E7F9 EA          =1  1990     ILSTBYT:MOV     A,PARAM1                ;Move byte into ACC
E7FA FC          =1  1991             MOV     PARAM3,A
E7FB C4          =1  1992             SWAP    A
E7FC F1EB        =1  1993             CALL    CONVHEX
E7FE FA          =1  1994             MOV     PARAM1,A
E7FF 12E5E8      =1  1995             CALL    ICO                     ;Save lower 4 bits in lower 4 of PARAM3
E802 EC          =1  1996             MOV     A,PARAM3                ;Needed because reg to reg moves invalid
E803 12E7EB      =1  1997             CALL    CONVHEX
E806 FA          =1  1998             MOV     PARAM1,A
E807 02E5E8      =1  1999             JMP     ICO
                 =1  2000 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

               =1   2001     ;*****************************************************************
               =1   2002     ;
               =1   2003     ;    NAME: PAINTER
               =1   2004     ;
               =1   2005     ;    ABSTRACT: Repaints the contents of LINBUF to the display.
               =1   2006     ;
               =1   2007     ;    INPUTS: PARAM6 (contains line length, LNLGTH)
               =1   2008     ;
               =1   2009     ;    OUTPUTS: None
               =1   2010     ;
               =1   2011     ;    VARIABLES MODIFIED: A, PARAM1, POINT1, PARAM6
               =1   2012     ;
               =1   2013     ;    ERROR EXITS: None
               =1   2014     ;
               =1   2015     ;    SUBROUTINES ACCESSED DIRECTLY: UPI_OUT
               =1   2016     ;
               =1   2017     ;
               =1   2018     ;*****************************************************************
E80A 7924      =1   2019     PAINTER:MOV     POINT1,#LINBUF
               =1   2020     REPAINT_2:
E80C E7        =1   2021             MOV     A,@POINT1
E80D FA        =1   2022             MOV     PARAM1,A
E80E 12E638    =1   2023             CALL    UPI_OUT
E811 09        =1   2024             INC     POINT1
E812 DFF8      =1   2025             DJNZ    PARAM6,REPAINT_2
E814 22        =1   2026             RET
               =1   2027 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                     =1      ;*****************************************************************
                     =1      ;
                     =1      ;    NAME: GETCHR
                     =1      ;
                     =1      ;    ABSTRACT: This routine returns one character from the line
                     =1      ;       buffer in CHARIN if a carraige return has bee received.
                     =1      ;       If no °CR° is present, it gets characters from the UPI and
                     =1      ;       fills the line buffer until a °CR° is encountered. It echos
                     =1      ;       each character, as it is received, to the display. If LIST
                     =1      ;       is on, it echoes the entire line to the serial port after a
                     =1      ;       °CR° is encountered.
                     =1      ;
                     =1      ;    INPUTS: CHRCNT, LNLGTH, LSTFLG, LINE_START
                     =1      ;
                     =1      ;    OUTPUTS: CHARIN
                     =1      ;
                     =1      ;    VARIABLES MODIFIED: A, PARAM1, PARAM2, LNLGTH, CHRCNT, C, CHARIN
                     =1      ;
                     =1      ;    ERROR EXITS: None
                     =1      ;
                     =1      ;    SUBROUTINES ACCESSED DIRECTLY: ITIME, UPI_CMD, INEWLINE, PAINTER,
                     =1      ;       UPI_OUT, ICI, ICO, SPACCO
                     =1      ;
                     =1      ;
                     =1      ;*****************************************************************
E815 E551            =1      GETCHR: MOV     A,CHRCNT               ;Move character counter into ACC
E817 B55442          =1              CJNE    A,LNLGTH,OUTCHR        ;Compare ACC to line length and jump to
                     =1                                             ;OUTCHR if not equal
E81A 7A00            =1              MOV     PARAM1,#SELECT_CON
E81C 12E625          =1              CALL    UPI_CMD
E81F E552            =1              MOV     A,LINE_START
E821 F554            =1              MOV     LNLGTH,A               ;Clear character count and line length
E823 F551            =1              MOV     CHRCNT,A
E825 2423            =1              ADD     A,#(LINBUF-1)          ;Initialize R0 as pointer to line buffer
E827 F8              =1              MOV     POINT0,A
E828 12E717          =1      CRWAIT: CALL    INEWLINE
E82B AF54            =1              MOV     PARAM6,LNLGTH          ;Re-paint the alpha-numeric display.
E82D BF0003          =1              CJNE    PARAM6,#00H,REPAINT
E830 02E835          =1              JMP     REPAINT_1
E833 110A            =1      REPAINT:CALL    PAINTER
                     =1      REPAINT_1:
E835 7AAD            =1              MOV     PARAM1,#('-'+BLINK)
E837 12E638          =1              CALL    UPI_OUT
E83A 12E5EB          =1              CALL    ICI
E83D F550            =1              MOV     CHARIN,A               ;Move input into character storage
E83F FA              =1              MOV     PARAM1,A               ;Move CHARIN into R2
E840 BA0D24          =1              CJNE    PARAM1,#CR,RUBOUT      ;Check for CR as input
E843 7424            =1              MOV     A,#LINBUF
E845 2554            =1              ADD     A,LNLGTH
E847 F8              =1              MOV     POINT0,A               ;Load R0 to next char in line buffer
E848 760D            =1              MOV     @POINT0,#CR            ;Load CR into line buffer
E84A 0554            =1              INC     LNLGTH
E84C E4              =1              CLR     A
E84D A201            =1              MOV     C,LSTFLG
E84F 92E6            =1              MOV     ACC.6,C
```

```
LOC  OBJ            LINE     SOURCE

E851 FA        =1   2083              MOV     PARAM1,A
E852 12E625    =1   2084              CALL    UPI_CMD                 ;Turn list mode on if selected
E855 12E717    =1   2085              CALL    INEWLINE
E858 AF54      =1   2086              MOV     PARAM6,LNLGTH
E85A 110A      =1   2087              CALL    PAINTER                 ;Echoes line a final time in list mode
E85C 7424      =1   2088     OUTCHR:  MOV     A,#LINBUF               ;Load A with base addr of storage array
E85E 2551      =1   2089              ADD     A,CHRCNT                ;Add character count to ACC
E860 F8        =1   2090              MOV     POINTO,A                ;RO used as indirect pointer to char.
E861 E6        =1   2091              MOV     A,@POINTO               ;Return char to GETCHR call routine in ACC
E862 F550      =1   2092              MOV     CHARIN,A                ;Move character pointer to by RO
E864 0551      =1   2093              INC     CHRCNT                  ;Increment character counter
E866 22        =1   2094              RET                             ;Exit from GETCHR
E867 BA7F18    =1   2095     RUBOUT:  CJNE    PARAM1,#RBOUT,LEGALI    ;Check for rub out as input
E86A E554      =1   2096              MOV     A,LNLGTH                ;Move line length into ACC
E86C B55202    =1   2097              CJNE    A,LINE_START,DELET      ;Check if any characters were input yet
E86F 80B7      =1   2098              JMP     CRWAIT                  ;CR wait loop
E871 7A08      =1   2099     DELET:   MOV     PARAM1,#BACKSP
E873 12E5E8    =1   2100              CALL    ICO                     ;Output back space
E876 12E5E6    =1   2101              CALL    SPACCO                  ;Output space
E879 7A08      =1   2102              MOV     PARAM1,#BACKSP
E87B 12E5E8    =1   2103              CALL    ICO                     ;Output back space
E87E 1554      =1   2104              DEC     LNLGTH                  ;Decrement line length
E880 80A6      =1   2105              JMP     CRWAIT                  ;CR wait loop
E882 E554      =1   2106     LEGALI:  MOV     A,LNLGTH
E884 B41702    =1   2107              CJNE    A,#LINMAX-1,TABKEY      ;Check that line does not exceed max
E887 809F      =1   2108              JMP     CRWAIT                  ;CR wait loop
E889 BA091A    =1   2109     TABKEY:  CJNE    PARAM1,#HORIZONTAL_TAB,INPUT
E88C 7424      =1   2110              MOV     A,#LINBUF
E88E 2554      =1   2111              ADD     A,LNLGTH
E890 F8        =1   2112              MOV     POINTO,A
E891 E554      =1   2113              MOV     A,LNLGTH
               =1   2114     MORE_SPACE:
E893 04        =1   2115              INC     A
E894 F554      =1   2116              MOV     LNLGTH,A
E896 7620      =1   2117              MOV     @POINTO,#' '
E898 08        =1   2118              INC     POINTO
E899 B41702    =1   2119              CJNE    A,#LINMAX-1,MORE_CONT
E89C 808A      =1   2120              JMP     CRWAIT
               =1   2121     MORE_CONT:
E89E 30E0F2    =1   2122              JNB     ACC.0,MORE_SPACE
E8A1 30E1EF    =1   2123              JNB     ACC.1,MORE_SPACE
E8A4 8082      =1   2124              JMP     CRWAIT
E8A6 E550      =1   2125     INPUT:   MOV     A,CHARIN
E8A8 30E503    =1   2126              JNB     ACC.5,INPUTOK
E8AB 30E600    =1   2127              JNB     ACC.6,INPUTOK
E8AE 7424      =1   2128     INPUTOK:         MOV A,#LINBUF           ;Load A with line buffer base addr
E8B0 2554      =1   2129              ADD     A,LNLGTH                ;Add line length to ACC
E8B2 F8        =1   2130              MOV     POINTO,A                ;POINTO used as pointer to array
E8B3 A650      =1   2131              MOV     @POINTO,CHARIN          ;Load input into storage array
E8B5 12E5E8    =1   2132              CALL    ICO                     ;Output input
E8B8 0554      =1   2133              INC     LNLGTH                  ;Increment line length counter
E8BA 0128      =1   2134              JMP     CRWAIT                  ;CR wait routine
               =1   2135 +1  $EJECT
```

```
 . .  ****************************************************
      :TOKE

  T   This routine inputs characters, ignoring spaces, until
      buffer is full (LNCNT).  If the characters are numbers
  L   en type is designated °number° and its value goes into
  ^   and VALHGH.  It compares the input token to the keyword table
      'ors if not found.  If found, it checks the next keyword
 .y   :o see if the token is a valid abreviation.  Assembler
      is that are not numbers will have the basic operand type
      ?t (B_0_T).

      one

  >   TOKSTR, B_0_T, A

      MODIFIED: A, POINTO, LINCNT, @POINTO, PARAM1, TEMP1,
 \N   I, DPTR, TOKSTR, B_0_T

      :TS: 01H (INVALID WORD i.e. token)

 '"   NES ACCESSED DIRECTLY: IERROR, GETCHR, IGETOKE, AZTEST,
      R, ALFNUM, STRING_SPACE

      ****************************************************
 .N       B_0_T
 ''       A,CHARIN                  ;Move char into ACC
          A,#' ',ALPHA              ;Loop on space inputs
 \L       GETCHR                    ;Get new input
 ''       IGETOKE                   ;Space loop
          POINTO,#STRGBF
 U.       LINCNT,#TOKSIZ+1
 ^'       A,#' '                    ;Load ACC with ASCII equiv of space
          @POINTO,A                 ;Fill buffer with spaces
 IN       POINTO                    ;Increment string buffer pointer
  '       LINCNT,SPFILL             ;Loop until string buffer is filled
          LINCNT,#TOKSIZ            ;Move length of string into R1
 IU       POINTO,#STRGBF            ;Move base addr of string buffer into R0
 '^       PARAM1,CHARIN             ;Move char into R2
    .     AZTEST                    ;See if char is a letter
 JL       STRFIL
 ''       NUMBER                    ;Jump to number if false
    -     ALFNUM                    ;See if char is letter or number
 U.       STRTST                    ;Jump to filler routine if non-numerical
 '`       A,PARAM1                  ;Save char in string buffer
          @POINTO,A                 ;Needed because reg to reg move invalid
 I        POINTO                    ;Increment string buffer pointer
 ''       TEMP1,POINTO              ;Save pointer from GETCHAR
    .L    GETCHR                    ;Get next input
 M  '     PARAM1,CHARIN             ;To pass param for ALFNUM
 '  /     POINTO,TEMP1              ;Restore pointer for GETOKE
    IZ    LINCNT,STRFIL             ;Get more char if line counter is not 0
 L  _L    ALFNUM                    ;Check for alpha-numeric character
```

```
LOC  OBJ          LINE     SOURCE

E8F8 5006     =1  2191              JNC     STRTST                  ;Loop until space is input
E8FA 1115     =1  2192              CALL    GETCHR                  ;Get next character
E8FC AA50     =1  2193              MOV     PARAM1,CHARIN           ;Setup for ALFNUM
E8FE 80F5     =1  2194              SJMP    SPWAIT
E900 7A00     =1  2195     STRTST: MOV     PARAM1,#00H
E902 12E9CD   =1  2196     STRTST1:CALL    STRING_SPACE            ;Compare STRGBF to the keyword table.
E905 7013     =1  2197              JNZ     GOOD_TOKE_FOUND
E907 400A     =1  2198              JC      CHECK_ABREV
E909 0A       =1  2199              INC     PARAM1
E90A BA68F5   =1  2200              CJNE    PARAM1,#(KEYTAB-TOKTBL+1),STRTST1
E90D 754301   =1  2201     TOKERR: MOV     ERRNUM,#01H             ;Invalid word
E910 02E3E4   =1  2202              JMP     IERROR
              =1  2203     CHECK_ABREV:
E913 0A       =1  2204              INC     PARAM1
E914 12E9CD   =1  2205              CALL    STRING_SPACE
E917 1A       =1  2206              DEC     PARAM1
E918 40F3     =1  2207              JC      TOKERR
              =1  2208     GOOD_TOKE_FOUND:
E91A EA       =1  2209              MOV     A,PARAM1
E91B 90E070   =1  2210              MOV     DPTR,#(TOKTBL - 1)
E91E 93       =1  2211              MOVC    A,@A+DPTR               ;Get token from table
E91F F548     =1  2212              MOV     TOKSTR,A                ;Put token in storage
E921 B44000   =1  2213              CJNE    A,#40H,GT0              ;Set basic operand type flag for
E924 4007     =1  2214     GT0:     JC      NOTBOT                  ;Tokens that are assembler operands which
E926 B49800   =1  2215              CJNE    A,#98H,GT1              ;are not numbers.
E929 5002     =1  2216     GT1:     JNC     NOTBOT
E92B D200     =1  2217              SETB    B_O_T
E92D E548     =1  2218     NOTBOT: MOV     A,TOKSTR
E92F 22       =1  2219              RET
              =1  2220 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

               =1   2221     ;***************************************************************
               =1   2222     ;
               =1   2223     ;    NAME: NUMBER
               =1   2224     ;
               =1   2225     ;    ABSTRACT: This routine checks to see if a number of characters
               =1   2226     ;       (1-24) is a valid hex number, converts it to a
               =1   2227     ;       16 bit binary number and gives it a number token if
               =1   2228     ;       is.  It ignores leading zeros and trailing 'Hs'.
               =1   2229     ;
               =1   2230     ;    INPUTS: A
               =1   2231     ;
               =1   2232     ;    OUTPUTS: TOKSTR, VALHGH, VALLOW
               =1   2233     ;
               =1   2234     ;    VARIABLES MODIFIED: VALLOW, VALHGH, PARAM2, A, B, TOKSTR
               =1   2235     ;
               =1   2236     ;    ERROR EXITS: None
               =1   2237     ;
               =1   2238     ;    SUBROUTINES ACCESSED DIRECTLY: NMTEST, HXTEST, GETCHR
               =1   2239     ;
               =1   2240     ;
               =1   2241     ;***************************************************************
E930 12E72D    =1   2242     NUMBER: CALL    NMTEST
E933 505F      =1   2243             JNC     SYMBOL                  ;Jump if char is not a number
E935 754A00    =1   2244             MOV     VALLOW,#00H             ;Initialize value storage
E938 754900    =1   2245             MOV     VALHGH,#00H
E93B 12E737    =1   2246     HEXSTR: CALL    HXTEST
E93E 502B      =1   2247             JNC     HTEST                   ;Jump if char is not a hex character
E940 12E72D    =1   2248             CALL    NMTEST                  ;Check for character=0 to 9
E943 5022      =1   2249             JNC     HEXCHR                  ;Load A into PARAM2 for hex char
E945 7B30      =1   2250             MOV     PARAM2,#'0'             ;Clear pointer
E947 E54A      =1   2251     RL4:    MOV     A,VALLOW
E949 75F010    =1   2252             MOV     B,#16                   ;To RL 4 places
E94C A4        =1   2253             MUL     AB
E94D F54A      =1   2254             MOV     VALLOW,A                ;ACC now holds VALLOW RL 4  places
E94F E550      =1   2255             MOV     A,CHARIN                ;Move last number entered into ACC
E951 9B        =1   2256             SUBB    A,PARAM2                ;Subtract ASCII equiv of 'A' or '0'
               =1   2257                                             ;as appropriate for hex or decimal
E952 254A      =1   2258             ADD     A,VALLOW                ;Add number to rotated VALLOW
E954 F54A      =1   2259             MOV     VALLOW,A                ;Store new value in VALLOW
E956 AAF0      =1   2260             MOV     PARAM1,B                ;Store upper 4 bits from rotate
E958 75F010    =1   2261             MOV     B,#10H
E95B E549      =1   2262             MOV     A,VALHGH                ;Move VALHGH into ACC
E95D A4        =1   2263             MUL     AB                      ;Rotate VALHGH 4 places to left
E95E 2A        =1   2264             ADD     A,PARAM1                ;Add upper 4 bits from VALLOW
E95F F549      =1   2265             MOV     VALHGH,A                ;Store new value in VALHGH
E961 1115      =1   2266             CALL    GETCHR                  ;Get next input
E963 AA50      =1   2267             MOV     PARAM1,CHARIN           ;Set up pass param for HXTEST
E965 80D4      =1   2268             SJMP    HEXSTR                  ;Loop until non hex char entered
E967 7B37      =1   2269     HEXCHR: MOV     PARAM2,#('A'-0AH)       ;Move ASCII equiv of 'A' into POINT1
E969 80DC      =1   2270             SJMP    RL4
E96B E550      =1   2271     HTEST:  MOV     A,CHARIN
E96D B44802    =1   2272             CJNE    A,#'H',NUMBER_1         ;See if char is 'H' and ignore if so
E970 1115      =1   2273             CALL    GETCHR
               =1   2274     NUMBER_1:
E972 E550      =1   2275             MOV     A,CHARIN                ;Look at next character
```

```
LOC  OBJ                   SOURCE

E974 B42C02      =1                  CJNE    A,#',',NUMBER_2         ;Check for valid delimiter - comma
E977 8015        =1                  SJMP    NUMBER_FOUND
                 =1        NUMBER_2:
E979 B40D02      =1                  CJNE    A,#CR,NUMBER_3          ;Check for valid delimiter - CR
E97C 8010        =1                  SJMP    NUMBER_FOUND
                 =1        NUMBER_3:
E97E B43D02      =1                  CJNE    A,#'=',NUMBER_4         ;Check for valid delimiter - equal sign
E981 800B        =1                  SJMP    NUMBER_FOUND
                 =1        NUMBER_4:
E983 B42002      =1                  CJNE    A,#' ',NUMBER_ERR       ;Check for valid delimiter - space
E986 8006        =1                  SJMP    NUMBER_FOUND
                 =1        NUMBER_ERR:
E988 754303      =1                  MOV     ERRNUM,#03H             ;Set up 'number req' error
E98B 02E3E4      =1                  JMP     IERROR
                 =1        NUMBER_FOUND:
E98E 754801      =1                  MOV     TOKSTR,#NUMBER_TOKE     ;Load toke storage with number token
E991 E548        =1                  MOV     A,TOKSTR                ;Load ACC with TOKEN
E993 22          =1                  RET
                 =1     +1 $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                    =1       ;*****************************************************************
                    =1       ;
                    =1       ;     NAME: SYMBOL
                    =1       ;
                    =1       ;     ABSTRACT: This routine checks a token against the symbol
                    =1       ;        table tokens (ie comma, equal sign, etc.), errors if
                    =1       ;        there is no match and returns the token in ACC if it is
                    =1       ;        found.
                    =1       ;
                    =1       ;     INPUTS: PARAM1
                    =1       ;
                    =1       ;     OUTPUTS: A, TOKSTR
                    =1       ;
                    =1       ;     VARIABLES MODIFIED: TOKSTR, A, DPTR, ERRNUM, CHARIN
                    =1       ;
                    =1       ;     ERROR EXITS: 01H (INVALID WORD)
                    =1       ;
                    =1       ;     SUBROUTINES ACCESSED DIRECTLY: IERROR, GETCHR
                    =1       ;
                    =1       ;
                    =1       ;*****************************************************************
E994 8A48           =1       SYMBOL: MOV     TOKSTR,PARAM1
E996 90E9AE         =1              MOV      DPTR,#SYMBOL_TBL
                    =1       SYM_TBL_SRCH:
E999 E4             =1              CLR      A
E99A 93             =1              MOVC     A,@A+DPTR
E99B 754301         =1              MOV      ERRNUM,#01H                  ;Invalid token (word)
E99E 601C           =1              JZ       ERRSET
E9A0 B54807         =1              CJNE     A,TOKSTR,NOT_MATCH_TBL
E9A3 A3             =1              INC      DPTR
E9A4 E4             =1              CLR      A
E9A5 93             =1              MOVC     A,@A+DPTR
E9A6 F548           =1              MOV      TOKSTR,A
E9A8 8015           =1              SJMP     SYMEND
                    =1       NOT_MATCH_TBL:
E9AA A3             =1              INC      DPTR
E9AB A3             =1              INC      DPTR
E9AC 80EB           =1              SJMP     SYM_TBL_SRCH
                    =1       SYMBOL_TBL:
E9AE 2C             =1              DB       ',',COMMA_TOKE
E9AF 02
E9B0 2F             =1              DB       '/',BAR_TOKE
E9B1 03
E9B2 3D             =1              DB       '=',EQUAL_TOKE
E9B3 04
E9B4 2B             =1              DB       '+',PLUS_TOKE
E9B5 05
E9B6 23             =1              DB       '#',POUND_TOKE
E9B7 06
E9B8 0D             =1              DB       CR,EOL_TOKE
E9B9 07
E9BA 00             =1              DB       0,0
E9BB 00
E9BC 02E3E4         =1       ERRSET: JMP     IERROR
E9BF BA0D06         =1       SYMEND: CJNE    PARAM1,#CR,LAB10        ;See if last input was a 'CR'
```

```
LOC  OBJ            LINE     SOURCE

E9C2 755020     =1  2343              MOV     CHARIN,#' '          ;Return a space to calling routine if 'CR'
E9C5 E548       =1  2344              MOV     A,TOKSTR             ;Load ACC with token
E9C7 22         =1  2345              RET                          ;Exit from GETOKE
E9C8 1115       =1  2346     LAB10:   CALL    GETCHR               ;Get next character if 'CR' wasn't last char
E9CA E548       =1  2347              MOV     A,TOKSTR             ;To return token in ACC
E9CC 22         =1  2348              RET                          ;Exit from GETOKE
                =1  2349 +1  $EJECT
```

```
LOC  OBJ            LINE      SOURCE

                =1  2350      ;****************************************************************
                =1  2351      ;
                =1  2352      ;     NAME: STRING_SPACE
                =1  2353      ;
                =1  2354      ;     ABSTRACT: This routine checks the contents of the string buffer
                =1  2355      ;        against the keyword table for any match (ie a valid abreviation
                =1  2356      ;        or an exact match) and returns to the calling routine.  There
                =1  2357      ;        are 4 places in every keyword and this routine matches for
                =1  2358      ;        spaces as well as characters. Carry and ACC are set
                =1  2359      ;        if match is exact, carry is set and ACC is cleared if match is
                =1  2360      ;        not exact (ie spaces do not match - could be an abrev.), both
                =1  2361      ;        carry and ACC are cleared if there is no match at all.
                =1  2362      ;
                =1  2363      ;     INPUTS: STRGBF, PARAM1 (token ordinal in KEYTAB)
                =1  2364      ;
                =1  2365      ;     OUTPUTS: Carry bit (C), A
                =1  2366      ;
                =1  2367      ;     VARIABLES MODIFIED: C, A, POINTO, STRGCT, DPTR, B, TEMP1
                =1  2368      ;
                =1  2369      ;     ERROR EXITS: None
                =1  2370      ;
                =1  2371      ;     SUBROUTINES ACCESSED DIRECTLY: None
                =1  2372      ;
                =1  2373      ;
                =1  2374      ;****************************************************************
                =1  2375      STRING_SPACE:
E9CD 783C       =1  2376              MOV     POINTO,#STRGBF  ;Load RO with address of string buffer
E9CF 755504     =1  2377              MOV     STRGCT,#TOKSIZ  ;Load counter with length of string
E9D2 90E0D4     =1  2378              MOV     DPTR,#(KEYTAB-4);Load DPTR with address of KEY TABLE
E9D5 75F004     =1  2379              MOV     B,#4
E9D8 EA         =1  2380              MOV     A,PARAM1        ;Load ACC with offset
E9D9 A4         =1  2381              MUL     AB              ;Multiply by 4 characters
E9DA C3         =1  2382              CLR     C
 9DB 2582       =1  2383              ADD     A,DPL           ;Add offset to base
 9DD F582       =1  2384              MOV     DPL,A
E9DF E5F0       =1  2385              MOV     A,B
E9E1 3583       =1  2386              ADDC    A,DPH
E9E3 F583       =1  2387              MOV     DPH,A
E9E5 E4         =1  2388      S_S_1:  CLR     A
E9E6 93         =1  2389              MOVC    A,@A+DPTR
E9E7 F556       =1  2390              MOV     TEMP1,A
E9E9 E6         =1  2391              MOV     A,@POINTO
E9EA B55609     =1  2392              CJNE    A,TEMP1,S_S_2
E9ED A3         =1  2393              INC     DPTR            ;Next key character
E9EE 08         =1  2394              INC     POINTO          ;Next string character
E9EF D555F3     =1  2395              DJNZ    STRGCT,S_S_1    ;Test the whole 4 char string
E9F2 D3         =1  2396              SETB    C               ;Match exactly including spaces
E9F3 E4         =1  2397              CLR     A
E9F4 F4         =1  2398              CPL     A
E9F5 22         =1  2399              RET
 9F6 B42003     =1  2400      S_S_2:  CJNE    A,#' ',S_S_3    ;Match but not exact (spaces)
 9F9 D3         =1  2401              SETB    C
 9FA E4         =1  2402              CLR     A
E9FB 22         =1  2403              RET
E9FC C3         =1  2404      S_S_3:  CLR     C               ;No match at all
```

```
LOC  OBJ            LINE     SOURCE

E9FD E4          =1  2405             CLR      A
E9FE 22          =1  2406             RET
                 =1  2407 +1  $EJECT
```

```
LOC  OBJ                SOURCE
                  =1         ;*****************************************************************
                  =1         ;
                  =1         ;    NAME: (I)PRINT_STRING
                  =1         ;
                  =1         ;    ABSTRACT: Prints a string from program memory.  At entry, PARAM1
                  =1         ;       and PARAM2 should point to the string.  The first element of
                  =1         ;       the string is the length (0-255), the rest of the elements are
                  =1         ;       output as ASCII characters.
                  =1         ;
                  =1         ;       WARNING: Calls to this routine may not be single-stepped through.
                  =1         ;
                  =1         ;    INPUTS: PARAM1(high byte), PARAM2(low byte)
                  =1         ;
                  =1         ;    OUTPUTS: None
                  =1         ;
                  =1         ;    VARIABLES MODIFIED: A, COUNT, DPTR, PARAM1
                  =1         ;
                  =1         ;    ERROR EXITS: None
                  =1         ;
                  =1         ;    SUBROUTINES ACCESSED DIRECTLY: ICO
                  =1         ;
                  =1         ;*****************************************************************
                  =1         IPRINT_STRING:
E9FF 8A83         =1                 MOV     DPH,PARAM1
EA01 8B82         =1                 MOV     DPL,PARAM2
EA03 E4           =1                 CLR     A                ;Counter:=string length.
EA04 93           =1                 MOVC    A,@A+DPTR
EA05 FF           =1                 MOV     COUNT,A
EA06 6009         =1                 JZ      PRINT_STRING_E   ;Exit if a null string or
                  =1         PRINT_STRING_1:
EA08 E4           =1                 CLR     A                ;else get the next element
EA09 A3           =1                 INC     DPTR
EA0A 93           =1                 MOVC    A,@A+DPTR
EA0B FA           =1                 MOV     PARAM1,A         ;and output it.
EA0C 12E5E8       =1                 CALL    ICO              ;Repeat loop until count=0.
EA0F DFF7         =1                 DJNZ    COUNT,PRINT_STRING_1
                  =1         PRINT_STRING_E:
EA11 22           =1                 RET                      ;Then return to the caller.
                  =1      +1 $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1  2447     ;****************************************************************
                =1  2448     ;
                =1  2449     ;    NAME: (I)DISPLAY_TOKEN
                =1  2450     ;
                =1  2451     ;    ABSTRACT: This routine displays an ASCII token using the token
                =1  2452     ;       value passed to it (PARAM1) to indicate which token to display.
                =1  2453     ;
                =1  2454     ;    INPUTS: PARAM1 (token to be displayed)
                =1  2455     ;
                =1  2456     ;    OUTPUTS: None
                =1  2457     ;
                =1  2458     ;    VARIABLES MODIFIED: PARAM2, DPTR, A, PARAM3, PARAM1
                =1  2459     ;
                =1  2460     ;    ERROR EXITS:  None
                =1  2461     ;
                =1  2462     ;    SUBROUTINES ACCESSED DIRECTLY: ICO
                =1  2463     ;
                =1  2464     ;****************************************************************
                =1  2465     IDISPLAY_TOKEN:
EA12 7B00       =1  2466             MOV     PARAM2,#00H
EA14 C3         =1  2467             CLR     C
                =1  2468     DTO_O:
EA15 90E071     =1  2469             MOV     DPTR,#TOKTBL
EA18 EB         =1  2470             MOV     A,PARAM2
EA19 93         =1  2471             MOVC    A,@A+DPTR
EA1A B50203     =1  2472             CJNE    A,2,DTO              ;2 is the direct addr of R2 which we call PARAM1
EA1D 02EA23     =1  2473             JMP     DT1
                =1  2474     DTO:
EA20 0B         =1  2475             INC     PARAM2
EA21 80F2       =1  2476             JMP     DTO_O
                =1  2477     DT1:
EA23 90E0D8     =1  2478             MOV     DPTR,#KEYTAB
                =1  2479     DT_LOOP:
EA26 A3         =1  2480             INC     DPTR
EA27 A3         =1  2481             INC     DPTR
EA28 A3         =1  2482             INC     DPTR
EA29 A3         =1  2483             INC     DPTR
EA2A DBFA       =1  2484             DJNZ    PARAM2,DT_LOOP
EA2C 7C04       =1  2485             MOV     PARAM3,#04H
EA2E E4         =1  2486     TOKLOP: CLR     A
EA2F 93         =1  2487             MOVC    A,@A+DPTR            ;Load ACC with first character of token
EA30 B42001     =1  2488             CJNE    A,#' ',TOK_WRITE
EA33 22         =1  2489             RET
                =1  2490     TOK_WRITE:
EA34 FA         =1  2491             MOV     PARAM1,A             ;To output character
EA35 12E5E8     =1  2492             CALL    ICO
EA38 A3         =1  2493             INC     DPTR
EA39 DCF3       =1  2494             DJNZ    PARAM3,TOKLOP        ;Loop if less than 4 characters output
EA3B 22         =1  2495             RET
                =1  2496     ;***** END OF DISPLAY_TOKEN *****
                =1  2497 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

               =1   2498     ;*****************************************************************
               =1   2499     ;
               =1   2500     ;    NAME: ASCII_TO_HEX (PARAM1)
               =1   2501     ;
               =1   2502     ;    ABSTRACT: Assumes that PARAM1 is an ASCII character representing
               =1   2503     ;       a hexidecimal digit and converts it to binary.  The result
               =1   2504     ;       is returned in the lower four bits of the accumulator.  The
               =1   2505     ;       upper bits are cleared.
               =1   2506     ;
               =1   2507     ;    INPUTS: PARAM1 (ASCII character)
               =1   2508     ;
               =1   2509     ;    OUTPUTS: A
               =1   2510     ;
               =1   2511     ;    VARIABLES MODIFIED: A
               =1   2512     ;
               =1   2513     ;    ERROR EXITS: None
               =1   2514     ;
               =1   2515     ;    SUBROUTINES ACCESSED DIRECTLY: None
               =1   2516     ;
               =1   2517     ;*****************************************************************
               =1   2518     IASCII_TO_HEX:
EA3C EA        =1   2519             MOV     A,PARAM1              ;Put ASCII character into ACC
EA3D 30E602    =1   2520             JNB     ACC.6,HEX1            ;Jump to HEX1 if CHAR < 40H
EA40 2409      =1   2521             ADD     A,#09H                ;Add nine if CHAR > 3FH
EA42 540F      =1   2522     HEX1:   ANL     A,#0FH                ;Mask lower 4 bits
EA44 22        =1   2523             RET
               =1   2524 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

               =1  2525     ;*****************************************************************
               =1  2526     ;
               =1  2527     ;     NAME: ITIME
               =1  2528     ;
               =1  2529     ;     ABSTRACT: TIME is a general purpose routine available through
               =1  2530     ;        the jump table. Parameter 1 and 2 are the high and low bytes
               =1  2531     ;        of a sixteen bit timer where each increment represents
               =1  2532     ;        100 uS as in PLM.
               =1  2533     ;        Time simply delays for the specified time and then returns.
               =1  2534     ;
               =1  2535     ;     INPUTS: PARAM1 (high byte), PARAM2 (low byte)
               =1  2536     ;
               =1  2537     ;     OUTPUTS: None
               =1  2538     ;
               =1  2539     ;     VARIABLES MODIFIED: A, DPTR, R5
               =1  2540     ;
               =1  2541     ;     ERROR EXITS: None
               =1  2542     ;
               =1  2543     ;     SUBROUTINES ACCESSED DIRECTLY: None
               =1  2544     ;
               =1  2545     ;
               =1  2546     ;*****************************************************************
               =1  2547
EA45 EA        =1  2548     ITIME:  MOV     A,PARAM1         ;Convert PARAM1 and PARAM2 into one 16-bit
EA46 F4        =1  2549             CPL     A                ;negative number in DPTR
EA47 F583      =1  2550             MOV     DPH,A
EA49 EB        =1  2551             MOV     A,PARAM2
EA4A F4        =1  2552             CPL     A
EA4B F582      =1  2553             MOV     DPL,A
EA4D A3        =1  2554             INC     DPTR
EA4E 7D2E      =1  2555     TIME1:  MOV     R5,#2EH          ;Setup and
EA50 DDFE      =1  2556             DJNZ    R5,$             ;Loop for 100 us
EA52 A3        =1  2557             INC     DPTR             ;Count out the 16-bit parameter
EA53 E582      =1  2558             MOV     A,DPL            ;Check DPTR for zero
EA55 4583      =1  2559             ORL     A,DPH
EA57 00        =1  2560             NOP
EA58 70F4      =1  2561             JNZ     TIME1
EA5A 22        =1  2562             RET
               =1  2563     ;*****************************************************************
                   2564 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                    2565 +1  $INCLUDE(:f1:DISCHA.INC)
               =1   2566     ;*****************************************************************
               =1   2567     ;
               =1   2568     ;    NAME: MEMORY_CMD
               =1   2569     ;
               =1   2570     ;    ABSTRACT: This routine saves the kind of memory operation
               =1   2571     ;       selected and checks for partitions and equal signs in order
               =1   2572     ;       to dicide whether a fill, load, display or block move is
               =1   2573     ;       requested.
               =1   2574     ;
               =1   2575     ;    INPUTS: TOKSTR
               =1   2576     ;
               =1   2577     ;    OUTPUTS: None
               =1   2578     ;
               =1   2579     ;    VARIABLES MODIFIED: A, TOKSAV, SELECT, PNTLOW, PNTHGH, B
               =1   2580     ;
               =1   2581     ;    ERROR EXITS: None
               =1   2582     ;
               =1   2583     ;    SUBROUTINES ACCESSED DIRECTLY: IGETOKE, IGET_PART, BMOVE,
               =1   2584     ;       IEOL_CHECK, DISMEM, LODMEM, FILLMEM
               =1   2585     ;
               =1   2586     ;
               =1   2587     ;*****************************************************************
EA5B E548      =1   2588     MEMORY_CMD:     MOV     A,TOKSTR
EA5D 5407      =1   2589             ANL     A,#07                   ;Last 3 bits of token determine selector
EA5F 85485B    =1   2590             MOV     TOKSAV,TOKSTR
EA62 F546      =1   2591             MOV     SELECT,A                ;Load selector
EA64 11BC      =1   2592             CALL    IGETOKE
EA66 12E7A2    =1   2593             CALL    IGET_PART               ;Partition? Returns 1 bit (C)=true if part.
EA69 855845    =1   2594             MOV     PNTLOW,PARTIT_LO_LOW
EA6C 855744    =1   2595             MOV     PNTHGH,PARTIT_LO_HIGH
EA6F 92F0      =1   2596             MOV     B.0,C
EA71 B4040B    =1   2597             CJNE    A,#EQUAL_TOKE,DIS_OR_ERR        ;Check for equal sign from GET_PART
EA74 30F00E    =1   2598             JNB     B.0,LODMEM              ;Single byte load (CBY addr = data)
EA77 11BC      =1   2599             CALL    IGETOKE
EA79 B48061    =1   2600             CJNE    A,#CBYTE_TOKE,FILLMEM   ;Block move (CBY addr TO addr =CBY addr)
EA7C 02EB58    =1   2601             JMP     BMOVE                   ;Fill mem. (CBY addr TO addr=data)
               =1   2602     DIS_OR_ERR:
EA7F 12E5BB    =1   2603             CALL    IEOL_CHECK
EA82 02EB02    =1   2604             JMP     DISMEM                  ;Display mem. (CBY addr TO addr-no equalsign)
               =1   2605 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

               =1  2606     ;*****************************************************************
               =1  2607     ;
               =1  2608     ;    NAME: LODMEM
               =1  2609     ;
               =1  2610     ;    ABSTRACT: The pointer will be set to memory address upon entry.
               =1  2611     ;       Parsing continues as long as new tokens are available on the
               =1  2612     ;       command line.  Each new token either supplies a new value which
               =1  2613     ;       goes into memory or a <CR> which terminates the command.  Commas
               =1  2614     ;       are expected between any two numbers and at the end of a line
               =1  2615     ;       when a continuation is desired.  When entry of data has gone
               =1  2616     ;       beyond one line (a continuation line) the line buffer is filled
               =1  2617     ;       with the message and address which tells the user what address
               =1  2618     ;       is currently being modified.
               =1  2619     ;
               =1  2620     ;    INPUTS: SELECT, PNTHGH, PNTLOW
               =1  2621     ;
               =1  2622     ;    OUTPUTS: Memory which was supposed to be accessed by the command
               =1  2623     ;       typed in at the console.
               =1  2624     ;
               =1  2625     ;    VARIABLES MODIFIED: PARAM1, A, POINTO, LINE_START
               =1  2626     ;
               =1  2627     ;    ERROR EXITS: None
               =1  2628     ;
               =1  2629     ;    SUBROUTINES ACCESSED DIRECTLY: IGETNUM, ISTORE, IGETOKE, INC_PNT,
               =1  2630     ;       ISAVE_AND_DISPLAY, IEOL_CHECK, IERROR
               =1  2631     ;
               =1  2632     ;
               =1  2633     ;*****************************************************************
EA85 12E769    =1  2634     LODMEM: CALL    IGETNUM
EA88 AA4A      =1  2635     LDLOOP: MOV     PARAM1,VALLOW          ;Load PARAM1 with data to be output
EA8A 12E672    =1  2636             CALL    ISTORE                 ;Output data into memory
EA8D 12E5C4    =1  2637             CALL    INC_PNT
EA90 11BC      =1  2638             CALL    IGETOKE                ;Get next token and character
EA92 B40242    =1  2639             CJNE    A,#COMMA_TOKE,EOLMEM   ;Jump to EOLMEM if token is not comma token
EA95 11BC      =1  2640             CALL    IGETOKE                ;Get next token and character after comma
EA97 B40738    =1  2641             CJNE    A,#EOL_TOKE,NUMMEN     ;Check if CR was entered
EA9A 7824      =1  2642             MOV     POINTO,#LINBUF
EA9C E546      =1  2643             MOV     A,SELECT               ;Choose first char, depending on type
EA9E 7652      =1  2644             MOV     @POINTO,#'R'           ;of memory access in progress
EAA0 B40002    =1  2645             CJNE    A,#(CBYTE_TOKE AND 07H),B_LAB_1
EAA3 7643      =1  2646             MOV     @POINTO,#'C'
EAA5 B40202    =1  2647     B_LAB_1:CJNE    A,#(DBYTE_TOKE AND 07H),B_LAB_2
EAA8 7644      =1  2648             MOV     @POINTO,#'D'
EAAA B40602    =1  2649     B_LAB_2:CJNE    A,#(XBYTE_TOKE AND 07H),B_LAB_3
EAAD 7658      =1  2650             MOV     @POINTO,#'X'
EAAF 08        =1  2651     B_LAB_3:INC     POINTO
EAB0 7642      =1  2652             MOV     @POINTO,#'B'
EAB2 08        =1  2653             INC     POINTO
EAB3 7659      =1  2654             MOV     @POINTO,#'Y'
EAB5 B40402    =1  2655             CJNE    A,#(RBIT_TOKE AND 07H),T_LAB
EAB8 7649      =1  2656             MOV     @POINTO,#'I'           ;Choose third char for bit or byte type
EABA 08        =1  2657     T_LAB:  INC     POINTO
EABB 7654      =1  2658             MOV     @POINTO,#'T'
EABD 08        =1  2659             INC     POINTO
EABE 7620      =1  2660             MOV     @POINTO,#' '
```

```
LOC  OBJ            LINE     SOURCE

EAC0 08         =1  2661              INC     POINTO
EAC1 AA44       =1  2662              MOV     PARAM1,PNTHGH
EAC3 12E7DD     =1  2663              CALL    ISAVE_AND_DISPLAY
EAC6 AA45       =1  2664              MOV     PARAM1,PNTLOW
EAC8 12E7DD     =1  2665              CALL    ISAVE_AND_DISPLAY
EACB 763D       =1  2666              MOV     @POINTO,#'='
EACD 75520A     =1  2667              MOV     LINE_START,#0AH
EAD0 11BC       =1  2668              CALL    IGETOKE                  ;Get next token and character
EAD2 B40102     =1  2669     NUMMEN: CJNE    A,#NUMBER_TOKE,EOLMEM    ;Check that a number was last char entered
                =1  2670
EAD5 80B1       =1  2671              JMP     LDLOOP                   ;Loop until CR entered
EAD7 02E5BB     =1  2672     EOLMEM: JMP     IEOL_CHECK
EADA 02E3E4     =1  2673     DISERR: JMP     IERROR
                =1  2674 +1  $EJECT
```

```
LOC  OBJ            LINE      SOURCE

               =1   2675      ;*****************************************************************
               =1   2676      ;
               =1   2677      ;    NAME: FILLMEM
               =1   2678      ;
               =1   2679      ;    ABSTRACT: This routine fills the memory selected with a single
               =1   2680      ;       value from PNTLOW and PNTHGH up to the high end of the
               =1   2681      ;       partition.
               =1   2682      ;
               =1   2683      ;    INPUTS: PNTLOW, PNTHGH, PARTIT_HI_LOW, PARTIT_HI_HIGH
               =1   2684      ;
               =1   2685      ;    OUTPUTS: Memory which was supposed to be accessed by the
               =1   2686      ;       command typed in at the console.
               =1   2687      ;
               =1   2688      ;    VARIABLES MODIFIED: ERRNUM, A, TEMP_LOW, VALLOW, PARAM1, C
               =1   2689      ;
               =1   2690      ;    ERROR EXITS: 1AH (TOKEN MUST BE A NUMBER)
               =1   2691      ;
               =1   2692      ;    SUBROUTINES ACCESSED DIRECTLY: IGETEOL, ISTORE, INC_PNT
               =1   2693      ;
               =1   2694      ;
               =1   2695      ;*****************************************************************
EADD 75431A    =1   2696      FILLMEM:MOV     ERRNUM,#1AH          ;Token must be a number
EAE0 B401F7    =1   2697              CJNE    A,#NUMBER_TOKE,DISERR
EAE3 854A47    =1   2698              MOV     TEMP_LOW,VALLOW
EAE6 12E773    =1   2699              CALL    IGETEOL
EAE9 85474A    =1   2700              MOV     VALLOW,TEMP_LOW
EAEC AA4A      =1   2701      FILLOOP:MOV     PARAM1,VALLOW        ;Load PARAM1 with single byte data
EAEE 12E672    =1   2702              CALL    ISTORE               ;Output data into memory
EAF1 C3        =1   2703              CLR     C
EAF2 E545      =1   2704              MOV     A,PNTLOW
EAF4 955A      =1   2705              SUBB    A,PARTIT_HI_LOW      ;Subtract pointer from ending address
EAF6 E544      =1   2706              MOV     A,PNTHGH
EAF8 9559      =1   2707              SUBB    A,PARTIT_HI_HIGH     ;to see if partition is full yet
EAFA 5005      =1   2708              JNC     FILL1                ;If not, continue filling
EAFC 12E5C4    =1   2709              CALL    INC_PNT
EAFF 80EB      =1   2710              JMP     FILLOOP
EB01 22        =1   2711      FILL1:  RET
               =1   2712 +1   $EJECT
```

```
LOC  OBJ                   SOURCE

                   =1      ;****************************************************************
                   =1      ;
                   =1      ;    NAME: DISMEM
                   =1      ;
                   =1      ;    ABSTRACT: This routine displays the data values of the selected
                   =1      ;       memory partition to the console.
                   =1      ;
                   =1      ;    INPUTS: PNTLOW, PNTHGH, PARTIT_HI_LOW, PARTIT_HI_HIGH
                   =1      ;
                   =1      ;    OUTPUTS: None
                   =1      ;
                   =1      ;    VARIABLES MODIFIED: COUNTR, A, DPTR, PARAM1, PARAM2
                   =1      ;
                   =1      ;    ERROR EXITS: None
                   =1      ;
                   =1      ;    SUBROUTINES ACCESSED DIRECTLY: INEWLINE, IDISPLAY_TOKEN, SPACCO,
                   =1      ;       ILSTWRD, ICO, IFETCH, ILSTBYT, IWAIT_FOR_USER, ICONTINUATION_LINE
                   =1      ;
                   =1      ;
                   =1      ;****************************************************************
EB02 755D01        =1      DISMEM: MOV     COUNTR,#1               ;Load counter with 1
EB05 155D          =1      DISLOP: DEC     COUNTR
EB07 E55D          =1              MOV     A,COUNTR
EB09 701E          =1              JNZ     DISFET                  ;Jump to DISFET if counter is not zero
EB0B 12E717        =1              CALL    INEWLINE
EB0E E546          =1              MOV     A,SELECT                ;Move selector into ACC
EB10 90EB51        =1              MOV     DPTR,#LAB23             ;Load DPTR with base of table
EB13 93            =1              MOVC    A,@A+DPTR
EB14 FA            =1              MOV     PARAM1,A                ;Setup for DISPLAY_TOKEN
EB15 5112          =1              CALL    IDISPLAY_TOKEN          ;Output token
EB17 12E5E6        =1              CALL    SPACCO                  ;Output space
EB1A AB45          =1              MOV     PARAM2,PNTLOW
 B1C AA44          =1              MOV     PARAM1,PNTHGH           ;Set-up for ILSTWRD
 B1E 12E7F4        =1              CALL    ILSTWRD                 ;Output address
 B21 7A3D          =1              MOV     PARAM1,#'='
EB23 12E5E8        =1              CALL    ICO                     ;Output an equal sign
EB26 755D04        =1              MOV     COUNTR,#4               ;Load counter with 4
EB29 12E66B        =1      DISFET: CALL    IFETCH                  ;to get memory location
EB2C FA            =1              MOV     PARAM1,A                ;Set-up for ILSTBYT
EB2D 12E7F9        =1              CALL    ILSTBYT
EB30 E545          =1              MOV     A,PNTLOW
EB32 B55A08        =1              CJNE    A,PARTIT_HI_LOW,COUNT1  ;See if PARTIT_LO_LOW=EALOW
EB35 E544          =1              MOV     A,PNTHGH
EB37 B55903        =1              CJNE    A,PARTIT_HI_HIGH,COUNT1 ;See if PARTIT_LO_HIGH=EAHGH
EB3A 02E3B0        =1              JMP     IWAIT_FOR_USER
EB3D E55D          =1      COUNT1: MOV     A,COUNTR
EB3F B40108        =1              CJNE    A,#1,NTLAST             ;See if counter = 1,
EB42 12E65D        =1              CALL    ICONTINUATION_LINE
EB45 12E5C4        =1      NOWAIT: CALL    INC_PNT
 B48 80BB          =1              JMP     DISLOP                  ;Loop until PNT is > EA
 4A 7A2C           =1      NTLAST: MOV     PARAM1,#','
 4C 12E5E8         =1              CALL    ICO                     ;To output a comma
 B4F 80F4          =1              JMP     NOWAIT
                   =1
EB51 80            =1      LAB23:  DB      CBYTE_TOKE
```

```
LOC  OBJ            LINE     SOURCE

EB52 81        =1  2768              DB      RBYTE_TOKE
EB53 82        =1  2769              DB      DBYTE_TOKE
EB54 00        =1  2770              DB      00
EB55 84        =1  2771              DB      RBIT_TOKE
EB56 00        =1  2772              DB      00
EB57 86        =1  2773              DB      XBYTE_TOKE
               =1  2774 +1  $EJECT
```

```
LOC  OBJ         SOURCE

                 ;******************************************************************
                 ;
                 ;     NAME: BMOVE
                 ;
                 ;     ABSTRACT: This routine will transfer CBYTE type memory from
                 ;        a specific location to another location in blocks of contiguous
                 ;        code.  It does not relocate addresses and it is possible
                 ;        to lose code by writing a block over the TOP address.  The
                 ;        pointer direction is changed depending on the direction of
                 ;        the move so that no change to the data will occur if the
                 ;        destination and source blocks overlap.
                 ;
                 ;     INPUTS: SELECT, PARTIT_HI_LOW, PARTIT_HI_HIGH, LENGTH_LOW,
                 ;        LENGTH_HIGH, PARTIT_LO_LOW, PARTIT_LO_HIGH
                 ;
                 ;     OUTPUTS: Memory which was supposed to be accessed by the
                 ;        command typed in at the console.
                 ;
                 ;     VARIABLES MODIFIED: A, ERRNUM, C, PCNTLO, PCNTHI, PNTLOW, C,
                 ;        PARAM1, PNTHGH
                 ;
                 ;     ERROR EXITS: 18H (CBYTE TYPE ONLY)
                 ;
                 ;     SUBROUTINES ACCESSED DIRECTLY: IGETNUM, SWAP_POINTERS, IFETCH,
                 ;        DEC_PNT, ISTORE
                 ;
                 ;
                 ;******************************************************************

EB58 E546        BMOVE:  MOV     A,SELECT
EB5A 754318              MOV     ERRNUM,#18H                       ;CBYTE type only
EB5D B40077              CJNE    A,#(CBYTE_TOKE AND 7),ERRMOD
EB60 12E769              CALL    IGETNUM
EB63 854A62              MOV     PCNTLO,VALLOW
EB66 854961              MOV     PCNTHI,VALHGH
EB69 C3                  CLR     C
EB6A E545                MOV     A,PNTLOW
EB6C 9562                SUBB    A,PCNTLO
EB6E E544                MOV     A,PNTHGH
EB70 9561                SUBB    A,PCNTHI
EB72 4032                JC      DOWN_MOVE
EB74 855A45              MOV     PNTLOW,PARTIT_HI_LOW
EB77 855944              MOV     PNTHGH,PARTIT_HI_HIGH
EB7A E562                MOV     A,PCNTLO
EB7C 2564                ADD     A,LENGTH_LOW
EB7E F562                MOV     PCNTLO,A
EB80 E561                MOV     A,PCNTHI
EB82 3563                ADDC    A,LENGTH_HIGH
EB84 F561                MOV     PCNTHI,A
EB86 12E5D7      UP_MOVE:CALL    SWAP_POINTERS
EB89 12E66B              CALL    IFETCH
EB8C FA                  MOV     PARAM1,A
EB8D 12E5CD              CALL    DEC_PNT
EB90 12E5D7              CALL    SWAP_POINTERS
EB93 12E672              CALL    ISTORE
```

```
LOC  OBJ            LINE     SOURCE

EB96 C3         =1  2830             CLR     C
EB97 E558       =1  2831             MOV     A,PARTIT_LO_LOW
EB99 9545       =1  2832             SUBB    A,PNTLOW
EB9B E557       =1  2833             MOV     A,PARTIT_LO_HIGH
EB9D 9544       =1  2834             SUBB    A,PNTHGH
EB9F 5025       =1  2835             JNC     BEND
EBA1 12E5CD     =1  2836             CALL    DEC_PNT
EBA4 80E0       =1  2837             JMP     UP_MOVE
                =1  2838     DOWN_MOVE:
EBA6 12E5D7     =1  2839             CALL    SWAP_POINTERS
EBA9 12E66B     =1  2840             CALL    IFETCH
EBAC FA         =1  2841             MOV     PARAM1,A
EBAD 12E5C4     =1  2842             CALL    INC_PNT
EBB0 12E5D7     =1  2843             CALL    SWAP_POINTERS
EBB3 12E672     =1  2844             CALL    ISTORE
EBB6 C3         =1  2845             CLR     C
EBB7 E545       =1  2846             MOV     A,PNTLOW
EBB9 955A       =1  2847             SUBB    A,PARTIT_HI_LOW
EBBB E544       =1  2848             MOV     A,PNTHGH
EBBD 9559       =1  2849             SUBB    A,PARTIT_HI_HIGH
EBBF 5005       =1  2850             JNC     BEND
EBC1 12E5C4     =1  2851             CALL    INC_PNT
EBC4 80E0       =1  2852             JMP     DOWN_MOVE
EBC6 22         =1  2853     BEND:   RET
                =1  2854 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                    =1       ;*****************************************************************
                    =1       ;
                    =1       ;     NAME: BR_CMD
                    =1       ;
                    =1       ;     ABSTRACT: This routine checks a token to see if it is a
                    =1       ;        breakpoint display or change.  If it is change, it sets the pamameters
                    =1       ;        of the range and clears or sets the breakpoints requested. (ABR is
                    =1       ;        a change only command). If it is a display command, each breakpoint
                    =1       ;        is output to the console. Reset is the default condition.
                    =1       ;        If the token is BR, the entire breakpoint RAM is cleared and then
                    =1       ;        breakpoints are added. If it is ABR, they are added without clearing
                    =1       ;        RAM first.
                    =1       ;
                    =1       ;     INPUTS: TOKSTR
                    =1       ;
                    =1       ;     OUTPUTS: Bits within the breakpoint hardware register.
                    =1       ;
                    =1       ;     VARIABLES MODIFIED: TOKSAV, A, ERRNUM, PARAM1, PARAM2, PARAM3, PARAM4,
                    =1       ;        LINE_START, POINTO, PNTLOW, PNTHGH, DPTR, VPC_LOW, VPC_HIGH,
                    =1       ;        ANY_BR_FLAG, FIRST_FLAG
                    =1       ;
                    =1       ;     ERROR EXITS: 19H (DISPLAY ONLY COMMAND)
                    =1       ;                  05H (EQUAL OR RETURN EXPECTED)
                    =1       ;                  0CH (NUMBER OR RESET REQUIRED)
                    =1       ;
                    =1       ;     SUBROUTINES ACCESSED DIRECTLY: IGETOKE, IERROR, IGET_PART, IEOL_CHECK
                    =1       ;        IGETEOL, LSSEQL, IDISPLAY_TOKEN, IWAIT_FOR_USER, INC_PNT,
                    =1       ;        ICONTINUATION_LINE, ILSTWRD, SPACCO, INEWLINE, ICO, IERROR
                    =1       ;        BRK_LINE_HDR, SETBRK, CLRBRK
                    =1       ;
                    =1       ;
                    =1       ;*****************************************************************
EBC7 85485B         =1       BR_CMD: MOV     TOKSAV,TOKSTR           ;Save last token for comparison
EBCA 11BC           =1               CALL    IGETOKE                 ;Get next token
EBCC B4070B         =1               CJNE    A,#EOL_TOKE,EQLMOD      ;Check if token is end of line
EBCF E55B           =1               MOV     A,TOKSAV                ;Move last token into ACC
EBD1 B4884D         =1               CJNE    A,#ABR_TOKE,LSTBRK      ;Jump to list mod if not ABR token
EBD4 754319         =1               MOV     ERRNUM,#19H             ;ABR is not a displayable command
EBD7 02E3E4         =1       ERRMOD: JMP     IERROR
EBDA 754305         =1       EQLMOD: MOV     ERRNUM,#05H             ;Equal or return expected
EBDD B404F7         =1               CJNE    A,#EQUAL_TOKE,ERRMOD    ;Error if '=' not entered here
EBE0 11BC           =1               CALL    IGETOKE
EBE2 B40130         =1               CJNE    A,#NUMBER_TOKE,RSTMOD
EBE5 E55B           =1               MOV     A,TOKSAV                ;Recall last token entered
EBE7 B48903         =1               CJNE    A,#BR_TOKE,NUMMOD       ;Check if it was break token
EBEA 12ECE1         =1               CALL    CLRBRK                  ;Clear breakpoints
EBED 12E7A2         =1       NUMMOD: CALL    IGET_PART
EBF0 12ECF2         =1               CALL    SETBRK
EBF3 E548           =1               MOV     A,TOKSTR                ;Recall present token
EBF5 B4021A         =1               CJNE    A,#COMMA_TOKE,ENDMOD    ;Check if comma was entered
EBF8 11BC           =1               CALL    IGETOKE
EBFA B407F0         =1               CJNE    A,#EOL_TOKE,NUMMOD      ;Check for EOL
EBFD 755204         =1               MOV     LINE_START,#04H
EC00 7824           =1               MOV     POINTO,#LINBUF
EC02 7641           =1               MOV     @POINTO,#'A'
```

```
LOC  OBJ                SOURCE

EC04 08          =1               INC     POINTO
EC05 7642        =1               MOV     @POINTO,#'B'
EC07 08          =1               INC     POINTO
EC08 7652        =1               MOV     @POINTO,#'R'
EC0A 08          =1               INC     POINTO
EC0B 763D        =1               MOV     @POINTO,#'='
EC0D 11BC        =1               CALL    IGETOKE
EC0F B407DB      =1               CJNE    A,#EOL_TOKE,NUMMOD
EC12 02E5BB      =1       ENDMOD: JMP     IEOL_CHECK
EC15 75430C      =1       RSTMOD: MOV     ERRNUM,#0CH             ;Number or reset required
EC18 B40EBC      =1               CJNE    A,#RESET_TOKE,ERRMOD    ;Check for reset entered
EC1B 12ECE1      =1               CALL    CLRBRK
EC1E 02E773      =1               JMP     IGETEOL
                 =1       ;*****************************************************************
EC21 E4          =1       LSTBRK: CLR     A
EC22 F545        =1               MOV     PNTLOW,A                ;Clear low byte of break pointer
EC24 F544        =1               MOV     PNTHGH,A                ;Clear high byte of break pointer
EC26 C202        =1               CLR     ANY_BR_FLAG
EC28 D203        =1               SETB    FIRST_FLAG
EC2A 90C000      =1       LAB2:   MOV     DPTR,#BRKOFF
EC2D 7A1F        =1               MOV     PARAM1,#MAXHGH
EC2F 7BFF        =1               MOV     PARAM2,#MAXLOW
EC31 AC44        =1               MOV     PARAM3,PNTHGH
EC33 AD45        =1               MOV     PARAM4,PNTLOW           ;Set up for LSSEQL test
EC35 12E74B      =1               CALL    LSSEQL                  ;Check that P??? <= MAX???
EC38 400D        =1               JC      LAB5B                   ;Exit if greater than
EC3A 200207      =1               JB      ANY_BR_FLAG,BRKEND      ;If any breakpoints were displayed
                 =1                                               ;don't display reset
EC3D 12ECD5      =1               CALL    BRK_LINE_HDR
EC40 7A0E        =1               MOV     PARAM1,#RESET_TOKE
EC42 5112        =1               CALL    IDISPLAY_TOKEN
EC44 02E3B0      =1       BRKEND: JMP     IWAIT_FOR_USER
EC47 E545        =1       LAB5B:  MOV     A,PNTLOW                ;Load ACC with break pointer low addr
EC49 2582        =1               ADD     A,DPL                   ;Add low addr of break to pointer
EC4B F582        =1               MOV     DPL,A                   ;Put new low addr back into DPL
EC4D 5002        =1               JNC     LAB5A
EC4F 0583        =1               INC     DPH                     ;Increment DPH if DPL had a carry
EC51 E544        =1       LAB5A:  MOV     A,PNTHGH
EC53 2583        =1               ADD     A,DPH
EC55 F583        =1               MOV     DPH,A
EC57 E0          =1               MOVX    A,@DPTR                 ;Load ACC with external RAM memory
EC58 30E005      =1               JNB     ACC.0,LAB3              ;Branch if break is set.
EC5B 12E5C4      =1               CALL    INC_PNT
EC5E 80CA        =1               JMP     LAB2
                 =1
EC60 85455E      =1       LAB3:   MOV     VPC_LOW,PNTLOW          ;Save break pointer low
EC63 85445F      =1               MOV     VPC_HIGH,PNTHGH         ;Save break pointer high
EC66 D202        =1               SETB    ANY_BR_FLAG
EC68 90C000      =1       BK1LOP: MOV     DPTR,#BRKOFF
EC6B AC44        =1               MOV     PARAM3,PNTHGH
EC6D AD45        =1               MOV     PARAM4,PNTLOW           ;Set up for LSSEQL
EC6F 12E74B      =1               CALL    LSSEQL                  ;Check that P??? <= MAX???
EC72 5019        =1               JNC     LSTOUT                  ;Jump to LSTOUT if greater than
EC74 E545        =1               MOV     A,PNTLOW                ;Load ACC with low addr of break pointer
EC76 2582        =1               ADD     A,DPL                   ;Add break RAM low addr offset to pointer low
```

```
LOC  OBJ            LINE     SOURCE

EC78 F582       =1  2965             MOV    DPL,A                  ;Put new addr back into DPL
EC7A 5002       =1  2966             JNC    LAB6A
EC7C 0583       =1  2967             INC    DPH                    ;Increment DPH if DPL produced a carry
EC7E E544       =1  2968     LAB6A:  MOV    A,PNTHGH
EC80 2583       =1  2969             ADD    A,DPH
EC82 F583       =1  2970             MOV    DPH,A
EC84 E0         =1  2971             MOVX   A,@DPTR                ;Load ACC with data in break RAM
EC85 20E005     =1  2972             JB     ACC.0,LSTOUT           ;Branch if break is off.
EC88 12E5C4     =1  2973             CALL   INC_PNT
EC8B 80DB       =1  2974             JMP    BK1LOP                 ;Loop until 0 occurs or end of break RAM
EC8D AC5F       =1  2975     LSTOUT: MOV    PARAM3,VPC_HIGH
EC8F AD5E       =1  2976             MOV    PARAM4,VPC_LOW         ;Set up for LSSEQL
EC91 12E74B     =1  2977             CALL   LSSEQL                 ;Check that SA??? <= MAX???
EC94 5094       =1  2978             JNC    LAB2                   ;Jump to BRKO and exit if true
EC96 200303     =1  2979             JB     FIRST_FLAG,LB_10
EC99 12E65D     =1  2980             CALL   ICONTINUATION_LINE
 C9C 12ECD5     =1  2981     LB_10:  CALL   BRK_LINE_HDR
 C9F C203       =1  2982             CLR    FIRST_FLAG
ECA1 AA5F       =1  2983             MOV    PARAM1,VPC_HIGH
ECA3 AB5E       =1  2984             MOV    PARAM2,VPC_LOW         ;Set up for ILSTWRD
ECA5 12E7F4     =1  2985             CALL   ILSTWRD                ;Output starting addr of one's in BRK RAM
ECA8 055E       =1  2986             INC    VPC_LOW                ;Increment starting address low
ECAA E55E       =1  2987             MOV    A,VPC_LOW
ECAC 7002       =1  2988             JNZ    LAB7                   ;Check for rollover
ECAE 055F       =1  2989             INC    VPC_HIGH               ;Increment starting addr high if true
ECB0 E55F       =1  2990     LAB7:   MOV    A,VPC_HIGH
ECB2 B54407     =1  2991             CJNE   A,PNTHGH,OUTOKE        ;Jump to OUTOKE if VPC_HIGH+1 <> PNTHGH
ECB5 E55E       =1  2992             MOV    A,VPC_LOW
ECB7 B54502     =1  2993             CJNE   A,PNTLOW,OUTOKE        ;Jump to OUTOKE if VPC_LOW+1 <> PNTLOW
ECBA 812A       =1  2994             JMP    LAB2                   ;Go process the end of line.
                =1  2995
ECBC 12E5E6     =1  2996     OUTOKE: CALL   SPACCO
 CBF 7A0D       =1  2997             MOV    PARAM1,#TO_TOKE
  C1 5112       =1  2998             CALL   IDISPLAY_TOKEN         ;Call display_token(to_toke).
  C3 12E5E6     =1  2999             CALL   SPACCO
 CC6 E545       =1  3000             MOV    A,PNTLOW               ;Load ACC with break pointer low
ECC8 14         =1  3001             DEC    A                      ;Decrement break pointer low
ECC9 FB         =1  3002             MOV    PARAM2,A               ;Set up for ILSTWRD
ECCA F4         =1  3003             CPL    A
ECCB AA44       =1  3004             MOV    PARAM1,PNTHGH          ;Set up for ILSTWRD
ECCD 7001       =1  3005             JNZ    LAB8                   ;Check for roll over
ECCF 1A         =1  3006             DEC    PARAM1                 ;Decrement break pointer high if true
ECD0 12E7F4     =1  3007     LAB8:   CALL   ILSTWRD
                =1  3008     ;
ECD3 812A       =1  3009             JMP    LAB2                   ;Continue the display of break RAM.
                =1  3010                                            ;as soon as the user hits a character
                =1  3011     BRK_LINE_HDR:
ECD5 12E717     =1  3012             CALL   INEWLINE
ECD8 7A89       =1  3013             MOV    PARAM1,#BR_TOKE
ECDA 5112       =1  3014             CALL   IDISPLAY_TOKEN
 CDC 7A3D       =1  3015             MOV    PARAM1,#'='
  DE 02E5E8     =1  3016             JMP    ICO
                =1  3017 +1  $eject
```

```
LOC  OBJ            LINE     SOURCE

               =1   3018
               =1   3019     ;*****END OF LSTBRK*****
               =1   3020
ECE1 7AFF      =1   3021     CLRBRK: MOV     PARAM1,#MAXLOW          ;Load PARAM1 with size of break RAM,low 8 bits
ECE3 7B20      =1   3022             MOV     PARAM2,#(MAXHGH+1)      ;Load PARAM2 with size of break RAM+1,high bits
ECE5 90C000    =1   3023             MOV     DPTR,#BRKOFF            ;Load DPTR with break RAM offset
ECE8 7401      =1   3024             MOV     A,#01H                  ;To clear the break condition.
ECEA F0        =1   3025     CLRLOP: MOVX    @DPTR,A                 ;Fill break RAM
ECEB A3        =1   3026             INC     DPTR                    ;Increment pointer at break RAM
ECEC DAFC      =1   3027             DJNZ    PARAM1,CLRLOP           ;Repeat loop until PARAM1=0
ECEE F0        =1   3028             MOVX    @DPTR,A                 ;Once more for PARAM1=0
ECEF DBF9      =1   3029             DJNZ    PARAM2,CLRLOP           ;Continue loop until PARAM2=0
ECF1 22        =1   3030             RET                             ;Exit from CLRBRK
               =1   3031     ;*****END OF CLRBRK*****
               =1   3032
ECF2 C3        =1   3033     SETBRK: CLR     C
ECF3 E55A      =1   3034             MOV     A,PARTIT_HI_LOW         ;Load ACC with ending addr low
ECF5 9558      =1   3035             SUBB    A,PARTIT_LO_LOW         ;To obtain number of locations to set
ECF7 F582      =1   3036             MOV     DPL,A                   ;Save low number in PARAM4
ECF9 E559      =1   3037             MOV     A,PARTIT_HI_HIGH        ;Load ACC with ending addr high
ECFB 20E726    =1   3038             JB      ACC.7,BRKERR
ECFE 9557      =1   3039             SUBB    A,PARTIT_LO_HIGH        ;Subtract starting addr high from ending addr
ED00 F583      =1   3040             MOV     DPH,A                   ;Save high break count in PARAM3
ED02 A3        =1   3041             INC     DPTR
ED03 0583      =1   3042             INC     DPH
ED05 AA83      =1   3043             MOV     PARAM1,DPH
ED07 AB82      =1   3044             MOV     PARAM2,DPL
ED09 90C000    =1   3045             MOV     DPTR,#BRKOFF
ED0C E557      =1   3046             MOV     A,PARTIT_LO_HIGH
ED0E 541F      =1   3047             ANL     A,#MAXHGH
ED10 FD        =1   3048             MOV     TEMP,A
ED11 E558      =1   3049             MOV     A,PARTIT_LO_LOW         ;Put starting addr low in ACC
ED13 2582      =1   3050             ADD     A,DPL                   ;Add break offset low
ED15 F582      =1   3051             MOV     DPL,A                   ;Put back into data pointer
ED17 ED        =1   3052             MOV     A,TEMP                  ;Load ACC with starting addr high
ED18 3583      =1   3053             ADDC    A,DPH                   ;Add break offset high
ED1A F583      =1   3054             MOV     DPH,A                   ;Load DPH with starting addr high + offset
ED1C E4        =1   3055     OUT1BK: CLR     A                       ;To output O'S
ED1D F0        =1   3056             MOVX    @DPTR,A                 ;Load break RAM
ED1E A3        =1   3057             INC     DPTR                    ;Increment break RAM pointer
ED1F DBFB      =1   3058             DJNZ    PARAM2,OUT1BK           ;Loop until count low=0
ED21 DAF9      =1   3059             DJNZ    PARAM1,OUT1BK           ;Loop until PARAM3=0
ED23 22        =1   3060             RET                             ;Exit from SETBRK
               =1   3061     ;*****END OF SETBRK*****
               =1   3062
ED24 75430D    =1   3063     BRKERR: MOV     ERRNUM,#0DH             ;7 is the error number for
               =1   3064                                             ;break range low > range high
ED27 02E3E4    =1   3065             JMP     IERROR                  ;Exit from break routine on error
               =1   3066
               =1   3067
               =1   3068 +1  $EJECT
```

```
LOC  OBJ           LINE      SOURCE

               =1  3069      ;******************************************************************
               =1  3070      ;
               =1  3071      ;     NAME: ACC_CMD/ PSW_CMD/ SP_CMD/ B_CMD
               =1  3072      ;
               =1  3073      ;     ABSTRACT: Displays or modifies the byte which is referenced
               =1  3074      ;        by the user register images passed to it.
               =1  3075      ;
               =1  3076      ;     INPUTS: None
               =1  3077      ;
               =1  3078      ;     OUTPUTS: Users version of the PC, DPTR, TMO, TM1
               =1  3079      ;
               =1  3080      ;     VARIABLES MODIFIED: PNTLOW, PNTHGH, SELECT, PARAM1
               =1  3081      ;
               =1  3082      ;     ERROR EXITS: None
               =1  3083      ;
               =1  3084      ;     SUBROUTINES ACCESSED DIRECTLY: ISIT_DISPLAY, IFETCH, ILSTBYT,
               =1  3085      ;        IWAIT_FOR_USER, ISTORE, KEY_BYTES
               =1  3086      ;
               =1  3087      ;
               =1  3088      ;******************************************************************
               =1  3089      ACC_CMD:
ED2A 7545E0    =1  3090              MOV     PNTLOW,#ACC
ED2D 02ED42    =1  3091              JMP     KEY_BYTE
               =1  3092      ;******************************************************************
               =1  3093      PSW_CMD:
ED30 7545D0    =1  3094              MOV     PNTLOW,#PSW
ED33 02ED42    =1  3095              JMP     KEY_BYTE
               =1  3096      ;******************************************************************
               =1  3097      SP_CMD:
ED36 754581    =1  3098              MOV     PNTLOW,#SP
ED39 02ED42    =1  3099              JMP     KEY_BYTE
               =1  3100      ;******************************************************************
               =1  3101      B_CMD:
 D3C 7545F0    =1  3102              MOV     PNTLOW,#B
 D3F 02ED42    =1  3103              JMP     KEY_BYTE
               =1  3104      ;******************************************************************
               =1  3105      KEY_BYTE:
ED42 12E784    =1  3106              CALL    ISIT_DISPLAY
ED45 754400    =1  3107              MOV     PNTHGH,#00H
ED48 754601    =1  3108              MOV     SELECT,#(RBYTE_TOKE AND 07H)    ;Set-up for fetch
ED4B 500A      =1  3109              JNC     CHANGE
ED4D 12E66B    =1  3110              CALL    IFETCH
ED50 FA        =1  3111              MOV     PARAM1,A                        ;Call ILSTBYT (result) to display it
ED51 12E7F9    =1  3112              CALL    ILSTBYT
ED54 02E3B0    =1  3113              JMP     IWAIT_FOR_USER
               =1  3114      CHANGE:
ED57 12E769    =1  3115              CALL    IGETNUM                         ;Get the numeric parameter
ED5A AA4A      =1  3116              MOV     PARAM1,VALLOW
ED5C 02E672    =1  3117              JMP     ISTORE
               =1  3118 +1   $EJECT
```

```
LOC  OBJ            LINE     SOURCE
               =1   3119     ;*****************************************************************
               =1   3120     ;
               =1   3121     ;     NAME: PC_CMD/ DPTR_CMD/ TMO_CMD/ TM1_CMD
               =1   3122     ;
               =1   3123     ;     ABSTRACT:  Decodes and exeutes those commands which display or alter
               =1   3124     ;        sixteen bit variables which have unique keywords to identify
               =1   3125     ;        them.
               =1   3126     ;
               =1   3127     ;     INPUTS: None
               =1   3128     ;
               =1   3129     ;     OUTPUTS: Users version of the PC, DPTR, TMO and TM1
               =1   3130     ;
               =1   3131     ;     VARIABLES MODIFIED: PARAM1, PARAM2, PNTLOW, TEMP_LOW, PNTHGH, A
               =1   3132     ;
               =1   3133     ;     ERROR EXITS: None
               =1   3134     ;
               =1   3135     ;     SUBROUTINES ACCESSED DIRECTLY: ISIT_DISPLAY, READ_PC, ILSTWRD,
               =1   3136     ;        WRITE_PC, IFETCH, ISTORE, IGETEOL, IGETNUM, IWAIT_FOR_USER,
               =1   3137     ;        KEYWORD_DISPLAY
               =1   3138     ;
               =1   3139     ;
               =1   3140     ;*****************************************************************
               =1   3141     PC_CMD:
ED5F 12E784    =1   3142             CALL    ISIT_DISPLAY
ED62 500C      =1   3143             JNC     PC_CHA
ED64 12EF9D    =1   3144             CALL    READ_PC                 ;Get the user program counter.
ED67 FB        =1   3145             MOV     PARAM2,A                ;And set up parameters to display it.
ED68 AAF0      =1   3146             MOV     PARAM1,B
ED6A 12E7F4    =1   3147             CALL    ILSTWRD
ED6D 02E3B0    =1   3148             JMP     IWAIT_FOR_USER
               =1   3149     PC_CHA:
ED70 12E769    =1   3150             CALL    IGETNUM
ED73 AA49      =1   3151             MOV     PARAM1,VALHGH
ED75 AB4A      =1   3152             MOV     PARAM2,VALLOW
ED77 12EFA8    =1   3153             CALL    WRITE_PC
ED7A 02E773    =1   3154             JMP     IGETEOL
               =1   3155     ;*****************************************************************
               =1   3156     DPTR_CMD:
ED7D 754583    =1   3157             MOV     PNTLOW,#DPH
ED80 754782    =1   3158             MOV     TEMP_LOW,#DPL
ED83 02ED95    =1   3159             JMP     KEYWORD_DISPLAY
               =1   3160     ;*****************************************************************
               =1   3161     TMO_CMD:
ED86 75458C    =1   3162             MOV     PNTLOW,#THO
ED89 75478A    =1   3163             MOV     TEMP_LOW,#TLO
ED8C 02ED95    =1   3164             JMP     KEYWORD_DISPLAY
               =1   3165     ;*****************************************************************
               =1   3166     TM1_CMD:
ED8F 75458D    =1   3167             MOV     PNTLOW,#TH1
ED92 75478B    =1   3168             MOV     TEMP_LOW,#TL1
               =1   3169     ;*****************************************************************
               =1   3170     KEYWORD_DISPLAY:
ED95 12E784    =1   3171             CALL    ISIT_DISPLAY
ED98 754601    =1   3172             MOV     SELECT,#(RBYTE_TOKE AND 07H)
ED9B 754400    =1   3173             MOV     PNTHGH,#0
```

```
LOC  OBJ            LINE     SOURCE

ED9E 5013       =1  3174             JNC     WCHANGE
EDA0 12E66B     =1  3175             CALL    IFETCH
EDA3 C547       =1  3176             XCH     A,TEMP_LOW
EDA5 F545       =1  3177             MOV     PNTLOW,A
EDA7 12E66B     =1  3178             CALL    IFETCH
EDAA FB         =1  3179             MOV     PARAM2,A
EDAB AA47       =1  3180             MOV     PARAM1,TEMP_LOW
EDAD 12E7F4     =1  3181             CALL    ILSTWRD
EDB0 02E3B0     =1  3182             JMP     IWAIT_FOR_USER          ;Wait for CR then start the monitor.
                =1  3183     WCHANGE:
EDB3 12E769     =1  3184             CALL    IGETNUM                 ;If it is, get the data to be loaded.
EDB6 AA49       =1  3185             MOV     PARAM1,VALHGH
EDB8 12E672     =1  3186             CALL    ISTORE
EDBB 854745     =1  3187             MOV     PNTLOW,TEMP_LOW
EDBE AA4A       =1  3188             MOV     PARAM1,VALLOW
EDC0 12E672     =1  3189             CALL    ISTORE
EDC3 02E773     =1  3190             JMP     IGETEOL                 ;Process end of line and return to the
                =1  3191     ;*******************************************************************
                    3192 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                    3193 +1  $INCLUDE(:F1:XQT.INC)
                 =1 3194     ;*****************************************************************
                 =1 3195     ;
                 =1 3196     ;    NAME: (I)BREAK
                 =1 3197     ;
                 =1 3198     ;    ABSTRACT: Control is transferred to this point when a break
                 =1 3199     ;       interrupt occurs.  The current user status is saved in the
                 =1 3200     ;       page of external RAM starting at 'RAMOFF' and control then
                 =1 3201     ;       passes to one of the return routines, STEP return and RUN
                 =1 3202     ;       return.
                 =1 3203     ;
                 =1 3204     ;    INPUTS: BREAK_STATUS, MON_FLAGS
                 =1 3205     ;
                 =1 3206     ;    OUTPUTS: LINE_START, CAUSE_IMAGE, UPI_DATA_IMAGE, all the users
                 =1 3207     ;       RAM and register image area.
                 =1 3208     ;
                 =1 3209     ;    VARIABLES MODIFIED: DPTR, SP, A, IE, POINTO, CAUSE_IMAGE,
                 =1 3210     ;       ERRNUM, C, B, PARAM1, LINE_START, UPI_DATA_IMAGE
                 =1 3211     ;
                 =1 3212     ;    ERROR EXITS: 16H (EXECUTION OVER VECTOR AT LOCATION 3)
                 =1 3213     ;
                 =1 3214     ;    SUBROUTINES ACCESSED DIRECTLY: ICSTS, UPI_IN, WRITE_PC, READ_PC,
                 =1 3215     ;       INIT_IO, UPI_OUT, SET_BAUD, UPI_CMD, SIGN_ON, STEP51_RETURN,
                 =1 3216     ;       UNBREAK, RUN_USER_RETURN
                 =1 3217     ;
                 =1 3218     ;
                 =1 3219     ;*****************************************************************
EDC6 C082        =1 3220     IBREAK: PUSH    DPL                          ;Save DPTR in the user stack.
EDC8 C083        =1 3221             PUSH    DPH
EDCA 90B0E0      =1 3222             MOV     DPTR,#(RAMOFF+ACC)
EDCD F0          =1 3223             MOVX    @DPTR,A                      ;Save user ACC.
EDCE 758283      =1 3224             MOV     DPL,#DPH
EDD1 D0E0        =1 3225             POP     ACC
EDD3 F0          =1 3226             MOVX    @DPTR,A                      ;Move user DPH from the stack to save area.
EDD4 1582        =1 3227             DEC     DPL
EDD6 D0E0        =1 3228             POP     ACC
EDD8 F0          =1 3229             MOVX    @DPTR,A                      ;Move user DPL from the stack to save area.
EDD9 7582A8      =1 3230             MOV     DPL,#IE                      ;Save the special function registers.
EDDC E5A8        =1 3231             MOV     A,IE
EDDE F0          =1 3232             MOVX    @DPTR,A
EDDF 75A800      =1 3233             MOV     IE,#00H
EDE2 758288      =1 3234             MOV     DPL,#TCON
EDE5 E588        =1 3235             MOV     A,TCON
EDE7 F0          =1 3236             MOVX    @DPTR,A
EDE8 758800      =1 3237             MOV     TCON,#0
EDEB 7582F0      =1 3238             MOV     DPL,#B                       ;Start with 'B'.
EDEE E5F0        =1 3239             MOV     A,B
EDF0 F0          =1 3240             MOVX    @DPTR,A
EDF1 7582B8      =1 3241             MOV     DPL,#IP
EDF4 E5B8        =1 3242             MOV     A,IP
EDF6 F0          =1 3243             MOVX    @DPTR,A
EDF7 758290      =1 3244             MOV     DPL,#P1
EDFA E590        =1 3245             MOV     A,P1
EDFC F0          =1 3246             MOVX    @DPTR,A
EDFD 7582B0      =1 3247             MOV     DPL,#P3
```

```
LOC  OBJ            LINE      SOURCE

EE00 E5B0       =1  3248                MOV     A,P3
EE02 F0         =1  3249                MOVX    @DPTR,A
EE03 7582D0     =1  3250                MOV     DPL,#PSW
EE06 E5D0       =1  3251                MOV     A,PSW
EE08 F0         =1  3252                MOVX    @DPTR,A
EE09 758298     =1  3253                MOV     DPL,#SCON
EE0C E598       =1  3254                MOV     A,SCON
EE0E F0         =1  3255                MOVX    @DPTR,A
EE0F 758281     =1  3256                MOV     DPL,#SP
EE12 E581       =1  3257                MOV     A,SP
EE14 14         =1  3258                DEC     A                       ;Compensate the SP for the break interrupt
EE15 14         =1  3259                DEC     A
EE16 F0         =1  3260                MOVX    @DPTR,A
EE17 75828C     =1  3261                MOV     DPL,#TH0
EE1A E58C       =1  3262                MOV     A,TH0
EE1C F0         =1  3263                MOVX    @DPTR,A
EE1D 75828D     =1  3264                MOV     DPL,#TH1
EE20 E58D       =1  3265                MOV     A,TH1
EE22 F0         =1  3266                MOVX    @DPTR,A
EE23 75828A     =1  3267                MOV     DPL,#TL0
EE26 E58A       =1  3268                MOV     A,TL0
EE28 F0         =1  3269                MOVX    @DPTR,A
EE29 75828B     =1  3270                MOV     DPL,#TL1
EE2C E58B       =1  3271                MOV     A,TL1
EE2E F0         =1  3272                MOVX    @DPTR,A
EE2F 758289     =1  3273                MOV     DPL,#TMOD
EE32 E589       =1  3274                MOV     A,TMOD
EE34 F0         =1  3275                MOVX    @DPTR,A                 ;Save the user internal RAM.
EE35 758200     =1  3276                MOV     DPL,#0                  ;Set DPTR to start of save area.
EE38 75D000     =1  3277                MOV     PSW,#0                  ;Select register bank 0.
EE3B E8         =1  3278                MOV     A,R0                    ;Save users R0 (our POINT0)
EE3C F0         =1  3279                MOVX    @DPTR,A
EE3D 7801       =1  3280                MOV     POINT0,#01H             ;Then save user RAM.
                =1  3281      BRK_LOOP:
EE3F A3         =1  3282                INC     DPTR
EE40 E6         =1  3283                MOV     A,@POINT0
EE41 F0         =1  3284                MOVX    @DPTR,A
EE42 08         =1  3285                INC     POINT0
EE43 B880F9     =1  3286                CJNE    POINT0,#128,BRK_LOOP
EE46 90B0FE     =1  3287                MOV     DPTR,#(RAMOFF+UPC+1)
EE49 D0E0       =1  3288                POP     ACC
EE4B F0         =1  3289                MOVX    @DPTR,A                 ;Get first byte of PC into 8155
EE4C 1582       =1  3290                DEC     DPL
EE4E D0E0       =1  3291                POP     ACC
EE50 F0         =1  3292                MOVX    @DPTR,A                 ;Save second byte of PC off stack
EE51 758107     =1  3293                MOV     SP,#STACK
EE54 7582FA     =1  3294                MOV     DPL,#MON_FLAGS
EE57 E0         =1  3295                MOVX    A,@DPTR
EE58 F520       =1  3296                MOV     20H,A                   ;Move the monitor flags storage area to the
                =1  3297                                                ;first eight bit locations.
EE5A 7582FB     =1  3298                MOV     DPL,#BREAK_STATUS
EE5D E0         =1  3299                MOVX    A,@DPTR                 ;See if break was invoked by the power
EE5E 6023       =1  3300                JZ      BREAK_CONTINUE          ;on and skip further checks if it was.
                =1  3301                                                ;If not continue.
EE60 90C000     =1  3302                MOV     DPTR,#BRKOFF            ;Find the cause of the break
```

```
LOC  OBJ                  SOURCE

EE63 E0          =1               MOVX     A,@DPTR
EE64 F560        =1               MOV      CAUSE_IMAGE,A
EE66 543C        =1               ANL      A,#03CH
EE68 7019        =1               JNZ      BREAK_CONTINUE
EE6A 12E602      =1               CALL     ICSTS                  ;No break set up-was it a keyboard entry?
EE6D 4009        =1               JC       BRKMORE
EE6F 754316      =1               MOV      ERRNUM,#16H            ;Execution over vector at loc 3
EE72 756004      =1               MOV      CAUSE_IMAGE,#4         ;Cause is guarded access.
EE75 02E3E4      =1               JMP      IERROR
                 =1       BRKMORE:
EE78 12E64C      =1               CALL     UPI_IN                 ;Else get the character
EE7B 547F        =1               ANL      A,#7FH
EE7D B41B63      =1               CJNE     A,#ESC,PRE_UNBREAK     ;Return to the user unless char is an ESCAPE.
EE80 756002      =1               MOV      CAUSE_IMAGE,#2         ;Cause is user abort.
                 =1       BREAK_CONTINUE:                         ;The interrupt is due to a valid break.
                 =1                                               ;Determine which one and reenter the
                 =1                                               ;monitor at the appropriate point.
EE83 75A800      =1               MOV      IE,#0                  ;Shut down all the interrupts while in the
EE86 758107      =1               MOV      SP,#STACK              ;Set up the monitor stack pointer
EE89 E560        =1               MOV      A,CAUSE_IMAGE
EE8B 20E409      =1               JB       ACC.4,BRK3             ;Always adjust for data break
EE8E 5428        =1               ANL      A,#28H
EE90 6015        =1               JZ       BRK4                   ;Bypass adjusting PC for any break
EE92 E560        =1               MOV      A,CAUSE_IMAGE          ;except PROG or  STEP
EE94 30E610      =1               JNB      ACC.6,BRK4             ;Check to see if NOP was forced on break.
                 =1                                               ;(i.e. PC is too big)
EE97 12EF9D      =1       BRK3:   CALL     READ_PC
EE9A C3          =1               CLR      C
EE9B 9401        =1               SUBB     A,#1
EE9D 5002        =1               JNC      BRK5
EE9F 15F0        =1               DEC      B
EEA1 FB          =1       BRK5:   MOV      PARAM2,A
EEA2 AAF0        =1               MOV      PARAM1,B
EEA4 12EFA8      =1               CALL     WRITE_PC
EEA7 12E386      =1       BRK4:   CALL     INIT_IO
EEAA 7A83        =1               MOV      PARAM1,#TOP_PORT
EEAC 12E625      =1               CALL     UPI_CMD
EEAF 7A00        =1               MOV      PARAM1,#0
EEB1 12E638      =1               CALL     UPI_OUT
EEB4 12E64C      =1               CALL     UPI_IN                 ;Clear UPIOBF
EEB7 12E386      =1               CALL     INIT_IO
EEBA 12F229      =1               CALL     SET_BAUD
EEBD A201        =1               MOV      C,LSTFLG
EEBF 755200      =1               MOV      LINE_START,#0
EEC2 E4          =1               CLR      A
EEC3 92E6        =1               MOV      ACC.6,C
EEC5 FA          =1               MOV      PARAM1,A
EEC6 12E625      =1               CALL     UPI_CMD
EEC9 90B0FB      =1               MOV      DPTR,#(RAMOFF+BREAK_STATUS)
EECC E0          =1               MOVX     A,@DPTR
EECD 7003        =1               JNZ      BRK1
EECF 02E2CC      =1               JMP      SIGN_ON
EED2 E560        =1       BRK1:   MOV      A,CAUSE_IMAGE
EED4 541E        =1               ANL      A,#1EH
EED6 6003        =1               JZ       BRK2                   ;Check for cause other than singlestep
```

```
LOC  OBJ            LINE     SOURCE

EED8 02F18E     =1  3358              JMP     RUN_USER_RETURN
EEDB E560       =1  3359     BRK2:    MOV     A,CAUSE_IMAGE
EEDD 30E503     =1  3360              JNB     ACC.5,PRE_UNBREAK       ;Reenter execution if not singlestep
EEE0 02F052     =1  3361              JMP     STEP51_RETURN           ;Return to the step command.
                =1  3362     PRE_UNBREAK:
EEE3 90B0F1     =1  3363              MOV     DPTR,#(RAMOFF+UPI_DATA_IMAGE)
EEE6 D2E7       =1  3364              SETB    ACC.7
EEE8 F0         =1  3365              MOVX    @DPTR,A                 ;escape
                =1  3366 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1  3367     ;*****************************************************************
                =1  3368     ;
                =1  3369     ;    NAME: UNBREAK
                =1  3370     ;
                =1  3371     ;    ABSTRACT: Restores the user status and starts execution of the
                =1  3372     ;       user program. CAUTION: This routine is position sensitive.
                =1  3373     ;       It is entered from BREAK as "in line" code.
                =1  3374     ;
                =1  3375     ;    INPUTS: All of the users registers and RAM images wil be used.,
                =1  3376     ;       TOP_STORE
                =1  3377     ;
                =1  3378     ;    OUTPUTS: MON_FLAGS
                =1  3379     ;
                =1  3380     ;    VARIABLES MODIFIED: A, DPTR, RO, B, PSW, SCON, SP, IP, THO,
                =1  3381     ;       TH1, TMOD, TCON, IE, IEO, ITO, PXO
                =1  3382     ;
                =1  3383     ;    ERROR EXITS: None
                =1  3384     ;
                =1  3385     ;    SUBROUTINES ACCESSED DIRECTLY: UPI_CMD, UPI_OUT
                =1  3386     ;
                =1  3387     ;*****************************************************************
EEE9 7A01       =1  3388     UNBREAK:MOV     PARAM1,#USART_MODE
EEEB 12E625     =1  3389             CALL    UPI_CMD
EEEE 7AFF       =1  3390             MOV     PARAM1,#0FFH
EEF0 12E638     =1  3391             CALL    UPI_OUT
EEF3 12E638     =1  3392             CALL    UPI_OUT               ;Output nulls to clr usart b/f reset in break
EEF6 7A83       =1  3393             MOV     PARAM1,#TOP_PORT
EEF8 12E625     =1  3394             CALL    UPI_CMD
EEFB 90B0F9     =1  3395             MOV     DPTR,#(RAMOFF+TOP_STORE)
EEFE E0         =1  3396             MOVX    A,@DPTR
EEFF FA         =1  3397             MOV     PARAM1,A
EF00 12E638     =1  3398             CALL    UPI_OUT
EF03 12E64C     =1  3399             CALL    UPI_IN                ;Clear UPIOBF
EF06 7A00       =1  3400             MOV     PARAM1,#SELECT_CON    ;Re-enable the console for I/O
EF08 12E625     =1  3401             CALL    UPI_CMD               ;then return
EF0B E520       =1  3402             MOV     A,20H                 ;Save the MON_FLAGS during execution.
EF0D 7582FA     =1  3403             MOV     DPL,#MON_FLAGS
EF10 F0         =1  3404             MOVX    @DPTR,A
EF11 787F       =1  3405             MOV     RO,#127               ;First restore the internal RAM.
EF13 75827F     =1  3406             MOV     DPL,#127
                =1  3407     UNBRK_LOOP:
EF16 E0         =1  3408             MOVX    A,@DPTR
EF17 F6         =1  3409             MOV     @RO,A
EF18 1582       =1  3410             DEC     DPL
EF1A D8FA       =1  3411             DJNZ    RO,UNBRK_LOOP
EF1C E0         =1  3412             MOVX    A,@DPTR
EF1D F6         =1  3413             MOV     @RO,A
EF1E 7582F0     =1  3414             MOV     DPL,#B
EF21 E0         =1  3415             MOVX    A,@DPTR
EF22 F5F0       =1  3416             MOV     B,A
EF24 758290     =1  3417             MOV     DPL,#P1
EF27 E0         =1  3418             MOVX    A,@DPTR
EF28 F590       =1  3419             MOV     P1,A
EF2A 7582B0     =1  3420             MOV     DPL,#P3
EF2D E0         =1  3421             MOVX    A,@DPTR
```

```
LOC  OBJ           LINE      SOURCE

EF2E 44C4       =1  3422              ORL     A,#0C4H
EF30 F5B0       =1  3423              MOV     P3,A
EF32 7582D0     =1  3424              MOV     DPL,#PSW
EF35 E0         =1  3425              MOVX    A,@DPTR
EF36 F5D0       =1  3426              MOV     PSW,A
EF38 758298     =1  3427              MOV     DPL,#SCON
EF3B E0         =1  3428              MOVX    A,@DPTR
EF3C F598       =1  3429              MOV     SCON,A
EF3E 758281     =1  3430              MOV     DPL,#SP
EF41 E0         =1  3431              MOVX    A,@DPTR
EF42 F581       =1  3432              MOV     SP,A
EF44 7582FD     =1  3433              MOV     DPL,#UPC
EF47 E0         =1  3434              MOVX    A,@DPTR
EF48 C0E0       =1  3435              PUSH    ACC               ;Restore first byte of PC to top of stack
EF4A A3         =1  3436              INC     DPTR
EF4B E0         =1  3437              MOVX    A,@DPTR
 F4C C0E0       =1  3438              PUSH    ACC               ;Second byte of PC
 F4E 7582B8     =1  3439              MOV     DPL,#IP
EF51 E0         =1  3440              MOVX    A,@DPTR
EF52 F5B8       =1  3441              MOV     IP,A
EF54 75828C     =1  3442              MOV     DPL,#TH0
EF57 E0         =1  3443              MOVX    A,@DPTR
EF58 F58C       =1  3444              MOV     TH0,A
EF5A 75828D     =1  3445              MOV     DPL,#TH1
EF5D E0         =1  3446              MOVX    A,@DPTR
EF5E F58D       =1  3447              MOV     TH1,A
EF60 75828A     =1  3448              MOV     DPL,#TL0
EF63 E0         =1  3449              MOVX    A,@DPTR
EF64 F58A       =1  3450              MOV     TL0,A
EF66 75828B     =1  3451              MOV     DPL,#TL1
EF69 E0         =1  3452              MOVX    A,@DPTR
EF6A F58B       =1  3453              MOV     TL1,A
 F6C 758289     =1  3454              MOV     DPL,#TMOD
 F6F E0         =1  3455              MOVX    A,@DPTR
 F70 F589       =1  3456              MOV     TMOD,A
EF72 758288     =1  3457              MOV     DPL,#TCON
EF75 E0         =1  3458              MOVX    A,@DPTR
EF76 F588       =1  3459              MOV     TCON,A
EF78 7582A8     =1  3460              MOV     DPL,#IE
EF7B E0         =1  3461              MOVX    A,@DPTR
EF7C 547E       =1  3462              ANL     A,#07EH           ;Leave overall enable and external 0 off until
                =1  3463                                        ;interrupt mode is established.
EF7E F5A8       =1  3464              MOV     IE,A              ;Set up IE.
EF80 758282     =1  3465              MOV     DPL,#DPL
EF83 E0         =1  3466              MOVX    A,@DPTR
EF84 C0E0       =1  3467              PUSH    ACC               ;Push user data pointer into the user stack.
EF86 0582       =1  3468              INC     DPL
EF88 E0         =1  3469              MOVX    A,@DPTR
EF89 C0E0       =1  3470              PUSH    ACC
EF8B 7582E0     =1  3471              MOV     DPL,#ACC          ;Restore the user A register.
 F8E E0         =1  3472              MOVX    A,@DPTR
 F8F D083       =1  3473              POP     DPH
 F91 D082       =1  3474              POP     DPL               ;Restore user data pointer.
EF93 C289       =1  3475              CLR     IE0               ;Set up the break logic interrupts.
EF95 D288       =1  3476              SETB    IT0
```

```
LOC  OBJ            LINE     SOURCE

EF97 D2B8       =1  3477              SETB    PX0
EF99 43A881     =1  3478              ORL     IE,#81H                 ;Edge mode, highest priority.
EF9C 32         =1  3479              RETI                            ;'Return' to the user.
                =1  3480 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1          ;*****************************************************************
                =1          ;
                =1          ;    NAME: READ_PC/WRITE_PC
                =1          ;
                =1          ;    ABSTRACT:
                =1          ;       READ_PC:  This routine returns a copy of the user program
                =1          ;       counter in A and B from the page of external RAM devoted to
                =1          ;       saving the user status.
                =1          ;
                =1          ;       WRITE_PC: this routine loads the user program counter
                =1          ;       with the parameter passed to it.
                =1          ;
                =1          ;    INPUTS: PARAM1 (high byte), PARAM2 (low byte)
                =1          ;
                =1          ;    OUTPUTS: ACC (low byte), B (high byte), users version of PC
                =1          ;
                =1          ;    VARIABLES MODIFIED: DPTR, A, B
                =1          ;
                =1          ;    ERROR EXITS: None
                =1          ;
                =1          ;    SUBROUTINES ACCESSED DIRECTLY: None
                =1          ;
                =1          ;
                =1          ;*****************************************************************
                =1          READ_PC:                          ;Set DPTR to point at the user PC in the
                =1                                            ;user stack.
EF9D 90B0FD     =1                  MOV     DPTR,#(RAMOFF+UPC)
EFA0 E0         =1                  MOVX    A,@DPTR
EFA1 F5F0       =1                  MOV     B,A               ;Load the user pc into B and A.
EFA3 A3         =1                  INC     DPTR
EFA4 E0         =1                  MOVX    A,@DPTR
EFA5 C5F0       =1                  XCH     A,B
EFA7 22         =1                  RET
                =1          WRITE_PC:                         ;Set the DPTR to point at the user PC in the
                =1                                            ;user stack.
EFA8 90B0FD     =1                  MOV     DPTR,#(RAMOFF+UPC)
EFAB EB         =1                  MOV     A,PARAM2          ;Write into the user PC.
EFAC F0         =1                  MOVX    @DPTR,A
EFAD A3         =1                  INC     DPTR
EFAE EA         =1                  MOV     A,PARAM1
EFAF F0         =1                  MOVX    @DPTR,A
EFB0 22         =1                  RET
                =1     +1   $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1  3524     ;*************************************************************
                =1  3525     ;
                =1  3526     ;    NAME: CHECK_FROM
                =1  3527     ;
                =1  3528     ;    ABSTRACT: This routine gets a token and if it is a 'from', it
                =1  3529     ;       will get the number and send it to the users PC.  It always
                =1  3530     ;       leaves this routine with a 'fresh' token whether it finds a
                =1  3531     ;       'from' or not.
                =1  3532     ;
                =1  3533     ;    INPUTS: None
                =1  3534     ;
                =1  3535     ;    OUTPUTS: TOKSTR
                =1  3536     ;
                =1  3537     ;    VARIABLES MODIFIED: PARAM1, PARAM2
                =1  3538     ;
                =1  3539     ;    ERROR EXITS: None
                =1  3540     ;
                =1  3541     ;    SUBROUTINES ACCESSED DIRECTLY: IGETOKE, IGETNUM, WRITE_PC
                =1  3542     ;
                =1  3543     ;
                =1  3544     ;*************************************************************
                =1  3545     CHECK_FROM:
EFB1 11BC       =1  3546            CALL    IGETOKE
EFB3 B4090B     =1  3547            CJNE    A,#FROM_TOKE,NOTFRM
EFB6 12E769     =1  3548            CALL    IGETNUM
EFB9 AA49       =1  3549            MOV     PARAM1,VALHGH
EFBB AB4A       =1  3550            MOV     PARAM2,VALLOW
EFBD F1A8       =1  3551            CALL    WRITE_PC
EFBF 11BC       =1  3552            CALL    IGETOKE
EFC1 22         =1  3553     NOTFRM: RET
                =1  3554 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1  3555     ;*****************************************************************
                =1  3556     ;
                =1  3557     ;    NAME: BREAK_VECTOR
                =1  3558     ;
                =1  3559     ;    ABSTRACT: This routine writes location 03 as a break
                =1  3560     ;       vector, and verifies that it was able to write.  This vector
                =1  3561     ;       does a long call to a service routine for all level zero
                =1  3562     ;       interrupts.  Level zero interrupts include:
                =1  3563     ;               UPI interrupts (keyboard closures, USART buffer
                =1  3564     ;                       empty or full, cassette characters rec'd)
                =1  3565     ;               Hardware breakpoints (PROG, DATA, GUARDED ACCESS,
                =1  3566     ;                       SINGLESTEP)
                =1  3567     ;
                =1  3568     ;    INPUTS: None
                =1  3569     ;
                =1  3570     ;    OUTPUTS: Code memory locations 3, 4 and 5
                =1  3571     ;
                =1  3572     ;    VARIABLES MODIFIED: DPTR, A, ERRNUM
                =1  3573     ;
                =1  3574     ;    ERROR EXITS: 17H (NO RAM AT LOCATION 3)
                =1  3575     ;
                =1  3576     ;    SUBROUTINES ACCESSED DIRECTLY: IERROR
                =1  3577     ;
                =1  3578     ;
                =1  3579     ;*****************************************************************
                =1  3580     BREAK_VECTOR:
EFC2 900003     =1  3581             MOV     DPTR,#0003H             ;Point to INTO vector address again
EFC5 7402       =1  3582             MOV     A,#02H                  ;Store a "LCALL" instruction
EFC7 F0         =1  3583             MOVX    @DPTR,A
EFC8 74E0       =1  3584             MOV     A,#HIGH(BREAK)          ;Store the high byte of address for "break"
EFCA A3         =1  3585             INC     DPTR
EFCB F0         =1  3586             MOVX    @DPTR,A
EFCC A3         =1  3587             INC     DPTR
EFCD 7403       =1  3588             MOV     A,#LOW(BREAK)           ;Store low byte of "break" address
EFCF F0         =1  3589             MOVX    @DPTR,A
EFD0 E4         =1  3590             CLR     A
EFD1 93         =1  3591             MOVC    A,@A+DPTR               ;Verify that the write did go into RAM
EFD2 B40301     =1  3592             CJNE    A,#LOW(BREAK),B_V_ERR   ;if not the same, go to error
EFD5 22         =1  3593             RET
                =1  3594     ;*******************************************************************
                =1  3595     B_V_ERR:
EFD6 754317     =1  3596             MOV     ERRNUM,#17H             ;No RAM at location 3
EFD9 02E3E4     =1  3597             JMP     IERROR
                =1  3598 +1  $EJECT
```

```
LOC  OBJ                SOURCE
                =1      ;*****************************************************************
                =1      ;
                =1      ;    NAME: STEP_CMD
                =1      ;
                =1      ;    ABSTRACT: STEP executes one or more instructions at a user
                =1      ;       selectable rate, breaking after each instruction.
                =1      ;       The monitor displays the contents of the PC, ACC,
                =1      ;       DPTR, SP and, optionally, a specified bit or byte.
                =1      ;
                =1      ;    INPUTS: None
                =1      ;
                =1      ;    OUTPUTS: BREAL_STATUS
                =1      ;
                =1      ;    VARIABLES MODIFIED: A, TOKSAV, DPTR, ERRNUM, PARAM1, BREAK_STATUS
                =1      ;
                =1      ;    ERROR EXITS: 03H (NUMBER EXPECTED)
                =1      ;                 09H (DECIMAL NUMBER EXPECTED)
                =1      ;
                =1      ;    SUROUTINES ACCESSED DIRECTLY: CHECK_FROM, IGETOKE, IGETEOL,
                =1      ;       BREAK_VECTOR, UPI_CMD, UPI_OUT, UNBREAK, IEOL_CHECK, IERROR
                =1      ;
                =1      ;*****************************************************************
                =1      STEP_CMD:
EFDC F1B1       =1              CALL    CHECK_FROM
EFDE 90B0F2     =1              MOV     DPTR,#(RAMOFF+SAVE_SEL)
EFE1 E4         =1              CLR     A
EFE2 F0         =1              MOVX    @DPTR,A                 ;Clear SAVE_SEL to avoid unwanted display..
EFE3 E548       =1              MOV     A,TOKSTR
EFE5 B40260     =1              CJNE    A,#COMMA_TOKE,STPEOL
EFE8 11BC       =1              CALL    IGETOKE
EFEA 54F8       =1              ANL     A,#0F8H                 ;Strip out the lower 3 bits
EFEC B4801F     =1              CJNE    A,#80H,DCLAUSE          ;and skip to process the delay clause if
                =1                                              ;not a display memory token.
EFEF 85485B     =1              MOV     TOKSAV,TOKSTR           ;Else proceed with display clause.
EFF2 12E769     =1              CALL    IGETNUM                 ;Save the address to be displayed in external
                =1                                              ;RAM.
EFF5 90B0F3     =1              MOV     DPTR,#(RAMOFF+ADDR_SAVE_HIGH)
EFF8 E549       =1              MOV     A,VALHGH
EFFA F0         =1              MOVX    @DPTR,A
EFFB A3         =1              INC     DPTR
EFFC E54A       =1              MOV     A,VALLOW
EFFE F0         =1              MOVX    @DPTR,A
EFFF 7582F2     =1              MOV     DPL,#SAVE_SEL
F002 E55B       =1              MOV     A,TOKSAV
F004 F0         =1              MOVX    @DPTR,A                 ;Save token to be displayed after STEP
F005 12E8BC     =1              CALL    IGETOKE
F008 B4023D     =1              CJNE    A,#COMMA_TOKE,STPEOL
F00B 12E8BC     =1              CALL    IGETOKE
F00E E548       =1       DCLAUSE:MOV    A,TOKSTR
F010 754303     =1              MOV     ERRNUM,#03H             ;Number expected
F013 B40139     =1              CJNE    A,#NUMBER_TOKE,EXERRO
F016 7409       =1              MOV     A,#9
F018 B54A00     =1              CJNE    A,VALLOW,LAB18
F01B 754309     =1       LAB18:  MOV    ERRNUM,#09H             ;Decimal number expected
F01E 402F       =1              JC      EXERRO                  ;Error unless number is less than 9.
```

```
LOC  OBJ            LINE     SOURCE

F020 E549       =1   3654               MOV     A,VALHGH
F022 702B       =1   3655               JNZ     EXERRO                ;Upper bits must be zero also.
F024 90B0F5     =1   3656               MOV     DPTR,#(RAMOFF+DELAY)
F027 E54A       =1   3657               MOV     A,VALLOW
F029 F0         =1   3658               MOVX    @DPTR,A
F02A 12E773     =1   3659               CALL    IGETEOL               ;Check that next entry is CR
                =1   3660       STPLOP:
F02D 74FF       =1   3661               MOV     A,#MULTISTEP
                =1   3662       STEP51:
F02F 90B0FB     =1   3663               MOV     DPTR,#(RAMOFF+BREAK_STATUS)
F032 F0         =1   3664               MOVX    @DPTR,A
F033 12EFC2     =1   3665               CALL    BREAK_VECTOR
F036 7A03       =1   3666               MOV     PARAM1,#GR_PORT
F038 12E625     =1   3667               CALL    UPI_CMD
F03B 7A08       =1   3668               MOV     PARAM1,#CLR_BRK_LATCHES ;Clear all break latches
F03D 12E638     =1   3669               CALL    UPI_OUT
F040 7A01       =1   3670               MOV     PARAM1,#SINGLE_BREAK
F042 12E638     =1   3671               CALL    UPI_OUT               ;Send it to the UPI data channel
F045 02EEE9     =1   3672               JMP     UNBREAK
F048 12E5BB     =1   3673       STPEOL: CALL    IEOL_CHECK
F04B 74FE       =1   3674               MOV     A,#SINGLESTEP
F04D 80E0       =1   3675               JMP     STEP51
F04F 02E3E4     =1   3676       EXERRO: JMP     IERROR
                =1   3677 +1    $EJECT
```

```
LOC  OBJ            LINE     SOURCE

               =1   3678     ;*****************************************************************
               =1   3679     ;
               =1   3680     ;    NAME: STEP51_RETURN
               =1   3681     ;
               =1   3682     ;    ABSTRACT: After the branch to UNBREAK in STEP_CMD, the user
               =1   3683     ;       execution has begun.  Exit from execution with the STEP_FLAG
               =1   3684     ;       set will result in a branch to STEP51_RETURN.
               =1   3685     ;
               =1   3686     ;    INPUTS: SAVE_SEL, BREAK_STATUS, DELAY, USER SP, ACC, DPTR,
               =1   3687     ;       ADDR_SAVE_HIGH, ADDR_SAVE_LOW
               =1   3688     ;
               =1   3689     ;    OUTPUTS: None
               =1   3690     ;
               =1   3691     ;    VARIABLES MODIFIED: PARAM1, PARAM2, ERRNUM, CAUSE_IMAGE, DPTR,
               =1   3692     ;
               =1   3693     ;    ERROR EXITS: 16H (EXECUTION ACROSS LOCATION 3)
               =1   3694     ;
               =1   3695     ;    SUBROUTINES ACCESSED DIRECTLY: INEWLINE, READ_PC, ICO, ILSTWRD,
               =1   3696     ;       SPACCO, ILSTBYT, IFETCH, ITIME, ICSTS, UPI_IN, ICI,
               =1   3697     ;       IWAIT_FOR_USER, IERROR
               =1   3698     ;
               =1   3699     ;
               =1   3700     ;*****************************************************************
               =1   3701     STEP51_RETURN:
F052 12E717    =1   3702             CALL    INEWLINE                ;Output a CR-LF.
F055 12EF9D    =1   3703             CALL    READ_PC                 ;Output the contents of the user PC to the
F058 AAF0      =1   3704             MOV     PARAM1,B                ;console.
F05A FB        =1   3705             MOV     PARAM2,A
F05B BAE00C    =1   3706             CJNE    PARAM1,#0E0H,NOT_STEP_THREE
F05E BB0309    =1   3707             CJNE    PARAM2,#3,NOT_STEP_THREE
F061 754316    =1   3708             MOV     ERRNUM,#16H             ;Adr 3 executed
F064 756004    =1   3709             MOV     CAUSE_IMAGE,#4          ;Cause is guarded access to loc 3
F067 02E3E4    =1   3710             JMP     IERROR
               =1   3711     NOT_STEP_THREE:
F06A 7A50      =1   3712             MOV     PARAM1,#'P'             ;Output PC label
F06C 12E5E8    =1   3713             CALL    ICO
F06F AAF0      =1   3714             MOV     PARAM1,B                ;Restore PC value to register for display.
F071 12E7F4    =1   3715             CALL    ILSTWRD                 ;Output address
F074 12E5E6    =1   3716             CALL    SPACCO                  ;Output space
F077 7A41      =1   3717             MOV     PARAM1,#'A'             ;Output user accumulator label
F079 12E5E8    =1   3718             CALL    ICO
F07C 90B0E0    =1   3719             MOV     DPTR,#(RAMOFF+ACC)
F07F E0        =1   3720             MOVX    A,@DPTR
F080 FA        =1   3721             MOV     PARAM1,A                ;Call ILSTBYT(user ACC).
F081 12E7F9    =1   3722             CALL    ILSTBYT
F084 12E5E6    =1   3723             CALL    SPACCO
F087 7A44      =1   3724             MOV     PARAM1,#'D'
F089 12E5E8    =1   3725             CALL    ICO                     ;Output DPTR label
F08C 90B082    =1   3726             MOV     DPTR,#(RAMOFF+DPL)
F08F E0        =1   3727             MOVX    A,@DPTR                 ;Displays the low and high byte of DPTR
F090 FB        =1   3728             MOV     PARAM2,A
F091 A3        =1   3729             INC     DPTR
F092 E0        =1   3730             MOVX    A,@DPTR
F093 FA        =1   3731             MOV     PARAM1,A
F094 12E7F4    =1   3732             CALL    ILSTWRD
```

```
LOC  OBJ          LINE     SOURCE

F097 12E5E6    =1  3733            CALL    SPACCO
F09A 7A53      =1  3734            MOV     PARAM1,#'S'          ;Output the SP label
F09C 12E5E8    =1  3735            CALL    ICO
F09F 90B081    =1  3736            MOV     DPTR,#(RAMOFF+SP)
F0A2 E0        =1  3737            MOVX    A,@DPTR
F0A3 FA        =1  3738            MOV     PARAM1,A
F0A4 12E7F9    =1  3739            CALL    ILSTBYT              ;Output the value of SP
F0A7 90B0F2    =1  3740            MOV     DPTR,#(RAMOFF+SAVE_SEL)
F0AA E0        =1  3741            MOVX    A,@DPTR              ;Get the select code saved in memory.
F0AB F55B      =1  3742            MOV     TOKSAV,A
F0AD 6022      =1  3743            JZ      STEP51_EXIT          ;Exit if no optional display.
F0AF 12E5E6    =1  3744            CALL    SPACCO               ;Output space
F0B2 7A28      =1  3745            MOV     PARAM1,#'('
F0B4 12E5E8    =1  3746            CALL    ICO                  ;Output left parentheses
F0B7 E55B      =1  3747            MOV     A,TOKSAV             ;Move saved token into ACC
F0B9 5407      =1  3748            ANL     A,#07H               ;Mask lower 3 bits
  BB F546      =1  3749            MOV     SELECT,A             ;Move lower 3 bits into selector for FETCH
  3D A3        =1  3750            INC     DPTR                 ;Fetch the saved address.
  BE E0        =1  3751            MOVX    A,@DPTR
F0BF F544      =1  3752            MOV     PNTHGH,A
F0C1 A3        =1  3753            INC     DPTR
F0C2 E0        =1  3754            MOVX    A,@DPTR
F0C3 F545      =1  3755            MOV     PNTLOW,A
               =1  3756                                         ;Fetch the memory byte the user wants
                                                                ;displayed.
F0C5 12E66B    =1  3757            CALL    IFETCH
F0C8 FA        =1  3758            MOV     PARAM1,A             ;And display it.
F0C9 12E7F9    =1  3759            CALL    ILSTBYT
F0CC 7A29      =1  3760            MOV     PARAM1,#')'          ;Output right parentheses
F0CE 12E5E8    =1  3761            CALL    ICO
               =1  3762    STEP51_EXIT:
F0D1 90B0FB    =1  3763            MOV     DPTR,#(RAMOFF+BREAK_STATUS)
F0D4 E0        =1  3764            MOVX    A,@DPTR
F0D5 B4FF2E    =1  3765            CJNE    A,#MULTISTEP,SSRET
  D8 90B0F5    =1  3766            MOV     DPTR,#(RAMOFF+DELAY)
   B E0        =1  3767            MOVX    A,@DPTR              ;Execute multiple single steps
  DC F55C      =1  3768            MOV     DLYCNT,A
F0DE E55C      =1  3769    STPDLY: MOV     A,DLYCNT
F0E0 600B      =1  3770     JZ     DLY_THRU
F0E2 155C      =1  3771            DEC     DLYCNT
F0E4 7A13      =1  3772            MOV     PARAM1,#13H
F0E6 7B88      =1  3773            MOV     PARAM2,#88H
F0E8 12EA45    =1  3774            CALL    ITIME                ;Delay for about 1/2 second per DLYCNT
F0EB 80F1      =1  3775            JMP     STPDLY               ;Loop until delay count = 0
               =1  3776    DLY_THRU:
F0ED 7A00      =1  3777            MOV     PARAM1,#00H
F0EF 7BA5      =1  3778            MOV     PARAM2,#0A5H
F0F1 12EA45    =1  3779            CALL    ITIME                ;Delays 16ms
F0F4 12E602    =1  3780            CALL    ICSTS
F0F7 4002      =1  3781            JC      STEP_STOP            ;No carry means no input pending
               =1  3782    STPLOP_REACH:
F0F9 012D      =1  3783            JMP     STPLOP
               =1  3784    STEP_STOP:
   B 12E64C    =1  3785            CALL    UPI_IN
  FE B41BF8    =1  3786            CJNE    A,#ESC,STPLOP_REACH
F101 12E5EB    =1  3787            CALL    ICI                  ;First esc stops step,2nd will exit.
```

```
LOC  OBJ            LINE     SOURCE

F104 80F3       =1  3788             JMP     STPLOP_REACH            ;Any key after 1st esc resumes step
                =1  3789     ;*****************************************************************
F106 12E3B0     =1  3790     SSRET:  CALL    IWAIT_FOR_USER
F109 02E2D6     =1  3791             JMP     START
F10C 02E3E4     =1  3792     EXERR1: JMP     IERROR
                =1  3793
                =1  3794 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                    =1       ;*****************************************************************
                    =1       ;
                    =1       ;    NAME: GO_CMD
                    =1       ;
                    =1       ;    ABSTRACT: This routine sets up conditions for entering user execution.
                    =1       ;       It looks for partition information and breakpoints and saves
                    =1       ;       an image of break enable hardware in software.
                    =1       ;
                    =1       ;    INPUTS: GR
                    =1       ;
                    =1       ;    OUTPUTS: GR, BREAK_STATUS
                    =1       ;
                    =1       ;    VARIABLES MODIFIED: A, ERRNUM, DPTR, PARAM1, PARAM2, GR
                    =1       ;
                    =1       ;    ERROR EXITS: OBH (BREAK ENABLE SYNTAX)
                    =1       ;
                    =1       ;    SUBROUTINES ACCESSED DIRECTLY: CHECK_FROM, IGETEOL, IGETOKE,
                    =1       ;       IEOL_CHECK, BREAK_VECTOR, UPI_CMD, UPI_OUT, UNBREAK, IPRINT_STRING,
                    =1       ;       READ_PC, ILSTWRD, IWAIT_FOR_USER
                    =1       ;
                    =1       ;*****************************************************************
                    =1       GO_CMD:
F10F 12EFB1         =1               CALL    CHECK_FROM
F112 6407           =1               XRL     A,#EOL_TOKE
F114 6053           =1               JZ      RUN_USER                ;If have the end of line token go to user
                    =1                                               ;emulation.
                    =1                                               ;If not then find out what kind of emulation
                    =1                                               ;is required.
                    =1
F116 E548           =1               MOV     A,TOKSTR                ; First restore the token.
F118 B4080C         =1               CJNE    A,#FOREVER_TOKE,NOTFOR
                    =1                                               ;See if token is FOREVER token
F11B 12E773         =1               CALL    IGETEOL                 ;Wait for CR after FOREVER
F11E 90B0F6         =1               MOV     DPTR,#(RAMOFF+GR)       ;Copy break enable image into hrdwr
F121 7409           =1               MOV     A,#NO_BREAK
F123 F0             =1               MOVX    @DPTR,A
F124 02F169         =1               JMP     RUN_USER                ;Jump to GO loop
F127 75430B         =1       NOTFOR: MOV     ERRNUM,#OBH             ;BRK enable syntax
F12A B40CDF         =1               CJNE    A,#TILL_TOKE,EXERR1     ;Jump to error routine if not TIL token
F12D 12E8BC         =1               CALL    IGETOKE
F130 B4D30C         =1               CJNE    A,#DATA_TOKE,NOTDAT     ;See if next input was data break
F133 12E773         =1               CALL    IGETEOL                 ;Make sure next input is CR
F136 90B0F6         =1               MOV     DPTR,#(RAMOFF+GR)       ;Copy break enable image into hrdwr
F139 740D           =1               MOV     A,#DATA_BREAK
F13B F0             =1               MOVX    @DPTR,A
F13C 02F169         =1               JMP     RUN_USER                ;Jump to GO loop
F13F 75430B         =1       NOTDAT: MOV     ERRNUM,#OBH             ;BRK enable syntax
F142 B4D5C7         =1               CJNE    A,#PROGRAM_TOKE,EXERR1  ;If program break not entered by now,
                    =1                                               ;Then error
F145 12E8BC         =1               CALL    IGETOKE
F148 B40B15         =1               CJNE    A,#OR_TOKE,PGMBRK       ;See if OR was entered in break sequence
F14B 12E8BC         =1               CALL    IGETOKE
F14E 75430B         =1               MOV     ERRNUM,#OBH             ;BRK enable syntax
F151 B4D3B8         =1               CJNE    A,#DATA_TOKE,EXERR1     ;Make sure data token was entered next
F154 12E773         =1               CALL    IGETEOL                 ;Make sure CR was entered last
```

```
LOC  OBJ            LINE     SOURCE

F157 90B0F6    =1   3850            MOV     DPTR,#(RAMOFF+GR)        ;Copy break enable image into sftwr
F15A 740F      =1   3851            MOV     A,#(DATA_BREAK OR PROGRAM_BREAK)
F15C F0        =1   3852            MOVX    @DPTR,A
F15D 02F169    =1   3853            JMP     RUN_USER
F160 12E5BB    =1   3854     PGMBRK: CALL   IEOL_CHECK
F163 90B0F6    =1   3855            MOV     DPTR,#(RAMOFF+GR)        ;Copy break enable image into sftwr
F166 740B      =1   3856            MOV     A,#PROGRAM_BREAK
F168 F0        =1   3857            MOVX    @DPTR,A
               =1   3858     RUN_USER:
F169 90B0FB    =1   3859            MOV     DPTR,#(RAMOFF+BREAK_STATUS)
F16C 74FB      =1   3860            MOV     A,#NOT_STEP
F16E F0        =1   3861            MOVX    @DPTR,A                  ;Clear the step flag to show a 'run' condition
F16F 12EFC2    =1   3862            CALL    BREAK_VECTOR
F172 7AF1      =1   3863            MOV     PARAM1,#HIGH(XEQT_MSG)
F174 7BA4      =1   3864            MOV     PARAM2,#LOW(XEQT_MSG)
F176 12E9FF    =1   3865            CALL    IPRINT_STRING
F179 7A03      =1   3866            MOV     PARAM1,#GR_PORT
F17B 12E625    =1   3867            CALL    UPI_CMD
F17E 7A08      =1   3868            MOV     PARAM1,#CLR_BRK_LATCHES ;Clear all break latches
F180 12E638    =1   3869            CALL    UPI_OUT
F183 90B0F6    =1   3870            MOV     DPTR,#(RAMOFF+GR)        ;Copy break enable image into hrdwr
F186 E0        =1   3871            MOVX    A,@DPTR
F187 FA        =1   3872            MOV     PARAM1,A
F188 12E638    =1   3873            CALL    UPI_OUT                  ;Send it to the UPI data channel
F18B 02EEE9    =1   3874            JMP     UNBREAK
               =1   3875     ;*************************************************************
               =1   3876     RUN_USER_RETURN:
F18E 7AF1      =1   3877            MOV     PARAM1,#HIGH(BREAK_MSG)
F190 7BB6      =1   3878            MOV     PARAM2,#LOW(BREAK_MSG)
F192 12E9FF    =1   3879            CALL    IPRINT_STRING
F195 12EF9D    =1   3880            CALL    READ_PC
F198 AAF0      =1   3881            MOV     PARAM1,B                 ;Display the user PC
F19A FB        =1   3882            MOV     PARAM2,A
F19B 12E7F4    =1   3883            CALL    ILSTWRD
F19E 12E3B0    =1   3884            CALL    IWAIT_FOR_USER           ;And goto the monitor.
F1A1 02E2D6    =1   3885            JMP     START
               =1   3886     ;*****************************************************************
               =1   3887     XEQT_MSG:
F1A4 11        =1   3888            DB      17,CR,LF,('EXECUTION BEGUN')
F1A5 0D
F1A6 0A
F1A7 45584543
F1AB 5554494F
F1AF 4E204245
F1B3 47554E
               =1   3889     BREAK_MSG:
F1B6 16        =1   3890            DB      22,CR,LF,('EXECUTION HALTED PC=')
F1B7 0D
F1B8 0A
F1B9 45584543
F1BD 5554494F
F1C1 4E204841
F1C5 4C544544
F1C9 2050433D
                    3891 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                         +1  $INCLUDE(:F1:MONFUN.INC)
                   =1        ;******************************************************************
                   =1        ;
                   =1        ;     NAME: LIST_CMD
                   =1        ;
                   =1        ;     ABSTRACT: This routine gets the 'keyword =' message and sets
                   =1        ;        up the LSTFLG to display tokens to the console and an auxilary
                   =1        ;        terminal.  Anytime display is called for. It will also terminate
                   =1        ;        any ISIS files with a control Z. List is on when LSTFLG = 1.
                   =1        ;
                   =1        ;     INPUTS: LSTFLG
                   =1        ;
                   =1        ;     OUTPUTS: LSTFLG
                   =1        ;
                   =1        ;     VARIABLES MODIFIED: LSTFLG, PARAM1, ERRNUM
                   =1        ;
                   =1        ;     ERROR EXITS: 08H (RESET OR ON REQUIRED)
                   =1        ;
                   =1        ;     SUBROUTINES ACCESSED DIRECTLY: ISIT_DISPLAY, IGETOKE,
                   =1        ;        IDISPLAY_TOKEN, ICO, UPI_CMD, INEWLINE, IWAIT_FOR_USER
                   =1        ;
                   =1        ;
                   =1        ;******************************************************************
                   =1        LIST_CMD:
F1CD 12E784        =1                CALL     ISIT_DISPLAY                ;Sets up 'keyword =' msg
F1D0 401E          =1                JC       DISPLAY_LIST                ;C=1 if display only
F1D2 12E8BC        =1                CALL     IGETOKE
F1D5 B40F03        =1                CJNE     A,#ON_TOKE,LIST_2           ;List turned on, no display
F1D8 D201          =1                SETB     LSTFLG
F1DA 22            =1                RET
F1DB 754308        =1        LIST_2: MOV      ERRNUM,#08H                 ;Reset or on required
F1DE B40E71        =1                CJNE     A,#RESET_TOKE,STATE_ERR     ;List turned off, no display
F1E1 7A01          =1                MOV      PARAM1,#USART_MODE
F1E3 12E625        =1                CALL     UPI_CMD
F1E6 C201          =1                CLR      LSTFLG
F1E8 7A1A          =1                MOV      PARAM1,#1AH
F1EA 12E5E8        =1                CALL     ICO                         ;Send cntrl-Z to close MDS file
F1ED 02E717        =1                JMP      INEWLINE                    ;Insure that control-z gets out before Usar
                             t Reset
                   =1        DISPLAY_LIST:
F1F0 7A0F          =1                MOV      PARAM1,#ON_TOKE             ;Display 'on' set up
F1F2 200102        =1                JB       LSTFLG,LIST_1
F1F5 7A0E          =1                MOV      PARAM1,#RESET_TOKE          ;Display 'reset' set up
F1F7 12EA12        =1        LIST_1: CALL     IDISPLAY_TOKEN
F1FA 02E3B0        =1                JMP      IWAIT_FOR_USER
                   =1    +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

               =1   3937     ;*****************************************************************
               =1   3938     ;
               =1   3939     ;    NAME: BAUD_CMD/ SET_BAUD
               =1   3940     ;
               =1   3941     ;    ABSTRACT: This routine will allow the user to display the
               =1   3942     ;       baud rate or change the baud rate to any legal value between
               =1   3943     ;       110 and 9600.  Default on power up is 2400.
               =1   3944     ;
               =1   3945     ;    INPUTS: BAUD_HIGH, BAUD_LOW
               =1   3946     ;
               =1   3947     ;    OUTPUTS: BAUD_HIGH, BAUD_LOW, BAUDKEY
               =1   3948     ;
               =1   3949     ;    VARIABLES MODIFIED: DPTR, ERRNUM, A, B, BAUD_HIGH, BAUD_LOW, BAUDKEY
               =1   3950     ;
               =1   3951     ;    ERROR EXITS: OAH (ILLEGAL BAUD VALUE)
               =1   3952     ;
               =1   3953     ;    SUBROUTINES ACCESSED DIRECTLY: ISIT_DISPLAY, IGETNUM, IERROR,
               =1   3954     ;       ILSTWRD, IWAIT_FOR_USER
               =1   3955     ;
               =1   3956     ;
               =1   3957     ;*****************************************************************
               =1   3958     BAUD_CMD:
F1FD 12E784    =1   3959             CALL     ISIT_DISPLAY
F200 4068      =1   3960             JC       BAUD_DISPLAY
F202 12E769    =1   3961             CALL     IGETNUM
F205 90F255    =1   3962             MOV      DPTR,#BAUD_RATE          ;Check table for a valid baud rate request.
F208 7800      =1   3963             MOV      POINTO,#00H
               =1   3964     BS_LOOP:
F20A E8        =1   3965             MOV      A,POINTO
F20B 93        =1   3966             MOVC     A,@A+DPTR
F20C B5493F    =1   3967             CJNE     A,VALHGH,BS_2
F20F E54A      =1   3968             MOV      A,VALLOW
F211 B80038    =1   3969             CJNE     POINTO,#00H,BM_1
               =1   3970                                               ;If POINTO=0, the lower 2 digits better be
               =1   3971                                               ;10 because the baud rate is 110.
F214 75430A    =1   3972             MOV      ERRNUM,#0AH              ;Illegal baud value
F217 B41038    =1   3973             CJNE     A,#10H,STATE_ERR
               =1   3974     PRE_SET_BAUD:
F21A 90B0F7    =1   3975             MOV      DPTR,#(RAMOFF+BAUD_HIGH)
F21D E549      =1   3976             MOV      A,VALHGH
F21F F0        =1   3977             MOVX     @DPTR,A
F220 A3        =1   3978             INC      DPTR
F221 E54A      =1   3979             MOV      A,VALLOW
F223 F0        =1   3980             MOVX     @DPTR,A
F224 7582FC    =1   3981             MOV      DPL,#BAUDKEY
F227 E8        =1   3982             MOV      A,POINTO
F228 F0        =1   3983             MOVX     @DPTR,A
               =1   3984     ;******************************************************************
               =1   3985     SET_BAUD:
F229 90B0FC    =1   3986             MOV      DPTR,#(RAMOFF+BAUDKEY)
F22C E0        =1   3987             MOVX     A,@DPTR
F22D 23        =1   3988             RL       A
F22E F5F0      =1   3989             MOV      B,A
F230 90F25C    =1   3990             MOV      DPTR,#TIMER_PRESET
F233 93        =1   3991             MOVC     A,@A+DPTR
```

```
LOC  OBJ            LINE     SOURCE

F234 C5F0       =1  3992              XCH     A,B
F236 A3         =1  3993              INC     DPTR
F237 93         =1  3994              MOVC    A,@A+DPTR
F238 C5F0       =1  3995              XCH     A,B                     ;Store the timer preset value.
F23A 90B805     =1  3996              MOV     DPTR,#(RAMIO+TIMER_HIGH)
F23D 4440       =1  3997              ORL     A,#CONTINUOUS_MODE
F23F F0         =1  3998              MOVX    @DPTR,A
F240 1582       =1  3999              DEC     DPL
F242 E5F0       =1  4000              MOV     A,B
F244 F0         =1  4001              MOVX    @DPTR,A
F245 90B800     =1  4002              MOV     DPTR,#RAMIO             ;Start - load timer
F248 74C0       =1  4003              MOV     A,#START_16_TIMER
F24A F0         =1  4004              MOVX    @DPTR,A
F24B 22         =1  4005              RET
F24C 60CC       =1  4006     BM_1:    JZ      PRE_SET_BAUD            ;Else the lower 2 digits better be 0
                =1  4007                                              ;because all the other rates end in 0.
  4E 08         =1  4008     BS_2:    INC     POINTO
  4F B496B8     =1  4009              CJNE    A,#HIGH(9600H),BS_LOOP
                =1  4010     STATE_ERR:
F252 02E3E4     =1  4011              JMP     IERROR
                =1  4012     BAUD_RATE:
F255 01         =1  4013              DB      HIGH(110H)
F256 03         =1  4014              DB      HIGH(300H)
F257 06         =1  4015              DB      HIGH(600H)
F258 12         =1  4016              DB      HIGH(1200H)
F259 24         =1  4017              DB      HIGH(2400H)
F25A 48         =1  4018              DB      HIGH(4800H)
F25B 96         =1  4019              DB      HIGH(9600H)
                =1  4020     TIMER_PRESET:
F25C 0470       =1  4021              DW      1136
F25E 01A1       =1  4022              DW      0417
F260 00D0       =1  4023              DW      0208
 262 0068       =1  4024              DW      0104
  64 0034       =1  4025              DW      0052
  66 001A       =1  4026              DW      0026
F268 000D       =1  4027              DW      0013
                =1  4028     ;*****************************************************************
                =1  4029     BAUD_DISPLAY:
F26A 90B0F7     =1  4030              MOV     DPTR,#(RAMOFF+BAUD_HIGH)
F26D E0         =1  4031              MOVX    A,@DPTR
F26E FA         =1  4032              MOV     PARAM1,A
F26F A3         =1  4033              INC     DPTR
F270 E0         =1  4034              MOVX    A,@DPTR
F271 FB         =1  4035              MOV     PARAM2,A
F272 12E7F4     =1  4036              CALL    ILSTWRD
F275 02E3B0     =1  4037              JMP     IWAIT_FOR_USER
                =1  4038 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE
               =1   4039     ;***************************************************************
               =1   4040     ;
               =1   4041     ;    NAME: TOP_CMD
               =1   4042     ;
               =1   4043     ;    ABSTRACT: This routine will set the top of memory to a value
               =1   4044     ;       requested by the user.  It will error for values > 7FFFH.
               =1   4045     ;       It will also list the current TOP value to the console upon
               =1   4046     ;       request.
               =1   4047     ;
               =1   4048     ;    INPUTS: TOP_STORE
               =1   4049     ;
               =1   4050     ;    OUTPUTS: TOP_STORE
               =1   4051     ;
               =1   4052     ;    VARIABLES MODIFIED: DPTR, A, B, PARAM1, ERRNUM
               =1   4053     ;
               =1   4054     ;    ERROR EXITS: 0DH (TOP VALUE > 7FFFH)
               =1   4055     ;
               =1   4056     ;    SUBROUTINES ACCESSED DIRECTLY: ISIT_DISPLAY, IGETNUM, ILSTBYT,
               =1   4057     ;       IWAIT_FOR_USER
               =1   4058     ;
               =1   4059     ;
               =1   4060     ;***************************************************************
F278 12E784    =1   4061     TOP_CMD:CALL    ISIT_DISPLAY
F27B 90B0F9    =1   4062             MOV     DPTR,#(RAMOFF+TOP_STORE)
F27E 401A      =1   4063             JC      TOP_DISPLAY
F280 12E769    =1   4064             CALL    IGETNUM
F283 E549      =1   4065             MOV     A,VALHGH              ;Do not allow top > 32k
F285 75430D    =1   4066             MOV     ERRNUM,#0DH           ;Top value > 7FFFH
F288 20E7C7    =1   4067             JB      ACC.7,STATE_ERR
F28B F5F0      =1   4068             MOV     B,A                   ;Check for the special case of 0000H
               =1   4069                                           ;otherwise the display should end
F28D 454A      =1   4070             ORL     A,VALLOW              ;with an FFH
F28F 6002      =1   4071             JZ      ST_1
F291 05F0      =1   4072             INC     B
               =1   4073     ST_1:
F293 E5F0      =1   4074             MOV     A,B
F295 90B0F9    =1   4075             MOV     DPTR,#(RAMOFF+TOP_STORE)
F298 F0        =1   4076             MOVX    @DPTR,A
F299 22        =1   4077             RET
               =1   4078     ;***************************************************************
               =1   4079     TOP_DISPLAY:
F29A E0        =1   4080             MOVX    A,@DPTR               ;Call listbyte(top).
F29B 6001      =1   4081             JZ      TOP_LIST_2
F29D 14        =1   4082             DEC     A
               =1   4083     TOP_LIST_2:
F29E FA        =1   4084             MOV     PARAM1,A
F29F 12E7F9    =1   4085             CALL    ILSTBYT
F2A2 90B0F9    =1   4086             MOV     DPTR,#(RAMOFF+TOP_STORE)
F2A5 E0        =1   4087             MOVX    A,@DPTR
F2A6 6008      =1   4088             JZ      TOP_LIST_0
F2A8 7AFF      =1   4089             MOV     PARAM1,#0FFH
F2AA 12E7F9    =1   4090             CALL    ILSTBYT
F2AD 02F2B5    =1   4091             JMP     TOP_LIST_1
               =1   4092     TOP_LIST_0:
F2B0 7A00      =1   4093             MOV     PARAM1,#00H
```

```
LOC  OBJ            LINE      SOURCE

F2B2 12E7F9      =1  4094              CALL    ILSTBYT
                 =1  4095      TOP_LIST_1:
F2B5 02E3B0      =1  4096              JMP     IWAIT_FOR_USER
                 =1  4097 +1   $EJECT
```

```
LOC  OBJ            LINE     SOURCE

               =1   4098     ;*****************************************************************
               =1   4099     ;
               =1   4100     ;    NAME: CAUSE_CMD
               =1   4101     ;
               =1   4102     ;    ABSTRACT: This routine will display the reason detected
               =1   4103     ;       for a break execution.  It is a display-only function.
               =1   4104     ;       The cause is determined and stored during BREAK.
               =1   4105     ;
               =1   4106     ;    INPUTS: CAUSE_IMAGE
               =1   4107     ;
               =1   4108     ;    OUTPUTS: None
               =1   4109     ;
               =1   4110     ;    VARIABLES MODIFIED: A, DPTR, COUNT, PARAM1, PARAM2, ERRNUM
               =1   4111     ;
               =1   4112     ;    ERROR EXITS: OEH (DISPLAY ONLY)
               =1   4113     ;
               =1   4114     ;    SUBROUTINES ACCESSED DIRECTLY: ISIT_DISPLAY, IPRINT_STRING,
               =1   4115     ;       IWAIT_FOR_USER
               =1   4116     ;
               =1   4117     ;
               =1   4118     ;*****************************************************************
               =1   4119     CAUSE_CMD:
F2B8 12E784    =1   4120             CALL    ISIT_DISPLAY
F2BB 75430E    =1   4121             MOV     ERRNUM,#0EH            ;Display only
F2BE 5092      =1   4122             JNC     STATE_ERR
F2C0 E560      =1   4123             MOV     A,CAUSE_IMAGE
F2C2 90F2DC    =1   4124             MOV     DPTR,#CAUSE_TAB
F2C5 7F05      =1   4125             MOV     COUNT,#5               ;Output the appropriate message.
               =1   4126     CL_LOOP:
F2C7 13        =1   4127             RRC     A                      ;Isolate the bit which indicates the
               =1   4128                                            ;cause of the break.
F2C8 20E004    =1   4129             JB      ACC.0,CL_0
F2CB A3        =1   4130             INC     DPTR
F2CC A3        =1   4131             INC     DPTR
F2CD DFF8      =1   4132             DJNZ    COUNT,CL_LOOP
               =1   4133     CL_0:
F2CF E4        =1   4134             CLR     A
F2D0 93        =1   4135             MOVC    A,@A+DPTR
F2D1 FA        =1   4136             MOV     PARAM1,A
F2D2 E4        =1   4137             CLR     A
F2D3 A3        =1   4138             INC     DPTR
F2D4 93        =1   4139             MOVC    A,@A+DPTR
F2D5 FB        =1   4140             MOV     PARAM2,A
F2D6 12E9FF    =1   4141             CALL    IPRINT_STRING
F2D9 02E3B0    =1   4142             JMP     IWAIT_FOR_USER
               =1   4143     CAUSE_TAB:
F2DC F2E8      =1   4144             DW      USER_MSG
F2DE F2F3      =1   4145             DW      GUARD_MSG
F2E0 F302      =1   4146             DW      PROG_MSG
F2E2 F310      =1   4147             DW      DATA_MSG
F2E4 F31B      =1   4148             DW      SINGLE_STEP_MSG
F2E6 F327      =1   4149             DW      NOBRK_MSG
               =1   4150     USER_MSG:
F2E8 0A        =1   4151             DB      10,('USER ABORT')
F2E9 55534552
```

```
LOC  OBJ            LINE     SOURCE

F2ED 2041424F
F2F1 5254
               =1   4152     GUARD_MSG:
F2F3 0E        =1   4153             DB      14,('GUARDED ACCESS')
F2F4 47554152
F2F8 44454420
F2FC 41434345
F300 5353
               =1   4154     PROG_MSG:
F302 0D        =1   4155             DB      13,('PROGRAM BREAK')
F303 50524F47
F307 52414D20
F30B 42524541
F30F 4B
               =1   4156     DATA_MSG:
F310 0A        =1   4157             DB      10,('DATA BREAK')
F311 44415441
F315 20425245
F319 414B
               =1   4158     SINGLE_STEP_MSG:
F31B 0B        =1   4159             DB      11,('SINGLE STEP')
F31C 53494E47
F320 4C452053
F324 544550
               =1   4160     NOBRK_MSG:
F327 0B        =1   4161             DB      11,('WHAT BREAK?')
F328 57484154
F32C 20425245
F330 414B3F
               =1   4162 +1  $EJECT
```

```
LOC  OBJ                  SOURCE

                =1        ;*****************************************************************
                =1        ;
                =1        ;    NAME: SEND_BYTE
                =1        ;
                =1        ;    ABSTRACT: This routine outputs one byte, in either hex or
                =1        ;       binary depending on the setting of the binary flag, to
                =1        ;       the USART. A new checksum is calculated and returned.
                =1        ;
                =1        ;    INPUTS: CHECKSUM, A
                =1        ;
                =1        ;    OUTPUTS: CHECKSUM
                =1        ;
                =1        ;    VARIABLES MODIFIED: A, PARAM1
                =1        ;
                =1        ;    ERROR EXITS: None
                =1        ;
                =1        ;    SUBROUTINES ACCESSED DIRECTLY: ICO, ILSTBYT
                =1        ;
                =1        ;
                =1        ;*****************************************************************
                =1        SEND_BYTE:
F333 CE         =1                XCH     A,CHECKSUM
F334 2E         =1                ADD     A,CHECKSUM
F335 CE         =1                XCH     A,CHECKSUM
F336 FA         =1                MOV     PARAM1,A
F337 200503     =1                JB      BINARY_FLG,SEND_BINARY
F33A 02E7F9     =1                JMP     ILSTBYT
                =1        SEND_BINARY:
F33D 02E5E8     =1                JMP     ICO
                =1     +1 $EJECT
```

```
LOC  OBJ            LINE     SOURCE

               =1   4193     ;*****************************************************************
               =1   4194     ;
               =1   4195     ;    NAME: HEXBIN
               =1   4196     ;
               =1   4197     ;    ABSTRACT: Reads two characters from the input device and
               =1   4198     ;       converts them to binary.  If the binary flag is set, then
               =1   4199     ;       one binary character is input.  This value is added to the
               =1   4200     ;       checksum byte and also returned to the calling routine.
               =1   4201     ;
               =1   4202     ;    INPUTS: BINARY_FLG, CHECKSUM
               =1   4203     ;
               =1   4204     ;    OUTPUTS: CHECKSUM
               =1   4205     ;
               =1   4206     ;    VARIABLES MODIFIED: PARAM1, A, TEMP
               =1   4207     ;
               =1   4208     ;    ERROR EXITS: None
               =1   4209     ;
               =1   4210     ;    SUBROUTINES ACCESSED DIRECTLY: UPI_IN, IASCII_TO_HEX, ICI
               =1   4211     ;
               =1   4212     ;
               =1   4213     ;*****************************************************************
F340 12E64C    =1   4214     HEXBIN: CALL    UPI_IN
F343 20050E    =1   4215             JB      BINARY_FLG,BINARY_LOAD
F346 FA        =1   4216             MOV     PARAM1,A
F347 12EA3C    =1   4217             CALL    IASCII_TO_HEX
F34A C4        =1   4218             SWAP    A                       ;Move the digit to the upper nibble.
F34B FD        =1   4219             MOV     TEMP,A                  ;Then save it in a temporary location.
F34C 12E5EB    =1   4220             CALL    ICI
F34F FA        =1   4221             MOV     PARAM1,A
F350 12EA3C    =1   4222             CALL    IASCII_TO_HEX
F353 4D        =1   4223             ORL     A,TEMP                  ;Then combine with previous digit.
               =1   4224     BINARY_LOAD:
F354 CE        =1   4225             XCH     A,CHECKSUM              ;Before returning the binary value
F355 2E        =1   4226             ADD     A,CHECKSUM              ;include it in checksum calculation.
F356 CE        =1   4227             XCH     A,CHECKSUM
F357 22        =1   4228             RET
               =1   4229 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1  4230     ;*****************************************************************
                =1  4231     ;
                =1  4232     ;    NAME: GET_TYPE
                =1  4233     ;
                =1  4234     ;    ABSTRACT: This routine looks for a colon from the cassette or
                =1  4235     ;       auxilary termimal input, gets the byte count, address and
                =1  4236     ;       file-type information contained in the header and does a checksum.
                =1  4237     ;
                =1  4238     ;    INPUTS: None
                =1  4239     ;
                =1  4240     ;    OUTPUTS: TYPE, PNTLOW, PNTHGH, COUNT, CHECKSUM
                =1  4241     ;
                =1  4242     ;    VARIABLES MODIFIED: A, CHECKSUM, COUNT, PNTHGH, PNTLOW, TYPE
                =1  4243     ;
                =1  4244     ;    ERROR EXITS: None
                =1  4245     ;
                =1  4246     ;    SUBROUTINES ACCESSED DIRECTLY: ICI, HEXBIN
                =1  4247     ;
                =1  4248     ;
                =1  4249     ;*****************************************************************
                =1  4250     GET_TYPE:
F358 12E64C     =1  4251             CALL    UPI_IN                  ;Scan for a colon.
F35B 547F       =1  4252             ANL     A,#7FH
F35D B43AF8     =1  4253             CJNE    A,#':',GET_TYPE
F360 E4         =1  4254             CLR     A
F361 FE         =1  4255             MOV     CHECKSUM,A
F362 7140       =1  4256             CALL    HEXBIN                  ;Load the byte count from
F364 FF         =1  4257             MOV     COUNT,A                 ;the next two characters of the record.
F365 7140       =1  4258             CALL    HEXBIN                  ;Load the load address
F367 F544       =1  4259             MOV     PNTHGH,A
F369 7140       =1  4260             CALL    HEXBIN
F36B F545       =1  4261             MOV     PNTLOW,A
F36D 7140       =1  4262             CALL    HEXBIN                  ;Load the record type.
F36F F565       =1  4263             MOV     TYPE,A
F371 22         =1  4264             RET
                =1  4265 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

               =1   4266     ;*****************************************************************
               =1   4267     ;
               =1   4268     ;    NAME: LOAD_HEX
               =1   4269     ;
               =1   4270     ;    ABSTRACT: Loads audio cassette data files (type 0) until EOF
               =1   4271     ;       is encountered.  Calculates a checksum, passes label (addr), writes
               =1   4272     ;       user PC, converts hex data to binary and returns.
               =1   4273     ;
               =1   4274     ;    INPUTS: None
               =1   4275     ;
               =1   4276     ;    OUTPUTS: Code memory locations addressed in the file being loaded.
               =1   4277     ;
               =1   4278     ;    VARIABLES MODIFIED: A, PARAM1, SELECT, PNTLOW, PNTHGH, ERRNUM
               =1   4279     ;
               =1   4280     ;    ERROR EXITS: None
               =1   4281     ;
               =1   4282     ;    SUBROUTINES ACCESSED DIRECTLY: GET_TYPE, HEXBIN, INIT_IO,
               =1   4283     ;       ISTORE, WRITE_PC, ITIME
               =1   4284     ;
               =1   4285     ;
               =1   4286     ;*****************************************************************
               =1   4287     LOAD_HEX:
F372 7158      =1   4288             CALL    GET_TYPE
F374 7019      =1   4289             JNZ     LH_7                ;If type is not zero (data record)
               =1   4290                                         ;then quit loading records.
F376 EF        =1   4291     LH_4:   MOV     A,COUNT             ;Load memory until the count gets
F377 600E      =1   4292             JZ      LH_6                ;to zero, COUNT=length read from file
F379 7140      =1   4293             CALL    HEXBIN
F37B FA        =1   4294             MOV     PARAM1,A
F37C 754600    =1   4295             MOV     SELECT,#0
F37F 12E672    =1   4296             CALL    ISTORE              ;Increment the load address.
F382 12E5C4    =1   4297             CALL    INC_PNT             ;Repeat the load loop until zero.
 385 DFEF      =1   4298             DJNZ    COUNT,LH_4
  87 7140      =1   4299     LH_6:   CALL    HEXBIN              ;The end of the record has been reached
  89 EE        =1   4300             MOV     A,CHECKSUM          ;so check the checksum field.
F38A 60E6      =1   4301             JZ      LOAD_HEX            ;Recall CHECKSUM from HEXBIN
F38C 02F447    =1   4302     LH_8:   JMP     LH_ERROR
F38F B401FA    =1   4303     LH_7:   CJNE    A,#1,LH_8           ;Look for EOF (type 1)
F392 7140      =1   4304             CALL    HEXBIN
F394 EE        =1   4305             MOV     A,CHECKSUM
F395 70F5      =1   4306             JNZ     LH_8
F397 12E386    =1   4307             CALL    INIT_IO
F39A AB45      =1   4308             MOV     PARAM2,PNTLOW       ;Write addr (label) to user PC
F39C AA44      =1   4309             MOV     PARAM1,PNTHGH
F39E 12EFA8    =1   4310             CALL    WRITE_PC
F3A1 7A07      =1   4311             MOV     PARAM1,#07H
F3A3 7B00      =1   4312             MOV     PARAM2,#00H
F3A5 12EA45    =1   4313             CALL    ITIME               ;Wait for 2 char times at 110 baud
F3A8 90A000    =1   4314             MOV     DPTR,#UPI_DATA      ;So no other chars get into the
F3AB E0        =1   4315             MOVX    A,@DPTR             ;Command buffer.  Flush output
 3AC 22        =1   4316             RET                         ;buffer flag.
               =1   4317 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

               =1  4318     ;*******************************************************************
               =1  4319     ;
               =1  4320     ;     NAME: STORE_HEX
               =1  4321     ;
               =1  4322     ;     ABSTRACT: This routine writes hex bytes on the cassette or to
               =1  4323     ;        the USART from memory.  It outputs all record marks and header
               =1  4324     ;        information and calculates a checksum.
               =1  4325     ;
               =1  4326     ;     INPUTS: BINARY_FLG, PARTIT_LO_LOW, PARTIT_LO_HIGH, PARTIT_HI_LOW,
               =1  4327     ;        PARTIT_HI_HIGH, Memory contents within the partition bounds.
               =1  4328     ;
               =1  4329     ;     OUTPUTS: None
               =1  4330     ;
               =1  4331     ;     VARIABLES MODIFIED: PARAM1, PARAM1, C, A, COUNT, TEMP, CHECKSUM,
               =1  4332     ;        SELECT, PNTHGH, PNTLOW, PARTIT_LO_LOW, PARTIT_LO_HIGH,
               =1  4333     ;        ERRNUM
               =1  4334     ;
               =1  4335     ;     ERROR EXITS: 14H (FILE READ/WRITE ERROR)
               =1  4336     ;
               =1  4337     ;     SUBROUTINES ACCESSED DIRECTLY: INEWLINE, ITIME, SEND_BYTE,
               =1  4338     ;        IFETCH, READ_PC, UPI_CMD, ICO, IERROR
               =1  4339     ;
               =1  4340     ;
               =1  4341     ;*******************************************************************
               =1  4342     STORE_HEX:
F3AD 200511    =1  4343             JB      BINARY_FLG,SH_6
F3B0 7A01      =1  4344             MOV     PARAM1,#01H            ;Delay 40 milliseconds.
F3B2 7B90      =1  4345             MOV     PARAM2,#90H
F3B4 12EA45    =1  4346             CALL    ITIME
F3B7 12E717    =1  4347             CALL    INEWLINE               ;Start sending record.
F3BA 7A13      =1  4348             MOV     PARAM1,#13H
F3BC 7B88      =1  4349             MOV     PARAM2,#88H            ;Delay 1/2 sec.
F3BE 12EA45    =1  4350             CALL    ITIME
F3C1 7A3A      =1  4351     SH_6:   MOV     PARAM1,#':'            ;Output the record mark.
F3C3 12E5E8    =1  4352             CALL    ICO
F3C6 C3        =1  4353             CLR     C                      ;Output hex records while sa<=ea.
F3C7 E55A      =1  4354             MOV     A,PARTIT_HI_LOW
F3C9 9558      =1  4355             SUBB    A,PARTIT_LO_LOW
F3CB FF        =1  4356             MOV     COUNT,A                ;(Save difference for later use).
F3CC E559      =1  4357             MOV     A,PARTIT_HI_HIGH
F3CE 9557      =1  4358             SUBB    A,PARTIT_LO_HIGH
F3D0 FD        =1  4359             MOV     TEMP,A                 ;Set count to 16 or the number of bytes
F3D1 403E      =1  4360             JC      SH_5                   ;left-whichever is less.
F3D3 ED        =1  4361             MOV     A,TEMP
F3D4 6002      =1  4362             JZ      SH_1
F3D6 7F0F      =1  4363             MOV     COUNT,#0FH
F3D8 EF        =1  4364     SH_1:   MOV     A,COUNT
F3D9 54F0      =1  4365             ANL     A,#0F0H
F3DB 6002      =1  4366             JZ      SH_2
F3DD 7F0F      =1  4367             MOV     COUNT,#0FH
F3DF 0F        =1  4368     SH_2:   INC     COUNT
F3E0 E4        =1  4369             CLR     A
F3E1 FE        =1  4370             MOV     CHECKSUM,A
F3E2 EF        =1  4371             MOV     A,COUNT
F3E3 7133      =1  4372             CALL    SEND_BYTE
```

```
LOC  OBJ            LINE     SOURCE

F3E5 E557      =1   4373             MOV    A,PARTIT_LO_HIGH
F3E7 7133      =1   4374             CALL   SEND_BYTE
F3E9 E558      =1   4375             MOV    A,PARTIT_LO_LOW
F3EB 7133      =1   4376             CALL   SEND_BYTE
F3ED E4        =1   4377             CLR    A
F3EE 7133      =1   4378             CALL   SEND_BYTE
               =1   4379     SH_3:                              ;Now go into a loop to output the data.
F3F0 754600    =1   4380             MOV    SELECT,#00H         ;Call fetch(0,sa).
F3F3 855744    =1   4381             MOV    PNTHGH,PARTIT_LO_HIGH
F3F6 855845    =1   4382             MOV    PNTLOW,PARTIT_LO_LOW
F3F9 12E66B    =1   4383             CALL   IFETCH
F3FC 7133      =1   4384             CALL   SEND_BYTE
F3FE E558      =1   4385             MOV    A,PARTIT_LO_LOW     ;Increment the address
F400 2401      =1   4386             ADD    A,#01H
F402 F558      =1   4387             MOV    PARTIT_LO_LOW,A
F404 5002      =1   4388             JNC    SH_4
 406 0557      =1   4389             INC    PARTIT_LO_HIGH
 408 DFE6      =1   4390     SH_4:   DJNZ   COUNT,SH_3          ;Decrement count and loop till zero.
F40A EE        =1   4391             MOV    A,CHECKSUM          ;Once done output the negation of the
F40B F4        =1   4392             CPL    A                   ;checksum.
F40C 04        =1   4393             INC    A
F40D 7133      =1   4394             CALL   SEND_BYTE           ;Then go output another record
F40F 809C      =1   4395             JMP    STORE_HEX
F411 E4        =1   4396     SH_5:   CLR    A
F412 FE        =1   4397             MOV    CHECKSUM,A
F413 7133      =1   4398             CALL   SEND_BYTE
F415 12EF9D    =1   4399             CALL   READ_PC
F418 C5F0      =1   4400             XCH    A,B
F41A 7133      =1   4401             CALL   SEND_BYTE
F41C E5F0      =1   4402             MOV    A,B
F41E 7133      =1   4403             CALL   SEND_BYTE
F420 E4        =1   4404             CLR    A
 421 04        =1   4405             INC    A
 422 7133      =1   4406             CALL   SEND_BYTE
 424 EE        =1   4407             MOV    A,CHECKSUM
F425 F4        =1   4408             CPL    A
F426 04        =1   4409             INC    A
F427 7133      =1   4410             CALL   SEND_BYTE
F429 7A01      =1   4411             MOV    PARAM1,#1
F42B 7B90      =1   4412             MOV    PARAM2,#90H
F42D 12EA45    =1   4413             CALL   ITIME
F430 12E717    =1   4414             CALL   INEWLINE
F433 20050A    =1   4415             JB     BINARY_FLG,SH_7     ;Skip control-Z if cassette operation.
F436 7A01      =1   4416             MOV    PARAM1,#USART_MODE  ;Select USART mode.
F438 12E625    =1   4417             CALL   UPI_CMD
F43B 7A1A      =1   4418             MOV    PARAM1,#1AH         ;Insert control Z to close MDS file
F43D 12E5E8    =1   4419             CALL   ICO
F440 7A13      =1   4420     SH_7:   MOV    PARAM1,#13H
F442 7B88      =1   4421             MOV    PARAM2,#88H
F444 02EA45    =1   4422             JMP    ITIME               ;Delay 1/2 sec to catch cntrl Z in list mode
               =1   4423     LH_ERROR:
 447 754314    =1   4424             MOV    ERRNUM,#14H         ;File read/write error
 44A 12E3E4    =1   4425             CALL   IERROR
               =1   4426 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1           ;****************************************************************
                =1           ;
                =1           ;    NAME: LOAD_CMD
                =1           ;
                =1           ;    ABSTRACT: This routine calls the routine LOAD_HEX which
                =1           ;       reads data files from the audio cassette in binary. It sets
                =1           ;       up the user messages and does checksums.
                =1           ;
                =1           ;    INPUTS: None
                =1           ;
                =1           ;    OUTPUTS: Code memory locations referenced by the file being loaded.
                =1           ;
                =1           ;    VARIABLES MODIFIED: PCNHTI, PCNTLO, BINARY_FLG, PARAM1, A,
                =1           ;       PARAM2
                =1           ;
                =1           ;    ERROR EXITS: None
                =1           ;
                =1           ;    SUBROUTINES ACCESSED DIRECTLY: IGETOKE, IPRINT_STRING,
                =1           ;       ICI, UPI_CMD, GET_TYPE, HEXBIN, LOAD_HEX, INIT_IO, ILSTWRD,
                =1           ;       IWAIT_FOR_USER
                =1           ;
                =1           ;
                =1           ;****************************************************************
                =1           LOAD_CMD:
F44D 12E8BC     =1                   CALL    IGETOKE                         ;Have a valid LOAD cmd
F450 854961     =1                   MOV     PCNTHI,VALHGH                   ;Save addr (label) field
F453 854A62     =1                   MOV     PCNTLO,VALLOW
F456 7AF5       =1                   MOV     PARAM1,#HIGH CASS_MSG           ;Set up °start cassette° msg
F458 7B2D       =1                   MOV     PARAM2,#LOW CASS_MSG
F45A 12E9FF     =1                   CALL    IPRINT_STRING
F45D 12E5EB     =1                   CALL    ICI                             ;Holds msg on display long enough to be see
                             n
F460 D205       =1                   SETB    BINARY_FLG                      ;Indicates a binary file
F462 7A02       =1                   MOV     PARAM1,#CASSETTE_READ
F464 12E625     =1                   CALL    UPI_CMD                         ;Select cassette mode
F467 E548       =1                   MOV     A,TOKSTR                        ;Restore original token
F469 B4012E     =1                   CJNE    A,#NUMBER_TOKE,FILE_DISPLAY     ;If not a number, need to get next
                =1           LOAD_LOOP:                                      ;Get number off cass and display it (direct
                             ory)
F46C 7158       =1                   CALL    GET_TYPE                        ;0=data file, 1=EOF, 2=file label record
F46E B402FB     =1                   CJNE    A,#2,LOAD_LOOP                  ;Is it the beginning of a file?
F471 E561       =1                   MOV     A,PCNTHI                        ;Yes, get the label (addr)
F473 B544F6     =1                   CJNE    A,PNTHGH,LOAD_LOOP
F476 E562       =1                   MOV     A,PCNTLO
F478 B545F1     =1                   CJNE    A,PNTLOW,LOAD_LOOP
F47B 7140       =1                   CALL    HEXBIN                          ;Convert to hex, calculate checksum
F47D EE         =1                   MOV     A,CHECKSUM
F47E 70C7       =1                   JNZ     LH_ERROR                        ;Checksum error
F480 7172       =1                   CALL    LOAD_HEX                        ;Read the data file from cassette
F482 12E386     =1                   CALL    INIT_IO
F485 90A000     =1                   MOV     DPTR,#UPI_DATA
F488 E0         =1                   MOVX    A,@DPTR                         ;Go back to console mode, clear OBF status
                             bit
F489 7AF5       =1                   MOV     PARAM1,#HIGH FILE_FOUND         ;Set up °File loaded° msg
F48B 7B3E       =1                   MOV     PARAM2,#LOW FILE_FOUND
```

```
LOC  OBJ            LINE     SOURCE

F48D 12E9FF     =1  4479              CALL    IPRINT_STRING
F490 AA61       =1  4480              MOV     PARAM1,PCNTHI                   ;Set up file number for display
F492 AB62       =1  4481              MOV     PARAM2,PCNTLO
F494 12E7F4     =1  4482              CALL    ILSTWRD
F497 02E3B0     =1  4483              JMP     IWAIT_FOR_USER                  ;Holds msg on display a short time
                =1  4484     FILE_DISPLAY:
F49A 7158       =1  4485              CALL    GET_TYPE                        ;Get here by saying LOAD <CR>
F49C B402FB     =1  4486              CJNE    A,#2,FILE_DISPLAY               ;Ask for directory, cant load w/o file #
F49F 12E386     =1  4487              CALL    INIT_IO
F4A2 90A000     =1  4488              MOV     DPTR,#UPI_DATA
F4A5 E0         =1  4489              MOVX    A,@DPTR                         ;Go back to console mode, clr OBF status bi
                             t
F4A6 7AF5       =1  4490              MOV     PARAM1,#HIGH NUM_FOUND          ;Sets up °first file found° msg
F4A8 7B4E       =1  4491              MOV     PARAM2,#LOW NUM_FOUND
F4AA 12E9FF     =1  4492              CALL    IPRINT_STRING
F4AD AA44       =1  4493              MOV     PARAM1,PNTHGH                   ;Set up file number (addr) for display
F4AF AB45       =1  4494              MOV     PARAM2,PNTLOW
F4B1 12E7F4     =1  4495              CALL    ILSTWRD
F4B4 02E3B0     =1  4496              JMP     IWAIT_FOR_USER                  ;Holds msg on display a short time
                =1  4497 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

               =1   4498     ;*****************************************************************
               =1   4499     ;
               =1   4500     ;    NAME: SAVE_CMD
               =1   4501     ;
               =1   4502     ;    ABSTRACT: This routine writes data in a user specified partition
               =1   4503     ;       to the audio cassette in binary using STORE_HEX which provides
               =1   4504     ;       address, type and checksum for each record. This procedure
               =1   4505     ;       takes care of all UPI set up.
               =1   4506     ;
               =1   4507     ;    INPUTS: Code memory within the partition
               =1   4508     ;
               =1   4509     ;    OUTPUTS: None
               =1   4510     ;
               =1   4511     ;    VARIABLES MODIFIED: PCNTHI, PCNTLO, PARAM1, PARAM2, BINARY_FLG
               =1   4512     ;       A, CHECKSUM
               =1   4513
               =1   4514     ;    ERROR EXITS: None
               =1   4515     ;
               =1   4516     ;    SUBROUTINES ACCESSED DIRECTLY: IGETNUM, IGETOKE, IGET_PART, IPRINT_STRING,
               =1   4517     ;       ICI, UPI_CMD, ICO, SEND_BYTE, IGET_COMMA, IEOL_CHECK, STORE_HEX
               =1   4518     ;
               =1   4519     ;
               =1   4520     ;*****************************************************************
               =1   4521     SAVE_CMD:
F4B7 12E769    =1   4522             CALL    IGETNUM
F4BA 854961    =1   4523             MOV     PCNTHI,VALHGH
F4BD 854A62    =1   4524             MOV     PCNTLO,VALLOW
F4C0 12E77A    =1   4525             CALL    IGET_COMMA
F4C3 12E8BC    =1   4526             CALL    IGETOKE
F4C6 12E7A2    =1   4527             CALL    IGET_PART
F4C9 12E5BB    =1   4528             CALL    IEOL_CHECK
F4CC 7AF5      =1   4529             MOV     PARAM1,#HIGH CASS_MSG
F4CE 7B2D      =1   4530             MOV     PARAM2,#LOW CASS_MSG
F4D0 12E9FF    =1   4531             CALL    IPRINT_STRING
F4D3 12E5EB    =1   4532             CALL    ICI
F4D6 D205      =1   4533             SETB    BINARY_FLG
F4D8 7A82      =1   4534             MOV     PARAM1,#CASSETTE_WRITE
F4DA 12E625    =1   4535             CALL    UPI_CMD                  ;Select cassette mode
F4DD 7A3A      =1   4536             MOV     PARAM1,#':'
F4DF 12E5E8    =1   4537             CALL    ICO
F4E2 E4        =1   4538             CLR     A
F4E3 FE        =1   4539             MOV     CHECKSUM,A
F4E4 7133      =1   4540             CALL    SEND_BYTE
F4E6 E561      =1   4541             MOV     A,PCNTHI
F4E8 7133      =1   4542             CALL    SEND_BYTE
F4EA E562      =1   4543             MOV     A,PCNTLO
F4EC 7133      =1   4544             CALL    SEND_BYTE
F4EE 7402      =1   4545             MOV     A,#2
F4F0 7133      =1   4546             CALL    SEND_BYTE
F4F2 EE        =1   4547             MOV     A,CHECKSUM
F4F3 F4        =1   4548             CPL     A
F4F4 04        =1   4549             INC     A
F4F5 7133      =1   4550             CALL    SEND_BYTE
F4F7 61AD      =1   4551             JMP     STORE_HEX
               =1   4552 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

               =1   4553     ;*****************************************************************
               =1   4554     ;
               =1   4555     ;    NAME: DOWNLOAD_CMD
               =1   4556     ;
               =1   4557     ;    ABSTRACT: This routine temporarily turns off the list mode,
               =1   4558     ;       selects the console, configures the UPI and loads hex files
               =1   4559     ;       from the auxilary terminal into memory.
               =1   4560     ;
               =1   4561     ;    INPUTS: None
               =1   4562     ;
               =1   4563     ;    OUTPUTS: Code memory location specified in the file being loaded.
               =1   4564     ;
               =1   4565     ;    VARIABLES MODIFIED: PARAM1, PARAM2, BINARY_FLG
               =1   4566     ;
               =1   4567     ;    ERROR EXITS: None
               =1   4568     ;
               =1   4569     ;    SUBROUTINES ACCESSED DIRECTLY: IPRINT_STRING, UPI_CMD,
               =1   4570     ;       LOAD_HEX
               =1   4571     ;
               =1   4572     ;
               =1   4573     ;*****************************************************************
               =1   4574     DOWNLOAD_CMD:
F4F9 C205      =1   4575             CLR     BINARY_FLG                ;Set "LIST = RESET"
F4FB 7A00      =1   4576             MOV     PARAM1,#SELECT_CON
F4FD 12E625    =1   4577             CALL    UPI_CMD
F500 7AF5      =1   4578             MOV     PARAM1,#HIGH LOAD_MSG
F502 7B23      =1   4579             MOV     PARAM2,#LOW LOAD_MSG
F504 12E9FF    =1   4580             CALL    IPRINT_STRING             ;Print loading msg
F507 7A01      =1   4581             MOV     PARAM1,#USART_MODE
F509 12E625    =1   4582             CALL    UPI_CMD                   ;Select USART mode
F50C 7172      =1   4583             CALL    LOAD_HEX
F50E 22        =1   4584             RET
               =1   4585 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

               =1   4586     ;*****************************************************************
               =1   4587     ;
               =1   4588     ;    NAME: UPLOAD_CMD
               =1   4589     ;
               =1   4590     ;    ABSTRACT: This routine gets a token and partition, turns off
               =1   4591     ;       list mode and outputs hex files to the console through the
               =1   4592     ;       UPI.
               =1   4593     ;
               =1   4594     ;    INPUTS: Code memory locations specified by the partition typed
               =1   4595     ;       by the user.
               =1   4596     ;
               =1   4597     ;    OUTPUTS: None
               =1   4598     ;
               =1   4599     ;    VARIABLES MODIFIED: PARAM1, BINARY_FLG, LSTFLG
               =1   4600     ;
               =1   4601     ;    ERROR EXITS: None
               =1   4602     ;
               =1   4603     ;    SUBROUTINES ACCESSED DIRECTLY: IGET_PART, IGETOKE,
               =1   4604     ;       UPI_CMD, STORE_HEX, IEOL_CHECK
               =1   4605     ;
               =1   4606     ;
               =1   4607     ;*****************************************************************
               =1   4608     UPLOAD_CMD:
F50F 12E8BC    =1   4609             CALL    IGETOKE
F512 12E7A2    =1   4610             CALL    IGET_PART
F515 12E5BB    =1   4611             CALL    IEOL_CHECK
F518 C205      =1   4612             CLR     BINARY_FLG
F51A C201      =1   4613             CLR     LSTFLG                  ;Set 'LIST = RESET'
F51C 7A40      =1   4614             MOV     PARAM1,#40H             ;Select Keybd/Disply with list on.
F51E 12E625    =1   4615             CALL    UPI_CMD
F521 61AD      =1   4616             JMP     STORE_HEX
               =1   4617     ;*****************************************************************
F523 09        =1   4618     LOAD_MSG:       DB      9,CR,LF,('LOADING')
F524 0D
F525 0A
F526 4C4F4144
F52A 494E47
F52D 10        =1   4619     CASS_MSG:       DB      16,CR,LF,('START CASSETTE')
F52E 0D
F52F 0A
F530 53544152
F534 54204341
F538 53534554
F53C 5445
F53E 0F        =1   4620     FILE_FOUND:     DB      15,CR,LF,('LOADED FILE  ')
F53F 0D
F540 0A
F541 4C4F4144
F545 45442046
F549 494C4520
F54D 20
F54E 13        =1   4621     NUM_FOUND:      DB      19,('FIRST FILE FOUND = ')
F54F 46495253
F553 54204649
F557 4C452046
```

```
LOC  OBJ            LINE     SOURCE

F55B 4F554E44
F55F 203D20
               =1   4622     ;*****************************************************************
               =1   4623     VERIFY_CMD:
F562 906009    =1   4624             MOV     DPTR,#6009H
F565 800D      =1   4625             SJMP    JMP_TAB_CHECKER
               =1   4626     TRANSFER_CMD:
F567 906006    =1   4627             MOV     DPTR,#6006H
F56A 8008      =1   4628             SJMP    JMP_TAB_CHECKER
               =1   4629     PROGRAM_CMD:
F56C 906003    =1   4630             MOV     DPTR,#6003H
F56F 8003      =1   4631             SJMP    JMP_TAB_CHECKER
               =1   4632     MODE_CMD:
F571 906000    =1   4633             MOV     DPTR,#6000H
               =1   4634     JMP_TAB_CHECKER:
F574 E4        =1   4635             CLR     A
F575 93        =1   4636             MOVC    A,@A+DPTR
F576 B40202    =1   4637             CJNE    A,#2,FAKE_BAD_CMD_ERR          ;Check for first byte of LJMP opcode
F579 E4        =1   4638             CLR     A
F57A 73        =1   4639             JMP     @A+DPTR
               =1   4640     FAKE_BAD_CMD_ERR:
F57B 754302    =1   4641             MOV     ERRNUM,#02H
F57E 02E3E4    =1   4642             JMP     IERROR
                    4643     ASMBASE:
                    4644     END
```

```
XREF SYMBOL TABLE LISTING
---- ------ ----- -------


NAME                 TYPE      VALUE AND REFERENCES

A_TOKE. . . . . . . N         0051H  450# 552
AB_TOKE . . . . . . N         005CH  451# 553
ABR_TOKE. . . . . . N         0088H  452# 554 916 2891
ACALL_TOKE. . . . . N         0012H  453# 555
ACC . . . . . . . . N DSEG    00E0H PREDEFINED  837 838 905 1127 1151 1299 1306 1338 1402 1601 1607 1616 2082 2122 2123
                                2126 2127 2520 2951 2972 3038 3090 3222 3225 3228 3288 3291 3323 3327 3348 3360 3364
                                3435 3438 3467 3470 3471 3719 4067 4129
ACC_CMD . . . . . . L CSEG    ED2AH  919 3089#
ACC_TOKE. . . . . . N         0098H  454# 556 918
ADD_TOKE. . . . . . N         0024H  455# 557
ADDC_TOKE . . . . . N         0023H  456# 558
ADDR_SAVE_HIGH. . . N         00F3H  401# 3635
ADDR_SAVE_LOW . . . N         00F4H  402#
AJMP_TOKE . . . . . N         0013H  457# 559
ALFNUM. . . . . . . L CSEG    E744H  1723# 2180 2190
ALPHA . . . . . . . L CSEG    E8C7H  2165 2168#
ANEND . . . . . . . L CSEG    E74AH  1725 1728#
ANL_TOKE. . . . . . N         0021H  458# 560
ANY_BR_FLAG . . . . L BSEG    0002H  437# 2927 2936 2957
ASM_PC_HIGH . . . . L DSEG    004BH  256# 830
ASM_PC_LOW. . . . . L DSEG    004CH  257# 831
ASM_TOKE. . . . . . N         00B0H  459# 561 920
ASMBASE . . . . . . L CSEG    F581H  921 933 4643#
ATA_TOKE. . . . . . N         000AH  224# 548
ATDPTR_TOKE . . . . N         005FH  447# 549
ATR0_TOKE . . . . . N         0052H  448# 550
ATR1_TOKE . . . . . N         0053H  449# 551
AZEND . . . . . . . L CSEG    E72CH  1701 1703#
AZTEST. . . . . . . L CSEG    E720H  1697# 1723 2177
B . . . . . . . . . N DSEG    00F0H PREDEFINED  836 901 906 1639 1640 2252 2260 2261 2379 2385 2596 2598 3102 3146 3238
                                3239 3333 3335 3414 3416 3509 3512 3704 3714 3881 3989 3992 3995 4000 4068 4072 4074
                                4400 4402
B_CMD . . . . . . . L CSEG    ED3CH  923 3101#
B_LAB_1 . . . . . . L CSEG    EAA5H  2645 2647#
B_LAB_2 . . . . . . L CSEG    EAAAH  2647 2649#
B_LAB_3 . . . . . . L CSEG    EAAFH  2649 2651#
B_O_T . . . . . . . L BSEG    0000H  277# 2163 2217
B_TOKE. . . . . . . N         009BH  460# 562 922
B_V_ERR . . . . . . L CSEG    EFD6H  3592 3595#
BACKSP. . . . . . . N         0008H  371# 2099 2102
BAR_TOKE. . . . . . N         0003H  220# 2335
BASE. . . . . . . . N         E000H  216# 296 297 298 299 300 301 302 303 305 306 307 308 309 310 311 312 313 314 315
                                316 317 320 347 1069
BAUD_CMD. . . . . . L CSEG    F1FDH  925 3958#
BAUD_DISPLAY. . . . L CSEG    F26AH  3960 4029#
BAUD_HIGH . . . . . N         00F7H  405# 817 3975 4030
BAUD_LOW. . . . . . N         00F8H  406#
BAUD_RATE . . . . . L CSEG    F255H  3962 4012#
BAUD_TOKE . . . . . N         00D0H  461# 563 924
BAUDKEY . . . . . . N         00FCH  410# 813 3981 3986
BEND. . . . . . . . L CSEG    EBC6H  2835 2850 2853#
```

```
NAME                 TYPE      VALUE AND REFERENCES

BINARY_FLG. . . . . L BSEG    0005H   440# 4188 4215 4343 4415 4458 4533 4575 4612
BINARY_LOAD . . . . L CSEG    F354H   4215 4224#
BITLOP. . . . . . . L CSEG    E6EEH   1628# 1632
BITROT. . . . . . . L CSEG    E6F5H   1628 1631#
BITSTR. . . . . . . L CSEG    E6F8H   1623 1633#
BK1LOP. . . . . . . L CSEG    EC68H   2958# 2974
BLINK . . . . . . . N         0080H   234# 1522 2069
BM_1. . . . . . . . L CSEG    F24CH   3969 4006#
BMOVE . . . . . . . L CSEG    EB58H   2601 2804#
BR_CMD. . . . . . . L CSEG    EBC7H   917 927 2887#
BR_TOKE . . . . . . N         0089H   462# 564 926 2899 3013
BREAK . . . . . . . L CSEG    E003H   328# 839 3584 3588 3592
BREAK_CONTINUE. . . L CSEG    EE83H   3300 3306 3317#
BREAK_MSG . . . . . L CSEG    F1B6H   3877 3878 3889#
BREAK_STATUS. . . . N         00FBH   409# 3298 3351 3663 3763 3859
 REAK_VECTOR. . . . L CSEG    EFC2H   3580# 3665 3862
  K_LINE_HDR. . . . L CSEG    ECD5H   2938 2981 3011#
 RK_LOOP. . . . . . L CSEG    EE3FH   3281# 3286
BRK1. . . . . . . . L CSEG    EED2H   3353 3355#
BRK2. . . . . . . . L CSEG    EEDBH   3357 3359#
BRK3. . . . . . . . L CSEG    EE97H   3323 3329#
BRK4. . . . . . . . L CSEG    EEA7H   3325 3327 3337#
BRK5. . . . . . . . L CSEG    EEA1H   3332 3334#
BRKEND. . . . . . . L CSEG    EC44H   2936 2941#
BRKERR. . . . . . . L CSEG    ED24H   3038 3063#
BRKMORE . . . . . . L CSEG    EE78H   3308 3312#
BRKOFF. . . . . . . N         C000H   392# 2929 2958 3023 3045 3302
BS_2. . . . . . . . L CSEG    F24EH   3967 4008#
BS_LOOP . . . . . . L CSEG    F20AH   3964# 4009
C_READ. . . . . . . L CSEG    E68CH   1575 1583#
C_TOKE. . . . . . . N         005EH   225# 565
CARSET. . . . . . . L CSEG    E72BH   1699 1700 1702#
 ASS_MSG. . . . . . L CSEG    F52DH   4454 4455 4529 4530 4619#
  SSETTE_READ . . . N         0002H   383# 4459
 ASSETTE_WRITE. . . N         0082H   384# 4534
CAUSE_CMD . . . . . L CSEG    F2B8H   929 4119#
CAUSE_IMAGE . . . . L DSEG    0060H   426# 832 3304 3310 3316 3322 3326 3355 3359 3709 4123
CAUSE_TAB . . . . . L CSEG    F2DCH   4124 4143#
CAUSE_TOKE. . . . . N         00D2H   463# 566 928
CBYTE_TOKE. . . . . N         0080H   226# 567 930 1574 2600 2645 2767 2806
CHANGE. . . . . . . L CSEG    ED57H   3109 3114#
CHANGE_CHECK. . . . L CSEG    E79BH   1841 1849#
CHARIN. . . . . . . L DSEG    0050H   261# 1006 2072 2092 2125 2131 2164 2176 2187 2193 2255 2267 2271 2275 2343
CHECK_ABREV . . . . L CSEG    E913H   2198 2203#
CHECK_EPROMS. . . . L CSEG    E3BAH   795 1068#
CHECK_ESC . . . . . L CSEG    E664H   1524# 1528
CHECK_FROM. . . . . L CSEG    EFB1H   3545# 3622 3817
CHECK_LOOP. . . . . L CSEG    E3BFH   1071# 1078
CHECK_OUT_OK. . . . L CSEG    E3E3H   1080 1091#
CHECKSUM. . . . . . N REG     R6      290# 1070 1074 1075 1079 4184 4185 4186 4225 4226 4227 4255 4300 4305 4370 4391
                                  4397 4407 4471 4539 4547
  RCNT. . . . . . . L DSEG    0051H   262# 1004 2053 2060 2089 2093
  I . . . . . . . . N         E009H   297#
CJNE_TOKE . . . . . N         0019H   464# 568
CL_LOOP . . . . . . L CSEG    F2C7H   4126# 4132
```

```
NAME                 TYPE      VALUE AND REFERENCES

CL_O. . . . . . . . . L CSEG   F2CFH  4129 4133#
CLR_BRK_LATCHES . . N          0008H  379# 1011 3668 3868
CLR_TOKE. . . . . . N          002AH  465# 569
CLRBRK. . . . . . . L CSEG     ECE1H  790 2900 2921 3021#
CLRLOP. . . . . . . L CSEG     ECEAH  3025# 3027 3029
CMDTAB. . . . . . . L CSEG     E30EH  891 892 915#
CO. . . . . . . . . N          E006H  296#
COMMA_TOKE. . . . . N          0002H  370# 1813 2334 2639 2904 3627 3645
CONTINUATION_LINE . N          E068H  315#
CONTINUOUS_MODE . . N          0040H  395# 3997
CONVHEX . . . . . . L CSEG     E7EBH  1932 1936 1959# 1993 1997
COPYRIGHT . . . . . L CSEG     E030H  350#
COUNT . . . . . . . N REG      R7     289# 2435 2443 4125 4132 4257 4291 4298 4356 4363 4364 4367 4368 4371 4390
COUNT1. . . . . . . L CSEG     EB3DH  2754 2756 2758#
COUNTR. . . . . . . L DSEG     005DH  423# 2733 2734 2735 2749 2758
CPL_TOKE. . . . . . N          002BH  466# 570
CR. . . . . . . . . N          000DH  372# 981 1156 1667 2074 2078 2279 2339 2342 3888 3890 4618 4619 4620
CRWAIT. . . . . . . L CSEG     E828H  2063# 2098 2105 2108 2120 2124 2134
CSTS. . . . . . . . N          E00CH  298#
CSTS_1. . . . . . . L CSEG     E609H  1337# 1338
DA_TOKE . . . . . . N          002CH  467# 572
DASM_TOKE . . . . . N          00B8H  468# 571 573 932
DATA_BREAK. . . . . N          000DH  387# 3838 3851
DATA_MSG. . . . . . L CSEG     F310H  4147 4156#
DATA_TOKE . . . . . N          00D3H  469# 574 3835 3848
DATECODE. . . . . . L CSEG     E046H  351#
DBYTE . . . . . . . L CSEG     E6BBH  1599 1605#
DBYTE_TOKE. . . . . N          0082H  470# 575 934 1605 2647 2769
DCLAUSE . . . . . . L CSEG     F00EH  3630 3647#
DEC_HIGH. . . . . . L CSEG     E5D6H  1242 1244#
DEC_PNT . . . . . . L CSEG     E5CDH  1239# 2827 2836
DEC_TOKE. . . . . . N          0035H  471# 576
DECODE. . . . . . . L CSEG     E2F1H  893 895# 912
DECODE_CALL . . . . L CSEG     E2E7H  888 890#
DELAY . . . . . . . N          00F5H  403# 3656 3766
DELET . . . . . . . L CSEG     E871H  2097 2099#
DIS_OR_ERR. . . . . L CSEG     EA7FH  2597 2602#
DISERR. . . . . . . L CSEG     EADAH  2673# 2697
DISFET. . . . . . . L CSEG     EB29H  2736 2750#
DISLOP. . . . . . . L CSEG     EB05H  2734# 2762
DISMEM. . . . . . . L CSEG     EB02H  2604 2733#
DISPLAY_LIST. . . . L CSEG     F1F0H  3917 3930#
DISPLAY_TOKEN . . . N          E059H  310#
DIV_TOKE. . . . . . N          0031H  472# 577
DJNZ_TOKE . . . . . N          0025H  473# 578
DLY_THRU. . . . . . L CSEG     F0EDH  3770 3776#
DLYCNT. . . . . . . L DSEG     005CH  422# 3768 3769 3771
DONT_WAIT . . . . . L CSEG     E666H  1521 1526#
DOWN_MOVE . . . . . L CSEG     EBA6H  2815 2838# 2852
DOWNLOAD_CMD. . . . L CSEG     F4F9H  937 4574#
DOWNLOAD_TOKE . . . N          00E0H  474# 579 936
DPH . . . . . . . . N DSEG     0083H PREDEFINED  834 1077 1141 1335 1340 1427 1435 1458 1466 1492 1495 1572 2386 2387
                                 2431 2550 2559 2946 2948 2949 2967 2969 2970 3040 3042 3043 3053 3054 3157 3221 3224
                                 3473
DPL . . . . . . . . N DSEG     0082H PREDEFINED  813 833 1142 1334 1341 1426 1436 1457 1467 1491 1496 1573 1602 1608 1621
```

```
NAME                 TYPE      VALUE AND REFERENCES

                                 1622 2383 2384 2432 2553 2558 2943 2944 2964 2965 3036 3044 3050 3051 3158 3220 3224
                                 3227 3230 3234 3238 3241 3244 3247 3250 3253 3256 3261 3264 3267 3270 3273 3276 3290
                                 3294 3298 3403 3406 3410 3414 3417 3420 3424 3427 3430 3433 3439 3442 3445 3448 3451
                                 3454 3457 3460 3465 3465 3468 3471 3474 3641 3726 3981 3999
DPTR_CMD. . . . . . L CSEG    ED7DH   939 3156#
DPTR_TOKE . . . . . N         00A1H   227# 580 938
DT_LOOP . . . . . . L CSEG    EA26H   2479# 2484
DT0 . . . . . . . . L CSEG    EA20H   2472 2474#
DT0_0 . . . . . . . L CSEG    EA15H   2468# 2476
DT1 . . . . . . . . L CSEG    EA23H   2473 2477#
ENDMOD. . . . . . . L CSEG    EC12H   2904 2918#
EOL_CHECK . . . . . N         E06EH   317#
EOL_ERROR . . . . . L CSEG    E5BFH   1206 1208#
EOL_TOKE. . . . . . N         0007H   223# 888 1206 1841 2339 2641 2889 2906 2917 3818
EOLMEM. . . . . . . L CSEG    EAD7H   2639 2669 2672#
EQLMOD. . . . . . . L CSEG    EBDAH   2889 2894#
EQUAL_TOKE. . . . . N         0004H   369# 1851 2336 2597 2895
ERR . . . . . . . . L CSEG    E6CDH   1597 1601 1607 1612#
ERRMOD. . . . . . . L CSEG    EBD7H   2806 2893# 2895 2920
ERRNUM. . . . . . . L DSEG    0043H   248# 821 913 1133 1139 1209 1595 1611 1801 1812 1850 1879 1897 2201 2288 2321 2696
                                 2805 2892 2894 2919 3063 3309 3596 3648 3652 3708 3832 3841 3847 3922 3972 4066 4121
                                 4424 4641
ERROR . . . . . . . N         E05FH   312#
ERROR_BEGIN . . . . L CSEG    E41BH   1140 1146#
ERROR_LOOP. . . . . L CSEG    E41DH   1149# 1151
ERROR_MSG . . . . . L CSEG    E426H   1081 1082 1130 1131 1155#
ERROR_TABLE . . . . L CSEG    E42DH   1087 1088 1137 1157#
ERROR_TEST. . . . . L CSEG    E40BH   1138# 1154
ERRSET. . . . . . . L CSEG    E9BCH   2322 2341#
ESC . . . . . . . . N         001BH   376# 1308 3315 3786
EXERR0. . . . . . . L CSEG    F04FH   3649 3653 3655 3676#
EXERR1. . . . . . . L CSEG    F10CH   3792# 3833 3842 3848
F0. . . . . . . . . N BSEG    00D5H PREDEFINED  1568 1571 1575 1587 1592 1603 1609 1623
FAKE_BAD_CMD_ERR. . L CSEG    F57BH   4637 4640#
FETCH . . . . . . . N         E04AH   305#
FETEND. . . . . . . L CSEG    E69DH   1582 1585 1589 1592# 1613 1630
FETERR. . . . . . . L CSEG    E6CAH   1581 1611#
FILE_DISPLAY. . . . L CSEG    F49AH   4462 4484# 4486
FILE_FOUND. . . . . L CSEG    F53EH   4477 4478 4620#
FILL1 . . . . . . . L CSEG    EB01H   2708 2711#
FILLMEM . . . . . . L CSEG    EADDH   2600 2696#
FILLOOP . . . . . . L CSEG    EAECH   2701# 2710
FIRST_FLAG. . . . . L BSEG    0003H   438# 2928 2979 2982
FOREVER_TOKE. . . . N         0008H   475# 582 3825
FROM_TOKE . . . . . N         0009H   476# 581 583 3547
GET_COMMA . . . . . N         E06BH   316#
GET_PART. . . . . . N         E065H   314#
GET_TYPE. . . . . . L CSEG    F358H   4250# 4253 4288 4464 4485
GETCHR. . . . . . . L CSEG    E815H   2053# 2166 2186 2192 2266 2273 2346
GETEOL. . . . . . . N         E053H   308#
GETNUM. . . . . . . N         E050H   307#
GETOKE. . . . . . . N         E056H   309#
GO_CMD. . . . . . . L CSEG    F10FH   941 3816#
GO_TOKE . . . . . . N         00C2H   477# 584 940
GOOD_TOKE_FOUND . . L CSEG    E91AH   2197 2208#
```

```
NAME                TYPE      VALUE AND REFERENCES

GR. . . . . . . . . N         00F6H  404# 799 3828 3837 3850 3855 3870
GR_PORT . . . . . . N         0003H  381# 1009 3666 3866
GT0 . . . . . . . . L CSEG    E924H  2213 2214#
GT1 . . . . . . . . L CSEG    E929H  2215 2216#
GUARD_MSG . . . . . L CSEG    F2F3H  4145 4152#
HEX1. . . . . . . . L CSEG    EA42H  2520 2522#
HEXBIN. . . . . . . L CSEG    F340H  4214# 4256 4258 4260 4262 4293 4299 4304 4470
HEXCHR. . . . . . . L CSEG    E967H  2249 2269#
HEXEND. . . . . . . L CSEG    E743H  1715 1719 1721#
HEXSTR. . . . . . . L CSEG    E93BH  2246# 2268
HORIZONTAL_TAB. . . N         0009H  374# 2109
HTEST . . . . . . . L CSEG    E96BH  2247 2271#
HXTEST. . . . . . . L CSEG    E737H  1713# 2246
IASCII_TO_HEX . . . L CSEG    EA3CH  339 2518# 4217 4222
IBREAK. . . . . . . L CSEG    EDC6H  328 3220#
ICI . . . . . . . . L CSEG    E5EBH  1298# 1525 2071 3787 4220 4457 4532
ICO . . . . . . . . L CSEG    E5E8H  332 1276# 1523 1668 1670 1846 1995 1999 2100 2103 2132 2442 2492 2748 2764 3016
                                3713 3718 3725 3735 3746 3761 3928 4191 4352 4419 4537
ICONTINUATION_LINE. L CSEG    E65DH  363 1520# 2760 2980
ICSTS . . . . . . . L CSEG    E602H  1045 1334# 1489 1527 3307 3780
IDISPLAY_TOKEN. . . L CSEG    EA12H  358 1844 2465# 2742 2940 2998 3014 3934
IE. . . . . . . . . N DSEG    00A8H PREDEFINED  3230 3231 3233 3320 3460 3464 3478
IE0 . . . . . . . . N BSEG    0089H PREDEFINED  3475
IEOL_CHECK. . . . . L CSEG    E5BBH  365 1205# 1806 2603 2672 2918 3673 3854 4528 4611
IERROR. . . . . . . L CSEG    E3E4H  360 914 1121# 1210 1612 1808 2202 2289 2341 2673 2893 3065 3311 3597 3676 3710 3792
                               4011 4425 4642
IFETCH. . . . . . . L CSEG    E66BH  353 1567# 2750 2825 2840 3110 3175 3178 3757 4383
IGET_COMMA. . . . . L CSEG    E77AH  364 1810# 4525
IGET_PART . . . . . L CSEG    E7A2H  362 1877# 2593 2901 4527 4610
IGETEOL . . . . . . L CSEG    E773H  356 1805# 2699 2922 3154 3190 3659 3827 3836 3849
IGETNUM . . . . . . L CSEG    E769H  355 1800# 1887 2634 2807 3115 3150 3184 3548 3633 3961 4064 4522
IGETOKE . . . . . . L CSEG    E8BCH  357 887 1800 1805 1811 1840 1885 1899 2163# 2167 2592 2599 2638 2640 2668 2888 2896
                                2905 2916 3546 3552 3628 3644 3646 3834 3844 3846 3918 4451 4526 4609
ILSTBYT . . . . . . L CSEG    E7F9H  337 1085 1134 1986 1990# 2752 3112 3722 3739 3759 4085 4090 4094 4189
ILSTWRD . . . . . . L CSEG    E7F4H  338 1986# 2746 2985 3007 3147 3181 3715 3732 3883 4036 4482 4495
INC_HIGH. . . . . . L CSEG    E5CCH  1234 1236#
INC_PNT . . . . . . L CSEG    E5C4H  1232# 2637 2709 2761 2842 2851 2952 2973 4297
INC_TOKE. . . . . . N         0037H  478# 585
INEWLINE. . . . . . L CSEG    E717H  335 1666# 1842 2063 2085 2737 3012 3702 3929 4347 4414
INIT_IO . . . . . . L CSEG    E386H  796 886 1002# 1124 3337 3343 4307 4474 4487
INPUT . . . . . . . L CSEG    E8A6H  2109 2125#
INPUTOK . . . . . . L CSEG    E8AEH  2126 2127 2128#
IP. . . . . . . . . N DSEG    00B8H PREDEFINED  3241 3242 3439 3441
IPRINT_STRING . . . L CSEG    E9FFH  340 862 1083 1089 1132 1143 2430# 3865 3879 4141 4456 4479 4492 4531 4580
ISAVE_AND_DISPLAY . L CSEG    E7DDH  359 1929# 2663 2665
ISIT_DISPLAY. . . . L CSEG    E784H  1837# 3106 3142 3171 3916 3959 4061 4120
ISTORE. . . . . . . L CSEG    E672H  354 1570# 2636 2702 2829 2844 3117 3186 3189 4296
ITO . . . . . . . . N BSEG    0088H PREDEFINED  3476
ITIME . . . . . . . L CSEG    EA45H  336 1019 1123 2548# 3774 3779 4313 4346 4350 4413 4422
IWAIT_FOR_USER. . . L CSEG    E3B0H  361 863 1041# 1144 2757 2941 3113 3148 3182 3790 3884 3935 4037 4096 4142 4483 4496
IWAIT_FOR_USER_1. . L CSEG    E3B4H  1044# 1046
JB_TOKE . . . . . . N         0027H  479# 586
JBC_TOKE. . . . . . N         0028H  480# 587
JC_TOKE . . . . . . N         0018H  481# 588
JMP_TAB_CHECKER . . L CSEG    F574H  4625 4628 4631 4634#
```

```
NAME                TYPE      VALUE AND REFERENCES

JMP_TOKE. . . . . . N         0032H  482# 589
JNB_TOKE. . . . . . N         0026H  483# 590
JNC_TOKE. . . . . . N         0017H  484# 591
JNZ_TOKE. . . . . . N         0015H  485# 592
JZ_TOKE . . . . . . N         0016H  486# 593
KEY_BYTE. . . . . . L CSEG    ED42H  3091 3095 3099 3103 3105#
KEYTAB. . . . . . . L CSEG    E0D8H  657# 2200 2378 2478
KEYWORD_DISPLAY . . L CSEG    ED95H  3159 3164 3170#
LAB1. . . . . . . . L CSEG    E761H  1761 1767#
LAB10 . . . . . . . L CSEG    E9C8H  2342 2346#
LAB18 . . . . . . . L CSEG    F01BH  3651 3652#
LAB1A . . . . . . . L CSEG    E764H  1766 1770#
LAB1B . . . . . . . L CSEG    E765H  1753 1771#
LAB2. . . . . . . . L CSEG    EC2AH  2929# 2953 2978 2994 3009
LAB23 . . . . . . . L CSEG    EB51H  2739 2767#
LAB3. . . . . . . . L CSEG    EC60H  2951 2955#
LAB5A . . . . . . . L CSEG    EC51H  2945 2947#
LAB5B . . . . . . . L CSEG    EC47H  2935 2942#
LAB6A . . . . . . . L CSEG    EC7EH  2966 2968#
LAB7. . . . . . . . L CSEG    ECB0H  2988 2990#
LAB8. . . . . . . . L CSEG    ECD0H  3005 3007#
LB_10 . . . . . . . L CSEG    EC9CH  2979 2981#
LCALL_TOKE. . . . . N         0010H  487# 594
LDLOOP. . . . . . . L CSEG    EA88H  2635# 2671
LEGALI. . . . . . . L CSEG    E882H  2095 2106#
LENGTH_HIGH . . . . L DSEG    0063H  429# 1896 2822
LENGTH_LOW. . . . . L DSEG    0064H  430# 1893 2819
LF. . . . . . . . . N         000AH  373# 981 1156 1669 3888 3890 4618 4619 4620
LFTROT. . . . . . . L CSEG    E711H  1644# 1645
LH_4. . . . . . . . L CSEG    F376H  4291# 4298
LH_6. . . . . . . . L CSEG    F387H  4292 4299#
LH_7. . . . . . . . L CSEG    F38FH  4289 4303#
LH_8. . . . . . . . L CSEG    F38CH  4302# 4303 4306
LH_ERROR. . . . . . L CSEG    F447H  4302 4423# 4472
LINBUF. . . . . . . L DSEG    0024H  242# 2019 2061 2075 2088 2110 2128 2642 2908
LINCNT. . . . . . . L DSEG    0053H  264# 2169 2173 2174 2189
LINE_START. . . . . L DSEG    0052H  263# 885 2058 2097 2667 2907 3346
LINMAX. . . . . . . N         0018H  232# 242 2107 2119
LIST_1. . . . . . . L CSEG    F1F7H  3932 3934#
LIST_2. . . . . . . L CSEG    F1DBH  3919 3922#
LIST_CMD. . . . . . L CSEG    F1CDH  943 3915#
LIST_TOKE . . . . . N         00D7H  488# 595 942
LJMP_TOKE . . . . . N         0011H  489# 596
LNLGTH. . . . . . . L DSEG    0054H  265# 1005 2054 2059 2064 2076 2079 2086 2096 2104 2106 2111 2113 2116 2129 2133
LOAD_CMD. . . . . . L CSEG    F44DH  945 4450#
LOAD_HEX. . . . . . L CSEG    F372H  4287# 4301 4473 4583
LOAD_LOOP . . . . . L CSEG    F46CH  4463# 4465 4467 4469
LOAD_MSG. . . . . . L CSEG    F523H  4578 4579 4618#
LOAD_TOKE . . . . . N         00E2H  490# 597 944
LODMEM. . . . . . . L CSEG    EA85H  2598 2634#
LSSEQL. . . . . . . L CSEG    E74BH  1753# 2934 2961 2977
LSTBRK. . . . . . . L CSEG    EC21H  2891 2924#
LSTBYT. . . . . . . N         E015H  301#
LSTFLG. . . . . . . L BSEG    0001H  278# 436 794 1125 1521 2081 3345 3920 3926 3932 4613
LSTOUT. . . . . . . L CSEG    EC8DH  2962 2972 2975#
```

```
NAME                TYPE      VALUE AND REFERENCES

LSTWRD. . . . . . . N         E018H  302#
MAXHGH. . . . . . . N         001FH  398# 2930 3022 3047
MAXLOW. . . . . . . N         00FFH  397# 2931 3021
MAXNUM_FLAG . . . . L BSEG    0004H  439# 1002 1753 1756 1771
MEMORY. . . . . . . L CSEG    E676H  1569 1572#
MEMORY_CMD. . . . . L CSEG    EA5BH  931 935 955 957 977 2588#
MODE_CMD. . . . . . L CSEG    F571H  947 4632#
MODE_TOKE . . . . . N         00B9H  491# 598 946
MON_FLAGS . . . . . N         00FAH  408# 3294 3403
MORE_CONT . . . . . L CSEG    E89EH  2119 2121#
MORE_SPACE. . . . . L CSEG    E893H  2114# 2122 2123
MOV_TOKE. . . . . . N         001FH  492# 599
MOVC_TOKE . . . . . N         001AH  493# 600
MOVX_TOKE . . . . . N         001BH  494# 601
MUL_TOKE. . . . . . N         0030H  495# 602
MULTISTEP . . . . . N         00FFH  414# 3661 3765
NEWLINE . . . . . . N         E00FH  299#
NEXT_ENTRY. . . . . L CSEG    E303H  897 908#
NMTEST. . . . . . . L CSEG    E72DH  1705# 1713 1726 2242 2248
NO_BREAK. . . . . . N         0009H  382# 800 1013 3829
NOBRK_MSG . . . . . L CSEG    F327H  4149 4160#
NOP_TOKE. . . . . . N         003BH  496# 603
NOT_MATCH_TBL . . . L CSEG    E9AAH  2323 2329#
NOT_STEP. . . . . . N         00FBH  412# 3860
NOT_STEP_THREE. . . L CSEG    F06AH  3706 3707 3711#
NOTBOT. . . . . . . L CSEG    E92DH  2214 2216 2218#
NOTDAT. . . . . . . L CSEG    F13FH  3835 3841#
NOTFOR. . . . . . . L CSEG    F127H  3825 3832#
NOTFRM. . . . . . . L CSEG    EFC1H  3547 3553#
NOWAIT. . . . . . . L CSEG    EB45H  2761# 2765
NTLAST. . . . . . . L CSEG    EB4AH  2759 2763#
NUM_FOUND . . . . . L CSEG    F54EH  4490 4491 4621#
NUMBER. . . . . . . L CSEG    E930H  2179 2242#
NUMBER_1. . . . . . L CSEG    E972H  2272 2274#
NUMBER_2. . . . . . L CSEG    E979H  2276 2278#
NUMBER_3. . . . . . L CSEG    E97EH  2279 2281#
NUMBER_4. . . . . . L CSEG    E983H  2282 2284#
NUMBER_ERR. . . . . L CSEG    E988H  2285 2287#
NUMBER_FOUND. . . . L CSEG    E98EH  2277 2280 2283 2286 2290#
NUMBER_OF_BYTES . . L DSEG    004DH  258#
NUMBER_TOKE . . . . N         0001H  219# 1802 1880 2291 2669 2697 2897 3649 4462
NUMEND. . . . . . . L CSEG    E736H  1709 1711#
NUMMEN. . . . . . . L CSEG    EAD2H  2641 2669#
NUMMOD. . . . . . . L CSEG    EBEDH  2899 2901# 2906 2917
OFST. . . . . . . . N         0010H  231# 447 448 449 451 453 455 456 457 458 464 465 466 467 471 472 473 478 479 480
                                481 482 483 484 485 486 487 489 492 493 494 495 496 499 500 503 516 517 518 519 520 521
                                523 524 527 528 538 539 540
ON_TOKE . . . . . . N         000FH  497# 604 3919 3931
OR_TOKE . . . . . . N         000BH  498# 605 3845
ORG_TOKE. . . . . . N         00D4H  228# 606
ORL_TOKE. . . . . . N         0022H  499# 607
OUR_CODE_HIGH . . . L DSEG    004EH  259#
OUR_CODE_LOW. . . . L DSEG    004FH  260#
OUTIBK. . . . . . . L CSEG    ED1CH  3055# 3058 3059
OUTCHR. . . . . . . L CSEG    E85CH  2054 2088#
```

```
NAME                TYPE     VALUE AND REFERENCES

OUTOKE. . . . . . . L CSEG   ECBCH  2991 2993 2996#
P1. . . . . . . . . N DSEG   0090H PREDEFINED  3244 3245 3417 3419
P3. . . . . . . . . N DSEG   00B0H PREDEFINED  3247 3248 3420 3423
PAINTER . . . . . . L CSEG   E80AH  2019# 2067 2087
PARAM1. . . . . . . N REG    R2     283# 808 811 860 892 912 1007 1009 1011 1013 1015 1017 1081 1084 1087 1121 1128
                                1130 1133 1141 1275 1433 1464 1522 1576 1580 1590 1639 1667 1669 1697 1705 1716 1762
                                1763 1767 1843 1845 1930 1935 1988 1990 1994 1998 2022 2056 2069 2073 2074 2083 2095
                                2099 2102 2109 2176 2182 2187 2193 2195 2199 2200 2204 2206 2209 2260 2264 2267 2316
                                2342 2380 2431 2441 2491 2519 2548 2635 2662 2664 2701 2741 2745 2747 2751 2763 2826
                                2841 2930 2939 2983 2997 3004 3006 3013 3015 3021 3027 3043 3059 3111 3116 3146 3151
                                3180 3185 3188 3335 3338 3340 3349 3388 3390 3393 3397 3400 3520 3549 3666 3668 3670
                                3704 3706 3712 3714 3717 3721 3724 3731 3734 3738 3745 3758 3760 3772 3777 3863 3866
                                3868 3872 3877 3881 3924 3927 3931 3933 4032 4084 4089 4093 4136 4187 4216 4221 4294
                                4309 4311 4344 4348 4351 4411 4416 4418 4420 4454 4459 4477 4480 4490 4493 4529 4534
                                4536 4576 4578 4581 4614
PARAM2. . . . . . . N REG    R3     284# 861 1018 1082 1088 1122 1131 1142 1759 1987 2250 2256 2269 2432 2466 2470 2475
                                2484 2551 2744 2931 2984 3002 3022 3029 3044 3058 3145 3152 3179 3334 3517 3550 3705
                                3707 3728 3773 3778 3864 3878 3882 4035 4140 4308 4312 4345 4349 4412 4421 4455 4478
                                4481 4491 4494 4530 4579
PARAM3. . . . . . . N REG    R4     285# 1754 1768 1991 1996 2485 2494 2932 2959 2975
PARAM4. . . . . . . N REG    R5     286# 1755 1760 2933 2960 2976
PARAM5. . . . . . . N REG    R6     287#
PARAM6. . . . . . . N REG    R7     288# 2025 2064 2065 2086
PARTIT_HI_HIGH. . . L DSEG   0059H  270# 1882 1889 1894 2707 2756 2817 2849 3037 4357
PARTIT_HI_LOW . . . L DSEG   005AH  271# 420 1881 1888 1891 2705 2754 2816 2847 3034 4354
PARTIT_LO_HIGH. . . L DSEG   0057H  268# 1884 1895 2595 2833 3039 3046 4358 4373 4381 4389
PARTIT_LO_LOW . . . L DSEG   0058H  269# 1883 1892 2594 2831 3035 3049 4355 4375 4382 4385 4387
PARTITION_E . . . . L CSEG   E7DBH  1886 1902#
PC_CHA. . . . . . . L CSEG   ED70H  3143 3149#
PC_CMD. . . . . . . L CSEG   ED5FH  949 3141#
PC_TOKE . . . . . . N        00A0H  229# 608 948
PCNTHI. . . . . . . L DSEG   0061H  427# 1252 1253 2809 2814 2821 2823 4452 4466 4480 4523 4541
PCNTLO. . . . . . . L DSEG   0062H  428# 1249 1250 2808 2812 2818 2820 4453 4468 4481 4524 4543
 GMBRK. . . . . . . L CSEG   F160H  3845 3854#
 LUS_TOKE . . . . . N        0005H  222# 2337
PNTHGH. . . . . . . L DSEG   0044H  249# 1235 1243 1251 1252 1572 1596 2595 2662 2706 2745 2755 2813 2817 2834 2848
                                2926 2932 2947 2956 2959 2968 2991 3004 3107 3173 3752 4259 4309 4381 4467 4493
PNTLOW. . . . . . . L DSEG   0045H  250# 1232 1233 1239 1240 1248 1249 1573 1600 1606 1614 1625 1633 1641 2594 2664
                                2704 2744 2753 2811 2816 2832 2846 2925 2933 2942 2955 2960 2963 2993 3000 3090 3094
                                3098 3102 3157 3162 3167 3177 3187 3755 4261 4308 4382 4469 4494
POINT0. . . . . . . N REG    R0     281# 1933 1934 1937 1938 2062 2077 2078 2090 2091 2112 2117 2118 2130 2131 2168
                                2171 2172 2175 2183 2184 2185 2188 2376 2391 2394 2642 2644 2646 2648 2650 2651 2652
                                2653 2654 2656 2657 2658 2659 2660 2661 2666 2908 2909 2910 2911 2912 2913 2914 2915
                                3280 3283 3285 3286 3963 3965 3969 3982 4008
POINT1. . . . . . . N REG    R1     282# 2019 2021 2024
POP_TOKE. . . . . . N        002DH  500# 609
POUND_TOKE. . . . . N        0006H  221# 2338
POWER_ON. . . . . . L CSEG   E274H  322 341 342 343 344 345 789#
PRE_SET_BAUD. . . . L CSEG   F21AH  3974# 4006
PRE_UNBREAK . . . . L CSEG   EEE3H  3315 3360 3362#
PRINT_STRING. . . . N        E01EH  303#
 RINT_STRING_1. . . L CSEG   EA08H  2437# 2443
 RINT_STRING_E. . . L CSEG   EA11H  2436 2444#
PROG_MSG. . . . . . L CSEG   F302H  4146 4154#
PROGRAM_BREAK . . . N        000BH  388# 3851 3856
```

```
NAME                TYPE      VALUE AND REFERENCES

PROGRAM_CMD . . . . L CSEG    F56CH  951 4629#
PROGRAM_TOKE. . . . N         00D5H  501# 610 950 3842
PSW . . . . . . . . N DSEG    00D0H PREDEFINED  835 3094 3250 3251 3277 3424 3426
PSW_CMD . . . . . . L CSEG    ED30H  953 3093#
PSW_TOKE. . . . . . N         0099H  502# 611 952
PUSH_TOKE . . . . . N         002FH  503# 612
PX0 . . . . . . . . N BSEG    00B8H PREDEFINED  3477
R0_TOKE . . . . . . N         0090H  504# 613
R1_TOKE . . . . . . N         0091H  505# 614
R2_TOKE . . . . . . N         0092H  506# 615
R3_TOKE . . . . . . N         0093H  507# 616
R4_TOKE . . . . . . N         0094H  508# 617
R5_TOKE . . . . . . N         0095H  509# 618
R6_TOKE . . . . . . N         0096H  510# 619
R7_TOKE . . . . . . N         0097H  511# 620
RAMIO . . . . . . . N         B800H  393# 3996 4002
RAMOFF. . . . . . . N         B000H  391# 799 805 817 1369 1400 1594 3222 3287 3351 3363 3395 3507 3516 3623 3635 3656
                                 3663 3719 3726 3736 3740 3763 3766 3828 3837 3850 3855 3859 3870 3975 3986 4030 4062
                                 4075 4086
RBIT. . . . . . . . L CSEG    E6CFH  1605 1613#
RBIT_TOKE . . . . . N         0084H  512# 621 954 1613 2655 2771
RBOUT . . . . . . . N         007FH  375# 2095
RBS_TOKE. . . . . . N         0000H  513# 622
RBYTE . . . . . . . L CSEG    E6A0H  1586 1594#
RBYTE_TOKE. . . . . N         0081H  514# 623 956 1599 2768 3108 3172
READ_PC . . . . . . L CSEG    EF9DH  3144 3329 3505# 3703 3880 4399
REG . . . . . . . . N         0040H  230# 447 448 449 451
REPAINT . . . . . . L CSEG    E833H  2065 2067#
REPAINT_1 . . . . . L CSEG    E835H  2066 2068#
REPAINT_2 . . . . . L CSEG    E80CH  2020# 2025
RESET_CMD . . . . . N         0004H  378# 1007
RESET_TOKE. . . . . N         000EH  515# 624 2920 2939 3923 3933
RET_TOKE. . . . . . N         003AH  516# 625
RETI_TOKE . . . . . N         0039H  517# 626
RHTROT. . . . . . . L CSEG    E701H  1637# 1638
RL_TOKE . . . . . . N         0034H  518# 627
RL4 . . . . . . . . L CSEG    E947H  2251# 2270
RLC_TOKE. . . . . . N         0033H  519# 628
RR_TOKE . . . . . . N         0038H  520# 629
RRC_TOKE. . . . . . N         0036H  521# 630
RSTMOD. . . . . . . L CSEG    EC15H  2897 2919#
RUBOUT. . . . . . . L CSEG    E867H  2074 2095#
RUN_USER. . . . . . L CSEG    F169H  3819 3831 3840 3853 3858#
RUN_USER_RETURN . . L CSEG    F18EH  3358 3876#
S_S_1 . . . . . . . L CSEG    E9E5H  2388# 2395
S_S_2 . . . . . . . L CSEG    E9F6H  2392 2400#
S_S_3 . . . . . . . L CSEG    E9FCH  2400 2404#
SAVE_AND_DISPLAY. . N         E05CH  311#
SAVE_CMD. . . . . . L CSEG    F4B7H  959 4521#
SAVE_SEL. . . . . . N         00F2H  400# 3623 3641 3740
SAVE_TOKE . . . . . N         00E3H  522# 631 958
SCON. . . . . . . . N DSEG    0098H PREDEFINED  3253 3254 3427 3429
SELECT. . . . . . . L DSEG    0046H  251# 1567 1570 1598 2591 2643 2738 2804 3108 3172 3749 4295 4380
SELECT_CON. . . . . N         0000H  235# 1015 2056 3400 4576
SEND_BINARY . . . . L CSEG    F33DH  4188 4190#
```

```
NAME                TYPE     VALUE AND REFERENCES

SEND_BYTE . . . . . L CSEG   F333H  4183# 4372 4374 4376 4378 4384 4394 4398 4401 4403 4406 4410 4540 4542 4544 4546
                                    4550
SET_BAUD. . . . . . L CSEG   F229H  816 3344 3985#
SETB_TOKE . . . . . N        0029H  523# 632
SETBRK. . . . . . . L CSEG   ECF2H  2902 3033#
SH_1. . . . . . . . L CSEG   F3D8H  4362 4364#
SH_2. . . . . . . . L CSEG   F3DFH  4366 4368#
SH_3. . . . . . . . L CSEG   F3F0H  4379# 4390
SH_4. . . . . . . . L CSEG   F408H  4388 4390#
SH_5. . . . . . . . L CSEG   F411H  4360 4396#
SH_6. . . . . . . . L CSEG   F3C1H  4343 4351#
SH_7. . . . . . . . L CSEG   F440H  4415 4420#
SIGN_ON . . . . . . L CSEG   E2CCH  859# 3354
SIGN_ON_MSG . . . . L CSEG   E36BH  860 861 892 980#
SINGLE_BREAK. . . . N        0001H  386# 3670
SINGLE_STEP_MSG . . L CSEG   F31BH  4148 4158#
SINGLESTEP. . . . . N        00FEH  413# 3674
SJMP_TOKE . . . . . N        0014H  524# 633
SP. . . . . . . . . N DSEG   0081H PREDEFINED  793 884 3098 3256 3257 3293 3321 3430 3432 3736
SP_CMD. . . . . . . L CSEG   ED36H  961 3097#
SP_TOKE . . . . . . N        009AH  525# 634 960
SPACCO. . . . . . . L CSEG   E5E6H  1086 1135 1275# 2101 2743 2996 2999 3716 3723 3733 3744
SPEFUN. . . . . . . L CSEG   E6DEH  1616 1621#
SPFILL. . . . . . . L CSEG   E8CCH  2170# 2173
SPWAIT. . . . . . . L CSEG   E8F5H  2190# 2194
SSRET . . . . . . . L CSEG   F106H  3765 3790#
ST_1. . . . . . . . L CSEG   F293H  4071 4073#
STACK . . . . . . . N        0007H  377# 884 3293 3321
START . . . . . . . L CSEG   E2D6H  884# 889 894 1145 1309 3791 3885
START_16_TIMER. . . N        00C0H  396# 4003
START_COMPARE . . . L CSEG   E756H  1754 1755 1757#
STATE_ERR . . . . . L CSEG   F252H  3923 3973 4010# 4067 4122
STEP_CMD. . . . . . L CSEG   EFDCH  963 3621#
STEP_STOP . . . . . L CSEG   F0FBH  3781 3784#
STEP_TOKE . . . . . N        00C1H  526# 635 962
STEP51. . . . . . . L CSEG   F02FH  3662# 3675
STEP51_EXIT . . . . L CSEG   F0D1H  3743 3762#
STEP51_RETURN . . . L CSEG   F052H  3361 3701#
STORE . . . . . . . N        E04DH  306#
STORE_HEX . . . . . L CSEG   F3ADH  4342# 4395 4551 4616
STORED_CHECK_SUM. . L CSEG   E049H  352#
STPDLY. . . . . . . L CSEG   F0DEH  3769# 3775
STPEOL. . . . . . . L CSEG   F048H  3627 3645 3673#
STPLOP. . . . . . . L CSEG   F02DH  3660# 3783
STPLOP_REACH. . . . L CSEG   F0F9H  3782# 3786 3788
STRFIL. . . . . . . L CSEG   E8E2H  2178 2180# 2189
STRGBF. . . . . . . L DSEG   003CH  243# 2168 2175 2376
STRGCT. . . . . . . L DSEG   0055H  266# 2377 2395
STRING_SPACE. . . . L CSEG   E9CDH  2196 2205 2375#
STRTST. . . . . . . L CSEG   E900H  2181 2191 2195#
STRTST1 . . . . . . L CSEG   E902H  2196# 2200
SUBB_TOKE . . . . . N        001EH  527# 636
SWAP_POINTERS . . . L CSEG   E5D7H  1247# 2824 2828 2839 2843
SWAP_TOKE . . . . . N        002EH  528# 637
SYM_TBL_SRCH. . . . L CSEG   E999H  2318# 2332
```

```
NAME                TYPE      VALUE AND REFERENCES

SYMBOL. . . . . . . L CSEG    E994H   2243 2316#
SYMBOL_TBL. . . . . L CSEG    E9AEH   2317 2333#
SYMEND. . . . . . . L CSEG    E9BFH   2328 2342#
T_LAB . . . . . . . L CSEG    EABAH   2655 2657#
TABKEY. . . . . . . L CSEG    E889H   2107 2109#
TCON. . . . . . . . N DSEG    0088H PREDEFINED  3234 3235 3237 3457 3459
TEMP. . . . . . . . N REG     R5      291# 3048 3052 4219 4223 4359 4361
TEMP_LOW. . . . . . L DSEG    0047H   252# 2698 2700 3158 3163 3168 3176 3180 3187
TEMPI . . . . . . . L DSEG    0056H   267# 1136 1140 1153 1625 1626 1627 1628 1633 1634 1635 1638 1641 1642 1643 1645
                                2185 2188 2390 2392
THO . . . . . . . . N DSEG    008CH PREDEFINED  3162 3261 3262 3442 3444
TH1 . . . . . . . . N DSEG    008DH PREDEFINED  3167 3264 3265 3445 3447
TILL_TOKE . . . . . N         000CH   529# 638 639 3833
TIME. . . . . . . . N         E012H   300#
TIME1 . . . . . . . L CSEG    EA4EH   2555# 2561
TIMER_HIGH. . . . . N         0005H   394# 3996
TIMER_PRESET. . . . L CSEG    F25CH   3990 4020#
TLO . . . . . . . . N DSEG    008AH PREDEFINED  3163 3267 3268 3448 3450
TL1 . . . . . . . . N DSEG    008BH PREDEFINED  3168 3270 3271 3451 3453
TMO_CMD . . . . . . L CSEG    ED86H   965 3161#
TMO_TOKE. . . . . . N         00A2H   530# 640 964
TM1_CMD . . . . . . L CSEG    ED8FH   967 3166#
TM1_TOKE. . . . . . N         00A3H   531# 641 966
TMOD. . . . . . . . N DSEG    0089H PREDEFINED  3273 3274 3454 3456
TO_TOKE . . . . . . N         000DH   532# 642 1886 2997
TOK_WRITE . . . . . L CSEG    EA34H   2488 2490#
TOKERR. . . . . . . L CSEG    E90DH   2201# 2207
TOKLOP. . . . . . . L CSEG    EA2EH   2486# 2494
TOKSAV. . . . . . . L DSEG    005BH   421# 1839 1843 2590 2887 2890 2898 3632 3642 3742 3747
TOKSIZ. . . . . . . N         0004H   233# 243 2169 2174 2377
TOKSTR. . . . . . . L DSEG    0048H   253# 897 1839 1878 2212 2218 2291 2292 2316 2323 2327 2344 2347 2588 2590 2887 2903
                                3626 3632 3647 3824 4461
TOKTBL. . . . . . . L CSEG    E071H   547# 2200 2210 2469
TOP_CMD . . . . . . L CSEG    F278H   969 4061#
TOP_DISPLAY . . . . L CSEG    F29AH   4063 4079#
TOP_LIST_0. . . . . L CSEG    F2B0H   4088 4092#
TOP_LIST_1. . . . . L CSEG    F2B5H   4091 4095#
TOP_LIST_2. . . . . L CSEG    F29EH   4081 4083#
TOP_PORT. . . . . . N         0083H   380# 808 3338 3393
TOP_STORE . . . . . N         00F9H   407# 3395 4062 4075 4086
TOP_TOKE. . . . . . N         00D6H   533# 643 968
TRANSFER_CMD. . . . L CSEG    F567H   971 4626#
TRANSFER_TOKE . . . N         00BAH   534# 644 970
TYPE. . . . . . . . L DSEG    0065H   431# 4263
UCI . . . . . . . . L CSEG    E619H   333 1398# 1399
UCSTS . . . . . . . L CSEG    E613H   334 1369# 1398
UNBREAK . . . . . . L CSEG    EEE9H   3388# 3672 3874
UNBRK_LOOP. . . . . L CSEG    EF16H   3407# 3411
UP_MOVE . . . . . . L CSEG    EB86H   2824# 2837
UPC . . . . . . . . N         00FDH   411# 3287 3433 3507 3516
UPI_C_1 . . . . . . L CSEG    E62CH   1429# 1432
UPI_CMD . . . . . . L CSEG    E625H   809 1008 1010 1016 1129 1425# 2057 2084 3339 3350 3389 3394 3401 3667 3867 3925
                                4417 4460 4535 4577 4582 4615
UPI_CONTROL . . . . N         A001H   389# 1336 1428 1459
UPI_DATA. . . . . . N         A000H   390# 797 1042 1493 4314 4475 4488
```

```
NAME                TYPE      VALUE AND REFERENCES

UPI_DATA_IMAGE. . . N         00F1H   399# 805 1369 1400 3363
UPI_IN. . . . . . . L CSEG    E64CH   810 1298 1489# 1490 3313 3342 3399 3785 4214 4251
UPI_INA . . . . . . L CSEG    E5F3H   1300 1301#
UPI_INB . . . . . . L CSEG    E5F8H   1303 1304#
UPI_INE . . . . . . L CSEG    E601H   1308 1310#
UPI_INR . . . . . . L CSEG    E5FCH   1302 1305 1307#
UPI_O_1 . . . . . . L CSEG    E63FH   1460# 1462
UPI_OUT . . . . . . L CSEG    E638H   812 1012 1014 1276 1457# 2023 2070 3341 3391 3392 3398 3669 3671 3869 3873
UPLOAD_CMD. . . . . L CSEG    F50FH   973 4608#
UPLOAD_TOKE . . . . N         00E1H   535# 645 972
USART_MODE. . . . . N         0001H   385# 3388 3924 4416 4581
USER_MSG. . . . . . L CSEG    F2E8H   4144 4150#
UTILTT_ERROR. . . . L CSEG    E778H   1802 1807# 1813 1851 1880 1898
VALHGH. . . . . . . L DSEG    0049H   254# 1882 1884 1889 2245 2262 2265 2809 3151 3185 3549 3636 3654 3967 3976 4065
                                 4452 4523
VALLOW. . . . . . . L DSEG    004AH   255# 1881 1883 1888 2244 2251 2254 2258 2259 2635 2698 2700 2701 2808 3116 3152
                                 3188 3550 3639 3651 3657 3968 3979 4070 4453 4524
VERIFY_CMD. . . . . L CSEG    F562H   975 4623#
VERIFY_TOKE . . . . N         00BBH   536# 646 974
VPC_HIGH. . . . . . L DSEG    005FH   425# 2956 2975 2983 2989 2990
VPC_LOW . . . . . . L DSEG    005EH   424# 2955 2976 2984 2986 2987 2992
WAIT_FOR_USER . . . N         E062H   313#
WCHANGE . . . . . . L CSEG    EDB3H   3174 3183#
WORKING_SPACE . . . L DSEG    0040H   244#
WRITE_PC. . . . . . L CSEG    EFA8H   3153 3336 3514# 3551 4310
X_WRT . . . . . . . L CSEG    E69CH   1591# 1646
XBYTE . . . . . . . L CSEG    E691H   1574 1586#
XBYTE_TOKE. . . . . N         0086H   537# 647 976 1586 2649 2773
XCH_TOKE. . . . . . N         001DH   538# 648
XCHD_TOKE . . . . . N         001CH   539# 649
XEQT_MSG. . . . . . L CSEG    F1A4H   3863 3864 3887#
XREAD . . . . . . . L CSEG    E697H   1588# 1604 1610
XRL_TOKE. . . . . . N         0020H   540# 650
XWRITE. . . . . . . L CSEG    E69BH   1587 1590# 1603 1609
ZTEST . . . . . . . L CSEG    E726H   1698 1700#

ASSEMBLY COMPLETE, NO ERRORS FOUND
```

```
ISIS-II MCS-51 MACRO ASSEMBLER X040
OBJECT MODULE PLACED IN :F3:SDKADM.HEX
ASSEMBLER INVOKED BY:  :F1:ASM51 :F1:SDKADM.SRC PRINT(:F2:SDKADM.LST) OBJECT(:F3:SDKADM.HEX) DATE(8,12,81) WORKFILES(:F3
                       :,:F3:) EP DB SB


LOC  OBJ            LINE     SOURCE

                       1     $NOMACRO
                       2     $XREF
                       3     $TITLE('SDK-51 ASSEMBLER/DISASSEMBLER INTEL PROPRIETARY VERS. #1.03')
                       4     ;*****************************************************************
                       5
                       6
                       7     ;       SDK-51 MONITOR  INTEL PROPRIETARY
                       8     ;       THIS SOFTWARE IS COPYRIGHTED UNDER INTIL PART NUMBER 162787-004
                       9
                      10     ;       VERSION 1.03            8-12-81;
                      10.5   ;
                      11     ;       NN  N  00000 TTTTTTT EEEEE    !!
                      11.5   ;       N N N  0   0    T    E        !!
                      12     ;       N N N  0   0    T    EEEE     !!
                      13     ;       N  NN  0   0    T    E        !!
                      14     ;       N  NN  0   0    T    E
                      15     ;       N   N  00000    T    EEEEE    !!
                      16
                      17
                      18     ;*****************************************************************
                      19
                      20
                      21     ;       COPYRIGHT (C) 1981 INTEL CORPORATION.;
                      22     ;       ALL RIGHTS RESERVED.
                      23
                      24     ;       NO PART OF THIS PROGRAM OR PUBLICATION MAY BE REPRODUCED,
                      25     ;       TRANSMITTED, TRANSCRIBED, STORED IN A RETRIEVAL SYSTEM, OR
                      26     ;       TRANSLATED INTO ANY LANGUAGE OR COMPUTER LANGUAGE, IN ANY
                      27     ;       FORM OR BY ANY MEANS, ELECTRONIC, MECHANICAL, MAGNETIC,
                      28     ;       OPTICAL, CHEMICAL, MANUAL OR OTHERWISE, WITHOUT THE PRIOR
                      29     ;       WRITTEN PERMISSION OF INTEL CORPORATION, 3065 BOWERS AVENUE,
                      30     ;       SANTA CLARA, CALIFORNIA 95051.
                      31
                      32
                      33
                      34     ;*****************************************************************
  F581                35     ASMBASE EQU     0F581H
F581                  36     ORG ASMBASE
F581 02F977           37             LJMP    ASSEMBLY_CMD
F584 02FCFD           38             LJMP    DISASSEMBLY_CMD
                      39     ;INCLUDE FOR COMMON.INC
                      40 +1  $NOLIST
                     145 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                         ;*******************************************************************
                         ;
                         ;       TABLE OF CONTENTS:
                         ;
                         ;       This listing contains a source file and 3 include files.
                         ;       Each include file contains a number of subroutines.  Each
                         ;       subroutine listed has its own 'header' block and begins on
                         ;       a new page.
                         ;       The files are as follows:
                         ;
                         ;       SDKADM.SRC (SOURCE FILE)
                         ;
                         ;                       MNEMONIC_TAB
                         ;                       TEMPORARY VARIABLES
                         ;                       FLAG ADDRESSES
                         ;                       CONSTANTS
                         ;                       INSTRUCTION_CODE
                         ;
                         ;                       ONE_BYTE_TAIL
                         ;                       MNEMONIC_FIRST_OPERAND
                         ;                       MNEMONIC_TWO_OPERANDS
                         ;                       MOVC_OPERANDS
                         ;                       THREE_OPERANDS
                         ;                       JUMP_OPERAND
                         ;                       JUMP_TWO_OPERANDS
                         ;                       JUMP_ABSOLUTE_OPERAND
                         ;                       JUMP_LONG_OPERAND
                         ;                       MNEMONIC_INSTRUCTION_TAIL
                         ;                       MNEMONIC_INSTR_LIST_TAIL
                         ;                       ASSEMBLY_CMD
                         ;
                         ;               ASM.INC (INCLUDE FILE)
                         ;
                         ;                       START_DIVIDE
                         ;                       CALCULATE_INSTRUCTION_VALUE
                         ;                       UPDATE_OUR_CODE
                         ;                       GET_FIRST_OPERAND
                         ;                       CHECK_AND_SET_EXP_FLAG/SET_EXP_16_FLAG/SET_EXP_FLAG/CHECK_EXP_FLAG
                         ;                               SET_POUND_FLAG/CHECK_AND_SET_SECOND_EXP_FLAG/SET_SLASH_EXP_
                         FLAG
                         ;                               SET_REL_FLAG/GET_SECOND_EXP
                         ;
                         ;               ASMA.INC (INCLUDE FILE)
                         ;
                         ;                       CHECK_AND_CHANGE_ASM_PC
                         ;                       CHANGE_TO_INSTRUCTION_OP
                         ;
                         ;               SDKDSM.INC (INCLUDE FILE)
                         ;
                         ;                       DISASSEMBLY_CMD
                         ;                       GET_HASH_VALUE
                         ;                       OPERAND_BYTE_CHECK
                         ;                       DISPLAY_OPERAND
                         ;                       DISPLAY_COMMA
                         ;                       DISASSEMBLE
```

```
LOC  OBJ            LINE     SOURCE

                     200     ;
                     201     ;****************************************************************
                     202 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                    ;***************************************************************************
                    ; *                                                                         *
                    ; * THIS MODULE CONTAINS THE TABLES USED TO IMPLEMENT ASSEMBLY AND          *
                    ; * DISASSEMBLY:                                                            *
                    ; *                                                                         *
                    ; *   INSTRUCTION$CODE - A table of 256 address entries, one per opcode.    *
                    ; *     Each entry codes up for its opcode the mnemonic, first operand and  *
                    ; *     second operand.  Specifically, the entry equals                     *
                    ; *       M + F*MNEMONIC$FACTOR + S*MNEMONIC$FACTOR*OPERAND$FACTOR          *
                    ; *     WHERE                                                               *
                    ; *       M is the ordinal of the mnemonic in MNEMONIC$TAB,                 *
                    ; *       F is 0 if there are no operands; otherwise F is one more than the *
                    ; *         ordinal of the first operand in the OPERAND$TAB, and            *
                    ; *       S is 0 if there is no second operand; otherwise S is one more than *
                    ; *         the ordinal of the second operand in the OPERAND$TAB.           *
                    ; *     The entry OFFFFH in this table indicates the opcode is undefined.   *
                    ; *                                                                         *
                    ; *   MNEMONIC$TAB - A symbol table listing all the mnemonics (operands     *
                    ; *     not included).  The value associated with each is the instruction   *
                    ; *     format, a number between 7 and 15 corresponding to the instruction  *
                    ; *     tail in the grammar appropriate to the mnemonic.  The instruction   *
                    ; *     format is also needed to disassemble the instruction.  The formats  *
                    ; *     are:                                                                *
                    ; *       7 - No operands (e.g. RETI)                                       *
                    ; *       8 - One operand (e.g. CLR A)                                      *
                    ; *       9 - Two operands (e.g. ADD A,RO)                                  *
                    ; *      10 - MOVC - Two operands (e.g. MOVC A,@A + DPTR)                   *
                    ; *      11 - CJNE - Three operands (e.g. CJNE @RO,#56H,42H)                *
                    ; *      12 - JUMP - Relative - One operand (e.g. JC 44H)                   *
                    ; *      13 - JUMP - Relative - Two operands (e.g. JNB 5H,45H)              *
                    ; *      14 - Absolute CALL and JUMP (e.g. ACALL 341H)                      *
                    ; *      15 - Long CALL and JUMP (e.g. LJMP 4536H)                          *
                    ; *                                                                         *
                    ; *     The first mnemonics in this table are long call and jump(15), next  *
                    ; *     are the absolute call and jump instructions(14), then jump-relative *
                    ; *     one-operand instructions(13), the CJNE three operand instructions   *
                    ; *     11), the MOVC instructions(10), the two operand instructions(9),    *
                    ; *     the jump-relative one-operand instructions(12), the one operand     *
                    ; *     instructions(8), and the no operand instructions(7).  The jump-     *
                    ; *     relative one-operand instructions are in between the two operand    *
                    ; *     instructions and the one operand instructions because in the action *
                    ; *     SELECT$INSTRUCTION$TAIL it has to be determined if the mnemonic is  *
                    ; *     JNB, JB, JBC, SETB, CLR, or CPL since these six instructions, if they*
                    ; *     have an expression, have a bit expression so BIT$EXP must be set.   *
                    ; *                                                                         *
                    ; *   OPERAND$TAB - A symbol table listing the operands.  No value is       *
                    ; *     associated with them.  Only the ordinal in the table is important.  *
                    ; *                                                                         *
                    ; ***************************************************************************

                    ;
                    ;
                    ;       DECLARE
                    ;       UNDEF LIT 'OFFFFH';
                    ;
```

```
LOC  OBJ            LINE     SOURCE

                     258     ;  DECLARE
                     259     ;    MNEMONIC$TAB$HEAD TABLE PUBLIC DATA(
                     260     ;      .MNEMONIC$TAB+0EDH,
                     261     ;      0FFFFH - 0EDH),
                     262     ;
  FFFF               263     MNE_UNDEF        EQU      0FFFFH
                     264
                     265     MNEMONIC_TAB:             ;(*) BYTE PUBLIC DATA(
                     266
F587 0F              267             DB       0FH      ; LCALL
F588 0F              268             DB       0FH      ; LJMP
F589 0E              269             DB       0EH      ; ACALL
F58A 0E              270             DB       0EH      ; AJMP
F58B 0C              271             DB       0CH      ; SJMP
F58C 0C              272             DB       0CH      ; JNZ
F58D 0C              273             DB       0CH      ; JZ
F58E 0C              274             DB       0CH      ; JNC
F58F 0C              275             DB       0CH      ; JC
F590 0B              276             DB       0BH      ; CJNE
F591 0A              277             DB       0AH      ; MOVC
F592 09              278             DB       09H      ; MOVX
F593 09              279             DB       09H      ; XCHD
F594 09              280             DB       09H      ; XCH
F595 09              281             DB       09H      ; SUBB
F596 09              282             DB       09H      ; MOV
F597 09              283             DB       09H      ; XRL
F598 09              284             DB       09H      ; ANL
F599 09              285             DB       09H      ; ORL
F59A 09              286             DB       09H      ; ADDC
F59B 09              287             DB       09H      ; ADD
F59C 0D              288             DB       0DH      ; DJNZ
F59D 0D              289             DB       0DH      ; JNB
F59E 0D              290             DB       0DH      ; JB
F59F 0D              291             DB       0DH      ; JBC
F5A0 08              292             DB       08H      ; SETB
F5A1 08              293             DB       08H      ; CLR
F5A2 08              294             DB       08H      ; CPL
F5A3 08              295             DB       08H      ; DA
F5A4 08              296             DB       08H      ; POP
F5A5 08              297             DB       08H      ; SWAP
F5A6 08              298             DB       08H      ; PUSH
F5A7 08              299             DB       08H      ; MUL
F5A8 08              300             DB       08H      ; DIV
F5A9 08              301             DB       08H      ; JMP(@A+DPTR)
F5AA 08              302             DB       08H      ; RLC
F5AB 08              303             DB       08H      ; RL
F5AC 08              304             DB       08H      ; DEC
F5AD 08              305             DB       08H      ; RRC
F5AE 08              306             DB       08H      ; INC
F5AF 08              307             DB       08H      ; RR
F5B0 07              308             DB       07H      ; RETI
F5B1 07              309             DB       07H      ; RET
F5B2 07              310             DB       07H      ; NOP
                     311
                     312      ;  DECLARE   ; ORDINALS OF MNEMONICS IN MNEMONIC$TAB
```

```
LOC  OBJ           LINE      SOURCE

                    313
                    314
  0000              315      MNE_LCALL        EQU      00
  0001              316      MNE_LJMP         EQU      01
  0002              317      MNE_ACALL        EQU      02
  0003              318      MNE_AJMP         EQU      03
  0004              319      MNE_SJMP         EQU      04
  0005              320      MNE_JNZ          EQU      05
  0006              321      MNE_JZ           EQU      06
  0007              322      MNE_JNC          EQU      07
  0008              323      MNE_JC           EQU      08
  0009              324      MNE_CJNE         EQU      09
  000A              325      MNE_MOVC         EQU      10
  000B              326      MNE_MOVX         EQU      11
  000C              327      MNE_XCHD         EQU      12
  000D              328      MNE_XCH          EQU      13
  000E              329      MNE_SUBB         EQU      14
  000F              330      MNE_MOV          EQU      15
  0010              331      MNE_XRL          EQU      16
  0011              332      MNE_ANL          EQU      17
  0012              333      MNE_ORL          EQU      18
  0013              334      MNE_ADDC         EQU      19
  0014              335      MNE_ADD          EQU      20
  0015              336      MNE_DJNZ         EQU      21
  0016              337      MNE_JNB          EQU      22
  0017              338      MNE_JB           EQU      23
  0018              339      MNE_JBC          EQU      24
  0019              340      MNE_SETB         EQU      25
  001A              341      MNE_CLR          EQU      26
  001B              342      MNE_CPL          EQU      27
  001C              343      MNE_DA           EQU      28
  001D              344      MNE_POP          EQU      29
  001E              345      MNE_SWAP         EQU      30
  001F              346      MNE_PUSH         EQU      31
  0020              347      MNE_MUL          EQU      32
  0021              348      MNE_DIV          EQU      33
  0022              349      MNE_JMP          EQU      34
  0023              350      MNE_RLC          EQU      35
  0024              351      MNE_RL           EQU      36
  0025              352      MNE_DEC          EQU      37
  0026              353      MNE_RRC          EQU      38
  0027              354      MNE_INC          EQU      39
  0028              355      MNE_RR           EQU      40
  0029              356      MNE_RETI         EQU      41
  002A              357      MNE_RET          EQU      42
  002B              358      MNE_NOP          EQU      43;
                    359      ;*******************************************************************
                    360      ; DECLARE   ; MNEMONIC_FACTOR (I.E. 44) TIMES ORDINAL+1 OF FIRST OPERANDS IN
                    361      ;                OPERAND_TAB.
                    362
  002C              363      A_OP1            EQU      0044
  0058              364      ATR0_OP1         EQU      0088
  0084              365      ATR1_OP1         EQU      0132
  00B0              366      R0_OP1           EQU      0176
  00DC              367      R1_OP1           EQU      0220
```

```
LOC  OBJ            LINE     SOURCE

 0108                368     R2_OP1          EQU     0264
 0134                369     R3_OP1          EQU     0308
 0160                370     R4_OP1          EQU     0352
 018C                371     R5_OP1          EQU     0396
 01B8                372     R6_OP1          EQU     0440
 01E4                373     R7_OP1          EQU     0484
 0210                374     AB_OP1          EQU     0528
 023C                375     DPTR_OP1        EQU     0572
 0268                376     C_OP1           EQU     0616
 0294                377     ATDPTR_OP1      EQU     660
 02C0                378     BYTE_EXP8_OP1   EQU     0704
 02EC                379     BIT_EXP8_OP1    EQU     0748
 0370                380     EXP16_OP1       EQU     0880
 039C                381     EXP11_OP1       EQU     0924
 03C8                382     REL8_OP1        EQU     0968
 03F4                383     ATA_PLUS_DPTR_OP1       EQU     1012;
                     384     ;DECLARE;OPERAND_FACTOR*MNEMONIC_FACTOR(I.E.1056)TIMESORDINALSOF
                     385     ;SECONDOPERANDSINOPERAND_TAB
                     386
 0420                387     A_OP2           EQU     01056
 0840                388     ATR0_OP2        EQU     02112
 0C60                389     ATR1_OP2        EQU     03168
 1080                390     R0_OP2          EQU     04224
 14A0                391     R1_OP2          EQU     05280
 18C0                392     R2_OP2          EQU     06336
 1CE0                393     R3_OP2          EQU     07392
 2100                394     R4_OP2          EQU     08448
 2520                395     R5_OP2          EQU     09504
 2940                396     R6_OP2          EQU     10560
 2D60                397     R7_OP2          EQU     11616
 39C0                398     C_OP2           EQU     14784
 3DE0                399     ATDPTR_OP2      EQU     15840
 4200                400     BYTE_EXP8_OP2   EQU     16896
 4620                401     BIT_EXP8_OP2    EQU     17952
 4A40                402     POUND_EXP_OP2   EQU     19008
 4E60                403     SLASH_EXP_OP2   EQU     20064
 5280                404     EXP16_OP2       EQU     21120
 5AC0                405     REL8_OP2        EQU     23232
 5EE0                406     ATA_PLUS_DPTR_OP2       EQU     24288
 6300                407     ATA_PLUS_PC_OP2 EQU     25344;
                     408 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                             ;********** TEMPORARY VARIABLES **********
                             ;***** DATA ADDRESSES *****

----                                 DSEG
005B                                 ORG      (PARTIT_HI_LOW+1)
005B                         INSTRUCTION_VALUE:      DS       1
005C                         ORDINAL:                DS       1
005D                         OLD_ASM_PC_HIGH:        DS       1
005E                         OLD_ASM_PC_LOW:         DS       1
005F                         INSTRUCTION:            DS       1
0060                         REL_OFFSET_HIGH:        DS       1
0061                         REL_OFFSET_LOW:         DS       1
0062                         TEMP_SEC:               DS       1
0063                         FIRST_OPER_ORDINAL:     DS       1
0064                         SECOND_OPER_ORDINAL:    DS       1
0065                         THIRD_OPER_ORDINAL:     DS       1
0066                         CURRENT_OPERAND:        DS       1
0067                         NO_OF_OPERANDS_PRINTED: DS       1
0068                         EXPRESSIONS_PRINTED:    DS       1
0069                         MEMORY_TRACE_ADDR_HIGH: DS       1
006A                         MEMORY_TRACE_ADDR_LOW:  DS       1
006B                         NUMBER_OF_OPERANDS:     DS       1
006C                         OPERAND_CHECK:          DS       1
006D                         MNEMONIC_ORDINAL:       DS       1
006E                         DIVIDEND_HIGH:  DS       1
006F                         DIVIDEND_LOW:   DS       1
0070                         DIVISOR:        DS       1
0071                         QUOTIENT_HIGH:  DS       1
0072                         QUOTIENT_LOW:   DS       1


                             ;***** FLAG ADDRESSES *****

----                                 BSEG
0002                                 ORG      (LSTFLG+1)
0002                         BIT_EXP:                DBIT     1
0003                         FIRST_EXP:              DBIT     1
0004                         SECOND_EXP:             DBIT     1
----                                 CSEG

                             ;***** CONSTANTS *****
  0016                       JUMP_END                EQU      22
  001B                       BIT_END                 EQU      27
  002C                       MNEMONIC_FACTOR         EQU      44
  0018                       OPERAND_FACTOR          EQU      24
  00A5                       UNDEFINED_OPCODE        EQU      0A5H

                             INSTRUCTION_CODE:                                     ;Hash Table
                             ;00
F5B3 002B                            DW       MNE_NOP
F5B5 039F                            DW       MNE_AJMP+EXP11_OP1
F5B7 0371                            DW       MNE_LJMP+EXP16_OP1
F5B9 0054                            DW       MNE_RR+A_OP1
                             ;04
F5BB 0053                            DW       MNE_INC+A_OP1
```

```
OC  OBJ          LINE     SOURCE

5BD 02E7          464              DW     MNE_INC+BYTE_EXP8_OP1
5BF 007F          465              DW     MNE_INC+ATR0_OP1
5C1 00AB          466              DW     MNE_INC+ATR1_OP1
                  467     ;08
5C3 00D7          468              DW     MNE_INC+R0_OP1
5C5 0103          469              DW     MNE_INC+R1_OP1
5C7 012F          470              DW     MNE_INC+R2_OP1
5C9 015B          471              DW     MNE_INC+R3_OP1
                  472     ;0C
5CB 0187          473              DW     MNE_INC+R4_OP1
5CD 01B3          474              DW     MNE_INC+R5_OP1
5CF 01DF          475              DW     MNE_INC+R6_OP1
5D1 020B          476              DW     MNE_INC+R7_OP1
                  477     ;10
5D3 5DC4          478              DW     MNE_JBC+BIT_EXP8_OP1+REL8_OP2
5D5 039E          479              DW     MNE_ACALL+EXP11_OP1
5D7 0370          480              DW     MNE_LCALL+EXP16_OP1
5D9 0052          481              DW     MNE_RRC+A_OP1
                  482     ;14
5DB 0051          483              DW     MNE_DEC+A_OP1
5DD 02E5          484              DW     MNE_DEC+BYTE_EXP8_OP1
5DF 007D          485              DW     MNE_DEC+ATR0_OP1
5E1 00A9          486              DW     MNE_DEC+ATR1_OP1
                  487     ;18
5E3 00D5          488              DW     MNE_DEC+R0_OP1
5E5 0101          489              DW     MNE_DEC+R1_OP1
5E7 012D          490              DW     MNE_DEC+R2_OP1
5E9 0159          491              DW     MNE_DEC+R3_OP1
                  492     ;1C
5EB 0185          493              DW     MNE_DEC+R4_OP1
5ED 01B1          494              DW     MNE_DEC+R5_OP1
5EF 01DD          495              DW     MNE_DEC+R6_OP1
5F1 0209          496              DW     MNE_DEC+R7_OP1
                  497     ;20
5F3 5DC3          498              DW     MNE_JB+BIT_EXP8_OP1+REL8_OP2
5F5 039F          499              DW     MNE_AJMP+EXP11_OP1
5F7 002A          500              DW     MNE_RET
5F9 0050          501              DW     MNE_RL+A_OP1
                  502     ;24
5FB 4A80          503              DW     MNE_ADD+A_OP1+POUND_EXP_OP2
5FD 4240          504              DW     MNE_ADD+A_OP1+BYTE_EXP8_OP2
5FF 0880          505              DW     MNE_ADD+A_OP1+ATR0_OP2
601 0CA0          506              DW     MNE_ADD+A_OP1+ATR1_OP2
                  507     ;28
603 10C0          508              DW     MNE_ADD+A_OP1+R0_OP2
605 14E0          509              DW     MNE_ADD+A_OP1+R1_OP2
607 1900          510              DW     MNE_ADD+A_OP1+R2_OP2
609 1D20          511              DW     MNE_ADD+A_OP1+R3_OP2
                  512     ;2C
60B 2140          513              DW     MNE_ADD+A_OP1+R4_OP2
60D 2560          514              DW     MNE_ADD+A_OP1+R5_OP2
60F 2980          515              DW     MNE_ADD+A_OP1+R6_OP2
611 2DA0          516              DW     MNE_ADD+A_OP1+R7_OP2
                  517     ;30
613 5DC2          518              DW     MNE_JNB+BIT_EXP8_OP1+REL8_OP2
```

```
LOC  OBJ            LINE     SOURCE

F615 039E            519              DW      MNE_ACALL+EXP11_OP1
F617 0029            520              DW      MNE_RETI
F619 004F            521              DW      MNE_RLC+A_OP1
                     522     ;34
F61B 4A7F            523              DW      MNE_ADDC+A_OP1+POUND_EXP_OP2
F61D 423F            524              DW      MNE_ADDC+A_OP1+BYTE_EXP8_OP2
F61F 087F            525              DW      MNE_ADDC+A_OP1+ATR0_OP2
F621 0C9F            526              DW      MNE_ADDC+A_OP1+ATR1_OP2
                     527     ;38
F623 10BF            528              DW      MNE_ADDC+A_OP1+R0_OP2
F625 14DF            529              DW      MNE_ADDC+A_OP1+R1_OP2
F627 18FF            530              DW      MNE_ADDC+A_OP1+R2_OP2
F629 1D1F            531              DW      MNE_ADDC+A_OP1+R3_OP2
                     532     ;3C
F62B 213F            533              DW      MNE_ADDC+A_OP1+R4_OP2
F62D 255F            534              DW      MNE_ADDC+A_OP1+R5_OP2
F62F 297F            535              DW      MNE_ADDC+A_OP1+R6_OP2
F631 2D9F            536              DW      MNE_ADDC+A_OP1+R7_OP2
                     537     ;40
F633 03D0            538              DW      MNE_JC+REL8_OP1
F635 039F            539              DW      MNE_AJMP+EXP11_OP1
F637 06F2            540              DW      MNE_ORL+BYTE_EXP8_OP1+A_OP2
F639 4D12            541              DW      MNE_ORL+BYTE_EXP8_OP1+POUND_EXP_OP2
                     542     ;44
F63B 4A7E            543              DW      MNE_ORL+A_OP1+POUND_EXP_OP2
F63D 423E            544              DW      MNE_ORL+A_OP1+BYTE_EXP8_OP2
F63F 087E            545              DW      MNE_ORL+A_OP1+ATR0_OP2
F641 0C9E            546              DW      MNE_ORL+A_OP1+ATR1_OP2
                     547     ;48
F643 10BE            548              DW      MNE_ORL+A_OP1+R0_OP2
F645 14DE            549              DW      MNE_ORL+A_OP1+R1_OP2
F647 18FE            550              DW      MNE_ORL+A_OP1+R2_OP2
F649 1D1E            551              DW      MNE_ORL+A_OP1+R3_OP2
                     552     ;4C
F64B 213E            553              DW      MNE_ORL+A_OP1+R4_OP2
F64D 255E            554              DW      MNE_ORL+A_OP1+R5_OP2
F64F 297E            555              DW      MNE_ORL+A_OP1+R6_OP2
F651 2D9E            556              DW      MNE_ORL+A_OP1+R7_OP2
                     557     ;50
F653 03CF            558              DW      MNE_JNC+REL8_OP1
F655 039E            559              DW      MNE_ACALL+EXP11_OP1
F657 06F1            560              DW      MNE_ANL+BYTE_EXP8_OP1+A_OP2
F659 4D11            561              DW      MNE_ANL+BYTE_EXP8_OP1+POUND_EXP_OP2
                     562     ;54
F65B 4A7D            563              DW      MNE_ANL+A_OP1+POUND_EXP_OP2
F65D 423D            564              DW      MNE_ANL+A_OP1+BYTE_EXP8_OP2
F65F 087D            565              DW      MNE_ANL+A_OP1+ATR0_OP2
F661 0C9D            566              DW      MNE_ANL+A_OP1+ATR1_OP2
                     567     ;58
F663 10BD            568              DW      MNE_ANL+A_OP1+R0_OP2
F665 14DD            569              DW      MNE_ANL+A_OP1+R1_OP2
F667 18FD            570              DW      MNE_ANL+A_OP1+R2_OP2
F669 1D1D            571              DW      MNE_ANL+A_OP1+R3_OP2
                     572     ;5C
F66B 213D            573              DW      MNE_ANL+A_OP1+R4_OP2
```

```
LOC  OBJ          LINE     SOURCE

F66D 255D          574              DW       MNE_ANL+A_OP1+R5_OP2
F66F 297D          575              DW       MNE_ANL+A_OP1+R6_OP2
F671 2D9D          576              DW       MNE_ANL+A_OP1+R7_OP2
                   577     ;60
F673 03CE          578              DW       MNE_JZ+REL8_OP1
F675 039F          579              DW       MNE_AJMP+EXP11_OP1
F677 06F0          580              DW       MNE_XRL+BYTE_EXP8_OP1+A_OP2
F679 4D10          581              DW       MNE_XRL+BYTE_EXP8_OP1+POUND_EXP_OP2
                   582     ;64
F67B 4A7C          583              DW       MNE_XRL+A_OP1+POUND_EXP_OP2
F67D 423C          584              DW       MNE_XRL+A_OP1+BYTE_EXP8_OP2
F67F 087C          585              DW       MNE_XRL+A_OP1+ATR0_OP2
F681 0C9C          586              DW       MNE_XRL+A_OP1+ATR1_OP2
                   587     ;68
F683 10BC          588              DW       MNE_XRL+A_OP1+R0_OP2
F685 14DC          589              DW       MNE_XRL+A_OP1+R1_OP2
 687 18FC          590              DW       MNE_XRL+A_OP1+R2_OP2
 689 1D1C          591              DW       MNE_XRL+A_OP1+R3_OP2
                   592     ;6C
F68B 213C          593              DW       MNE_XRL+A_OP1+R4_OP2
F68D 255C          594              DW       MNE_XRL+A_OP1+R5_OP2
F68F 297C          595              DW       MNE_XRL+A_OP1+R6_OP2
F691 2D9C          596              DW       MNE_XRL+A_OP1+R7_OP2
                   597     ;70
F693 03CD          598              DW       MNE_JNZ+REL8_OP1
F695 039E          599              DW       MNE_ACALL+EXP11_OP1
F697 489A          600              DW       MNE_ORL+C_OP1+BIT_EXP8_OP2
F699 0416          601              DW       MNE_JMP+ATA_PLUS_DPTR_OP1
                   602     ;74
F69B 4A7B          603              DW       MNE_MOV+A_OP1+POUND_EXP_OP2
F69D 4D0F          604              DW       MNE_MOV+BYTE_EXP8_OP1+POUND_EXP_OP2
F69F 4AA7          605              DW       MNE_MOV+ATR0_OP1+POUND_EXP_OP2
F6A1 4AD3          606              DW       MNE_MOV+ATR1_OP1+POUND_EXP_OP2
                   607     ;78
 6A3 4AFF          608              DW       MNE_MOV+R0_OP1+POUND_EXP_OP2
F6A5 4B2B          609              DW       MNE_MOV+R1_OP1+POUND_EXP_OP2
F6A7 4B57          610              DW       MNE_MOV+R2_OP1+POUND_EXP_OP2
F6A9 4B83          611              DW       MNE_MOV+R3_OP1+POUND_EXP_OP2
                   612     ;7C
F6AB 4BAF          613              DW       MNE_MOV+R4_OP1+POUND_EXP_OP2
F6AD 4BDB          614              DW       MNE_MOV+R5_OP1+POUND_EXP_OP2
F6AF 4C07          615              DW       MNE_MOV+R6_OP1+POUND_EXP_OP2
F6B1 4C33          616              DW       MNE_MOV+R7_OP1+POUND_EXP_OP2
                   617     ;80
F6B3 03CC          618              DW       MNE_SJMP+REL8_OP1
F6B5 039F          619              DW       MNE_AJMP+EXP11_OP1
F6B7 4899          620              DW       MNE_ANL+C_OP1+BIT_EXP8_OP2
F6B9 6336          621              DW       MNE_MOVC+A_OP1+ATA_PLUS_PC_OP2
                   622     ;84
F6BB 0231          623              DW       MNE_DIV+AB_OP1
 6BD 44CF          624              DW       MNE_MOV+BYTE_EXP8_OP1+BYTE_EXP8_OP2
 6BF 0B0F          625              DW       MNE_MOV+BYTE_EXP8_OP1+ATR0_OP2
 6C1 0F2F          626              DW       MNE_MOV+BYTE_EXP8_OP1+ATR1_OP2
                   627     ;88
F6C3 134F          628              DW       MNE_MOV+BYTE_EXP8_OP1+R0_OP2
```

```
LOC  OBJ            LINE     SOURCE

F6C5 176F            629               DW      MNE_MOV+BYTE_EXP8_OP1+R1_OP2
F6C7 1B8F            630               DW      MNE_MOV+BYTE_EXP8_OP1+R2_OP2
F6C9 1FAF            631               DW      MNE_MOV+BYTE_EXP8_OP1+R3_OP2
                     632      ;8C
F6CB 23CF            633               DW      MNE_MOV+BYTE_EXP8_OP1+R4_OP2
F6CD 27EF            634               DW      MNE_MOV+BYTE_EXP8_OP1+R5_OP2
F6CF 2C0F            635               DW      MNE_MOV+BYTE_EXP8_OP1+R6_OP2
F6D1 302F            636               DW      MNE_MOV+BYTE_EXP8_OP1+R7_OP2
                     637      ;90
F6D3 54CB            638               DW      MNE_MOV+DPTR_OP1+EXP16_OP2
F6D5 039E            639               DW      MNE_ACALL+EXP11_OP1
F6D7 3CBB            640               DW      MNE_MOV+BIT_EXP8_OP1+C_OP2
F6D9 5F16            641               DW      MNE_MOVC+A_OP1+ATA_PLUS_DPTR_OP2
                     642      ;94
F6DB 4A7A            643               DW      MNE_SUBB+A_OP1+POUND_EXP_OP2
F6DD 423A            644               DW      MNE_SUBB+A_OP1+BYTE_EXP8_OP2
F6DF 087A            645               DW      MNE_SUBB+A_OP1+ATR0_OP2
F6E1 0C9A            646               DW      MNE_SUBB+A_OP1+ATR1_OP2
                     647      ;98
F6E3 10BA            648               DW      MNE_SUBB+A_OP1+R0_OP2
F6E5 14DA            649               DW      MNE_SUBB+A_OP1+R1_OP2
F6E7 18FA            650               DW      MNE_SUBB+A_OP1+R2_OP2
F6E9 1D1A            651               DW      MNE_SUBB+A_OP1+R3_OP2
                     652      ;9C
F6EB 213A            653               DW      MNE_SUBB+A_OP1+R4_OP2
F6ED 255A            654               DW      MNE_SUBB+A_OP1+R5_OP2
F6EF 297A            655               DW      MNE_SUBB+A_OP1+R6_OP2
F6F1 2D9A            656               DW      MNE_SUBB+A_OP1+R7_OP2
                     657      ;A0
F6F3 50DA            658               DW      MNE_ORL+C_OP1+SLASH_EXP_OP2
F6F5 039F            659               DW      MNE_AJMP+EXP11_OP1
F6F7 4897            660               DW      MNE_MOV+C_OP1+BIT_EXP8_OP2
F6F9 0263            661               DW      MNE_INC+DPTR_OP1
                     662      ;A4
F6FB 0230            663               DW      MNE_MUL+AB_OP1
F6FD FFFF            664               DW      MNE_UNDEF
F6FF 4267            665               DW      MNE_MOV+ATR0_OP1+BYTE_EXP8_OP2
F701 4293            666               DW      MNE_MOV+ATR1_OP1+BYTE_EXP8_OP2
                     667      ;A8
F703 42BF            668               DW      MNE_MOV+R0_OP1+BYTE_EXP8_OP2
F705 42EB            669               DW      MNE_MOV+R1_OP1+BYTE_EXP8_OP2
F707 4317            670               DW      MNE_MOV+R2_OP1+BYTE_EXP8_OP2
F709 4343            671               DW      MNE_MOV+R3_OP1+BYTE_EXP8_OP2
                     672      ;AC
F70B 436F            673               DW      MNE_MOV+R4_OP1+BYTE_EXP8_OP2
F70D 439B            674               DW      MNE_MOV+R5_OP1+BYTE_EXP8_OP2
F70F 43C7            675               DW      MNE_MOV+R6_OP1+BYTE_EXP8_OP2
F711 43F3            676               DW      MNE_MOV+R7_OP1+BYTE_EXP8_OP2
                     677      ;B0
F713 50D9            678               DW      MNE_ANL+C_OP1+SLASH_EXP_OP2
F715 039E            679               DW      MNE_ACALL+EXP11_OP1
F717 0307            680               DW      MNE_CPL+BIT_EXP8_OP1
F719 0283            681               DW      MNE_CPL+C_OP1
                     682      ;B4
F71B 4A75            683               DW      MNE_CJNE+A_OP1+POUND_EXP_OP2
```

```
LOC  OBJ             LINE     SOURCE

F71D 4235             684              DW      MNE_CJNE+A_OP1+BYTE_EXP8_OP2
F71F 4AA1             685              DW      MNE_CJNE+ATR0_OP1+POUND_EXP_OP2
F721 4ACD             686              DW      MNE_CJNE+ATR1_OP1+POUND_EXP_OP2
                      687     ;B8
F723 4AF9             688              DW      MNE_CJNE+R0_OP1+POUND_EXP_OP2
F725 4B25             689              DW      MNE_CJNE+R1_OP1+POUND_EXP_OP2
F727 4B51             690              DW      MNE_CJNE+R2_OP1+POUND_EXP_OP2
F729 4B7D             691              DW      MNE_CJNE+R3_OP1+POUND_EXP_OP2
                      692     ;BC
F72B 4BA9             693              DW      MNE_CJNE+R4_OP1+POUND_EXP_OP2
F72D 4BD5             694              DW      MNE_CJNE+R5_OP1+POUND_EXP_OP2
F72F 4C01             695              DW      MNE_CJNE+R6_OP1+POUND_EXP_OP2
F731 4C2D             696              DW      MNE_CJNE+R7_OP1+POUND_EXP_OP2
                      697     ;C0
F733 02DF             698              DW      MNE_PUSH+BYTE_EXP8_OP1
F735 039F             699              DW      MNE_AJMP+EXP11_OP1
F737 0306             700              DW      MNE_CLR+BIT_EXP8_OP1
F739 0282             701              DW      MNE_CLR+C_OP1
                      702     ;C4
F73B 004A             703              DW      MNE_SWAP+A_OP1
F73D 4239             704              DW      MNE_XCH+A_OP1+BYTE_EXP8_OP2
F73F 0879             705              DW      MNE_XCH+A_OP1+ATR0_OP2
F741 0C99             706              DW      MNE_XCH+A_OP1+ATR1_OP2
                      707     ;C8
F743 10B9             708              DW      MNE_XCH+A_OP1+R0_OP2
F745 14D9             709              DW      MNE_XCH+A_OP1+R1_OP2
F747 18F9             710              DW      MNE_XCH+A_OP1+R2_OP2
F749 1D19             711              DW      MNE_XCH+A_OP1+R3_OP2
                      712     ;CC
F74B 2139             713              DW      MNE_XCH+A_OP1+R4_OP2
F74D 2559             714              DW      MNE_XCH+A_OP1+R5_OP2
F74F 2979             715              DW      MNE_XCH+A_OP1+R6_OP2
F751 2D99             716              DW      MNE_XCH+A_OP1+R7_OP2
                      717     ;D0
F753 02DD             718              DW      MNE_POP+BYTE_EXP8_OP1
F755 039E             719              DW      MNE_ACALL+EXP11_OP1
F757 0305             720              DW      MNE_SETB+BIT_EXP8_OP1
F759 0281             721              DW      MNE_SETB+C_OP1
                      722     ;D4
F75B 0048             723              DW      MNE_DA+A_OP1
F75D 5D95             724              DW      MNE_DJNZ+BYTE_EXP8_OP1+REL8_OP2
F75F 0878             725              DW      MNE_XCHD+A_OP1+ATR0_OP2
F761 0C98             726              DW      MNE_XCHD+A_OP1+ATR1_OP2
                      727     ;D8
F763 5B85             728              DW      MNE_DJNZ+R0_OP1+REL8_OP2
F765 5BB1             729              DW      MNE_DJNZ+R1_OP1+REL8_OP2
F767 5BDD             730              DW      MNE_DJNZ+R2_OP1+REL8_OP2
F769 5C09             731              DW      MNE_DJNZ+R3_OP1+REL8_OP2
                      732     ;DC
F76B 5C35             733              DW      MNE_DJNZ+R4_OP1+REL8_OP2
F76D 5C61             734              DW      MNE_DJNZ+R5_OP1+REL8_OP2
F76F 5C8D             735              DW      MNE_DJNZ+R6_OP1+REL8_OP2
F771 5CB9             736              DW      MNE_DJNZ+R7_OP1+REL8_OP2
                      737     ;E0
F773 3E17             738              DW      MNE_MOVX+A_OP1+ATDPTR_OP2
```

```
LOC  OBJ            LINE     SOURCE

F775 039F            739              DW      MNE_AJMP+EXP11_OP1
F777 0877            740              DW      MNE_MOVX+A_OP1+ATR0_OP2
F779 0C97            741              DW      MNE_MOVX+A_OP1+ATR1_OP2
                     742     ;E4
F77B 0046            743              DW      MNE_CLR+A_OP1
F77D 423B            744              DW      MNE_MOV+A_OP1+BYTE_EXP8_OP2
F77F 087B            745              DW      MNE_MOV+A_OP1+ATR0_OP2
F781 0C9B            746              DW      MNE_MOV+A_OP1+ATR1_OP2
                     747     ;E8
F783 10BB            748              DW      MNE_MOV+A_OP1+R0_OP2
F785 14DB            749              DW      MNE_MOV+A_OP1+R1_OP2
F787 18FB            750              DW      MNE_MOV+A_OP1+R2_OP2
F789 1D1B            751              DW      MNE_MOV+A_OP1+R3_OP2
                     752     ;EC
F78B 213B            753              DW      MNE_MOV+A_OP1+R4_OP2
F78D 255B            754              DW      MNE_MOV+A_OP1+R5_OP2
F78F 297B            755              DW      MNE_MOV+A_OP1+R6_OP2
F791 2D9B            756              DW      MNE_MOV+A_OP1+R7_OP2
                     757     ;F0
F793 06BF            758              DW      MNE_MOVX+ATDPTR_OP1+A_OP2
F795 039E            759              DW      MNE_ACALL+EXP11_OP1
F797 0483            760              DW      MNE_MOVX+ATR0_OP1+A_OP2
F799 04AF            761              DW      MNE_MOVX+ATR1_OP1+A_OP2
                     762     ;F4
F79B 0047            763              DW      MNE_CPL+A_OP1
F79D 06EF            764              DW      MNE_MOV+BYTE_EXP8_OP1+A_OP2
F79F 0487            765              DW      MNE_MOV+ATR0_OP1+A_OP2
F7A1 04B3            766              DW      MNE_MOV+ATR1_OP1+A_OP2
                     767     ;F8
F7A3 04DF            768              DW      MNE_MOV+R0_OP1+A_OP2
F7A5 050B            769              DW      MNE_MOV+R1_OP1+A_OP2
F7A7 0537            770              DW      MNE_MOV+R2_OP1+A_OP2
F7A9 0563            771              DW      MNE_MOV+R3_OP1+A_OP2
                     772     ;FC
F7AB 058F            773              DW      MNE_MOV+R4_OP1+A_OP2
F7AD 05BB            774              DW      MNE_MOV+R5_OP1+A_OP2
F7AF 05E7            775              DW      MNE_MOV+R6_OP1+A_OP2
F7B1 0613            776              DW      MNE_MOV+R7_OP1+A_OP2;
                     777     ;*****************************************************************
                     778 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                     779     ;*****************************************************************
                     780     ;
                     781     ;    NAME: ONE_BYTE_TAIL/ MNEMONIC_SECOND_OPERAND_TAIL
                     782     ;
                     783     ;    ABSTRACT: This routine finds the opcode in the hash table which
                     784     ;       matches the token entered and sets the NUMBER_OF_BYTES according
                     785     ;       to the expression flags.  These are all one byte instructions
                     786     ;       regardless of actual NUMBER_OF_BYTES setting. Opcodes include
                     787     ;       NOP, RET etc..
                     788     ;
                     789     ;    INPUTS: None
                     790     ;
                     791     ;    OUTPUTS: OUR_CODE_LOW, OUR_CODE_HIGH
                     792     ;
                     793     ;    VARIABLES MODIFIED: None
                     794     ;
                     795     ;    ERROR EXITS: None
                     796     ;
                     797     ;    SUBROUTINES ACCESSED DIRECTLY: CALCULATE_INSTRUCTION_VALUE,
                     798     ;       CHECK_EXP_FLAG
                     799     ;
                     800     ;****************************************************************
                     801     ONE_BYTE_TAIL:
                     802     MNEMONIC_SECOND_OPERAND_TAIL:
F7B3 12FA00          803            CALL    CALCULATE_INSTRUCTION_VALUE
F7B6 02FAC9          804            JMP     CHECK_EXP_FLAG
                     805 +1  $EJECT
```

```
LOC  OBJ            SOURCE

                    ;*****************************************************************
                    ;
                    ;    NAME: MNEMONIC_FIRST_OPERAND
                    ;
                    ;    ABSTRACT: This routine sets flags to indicate how to assemble
                    ;       one byte instructions with one operand.  It gets a hash
                    ;       value and passes the expression or expressions to run time
                    ;       routines.  Instructions include: CLR A, INC A, JMP @A+DPTR,
                    ;       etc.
                    ;
                    ;    INPUTS: None
                    ;
                    ;    OUTPUTS: NUMBER_OF_BYTES, ORDINAL, OUR_CODE_HIGH, OUR_CODE_LOW,
                    ;       VALLOW
                    ;
                    ;    VARIABLES MODIFIED: A, ORDINAL, NUMBER_OF_BYTES
                    ;
                    ;    ERROR EXITS: 10H (ASSEMBLY SYNTAX ERROR)
                    ;
                    ;    SUBROUTINES ACCESSED DIRECTLY: GETOKE, ONE_BYTE_TAIL,
                    ;       UPDATE_OUR_CODE, CALCULATE_INSTRUCTION_VALUE,
                    ;       GET_FIRST_OPERAND,  CHECK_AND_SET_EXP_FLAG
                    ;
                    ;*****************************************************************
                    MNEMONIC_FIRST_OPERAND:
F7B9 12E056                 CALL    GETOKE
F7BC B40A14                 CJNE    A,#ATA_TOKE,MF00                ;Check for @A+DPTR
F7BF 12E056                 CALL    GETOKE
F7C2 B4056D                 CJNE    A,#PLUS_TOKE,ASERR
F7C5 12E056                 CALL    GETOKE
F7C8 B4A167                 CJNE    A,#DPTR_TOKE,ASERR
F7CB 755C17                 MOV     ORDINAL,#17H
F7CE 12FA28                 CALL    UPDATE_OUR_CODE
F7D1 80E0                   JMP     ONE_BYTE_TAIL
F7D3 300005         MF00:   JNB     B_O_T,MF01
F7D6 12FA63                 CALL    GET_FIRST_OPERAND
F7D9 80D8                   JMP     ONE_BYTE_TAIL
F7DB B4A10D         MF01:   CJNE    A,#DPTR_TOKE,MF02
F7DE 755C0D                 MOV     ORDINAL,#0DH
F7E1 12FA28                 CALL    UPDATE_OUR_CODE
F7E4 12FA00                 CALL    CALCULATE_INSTRUCTION_VALUE
F7E7 754D01                 MOV     NUMBER_OF_BYTES,#01H
F7EA 22                     RET
F7EB B40144         MF02:   CJNE    A,#NUMBER_TOKE,ASERR
F7EE 12FAA8                 CALL    CHECK_AND_SET_EXP_FLAG
F7F1 02FA00                 JMP     CALCULATE_INSTRUCTION_VALUE
                +1  $EJECT
```

```
LOC  OBJ            SOURCE

                    ;*******************************************************************
                    ;
                    ;    NAME: MNEMONIC_TWO_OPERANDS
                    ;
                    ;    ABSTRACT: This routine sets flags to indicate how to assemble
                    ;       two operand instructions with 2 or 3 bytes.  It gets a hash
                    ;       value and passes the expression or expressions to run time
                    ;       routines.  Instructions include: MOV DPTR,#<addr>,
                    ;       MOV <data addr>,<data addr>.
                    ;
                    ;    INPUTS: None
                    ;
                    ;    OUTPUTS: NUMBER_OF_BYTES, ORDINAL, OUR_CODE_LOW, OUR_CODE_HIGH,
                    ;       TEMP_SEC, VALLOW
                    ;
                    ;    VARIABLES MODIFIED: A, ORDINAL, TEMP_SEC, ERRNUM
                    ;
                    ;    ERROR EXITS: 03H (NUMBER EXPECTED)
                    ;                 10H (ASSEMBLY SYNTAX)
                    ;
                    ;    SUBROUTINES ACCESSED DIRECTLY: GETOKE, UPDATE_OUR_CODE, GET_COMMA,
                    ;       GETNUM, MNEMONIC_SECOND_OPERAND_TAIL, CALCULATE_INSTRUCTION_VALUE,
                    ;       GET_SECOND_OPERAND, SET_POUND_EXP_FLAG, SET_SLASH_EXP_FLAG,
                    ;       CHECK_AND_SET_SECOND_EXP_FLAG
                    ;
                    ;******************************************************************
                    MNEMONIC_TWO_OPERANDS:
F7F4 12E056                 CALL    GETOKE
F7F7 B4A118                 CJNE    A,#DPTR_TOKE,MT00
F7FA 755C0D                 MOV     ORDINAL,#0DH
F7FD 12FA28                 CALL    UPDATE_OUR_CODE
F800 12E06B                 CALL    GET_COMMA
F803 12E056                 CALL    GETOKE
F806 B40629                 CJNE    A,#POUND_TOKE,ASERR
F809 12E050                 CALL    GETNUM
F80C 12FAB8                 CALL    SET_EXP_16_FLAG
F80F 02FA00                 JMP     CALCULATE_INSTRUCTION_VALUE
F812 300006         MT00:   JNB     B_O_T,MFT00                     ;MNEMONIC_FIRST_TWO_OPERANDS
F815 12FA63                 CALL    GET_FIRST_OPERAND
F818 02F824                 JMP     MT01
F81B B40114         MFT00:  CJNE    A,#NUMBER_TOKE,ASERR
F81E 12FAC1                 CALL    SET_EXP_FLAG
F821 854A62                 MOV     TEMP_SEC,VALLOW
F824 12E06B         MT01:   CALL    GET_COMMA
F827 12E056                 CALL    GETOKE                          ;MNEMONIC_SECOND_OPERAND
F82A 30000B                 JNB     B_O_T,MS00
F82D 12FB0D                 CALL    GET_SECOND_OPERAND
F830 8081                   JMP     MNEMONIC_SECOND_OPERAND_TAIL
F832 754310         ASERR:  MOV     ERRNUM,#10H                     ;Assembly syntax
F835 02E05F                 JMP     ERROR
F838 E548           MS00:   MOV     A,TOKSTR
F83A B40609                 CJNE    A,#POUND_TOKE,MS01
F83D 12FAE8                 CALL    SET_POUND_EXP_FLAG
F840 12E050                 CALL    GETNUM
F843 02F7B3                 JMP     MNEMONIC_SECOND_OPERAND_TAIL
```

```
LOC  OBJ            LINE     SOURCE

F846 B40309          908     MS01:   CJNE    A,#BAR_TOKE,MS02
F849 12FAFC          909             CALL    SET_SLASH_EXP_FLAG
F84C 12E050          910             CALL    GETNUM
F84F 02F7B3          911             JMP     MNEMONIC_SECOND_OPERAND_TAIL
F852 754303          912     MS02:   MOV     ERRNUM,#03H
F855 B4016A          913             CJNE    A,#NUMBER_TOKE,TOERR
F858 12FAF0          914             CALL    CHECK_AND_SET_SECOND_EXP_FLAG
F85B 02F7B3          915             JMP     MNEMONIC_SECOND_OPERAND_TAIL
                     916 +1  $EJECT
```

```
LOC  OBJ           SOURCE

                   ;*****************************************************************
                   ;
                   ;    NAME: MOVC_OPERANDS
                   ;
                   ;    ABSTRACT: This routine divides operands into one of two possible
                   ;       cases and modifies the hash value. Instructions are
                   ;       MOVC A,@A+DPTR and MOVC A,@A+PC.
                   ;
                   ;    INPUTS: None
                   ;
                   ;    OUTPUTS: ORDINAL, OUR_CODE_LOW, OUR_CODE_HIGH
                   ;
                   ;    VARIABLES MODIFIED: A, ORDINAL
                   ;
                   ;    ERROR EXITS: 10H (ASSEMBLY SYNTAX)
                   ;
                   ;    SUBROUTINES ACCESSED DIRECTLY: GETOKE, GET_FIRST_OPERAND, GET_COMMA,
                   ;       UPDATE_OUR_CODE, ONE_BYTE_TAIL
                   ;
                   ;*****************************************************************
                   MOVC_OPERANDS:
F85E 12E056                CALL    GETOKE
F861 3000CE                JNB     B_O_T,ASERR
F864 12FA63                CALL    GET_FIRST_OPERAND
F867 12E06B                CALL    GET_COMMA
F86A 12E056                CALL    GETOKE                  ;MOVC_TAIL
F86D B40AC2                CJNE    A,#ATA_TOKE,ASERR
F870 12E056                CALL    GETOKE
F873 12E056                CALL    GETOKE
F876 B4A109                CJNE    A,#DPTR_TOKE,MTO
F879 755C17                MOV     ORDINAL,#17H
F87C 12FA28                CALL    UPDATE_OUR_CODE
F87F 02F7B3                JMP     ONE_BYTE_TAIL
 882 B4A0AD        MTO:    CJNE    A,#PC_TOKE,ASERR
 885 755C18                MOV     ORDINAL,#18H
F888 12FA28                CALL    UPDATE_OUR_CODE
F88B 02F7B3                JMP     ONE_BYTE_TAIL
              +1   $EJECT
```

```
LOC  OBJ            SOURCE

                    ;****************************************************************
                    ;
                    ;    NAME: THREE_OPERANDS
                    ;
                    ;    ABSTRACT: This routine parses the opcodes and modifies the
                    ;       hash value accordingly.  It saves the data address or
                    ;       immediate data field and the destination address.  Instructions
                    ;       are CJNE @R0,#<data>,<addr>; CJNE @R1,#<data>,<addr>;
                    ;       CJNE A,#<data>,<addr>; CJNE A,<data>,<addr>; CJNE Rn,#<data>,<data>
                    ;
                    ;    INPUTS: None
                    ;
                    ;    OUTPUTS: ORIDNAL, OUR_CODE_LOW, OUR_CODE_HIGH, VALLOW, TEMP_SEC,
                    ;       NUMBER_OF_BYTES
                    ;
                    ;    VARIABLES MODIFIED: NUMBER_OF_BYTES, TEMP_SEC, A
                    ;
                    ;    ERROR EXITS: 10H (ASSEMBLY SYNTAX)
                    ;                 03H (NUMBER EXPECTED)
                    ;
                    ;    SUBROUTINES ACCESSED DIRECTLY: GETOKE, GET_FIRST_OPERAND,
                    ;       GET_COMMA, SET_POUND_EXP_FLAG, CHECK_AND_SET_SECOND_EXP_FLAG,
                    ;       GETNUM, CALCULATE_INSTRUCTION_VALUE, ERROR
                    ;
                    ;****************************************************************
                    THREE_OPERANDS:
F88E 12E056                 CALL    GETOKE
F891 30009E                 JNB     B_O_T,ASERR
F894 12FA63                 CALL    GET_FIRST_OPERAND
F897 12E06B                 CALL    GET_COMMA
F89A 12E056                 CALL    GETOKE                          ;SECOND_THREE_OPERANDS
F89D B40609                 CJNE    A,#POUND_TOKE,ST01
F8A0 12FAE8                 CALL    SET_POUND_EXP_FLAG
F8A3 12E050                 CALL    GETNUM
F8A6 02F8B2                 JMP     STORET
F8A9 754303         ST01:   MOV     ERRNUM,#03H                     ;Number expected
F8AC B40113                 CJNE    A,#NUMBER_TOKE,TOERR
F8AF 12FAF0                 CALL    CHECK_AND_SET_SECOND_EXP_FLAG
F8B2 854A62         STORET: MOV     TEMP_SEC,VALLOW
F8B5 12E06B                 CALL    GET_COMMA
F8B8 12E050                 CALL    GETNUM
F8BB 12FA00                 CALL    CALCULATE_INSTRUCTION_VALUE
F8BE 754D05                 MOV     NUMBER_OF_BYTES,#05H
F8C1 22                     RET
F8C2 02E05F         TOERR:  JMP     ERROR
F8C5 22                     RET
                +1  $EJECT
```

```
LOC  OBJ            SOURCE

                    ;*****************************************************************
                    ;
                    ;    NAME: JUMP_OPERAND
                    ;
                    ;    ABSTRACT: This routine gets the destination for a jump from
                    ;       the command line and sets the relative operand flag to
                    ;       indicate the method of assembly. Instructions are SJMP<addr>,
                    ;       JNC<addr>, JC<addr>, JZ<addr>, JNZ<addr>.
                    ;
                    ;    INPUTS: None
                    ;
                    ;    OUTPUTS: OUR_CODE_LOW, OUR_CODE_HIGH, VALLOW
                    ;
                    ;    VARIABLES MODIFIED: None
                    ;
                    ;    ERROR EXITS: None
                    ;
                    ;    SUBROUTINES ACCESSED DIRECTLY: GETNUM, SET_REL_FLAG,
                    ;       CALCULATE_INSTRUCTION_VALUE
                    ;
                    ;*****************************************************************
                    JUMP_OPERAND:
F8C6 12E050                 CALL    GETNUM
F8C9 12FB04                 CALL    SET_REL_FLAG
F8CC 02FA00                 JMP     CALCULATE_INSTRUCTION_VALUE
               +1   $EJECT
```

```
LOC  OBJ             SOURCE
                     ;******************************************************************
                     ;
                     ;    NAME: JUMP_TWO_OPERANDS
                     ;
                     ;    ABSTRACT: This routine gets an expression for an address bit
                     ;       which will be tested by the jump.  It modifies OUR_CODE and
                     ;       REL_FLAG to indicate proper means of assembly, then gets
                     ;       the destination address. Instructions are JB<bit addr>,<addr>;
                     ;       JBC<bit addr>,<addr>; JNB<bit addr>,<addr>; DJNZ<bit addr>,<addr>;
                     ;       DJNZ Rn,<addr>.
                     ;
                     ;    INPUTS: B_O_T
                     ;
                     ;    OUTPUTS: NUMBER_OF_BYTES, TEMP_SEC, OUR_CODE_LOW, OUR_CODE_HIGH,
                     ;       VALLOW
                     ;
                     ;    VARIABLES MODIFIED: NUMBER_OF_BYTES, TEMP_SEC
                     ;
                     ;    ERROR EXITS: None
                     ;
                     ;    SUBROUTINES ACCESSED DIRECTLY: GETOKE, GET_FIRST_OPERAND,
                     ;       SET_REL_FLAG, CALCULATE_INSTRUCTION_VALUE, CHECK_AND_SET_EXP_FLAG,
                     ;       GET_COMMA, GETNUM
                     ;
                     ;******************************************************************
                     JUMP_TWO_OPERANDS:
F8CF 12E056                  CALL    GETOKE
F8D2 30000C                  JNB     B_O_T,JT00
F8D5 12FA63                  CALL    GET_FIRST_OPERAND
F8D8 12FB04                  CALL    SET_REL_FLAG
F8DB 12FA00                  CALL    CALCULATE_INSTRUCTION_VALUE
F8DE 02F8F0                  JMP     JTRET
F8E1 B401DE          JT00:   CJNE    A,#NUMBER_TOKE,TOERR
F8E4 12FAA8                  CALL    CHECK_AND_SET_EXP_FLAG
F8E7 12FB04                  CALL    SET_REL_FLAG
F8EA 12FA00                  CALL    CALCULATE_INSTRUCTION_VALUE
F8ED 754D05                  MOV     NUMBER_OF_BYTES,#05H
F8F0 854A62          JTRET:  MOV     TEMP_SEC,VALLOW
F8F3 12E06B                  CALL    GET_COMMA
F8F6 02E050                  JMP     GETNUM
                 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                    1069     ;******************************************************************
                    1070     ;
                    1071     ;    NAME: JUMP_ABSOLUTE_OPERAND
                    1072     ;
                    1073     ;    ABSTRACT: This routine gets the destination address and
                    1074     ;       modifies OUR_CODE to indicate that the upper 3 bits of
                    1075     ;       address must be included in the final opcode.  Instructions
                    1076     ;       of this type are AJMP <addr>, ACALL <addr>.
                    1077     ;
                    1078     ;    INPUTS: None
                    1079     ;
                    1080     ;    OUTPUTS: ORDINAL, NUMBER_OF_BYTES, OUR_CODE_LOW, OUR_CODE_HIGH,
                    1081     ;       VALLOW, VALHGH
                    1082     ;
                    1083     ;    VARIABLES MODIFIED: ORDINAL, NUMBER_OF_BYTES
                    1084     ;
                    1085     ;    ERROR EXITS: None
                    1086     ;
                    1087     ;    SUBROUTINES ACCESSED DIRECTLY: GETNUM, UPDATE_OUR_CODE,
                    1088     ;       CALCULATE_INSTRUCTION_VALUE
                    1089     ;
                    1090     ;******************************************************************
                    1091     JUMP_ABSOLUTE_OPERAND:
F8F9 12E050         1092             CALL    GETNUM
F8FC 755C15         1093             MOV     ORDINAL,#15H                 ;SET_EXP_11_FLAG
F8FF 12FA28         1094             CALL    UPDATE_OUR_CODE              ;2K page jump
F902 754D06         1095             MOV     NUMBER_OF_BYTES,#06H         ;Absolute instruction
F905 02FA00         1096             JMP     CALCULATE_INSTRUCTION_VALUE
                    1097 +1  $EJECT
```

```
LOC  OBJ            SOURCE

                    ;*******************************************************************
                    ;
                    ;    NAME: JUMP_LONG_OPERAND
                    ;
                    ;    ABSTRACT: This routine gets the destination address and sets
                    ;       the 16 bit expression flag.  It then searches the hash table
                    ;       for a matching opcode.  Instructions are LCALL <addr> and
                    ;       LJMP <addr>.
                    ;
                    ;    INPUTS: None
                    ;
                    ;    OUTPUTS: ORDINAL, NUMBER_OF_BYTES, OUR_CODE_LOW, OUR_CODE_HIGH,
                    ;       VALHGH, VALLOW
                    ;
                    ;    VARIABLES MODIFIED: None
                    ;
                    ;    ERROR EXITS: None
                    ;
                    ;    SUBROUTINES ACCESSED DIRECTLY: GETNUM, SET_EXP_16_FLAG,
                    ;       CALCULATE_INSTRUCTION_VALUE
                    ;
                    ;*******************************************************************
                    JUMP_LONG_OPERAND:
F908 12E050                 CALL    GETNUM
F90B 12FAB8                 CALL    SET_EXP_16_FLAG
F90E 02FA00                 JMP     CALCULATE_INSTRUCTION_VALUE
                    $EJECT
```

```
LOC  OBJ         SOURCE

                 ;*************************************************************
                 ;
                 ;    NAME: MNEMONIC_INSTRUCTION_TAIL
                 ;
                 ;    ABSTRACT: This routine selects the type of instruction as determined
                 ;       by the MNEMONIC INSTRUCTION TABLE and calls the handler for the
                 ;       type specified.  The handler completes the parsing of the command
                 ;       line and does the hash table look-up.
                 ;
                 ;    INPUTS: INSTRUCTION_VALUE
                 ;
                 ;    OUTPUTS: ORDINAL, VALLOW, VALHGH, TEMP_SEC, NUMBER_OF_BYTES,
                 ;       OUR_CODE_LOW, OUR_CODE_HIGH
                 ;
                 ;    VARIABLES MODIFIED: DPTR, A, C, B, ERRNUM
                 ;
                 ;    ERROR EXITS: 10H (ASSEMBLY SYNTAX)
                 ;
                 ;    SUBROUTINES ACCESSED DIRECTLY: ONE_BYTE_TAIL, MNEMONIC_FIRST_OPERAND,
                 ;       MNEMONIC_TWO_OPERANDS, MOVC_OPERANDS, THREE_OPERANDS, JUMP_OPERAND,
                 ;       JUMP_TWO_OPERANDS, JUMP_ABSOLUTE_OPERAND, JUMP_LONG_OPERAND
                 ;
                 ;*************************************************************
                 MNEMONIC_INSTRUCTION_TAIL:
F911 754310              MOV     ERRNUM,#10H
F914 90F921              MOV     DPTR,#MIT_JMP_TBL
F917 E55B                MOV     A,INSTRUCTION_VALUE
F919 C3                  CLR     C
F91A 9407                SUBB    A,#07H
F91C 75F003              MOV     B,#03H
F91F A4                  MUL     AB                      ;Mult by 3, each tbl entry is 3 bytes
F920 73                  JMP     @A+DPTR
                 MIT_JMP_TBL:
F921 02F7B3              LJMP    ONE_BYTE_TAIL
F924 02F7B9              LJMP    MNEMONIC_FIRST_OPERAND
F927 02F7F4              LJMP    MNEMONIC_TWO_OPERANDS
F92A 02F85E              LJMP    MOVC_OPERANDS
F92D 02F88E              LJMP    THREE_OPERANDS
F930 02F8C6              LJMP    JUMP_OPERAND
F933 02F8CF              LJMP    JUMP_TWO_OPERANDS
F936 02F8F9              LJMP    JUMP_ABSOLUTE_OPERAND
F939 02F908              LJMP    JUMP_LONG_OPERAND
              +1 $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                    1168     ;****************************************************************
                    1169     ;
                    1170     ;    NAME: MNEMONIC_INSTR_LIST_TAIL
                    1171     ;
                    1172     ;    ABSTRACT: This routine sets up information to be used in later
                    1173     ;       processing of the mnemonic by deciphering the information
                    1174     ;       in MNEMONIC_TAB then the call to MNEMONIC_INSTRUCTION_TAIL and
                    1175     ;       CHANGE_TO_INSTRUCTION_OP completes the assembly.
                    1176     ;
                    1177     ;    INPUTS: TOKSTR, ASM_PC_LOW, ASM_PC_HIGH
                    1178     ;
                    1179     ;    OUTPUTS: Code memory locations pointed to by ASM_PC.
                    1180     ;
                    1181     ;    VARIABLES MODIFIED: BIT_EXP, FIRST_EXP, SECOND_EXP, A, C, DPTR,
                    1182     ;       INSTRUCTION_VALUE, OUR_CODE_LOW
                    1183     ;
                    1184     ;    ERROR EXITS: None
                    1185     ;
                    1186     ;    SUBROUTINES ACCESSED DIRECTLY: MNEMONIC_INSTRUCTION_TAIL,
                    1187     ;       CHANGE_TO_INSTRUCTION_OP
                    1188     ;
                    1189     ;****************************************************************
                    1190     MNEMONIC_INSTR_LIST_TAIL:
F93C C202           1191             CLR     BIT_EXP                         ;MNEMONIC_INSTR
F93E C203           1192             CLR     FIRST_EXP                       ;Initialize flags
F940 C204           1193             CLR     SECOND_EXP
F942 754D00         1194             MOV     NUMBER_OF_BYTES,#00H
F945 754E00         1195             MOV     OUR_CODE_HIGH,#00H              ;SELECT_INSTRUCTION_TAIL
F948 C3             1196             CLR     C
F949 E548           1197             MOV     A,TOKSTR
F94B 9410           1198             SUBB    A,#OFST
F94D F54F           1199             MOV     OUR_CODE_LOW,A
F94F 90F587         1200             MOV     DPTR,#MNEMONIC_TAB
F952 93             1201             MOVC    A,@A+DPTR
F953 F55B           1202             MOV     INSTRUCTION_VALUE,A             ;Search for corresponding match using
F955 7416           1203             MOV     A,#JUMP_END                     ;look-up and check to see if mnemonic
F957 B54F0C         1204             CJNE    A,OUR_CODE_LOW,OUR_GTRTHN       ;is JBC, JB, JNB, CPL or SETB.  If is,
                    1205     CONT_OUR_CODE:
F95A C3             1206             CLR     C                               ;then set BIT_EXP.  If OUR_CODE=#JUMP_EN
F95B E54F           1207             MOV     A,OUR_CODE_LOW                  ;Then fall thru or check for OUR_CODE>#JUMP
                             _END
F95D 941C           1208             SUBB    A,#(BIT_END+1)                  ;Check that OUR_CODE<=BIT_END
F95F 5007           1209             JNC     END_SELECT_INSTRUCTION_TAIL
F961 D202           1210             SETB    BIT_EXP
F963 02F968         1211             JMP     END_SELECT_INSTRUCTION_TAIL
                    1212     OUR_GTRTHN:
F966 40F2           1213             JC      CONT_OUR_CODE
                    1214     END_SELECT_INSTRUCTION_TAIL:
F968 E54F           1215             MOV     A,OUR_CODE_LOW
F96A B42B03         1216             CJNE    A,#2BH,MI0
F96D 02F972         1217             JMP     MI1
F970 5035           1218     MI0:    JNC     AMTERR
F972 3111           1219     MI1:    CALL    MNEMONIC_INSTRUCTION_TAIL
F974 02FB82         1220             JMP     CHANGE_TO_INSTRUCTION_OP
                    1221 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                    1222     ;*****************************************************************
                    1223     ;
                    1224     ;     NAME: ASSEMBLY_CMD
                    1225     ;
                    1226     ;     ABSTRACT: This routine parses the rest of the command line
                    1227     ;        for ORG or carraige return and enters the ASM mode.  Once
                    1228     ;        in ASM mode, control remains here in a loop assembling
                    1229     ;        instructions until a carraige return is found on a line by
                    1230     ;        itself.
                    1231     ;
                    1232     ;     INPUTS: None
                    1233     ;
                    1234     ;     OUTPUTS: Code memeory locations pointed to in ORG clause or
                    1235     ;        pre-existing ASM_PC setting.
                    1236     ;
                    1237     ;     VARIABLES MODIFIED: ASM_PC_HIGH, ASM_PC_LOW, A, POINTO, PARAM1,
                    1238     ;        ERRNUM
                    1239     ;
                    1240     ;     ERROR EXITS: 10H (ASSEMBY SYNTAX)
                    1241     ;
                    1242     ;     SUBROUTINES ACCESSED DIRECTLY: GETOKE, NEWLINE, GETNUM,
                    1243     ;        SAVE_AND_DISPLAY, ERROR, MNEMONIC_INSTR_LIST_TAIL, GETEOL
                    1244     ;
                    1245     ;*****************************************************************
                    1246     ASSEMBLY_CMD:
F977 755205         1247             MOV     LINE_START,#05H
F97A 12E056         1248             CALL    GETOKE
F97D B4D40F         1249             CJNE    A,#ORG_TOKE,AMTO
F980 12E050         1250             CALL    GETNUM                  ;Get past address
F983 85494B         1251             MOV     ASM_PC_HIGH,VALHGH
F986 854A4C         1252             MOV     ASM_PC_LOW,VALLOW
F989 12E00F         1253             CALL    NEWLINE
F98C 12E056         1254             CALL    GETOKE
F98F B40715         1255     AMTO:   CJNE    A,#EOL_TOKE,AMTERR
F992 7824           1256     AMT1:   MOV     POINTO,#LINBUF
F994 AA4B           1257             MOV     PARAM1,ASM_PC_HIGH
F996 12E05C         1258             CALL    SAVE_AND_DISPLAY
F999 AA4C           1259             MOV     PARAM1,ASM_PC_LOW
F99B 12E05C         1260             CALL    SAVE_AND_DISPLAY
F99E 7620           1261             MOV     @POINTO,#' '
F9A0 12E056         1262             CALL    GETOKE
F9A3 B40707         1263             CJNE    A,#EOL_TOKE,AMT2
F9A6 22             1264             RET
F9A7 754310         1265     AMTERR: MOV     ERRNUM,#10H             ;Assembly syntax
F9AA 02E05F         1266             JMP     ERROR
F9AD 313C           1267     AMT2:   CALL    MNEMONIC_INSTR_LIST_TAIL
F9AF 12E053         1268             CALL    GETEOL
F9B2 80DE           1269             JMP     AMT1
                    1270
                    1271 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                      +1   $INCLUDE(:F1:ASM.INC)
                 =1        ;*****************************************************************
                 =1        ;
                 =1        ;       This is the include file called ASM.INC.  It contains the
                 =1        ;       following subroutines in order:
                 =1        ;
                 =1        ;               START_DIVIDE
                 =1        ;               CALCULATE_INSTRUCTION_VALUE
                 =1        ;               UPDATE_OUR_CODE
                 =1        ;               GET_FIRST_OPERAND
                 =1        ;               CHECK_AND_SET_EXP_FLAG
                 =1        ;                       SET_EXP_16_FLAG
                 =1        ;                       SET_EXP_FLAG
                 =1        ;                       CHECK_EXP_FLAG
                 =1        ;                       SET_POUND_EXP_FLAG
                 =1        ;                       CHECK_AND_SET_SECOND_EXP_FLAG
                 =1        ;                       SET_SLASH_EXP_FLAG
                 =1        ;                       SET_REL_FLAG
                 =1        ;               GET_SECOND_EXP
                 =1        ;
                 =1        ;*****************************************************************
                 =1   +1   $EJECT
```

```
LOC  OBJ          LINE     SOURCE

             =1         ;/**************************************************************************
             =1         ; *                                                                        *
             =1         ; * This module contains most procedures needed to implement the           *
             =1         ; * assembler which processes the ASM command. The rest are contained      *
             =1         ; * in the ASMA module.                                                    *
             =1         ; *                                                                        *
             =1         ; *   INSTRUCTION_VALUE - Public variable used at parse time.  The         *
             =1         ; *     instruction is assembled into it.                                  *
             =1         ; *                                                                        *
             =1         ; *------------------------------------------------------------------------*
             =1         ; *                                                                        *
             =1         ; * The assembler consists of three pieces:                                *
             =1         ; * - Tables in the module ASM_TBL code which contain the details of the   *
             =1         ; *   8051 assembly language,                                              *
             =1         ; * - Parse time procedures in this module use these tables to:            *
             =1         ; *   -Set up flags and variables to control actual memory     *
             =1         ; *    writing operations, search the tables for matched to the hashed     *
             =1         ; *    command line.
             =1         ; *   - Assemble the instruction as if any expression, immediate data, or  *
             =1         ; *       jump addresses are zero (they are evaluated at run-time).        *
             =1         ; * - Procedures selected by the above parse time procedures determine:    *
             =1         ; *   - What the instruction format is,                                    *
             =1         ; *   - How to combine the expressions, immediate data, or jump addresses  *
             =1         ; *       (if any) after being calculated with the instruction value       *
             =1         ; *       assembled at parse time to create the final result of the        *
             =1         ; *       assembly in memory.                                              *
             =1         ; *                                                                        *
             =1         ; * The opcode is found by generating a hash value as the parser scans the *
             =1         ; * instruction.  How the hash value is calculated is discussed in ASM_TBL. *
             =1         ; * All the hash values are stored in the table, #INSTRUCTION_CODE, and the *
             =1         ; * ordinal corresponding to a hash value is the opcode for that instruction.*
             =1         ; * Except for absolute instructions, in which case the opcode is further   *
             =1         ; * calculated in CHANGE_TO_INSTRUCTION_OP, NUMBER_OF_BYTES contains either *
             =1         ; * the actual number of bytes in the instruction or a code to enable      *
             =1         ; * CHANGE_TO_INSTRUCTION_OP to write the correct number of bytes in the   *
             =1         ; * correct order.  See CHANGE_TO_INSTRUCTION_OP for more details.         *
             =1         ; *                                                                        *
             =1         ; * Parsing the command line leaves the opcode in INSTRUCTION_VALUE at run  *
             =1         ; * time.  CHANGE_TO_INSTRUCTION_OP is called after each command line      *
             =1         ; * to process the type of instruction appropriately to write it out to    *
             =1         ; * memory.  Relative offsets and 2K jump or calls are generated here.     *
             =1         ; *                                                                        *
             =1         ; * Details on the use of the tables in the assembly can be found in the   *
             =1         ; * documentation in the ASM_TBL module.                                   *
             =1         ; *                                                                        *
             =1         ; *------------------------------------------------------------------------*
             =1         ; *                                                                        *
             =1         ; * In the operand_table the basic operands(ex. C,A,R0-R7,etc.) have the   *
             =1         ; * ordinal+1 values of 1-15 but the values 16-24 were used to represent   *
             =1         ; * certain expressions as follows:                                        *
             =1         ; *                                                                        *
             =1         ; *   16 - BYTE EXP8                  21 - EXP11                           *
             =1         ; *   17 - BIT EXP8                   22 - RELATIVE OFFSET EXPRESSION      *
             =1         ; *   18 - IMMEDIATE(#) EXP8          23 - @A+DPTR                         *
             =1         ; *   19 - COMPLEMENT(/) EXP8         24 - @A+PC                           *
```

```
LOC  OBJ            LINE     SOURCE

              =1  1349      ; *   20 - EXP16                                                          *
              =1  1350      ; *                                                                       *
              =1  1351      ; *-----------------------------------------------------------------------*
              =1  1352      ; *                                                                       *
              =1  1353      ; * A problem arose which made the software more involved: determining if  *
              =1  1354      ; * the eight bit expression was a bit or byte expression. Since disassembly *
              =1  1355      ; * uses the same tables as assembly the hash values had to be precise.    *
              =1  1356      ; * The following instructions had bit expressions:                      *
              =1  1357      ; *                                                                       *
              =1  1358      ; *   JBC  BIT EXP,CODE EXP       ORL C,BIT EXP        MOV BIT EXP,C       *
              =1  1359      ; *   JB   BIT EXP,CODE EXP       ANL C,BIT EXP                            *
              =1  1360      ; *   JNB  BIT EXP,CODE EXP       MOV C,BIT EXP                            *
              =1  1361      ; *   CLR  BIT EXP,CODE EXP                                                *
              =1  1362      ; *   CPL  BIT EXP,CODE EXP                                                *
              =1  1363      ; *   SETB BIT EXP,CODE EXP                                                *
              =1  1364      ; *                                                                       *
              =1  1365      ; * In the first group, if the mnemonic was one of those six mnemonics the *
              =1  1366      ; * BIT_EXP FLAG was set and if an expression was found we know it was a bit *
              =1  1367      ; * expression. The second group was a little more difficult.  If the first *
              =1  1368      ; * operand of a two operand instruction was found to be a 'C' the BIT_EXP  *
              =1  1369      ; * flag was set and then if the second operand was an expression we knew it *
              =1  1370      ; * was a bit expression.  The third group was the real problem.  If the    *
              =1  1371      ; * second operand of a two operand instruction was a 'C' and the first    *
              =1  1372      ; * operand had been an expression then the hash value was re-calculated to *
              =1  1373      ; * indicate a bit expression.                                            *
              =1  1374      ; *                                                                       *
              =1  1375      ; ***********************************************************************/
              =1  1376 +1   $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1  1377     ;*****************************************************************
                =1  1378     ;
                =1  1379     ;    NAME: START_DIVIDE
                =1  1380     ;
                =1  1381     ;    ABSTRACT: This is a software divide routine.  Inputs are an 8-bit
                =1  1382     ;       divisor and a 16-bit dividend.  The quotient is 16-bits and
                =1  1383     ;       the remainder is truncated to 8 bits.
                =1  1384     ;
                =1  1385     ;    INPUTS: DIVIDEND_HIGH, DIVIDEND_LOW, DIVISOR
                =1  1386     ;
                =1  1387     ;    OUTPUTS: QUOTIENT_HIGH, QUOTIENT_LOW
                =1  1388     ;
                =1  1389     ;    VARIABLES MODIFIED: A, PARAM6, DIVIDEND_LOW, QUOTIENT_HIGH,
                =1  1390     ;       PARAM5, PARAM4, C, DIVIDEND_HIGH, QUOTIENT_LOW
                =1  1391     ;
                =1  1392     ;    ERROR EXITS: None
                =1  1393     ;
                =1  1394     ;    SUBROUTINES ACCESSED DIRECTLY: None
                =1  1395     ;
                =1  1396     ;*****************************************************************
                =1  1397     START_DIVIDE:
F9B4 E570       =1  1398             MOV     A,DIVISOR
F9B6 7F09       =1  1399             MOV     PARAM6,#09H
F9B8 7E00       =1  1400             MOV     PARAM5,#00H
F9BA 7D00       =1  1401             MOV     PARAM4,#00H
                =1  1402     DIVIDE_1:
F9BC C3         =1  1403             CLR C
                =1  1404     DIVIDE_2:
F9BD E56E       =1  1405             MOV     A,DIVIDEND_HIGH
F9BF 4011       =1  1406             JC      SUBTRACT_WITH_C         ;Carry occurs from rotate
F9C1 6021       =1  1407             JZ      ROTATE                  ;Rotate quotient and dividend if zero
F9C3 9570       =1  1408             SUBB    A,DIVISOR
F9C5 401D       =1  1409             JC      ROTATE                  ;A carry means divisor is larger than dividend
F9C7 F56E       =1  1410             MOV     DIVIDEND_HIGH,A         ;Replace DIVIDEND_HIGH with new number
F9C9 EE         =1  1411             MOV     A,PARAM5                ;PARAM5 holds lower byte of quotient
F9CA 2401       =1  1412             ADD     A,#01H                  ;Increment quotient
F9CC 5001       =1  1413             JNC     DIVIDE_3
F9CE 0D         =1  1414             INC     PARAM4                  ;High counter incremented if carry occurs
                =1  1415     DIVIDE_3:
F9CF FE         =1  1416             MOV     PARAM5,A                ;Replace with new quotient
F9D0 80EA       =1  1417             JMP     DIVIDE_1                ;Loop
                =1  1418     SUBTRACT_WITH_C:
F9D2 EE         =1  1419             MOV     A,PARAM5
F9D3 2401       =1  1420             ADD     A,#01H
F9D5 5001       =1  1421             JNC     DIVIDE_4
F9D7 0D         =1  1422             INC     PARAM4
                =1  1423     DIVIDE_4:
F9D8 FE         =1  1424             MOV     PARAM5,A                ;Quotient always incremented if carry set
F9D9 C3         =1  1425             CLR     C
F9DA E56E       =1  1426             MOV     A,DIVIDEND_HIGH
F9DC 9570       =1  1427             SUBB    A,DIVISOR
F9DE F56E       =1  1428             MOV     DIVIDEND_HIGH,A         ;Subtract divisor from dividend
F9E0 40DA       =1  1429             JC      DIVIDE_1                ;Jump to subtract with no carry if carry is set
F9E2 80EE       =1  1430             JMP     SUBTRACT_WITH_C         ;Loop in subtract with C if no carry
F9E4 DF05       =1  1431     ROTATE: DJNZ    PARAM6,ROTATE_CONTINUE  ;PARAM6 counts number of rotates
```

```
LOC  OBJ            LINE      SOURCE

F9E6 8D71       =1  1432              MOV     QUOTIENT_HIGH,PARAM4
F9E8 8E72       =1  1433              MOV     QUOTIENT_LOW,PARAM5
F9EA 22         =1  1434              RET                             ;Exit from divide routine
                =1  1435      ROTATE_CONTINUE:
F9EB C3         =1  1436              CLR     C
F9EC EE         =1  1437              MOV     A,PARAM5
F9ED 33         =1  1438              RLC     A                       ;Rotate ;pwer byte of quotient first
F9EE FE         =1  1439              MOV     PARAM5,A                ;Replace with new quotient low
F9EF ED         =1  1440              MOV     A,PARAM4
F9F0 33         =1  1441              RLC     A                       ;Rotate upper byte with MSB from lower
F9F1 FD         =1  1442              MOV     PARAM4,A                ;byte into LSB of upper byte
F9F2 C3         =1  1443              CLR     C
F9F3 E56F       =1  1444              MOV     A,DIVIDEND_LOW          ;Rotate dividend with every rotate of
                =1  1445                                              ;quotient
F9F5 33         =1  1446              RLC     A
F9F6 F56F       =1  1447              MOV     DIVIDEND_LOW,A
F9F8 E56E       =1  1448              MOV     A,DIVIDEND_HIGH
F9FA 33         =1  1449              RLC     A
F9FB F56E       =1  1450              MOV     DIVIDEND_HIGH,A
F9FD 80BE       =1  1451              SJMP    DIVIDE_2                ;Loop
F9FF 22         =1  1452              RET                             ;End of divide routines
                =1  1453 +1   $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1  1454     ;*****************************************************************
                =1  1455     ;
                =1  1456     ;     NAME: CALCULATE_INSTRUCTION_VALUE
                =1  1457     ;
                =1  1458     ;     ABSTRACT: Parse-time action to assemble the instruction just parsed
                =1  1459     ;        into the public variable INSTRUCTION_VALUE.  The values may be
                =1  1460     ;        calculated and filled in at run-time.  Using the hash value,
                =1  1461     ;        the #INSTRUCTION_CODE table is searched for a corresponding match.
                =1  1462     ;        If one is found, the ordinal of the match (INSTRUCTION_VALUE) is
                =1  1463     ;        the opcode of the instruction.  If one is not found, an error is issued
                =1  1464     ;        and processing stops.
                =1  1465     ;
                =1  1466     ;     INPUTS: OUR_CODE_LOW, OUR_CODE_HIGH
                =1  1467     ;
                =1  1468     ;     OUTPUTS: INSTRUCTION, OUR_CODE_LOW, OUR_CODE_HIGH
                =1  1469     ;
                =1  1470     ;     VARIABLES MODIFIED: DPTR, A, ERRNUM, C, INSTRUCTION
                =1  1471     ;
                =1  1472     ;     ERROR EXITS: 10H (ASSEMBLY SYNTAX)
                =1  1473     ;
                =1  1474     ;     SUBROUTINES ACCESSED DIRECTLY: ERROR
                =1  1475     ;
                =1  1476     ;*****************************************************************
                =1  1477     CALCULATE_INSTRUCTION_VALUE:
FA00 90F5B3     =1  1478             MOV     DPTR,#INSTRUCTION_CODE
FA03 755F00     =1  1479             MOV     INSTRUCTION,#00H
                =1  1480     INST_VALUE_LOOP:
FA06 E4         =1  1481             CLR     A
FA07 93         =1  1482             MOVC    A,@A+DPTR
FA08 055F       =1  1483             INC     INSTRUCTION
FA0A A3         =1  1484             INC     DPTR
FA0B B54E09     =1  1485             CJNE    A,OUR_CODE_HIGH,CHECK_AND_INC_HASH_TAB
FA0E E4         =1  1486             CLR     A
FA0F 93         =1  1487             MOVC    A,@A+DPTR
FA10 A3         =1  1488             INC     DPTR
FA11 B54F04     =1  1489             CJNE    A,OUR_CODE_LOW,CHECK_HASH_TAB   ;Second byte is high byte (CS)
FA14 155F       =1  1490             DEC     INSTRUCTION
FA16 22         =1  1491             RET
                =1  1492     CHECK_AND_INC_HASH_TAB:
FA17 A3         =1  1493             INC     DPTR
                =1  1494     CHECK_HASH_TAB:
FA18 E583       =1  1495             MOV     A,DPH
FA1A B4F7E9     =1  1496             CJNE    A,#HIGH(INSTRUCTION_CODE+200H),INST_VALUE_LOOP
FA1D E582       =1  1497             MOV     A,DPL
FA1F B4B3E4     =1  1498             CJNE    A,#LOW(INSTRUCTION_CODE+200H),INST_VALUE_LOOP
FA22 754310     =1  1499             MOV     ERRNUM,#10H                     ;Assembly syntax
FA25 02E05F     =1  1500             JMP     ERROR
                =1  1501 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

               =1   1502     ;*****************************************************************
               =1   1503     ;
               =1   1504     ;     NAME: UPDATE_OUR_CODE
               =1   1505     ;
               =1   1506     ;     ABSTRACT: Local procedure used to determine whether to use
               =1   1507     ;        #MNEMONIC_FACTOR (first operand) or #OPERAND_FACTOR*#MNEMONIC_FACTOR
               =1   1508     ;        (second operand) and then update the hash value, OUR_CODE.
               =1   1509     ;
               =1   1510     ;     INPUTS: OUR_CODE_LOW, OUR_CODE_HIGH, ORDINAL
               =1   1511     ;
               =1   1512     ;     OUTPUTS: OUR_CODE_LOW, OUR_CODE_HIGH
               =1   1513     ;
               =1   1514     ;     VARIABLES MODIFIED: A, B, OUR_CODE_HIGH, OUR_CODE_LOW, PARAM6
               =1   1515     ;
               =1   1516     ;     ERROR EXITS: None
               =1   1517     ;
               =1   1518     ;     SUBROUTINES ACCESSED DIRECTLY: None
               =1   1519     ;
               =1   1520     ;*****************************************************************
               =1   1521     UPDATE_OUR_CODE:
FA28 E54E      =1   1522             MOV     A,OUR_CODE_HIGH
FA2A 7017      =1   1523             JNZ     UL0
FA2C 742C      =1   1524             MOV     A,#MNEMONIC_FACTOR                ;Set-up for <=
FA2E 855CF0    =1   1525             MOV     B,ORDINAL                         ;Set-up for MUL AB
FA31 B54F0D    =1   1526             CJNE    A,OUR_CODE_LOW,UPDATE_LSSTHN      ;Fall through if"=",or check
               =1   1527     CONT_UPDATE_LSSTHN:                                 ;for "<"
FA34 A4        =1   1528             MUL     AB                                ;ORDINAL * #MNEMONIC_FACTOR
FA35 85F04E    =1   1529             MOV     OUR_CODE_HIGH,B                   ;Save high order byte
FA38 254F      =1   1530             ADD     A,OUR_CODE_LOW                    ;Add old OUR_CODE
FA3A 5002      =1   1531             JNC     UL1
FA3C 054E      =1   1532             INC     OUR_CODE_HIGH
FA3E F54F      =1   1533     UL1:    MOV     OUR_CODE_LOW,A                    ;Replace with new code
FA40 22        =1   1534             RET                                       ;Exit
               =1   1535     UPDATE_LSSTHN:
FA41 50F1      =1   1536             JNC     CONT_UPDATE_LSSTHN
FA43 742C      =1   1537     UL0:    MOV     A,#MNEMONIC_FACTOR
FA45 855CF0    =1   1538             MOV     B,ORDINAL
FA48 A4        =1   1539             MUL     AB                                ;ORDINAL * #MNEMONIC_FACTOR
FA49 AFF0      =1   1540             MOV     PARAM6,B
FA4B 75F018    =1   1541             MOV     B,#OPERAND_FACTOR
FA4E A4        =1   1542             MUL     AB
FA4F 254F      =1   1543             ADD     A,OUR_CODE_LOW                    ;Add old OUR_CODE
FA51 F54F      =1   1544             MOV     OUR_CODE_LOW,A                    ;Replace with new OUR_CODE
FA53 E5F0      =1   1545             MOV     A,B
FA55 354E      =1   1546             ADDC    A,OUR_CODE_HIGH                   ;Add high byte of OUR_CODE
FA57 F54E      =1   1547             MOV     OUR_CODE_HIGH,A
FA59 EF        =1   1548             MOV     A,PARAM6                          ;Recall upper byte of first mult.
FA5A 75F018    =1   1549             MOV     B,#OPERAND_FACTOR
FA5D A4        =1   1550             MUL     AB
FA5E 254E      =1   1551             ADD     A,OUR_CODE_HIGH                   ;Add upper byte if second multiply
               =1   1552                                                       ;to upper byte of first multiply
               =1   1553                                                       ;Multiplied by OPER_FACTOR
FA60 F54E      =1   1554             MOV     OUR_CODE_HIGH,A
FA62 22        =1   1555             RET                                       ;Exit
               =1   1556 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

               =1   1557     ;*****************************************************************
               =1   1558     ;
               =1   1559     ;    NAME: GET_FIRST_OPERAND
               =1   1560     ;
               =1   1561     ;    ABSTRACT: (ORDINAL + 1)*MNEMONIC_FACTOR is added to OUR_CODE
               =1   1562     ;       (the hash value). If the operand was a 'C', then BIT_EXP is
               =1   1563     ;       set to 1 (true).
               =1   1564     ;
               =1   1565     ;    INPUTS: TOKSTR, OUR_CODE_LOW, OUR_CODE_HIGH
               =1   1566     ;
               =1   1567     ;    OUTPUTS: BIT_EXP, OUR_CODE_LOW, OUR_CODE_HIGH
               =1   1568     ;
               =1   1569     ;    VARIABLES MODIFIED: B, A, C, OUR_CODE_LOW, OUR_CODE_HIGH, PARAM6,
               =1   1570     ;       BIT_EXP
               =1   1571     ;
               =1   1572     ;    ERROR EXITS: None
               =1   1573     ;
               =1   1574     ;    SUBROUTINES ACCESSED DIRECTLY: None
               =1   1575     ;
               =1   1576     ;*****************************************************************
               =1   1577     GET_FIRST_OPERAND:
FA63 75F02C    =1   1578             MOV     B,#MNEMONIC_FACTOR
FA66 E548      =1   1579             MOV     A,TOKSTR
FA68 C3        =1   1580             CLR     C
FA69 9490      =1   1581             SUBB    A,#90H
FA6B 401B      =1   1582             JC      FIRST_NOT_REGISTER
FA6D 9408      =1   1583             SUBB    A,#08H
FA6F 5017      =1   1584             JNC     FIRST_NOT_REGISTER      ;Check if TOKSTR=REGISTER token(0-7)
FA71 E548      =1   1585             MOV     A,TOKSTR
FA73 C3        =1   1586             CLR     C
FA74 948C      =1   1587             SUBB    A,#8CH
FA76 A4        =1   1588             MUL     AB
FA77 254F      =1   1589             ADD     A,OUR_CODE_LOW
 A79 F54F      =1   1590             MOV     OUR_CODE_LOW,A
 A7B 5002      =1   1591             JNC     GE_FI_OP_1
FA7D 054E      =1   1592             INC     OUR_CODE_HIGH
               =1   1593     GE_FI_OP_1:
FA7F E5F0      =1   1594             MOV     A,B
FA81 254E      =1   1595             ADD     A,OUR_CODE_HIGH
FA83 F54E      =1   1596             MOV     OUR_CODE_HIGH,A
FA85 02FAA0    =1   1597             JMP     SET_BIT_EXP
               =1   1598     FIRST_NOT_REGISTER:
FA88 7410      =1   1599             MOV     A,#OFST
FA8A 2440      =1   1600             ADD     A,#REG
FA8C FF        =1   1601             MOV     PARAM6,A
FA8D E548      =1   1602             MOV     A,TOKSTR
FA8F C3        =1   1603             CLR     C
FA90 9F        =1   1604             SUBB    A,PARAM6
FA91 A4        =1   1605             MUL     AB
FA92 254F      =1   1606             ADD     A,OUR_CODE_LOW
FA94 F54F      =1   1607             MOV     OUR_CODE_LOW,A
 A96 5002      =1   1608             JNC     GE_FI_OP_2
 A98 054E      =1   1609             INC     OUR_CODE_HIGH
               =1   1610     GE_FI_OP_2:
FA9A E5F0      =1   1611             MOV     A,B
```

```
LOC  OBJ            LINE     SOURCE

FA9C 254E       =1  1612            ADD     A,OUR_CODE_HIGH
FA9E F54E       =1  1613            MOV     OUR_CODE_HIGH,A
                =1  1614     SET_BIT_EXP:
FAA0 E548       =1  1615            MOV     A,TOKSTR
FAA2 B45E02     =1  1616            CJNE    A,#C_TOKE,END_FIRST_OPERAND
FAA5 D202       =1  1617            SETB    BIT_EXP
                =1  1618     END_FIRST_OPERAND:
FAA7 22         =1  1619            RET                         ;Exit
                =1  1620 +1  $EJECT
```

```
LOC  OBJ          LINE     SOURCE

             =1   1621     ;*****************************************************************
             =1   1622     ;
             =1   1623     ;    NAME: CHECK_AND_SET_EXP_FLAG, SET_EXP_16_FLAG, SET_EXP_FLAG,
             =1   1624     ;       CHECK_EXP_FLAG, SET_POUND_EXP_FLAG, CHECK_AND_SET_SECOND_EXP_FLAG,
             =1   1625     ;       SET_SLASH_EXP_FLAG, SET_REL_FLAG
             =1   1626     ;
             =1   1627     ;    ABSTRACT:
             =1   1628     ;       CHECK_AND_SET_EXP_FLAG: Parse-time action to check to see if
             =1   1629     ;       BIT_EXP is set(1). If so, the EXP8 is a bit EXP8 (eight-bit
             =1   1630     ;       expression), otherwise is ti a byte EXP8.  The ordinal is set
             =1   1631     ;       appropriately and UPDATE_OUR_CODE is called to update the
             =1   1632     ;       hash value, OUR_CODE.  The FIRST_EXP flag is set(1) to signify
             =1   1633     ;       that the first operand was an expression of some sort.
             =1   1634     ;       NUMBER_OF_BYTES is set to 2 to signify that it is a two byte
             =1   1635     ;       instruction so far.
             =1   1636     ;
             =1   1637     ;       SET_EXP_16_FLAG:  Parse-time action to set the ordinal to 20 to
             =1   1638     ;       show that the operand has an EXP16 ad then cal UPDATE_OUR_CODE to
             =1   1639     ;       update the hash value, OUR_CODE.  SET_NUMBER_OF_BYTES equal to
             =1   1640     ;       7 to signify that the instruction was a long jump or call or
             =1   1641     ;       MOV DPTR,EXP16.
             =1   1642     ;
             =1   1643     ;       SET_EXP_FLAG:  Parse-time prodecure to set the  ordinal equal to
             =1   1644     ;       16 to show that the operand was a byte EXP8 expression ad call
             =1   1645     ;       UPDATE_OUR_CODE to update the hash value, OUR_CODE.  Set the
             =1   1646     ;       FIRST_EXP flag to show that the first operand was an expression
             =1   1647     ;       of some sort.
             =1   1648     ;
             =1   1649     ;       CHECK_EXP_FLAG:  Parse-time action that checks the FIRST_EXP
             =1   1650     ;       flag and the SECOND_EXP flag.  by determining which are set
             =1   1651     ;       and which are not, NUMBER_OF_BYTES is set according to the
             =1   1652     ;       number of bytes in the instruction.
             =1   1653     ;               FIRST_EXP        SECOND_EXP       NUMBER_OF_BYTES
             =1   1654     ;               0                0                1
             =1   1655     ;               0                1                2
             =1   1656     ;               1                0                2
             =1   1657     ;               1                1                3
             =1   1658     ;
             =1   1659     ;       SET_POUND_EXP_FLAG:  Parse-time action to set the ordinal equal
             =1   1660     ;       to 18 to show that the operand was an immediate(#) expression.
             =1   1661     ;       update the hash value, OUR_CODE, by calling UPDATE_OUR_CODE.
             =1   1662     ;       SECOND_EXP flag is set to signify that the second operand was an
             =1   1663     ;       expression of some sort.
             =1   1664     ;
             =1   1665     ;       CHECK_AND_SET_SECOND_EXP_FLAG:  Parse-time action to set the
             =1   1666     ;       SECOND_EXP flag to signify that the second operand was an expression
             =1   1667     ;       of some sort.  The BIT_EXP flag is checked.  If set, the ordinal
             =1   1668     ;       is set equal to 17 to show that the operand was a bit EXP8.  If
             =1   1669     ;       it was not set, the ordinal is set to 16 to show that the operand
             =1   1670     ;       was a byte EXP8.  The hash value is updated by calling UPDATE_OUR_CODE.
             =1   1671     ;
             =1   1672     ;       SET_SLASH_EXP_FLAG:  Parse-time action to set the ordinal equal to 19
             =1   1673     ;       to show that the operand was the complement(/) of a bit expression.
             =1   1674     ;       update the hash value, OUR_CODE, by calling UPDATE_OUR_CODE.
             =1   1675     ;       SECOND_EXP is set to signify that the second operand was an expression
```

```
LOC  OBJ            LINE      SOURCE

               =1  1676      ;       of some sort.
               =1  1677      ;
               =1  1678      ;       SET_REL_FLAG:  Parse-time action to set the ordinal equal to 22 to
               =1  1679      ;       show that the operand was a relative offset(EXP8).  The hash value,
               =1  1680      ;       OUR_CODE, is updated by calling UPDATE_OUR_CODE.  Set NUMBER_OF_BYTES
               =1  1681      ;       equal to 4 to signify that it was a jump instruction with a relative
               =1  1682      ;       operand.
               =1  1683      ;
               =1  1684      ;    INPUTS: BIT_EXP, OUR_CODE_LOW, OUR_CODE_HIGH, FIRST_EXP, SECOND_EXP
               =1  1685      ;
               =1  1686      ;    OUTPUTS: NUMBER_OF_BYTES, ORDINAL, FIRST_EXP, SECOND_EXP, OUR_CODE_LOW,
               =1  1687      ;       OUR_CODE_HIGH
               =1  1688      ;
               =1  1689      ;    VARIABLES MODIFIED: ORDINAL, FIRST_EXP, NUMBER_OF_BYTES, SECOND_EXP,
               =1  1690      ;       A, C, B, DPTR
               =1  1691      ;
               =1  1692      ;    ERROR EXITS: None
               =1  1693      ;
               =1  1694      ;    SUBROUTINES ACCESSED DIRECTLY: UPDATE_OUR_CODE
               =1  1695      ;
               =1  1696      ;*****************************************************************
               =1  1697      CHECK_AND_SET_EXP_FLAG:
FAA8 755C10    =1  1698              MOV     ORDINAL,#10H             ;In case no bit 8
FAAB 300202    =1  1699              JNB     BIT_EXP,NO_BIT_8
FAAE 055C      =1  1700              INC     ORDINAL                  ;Bit 8 occurance
               =1  1701      NO_BIT_8:
FAB0 5128      =1  1702              CALL    UPDATE_OUR_CODE
FAB2 D203      =1  1703              SETB    FIRST_EXP
FAB4 754D02    =1  1704              MOV     NUMBER_OF_BYTES,#02H     ;Two bytes so far
FAB7 22        =1  1705              RET                              ;Exit
               =1  1706      ;******************************************************************
               =1  1707      SET_EXP_16_FLAG:
FAB8 755C14    =1  1708              MOV     ORDINAL,#14H
FABB 5128      =1  1709              CALL    UPDATE_OUR_CODE
FABD 754D07    =1  1710              MOV     NUMBER_OF_BYTES,#07H     ;To signify an EXP16 instruction
FAC0 22        =1  1711              RET                              ;Exit
               =1  1712      ;******************************************************************
               =1  1713      SET_EXP_FLAG:
FAC1 755C10    =1  1714              MOV     ORDINAL,#10H
FAC4 5128      =1  1715              CALL    UPDATE_OUR_CODE
FAC6 D203      =1  1716              SETB    FIRST_EXP                ;First operand of an exrpession
FAC8 22        =1  1717              RET
               =1  1718      ;******************************************************************
               =1  1719      CHECK_EXP_FLAG:
FAC9 E4        =1  1720              CLR     A
FACA A203      =1  1721              MOV     C,FIRST_EXP
FACC 33        =1  1722              RLC     A
FACD A204      =1  1723              MOV     C,SECOND_EXP
FACF 33        =1  1724              RLC     A
FAD0 75F004    =1  1725              MOV     B,#04H
FAD3 90FAD8    =1  1726              MOV     DPTR,#EXP_FLAG_TABLE
FAD6 A4        =1  1727              MUL     AB
FAD7 73        =1  1728              JMP     @A+DPTR
               =1  1729      EXP_FLAG_TABLE:
FAD8 754D01    =1  1730              MOV     NUMBER_OF_BYTES,#01H
```

```
LOC  OBJ            LINE      SOURCE

FADB 22         =1  1731              RET
FADC 754D02     =1  1732              MOV     NUMBER_OF_BYTES,#02H
FADF 22         =1  1733              RET
FAE0 754D02     =1  1734              MOV     NUMBER_OF_BYTES,#02H
FAE3 22         =1  1735              RET
FAE4 754D03     =1  1736              MOV     NUMBER_OF_BYTES,#03H
FAE7 22         =1  1737              RET                              ;Exit
                =1  1738      ;*****************************************************************
                =1  1739      SET_POUND_EXP_FLAG:
FAE8 755C12     =1  1740              MOV     ORDINAL,#12H
FAEB 5128       =1  1741              CALL    UPDATE_OUR_CODE
FAED D204       =1  1742              SETB    SECOND_EXP
FAEF 22         =1  1743              RET                              ;Exit
                =1  1744      ;*****************************************************************
                =1  1745      CHECK_AND_SET_SECOND_EXP_FLAG:
FAF0 D204       =1  1746              SETB    SECOND_EXP
FAF2 7410       =1  1747              MOV     A,#10H
FAF4 300201     =1  1748              JNB     BIT_EXP,SECOND_NO_BIT_8
FAF7 04         =1  1749              INC     A
                =1  1750      SECOND_NO_BIT_8:
FAF8 F55C       =1  1751              MOV     ORDINAL,A
FAFA 4128       =1  1752              JMP     UPDATE_OUR_CODE
                =1  1753      ;*****************************************************************
                =1  1754      SET_SLASH_EXP_FLAG:
FAFC 755C13     =1  1755              MOV     ORDINAL,#13H             ;Complement of a bit expression
FAFF 5128       =1  1756              CALL    UPDATE_OUR_CODE
FB01 D204       =1  1757              SETB    SECOND_EXP
FB03 22         =1  1758              RET                              ;Exit
                =1  1759      ;*****************************************************************
                =1  1760      SET_REL_FLAG:
FB04 755C16     =1  1761              MOV     ORDINAL,#16H             ;Relative offset
FB07 5128       =1  1762              CALL    UPDATE_OUR_CODE
FB09 754D04     =1  1763              MOV     NUMBER_OF_BYTES,#04H     ;Jump instruction with relative operand
FB0C 22         =1  1764              RET                              ;Exit
                =1  1765 +1   $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1  1766     ;*****************************************************************
                =1  1767     ;
                =1  1768     ;    NAME: GET_SECOND_EXP
                =1  1769     ;
                =1  1770     ;    ABSTRACT: (#MNEMONIC_FACTOR* #OPERAND_FACTOR) is added to the
                =1  1771     ;       hash value, OUR_CODE.  If the operand was a 'C', then OUR_CODE
                =1  1772     ;       must be re-calculated to allow for a bit EXP8 instead of a byte
                =1  1773     ;       EXP8.
                =1  1774     ;
                =1  1775     ;    INPUTS: OUR_CODE_LOW, OUR_CODE_HIGH, TOKSTR
                =1  1776     ;
                =1  1777     ;    OUTPUTS: OUR_CODE_LOW, OUR_CODE_HIGH
                =1  1778     ;
                =1  1779     ;    VARIABLES MODIFIED: B, A, C, PARAM6, OUR_CODE_LOW, OUR_CODE_HIGH
                =1  1780     ;
                =1  1781     ;    ERROR EXITS: None
                =1  1782     ;
                =1  1783     ;    SUBROUTINES ACCESSED DIRECTLY: None
                =1  1784     ;
                =1  1785     ;*****************************************************************
                =1  1786     GET_SECOND_OPERAND:
FB0D 75F02C     =1  1787             MOV     B,#MNEMONIC_FACTOR
FB10 E548       =1  1788             MOV     A,TOKSTR
FB12 C3         =1  1789             CLR     C
FB13 9490       =1  1790             SUBB    A,#90H
FB15 4025       =1  1791             JC      SECOND_NOT_REGISTER
FB17 9408       =1  1792             SUBB    A,#08H
FB19 5021       =1  1793             JNC     SECOND_NOT_REGISTER     ;Check if TOKSTR=REGISTER token(0-7)
FB1B E548       =1  1794             MOV     A,TOKSTR
FB1D C3         =1  1795             CLR     C
FB1E 948C       =1  1796             SUBB    A,#8CH
FB20 A4         =1  1797             MUL     AB
FB21 AFF0       =1  1798             MOV     PARAM6,B
FB23 75F018     =1  1799             MOV     B,#OPERAND_FACTOR
FB26 A4         =1  1800             MUL     AB
FB27 254F       =1  1801             ADD     A,OUR_CODE_LOW
FB29 F54F       =1  1802             MOV     OUR_CODE_LOW,A
FB2B E5F0       =1  1803             MOV     A,B
FB2D 354E       =1  1804             ADDC    A,OUR_CODE_HIGH
FB2F F54E       =1  1805             MOV     OUR_CODE_HIGH,A
FB31 EF         =1  1806             MOV     A,PARAM6
FB32 75F018     =1  1807             MOV     B,#OPERAND_FACTOR
FB35 A4         =1  1808             MUL     AB
FB36 254E       =1  1809             ADD     A,OUR_CODE_HIGH
FB38 F54E       =1  1810             MOV     OUR_CODE_HIGH,A
FB3A 8023       =1  1811             SJMP    OPERAND_C
                =1  1812     SECOND_NOT_REGISTER:
FB3C 7410       =1  1813             MOV     A,#OFST
FB3E 2440       =1  1814             ADD     A,#REG
FB40 FF         =1  1815             MOV     PARAM6,A
FB41 E548       =1  1816             MOV     A,TOKSTR
FB43 C3         =1  1817             CLR     C
FB44 9F         =1  1818             SUBB    A,PARAM6
FB45 A4         =1  1819             MUL     AB
FB46 AFF0       =1  1820             MOV     PARAM6,B
```

```
LOC  OBJ            LINE     SOURCE

FB48 75F018       =1  1821              MOV     B,#OPERAND_FACTOR
FB4B A4           =1  1822              MUL     AB
FB4C 254F         =1  1823              ADD     A,OUR_CODE_LOW
FB4E F54F         =1  1824              MOV     OUR_CODE_LOW,A
FB50 E5F0         =1  1825              MOV     A,B
FB52 354E         =1  1826              ADDC    A,OUR_CODE_HIGH
FB54 F54E         =1  1827              MOV     OUR_CODE_HIGH,A
FB56 EF           =1  1828              MOV     A,PARAM6
FB57 75F018       =1  1829              MOV     B,#OPERAND_FACTOR
FB5A A4           =1  1830              MUL     AB
FB5B 254E         =1  1831              ADD     A,OUR_CODE_HIGH
FB5D F54E         =1  1832              MOV     OUR_CODE_HIGH,A
                  =1  1833     OPERAND_C:
FB5F E54E         =1  1834              MOV     A,OUR_CODE_HIGH
FB61 B43C08       =1  1835              CJNE    A,#03CH,END_SECOND_OPERAND
FB64 E54F         =1  1836              MOV     A,OUR_CODE_LOW
FB66 B48F03       =1  1837              CJNE    A,#08FH,END_SECOND_OPERAND
FB69 754FBB       =1  1838              MOV     OUR_CODE_LOW,#0BBH
                  =1  1839     END_SECOND_OPERAND:
FB6C 22           =1  1840              RET                         ;EXIT
                      1841 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                    1842 +1  $INCLUDE(:F1:ASMA.INC)
               =1   1843     ;*****************************************************************
               =1   1844     ;
               =1   1845     ;       This is the include file called ASMA.INC.  It contains the
               =1   1846     ;       following subroutines in order:
               =1   1847     ;
               =1   1848     ;               CHECK_AND_CHANGE_ASM_PC
               =1   1849     ;               CHANGE_TO_INSTRUCTION_OP
               =1   1850     ;
               =1   1851     ;*****************************************************************
               =1   1852 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE
               =1   1853     ;***************************************************************************
               =1   1854     ;
               =1   1855     ;    NAME: CHECK_AND_CHANGE_ASM_PC
               =1   1856     ;
               =1   1857     ;    ABSTRACT: Change the ASM_PC according to NUMBER_OF_BYTES
               =1   1858     ;       and check to make sure it does not wrap around.
               =1   1859     ;
               =1   1860     ;    INPUTS: NUMBER_OF_BYTES, ASM_PC_LOW, ASM_PC_HIGH
               =1   1861     ;
               =1   1862     ;    OUTPUTS: ASM_PC_LOW, ASM_PC_HIGH
               =1   1863     ;
               =1   1864     ;    VARIABLES MODIFIED: A, PARAM1, ASM_PC_HIGH, ASM_PC_LOW, ERRNUM
               =1   1865     ;
               =1   1866     ;    ERROR EXITS: 13H (ASM PC>OFFFH)
               =1   1867     ;
               =1   1868     ;    SUBROUTINES ACCESSED DIRECTLY: ERROR
               =1   1869     ;
               =1   1870     ;***************************************************************************
               =1   1871
               =1   1872     CHECK_AND_CHANGE_ASM_PC:
FB6D E54D      =1   1873             MOV     A,NUMBER_OF_BYTES
FB6F 254C      =1   1874             ADD     A,ASM_PC_LOW
FB71 FA        =1   1875             MOV     PARAM1,A                ;Save to put in ASM_PC_LOW
FB72 E4        =1   1876             CLR     A
FB73 354B      =1   1877             ADDC    A,ASM_PC_HIGH           ;Add 1 to ASM_PC_HIGH if carry set
FB75 5006      =1   1878             JNC     CHANGE_ASM_PC_1         ;Error if carry set after add
FB77 754313    =1   1879             MOV     ERRNUM,#13H             ;ASM PC > OFFFH
FB7A 02E05F    =1   1880             JMP     ERROR
               =1   1881     CHANGE_ASM_PC_1:
FB7D F54B      =1   1882             MOV     ASM_PC_HIGH,A
FB7F 8A4C      =1   1883             MOV     ASM_PC_LOW,PARAM1       ;Replace ASM_PC with new value
FB81 22        =1   1884             RET
               =1   1885 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                             ;*****************************************************************************
                             ;
                             ;       NAME: CHANGE_TO_INSTRUCTION_OP
                             ;
                             ;       ABSTRACT: Run time action used to process the one, two or three bytes of
                             ;          the assembled instruction and write it out to memory.  The assembly
                             ;          program counter (ASM_PC) is updated according to the number of bytes
                             ;          in the instruction.  A case statement will take care of all the
                             ;          different types of instructions.  The byte(s) of the instruction are
                             ;          stored in the appropriate order in a working area, WORKING_SPACE (3).
                             ;          The opcode is always put in the fist byte.  If the instruction is
                             ;          other than a one byte instruction, the other bytes are obtained from
                             ;          VALLOW, VALHGH or TEMP_SEC as necessary.  NUMBER_OF_BYTES is updated
                             ;          to reflect the number of bytes in the instruction to be written out
                             ;          to memory and the ASM_PC is updated.  The individual cases are as
                             ;          follows:
                             ;
                             ;          Case 1:  One byte instructions (ex. NOP)
                             ;
                             ;          Case 2:  Two byte instructions (ex. MOV R7,#DATA)
                             ;                   Put expression in second byte.
                             ;
                             ;          Case 3:  Three byte instructions (ex. MOV EXP8,#EXP)
                             ;                   Put the first expression in the second byte.
                             ;                   put the second expression in the third byte.
                             ;
                             ;          Case 4:  Jump instruction with one relative operand (ex. JC REL. OPER.)
                             ;                   Calculate the relative offset and put it in the second byte.
                             ;
                             ;          Case 5:  Jump instruction with an expression as the first operand
                             ;                   and a relative operand as the second operand
                             ;                   (ex. JNB EXP8,REL. OPER.)
                             ;                   Put the expression in the second byte, calculate the relative
                             ;                   offset and put it in the third byte.
                             ;
                             ;          Case 6:  Absolute call or jump instruction (ex. ACALL EXP11).
                             ;                   Calculate the 2K jump or call and incorporate it into the
                             ;                   opcode.  Put the  lower 8 bits of EXP11 in the second byte.
                             ;
                             ;          Case 7:  Long jump or call instruction or MOV DPTR,EXP16
                             ;                   (ex. LJMP EXP16).
                             ;                   The high byte of EXP16 is put in the second byte.  The low
                             ;                   byte of EXP16 is put in the third byte.
                             ;
                             ;        INPUTS: VALHGH, VALLOW, TEMP_SEC, INSTRUCTION_VALUE
                             ;
                             ;        OUTPUTS: Memory at address of ASM_PC
                             ;
                             ;        VARIABLES MODIFIED: NUMBER_OF_BYTES, REL_OFFSET_LOW, REL_OFFSET_HIGH,
                             ;           A, ERRNUM, OLD_ASM_PC_HIGH, OLD_ASM_PC_LOW, POINTO, TEMP_SEC, C,
                             ;           TEMP_LOW, SELECT, PNTLOW, PNTHGH, ASM_PC_HIGH, ASM_PC_LOW
                             ;
                             ;        ERROR EXITS: 10H (ASSEMBLY SYNTAX)
                             ;                     11H (ADDRESS OUT OF RANGE-11 BIT ABSOLUTE OFFSET)
                             ;                     12H (ADDRESS OUT OF RANGE-8 BIT RELATIVE OFFSET)
```

```
LOC  OBJ           LINE      SOURCE

                =1  1941     ;
                =1  1942     ;     SUBROUTINES ACCESSED DIRECTLY: CHECK_AND_CHANGE_ASM_PC, ERROR
                =1  1943     ;
                =1  1944     ;*************************************************************************
                =1  1945     CHANGE_TO_INSTRUCTION_OP:
FB82 854B5D     =1  1946             MOV     OLD_ASM_PC_HIGH,ASM_PC_HIGH
FB85 854C5E     =1  1947             MOV     OLD_ASM_PC_LOW,ASM_PC_LOW
FB88 E54D       =1  1948             MOV     A,NUMBER_OF_BYTES
FB8A B40109     =1  1949             CJNE    A,#01H,CHANGE_CASE_2            ;Change case 1
FB8D 716D       =1  1950             CALL    CHECK_AND_CHANGE_ASM_PC         ;Update ASM PC
FB8F 7840       =1  1951             MOV     POINT0,#WORKING_SPACE           ;Get opcode
FB91 A65F       =1  1952             MOV     @POINT0,INSTRUCTION
FB93 02FCD4     =1  1953             JMP     CHANGE_END
                =1  1954     CHANGE_CASE_2:
FB96 B4020C     =1  1955             CJNE    A,#02H,CHANGE_CASE_3
FB99 716D       =1  1956             CALL    CHECK_AND_CHANGE_ASM_PC
FB9B 7840       =1  1957             MOV     POINT0,#WORKING_SPACE
FB9D A65F       =1  1958             MOV     @POINT0,INSTRUCTION             ;Put opcode in 1st byte
FB9F 08         =1  1959             INC     POINT0
FBA0 A64A       =1  1960             MOV     @POINT0,VALLOW
FBA2 02FCD4     =1  1961             JMP     CHANGE_END
                =1  1962     CHANGE_CASE_3:
FBA5 B4031B     =1  1963             CJNE    A,#03H,CHANGE_CASE_4
FBA8 716D       =1  1964             CALL    CHECK_AND_CHANGE_ASM_PC
FBAA 7840       =1  1965             MOV     POINT0,#WORKING_SPACE
FBAC A65F       =1  1966             MOV     @POINT0,INSTRUCTION             ;Put opcode in 1st byte
FBAE E55F       =1  1967             MOV     A,INSTRUCTION
FBB0 B48506     =1  1968             CJNE    A,#85H,CASE_3_MORE
FBB3 E562       =1  1969             MOV     A,TEMP_SEC
FBB5 C54A       =1  1970             XCH     A,VALLOW
FBB7 F562       =1  1971             MOV     TEMP_SEC,A
                =1  1972     CASE_3_MORE:
FBB9 7841       =1  1973             MOV     POINT0,#(WORKING_SPACE+1)
FBBB A662       =1  1974             MOV     @POINT0,TEMP_SEC
FBBD 08         =1  1975             INC     POINT0
FBBE A64A       =1  1976             MOV     @POINT0,VALLOW
FBC0 02FCD4     =1  1977             JMP     CHANGE_END
                =1  1978     CHANGE_CASE_4:
FBC3 B40460     =1  1979             CJNE    A,#04H,CHANGE_CASE_5
FBC6 754312     =1  1980             MOV     ERRNUM,#12H                     ;Adr out of range-8 bit
FBC9 754D02     =1  1981             MOV     NUMBER_OF_BYTES,#02H            ;2 byte instruction
FBCC 716D       =1  1982             CALL    CHECK_AND_CHANGE_ASM_PC
FBCE 854A61     =1  1983             MOV     REL_OFFSET_LOW,VALLOW
FBD1 854960     =1  1984             MOV     REL_OFFSET_HIGH,VALHGH          ;Move value into relative offset
FBD4 E560       =1  1985             MOV     A,REL_OFFSET_HIGH
FBD6 B54B03     =1  1986             CJNE    A,ASM_PC_HIGH,CHANGE_CASE_4A
FBD9 02FBE1     =1  1987             JMP     CHANGE_CASE_4AA
                =1  1988     CHANGE_CASE_4A:
FBDC 4024       =1  1989             JC      BACKWARD_JUMP_CASE_4
FBDE 02FBE8     =1  1990             JMP     FORWARD_JUMP_CASE_4
                =1  1991     CHANGE_CASE_4AA:
FBE1 E561       =1  1992             MOV     A,REL_OFFSET_LOW
FBE3 B54C00     =1  1993             CJNE    A,ASM_PC_LOW,CHANGE_CASE_4C
                =1  1994     CHANGE_CASE_4C:
FBE6 401A       =1  1995             JC      BACKWARD_JUMP_CASE_4            ;Jump if rel. offset if < ASM_PC
```

```
LOC  OBJ          LINE      SOURCE

             =1   1996      FORWARD_JUMP_CASE_4:
FBE8 C3      =1   1997              CLR     C
FBE9 E561    =1   1998              MOV     A,REL_OFFSET_LOW
FBEB 954C    =1   1999              SUBB    A,ASM_PC_LOW
FBED F561    =1   2000              MOV     REL_OFFSET_LOW,A
FBEF E560    =1   2001              MOV     A,REL_OFFSET_HIGH
FBF1 954B    =1   2002              SUBB    A,ASM_PC_HIGH                ;Subtract ASM_PC from relative offset
FBF3 7067    =1   2003              JNZ     CHANGE_ERROR                 ;Error if relative offset > 0FFH
FBF5 747F    =1   2004              MOV     A,#7FH
FBF7 B56100  =1   2005              CJNE    A,REL_OFFSET_LOW,CHANGE_CASE_4D
             =1   2006      CHANGE_CASE_4D:
FBFA 4060    =1   2007              JC      CHANGE_ERROR                 ;Error if relative offset > 7FH
FBFC 7841    =1   2008              MOV     POINT0,#(WORKING_SPACE+1)
FBFE A661    =1   2009              MOV     @POINT0,REL_OFFSET_LOW       ;Move offset into WORKING_SPACE (1)
FC00 801D    =1   2010              SJMP    CHANGE_CASE_4_END
             =1   2011
             =1   2012      BACKWARD_JUMP_CASE_4:
FC02 C3      =1   2013              CLR     C
FC03 E54C    =1   2014              MOV     A,ASM_PC_LOW
FC05 9561    =1   2015              SUBB    A,REL_OFFSET_LOW
FC07 F561    =1   2016              MOV     REL_OFFSET_LOW,A
FC09 E54B    =1   2017              MOV     A,ASM_PC_HIGH
FC0B 9560    =1   2018              SUBB    A,REL_OFFSET_HIGH            ;Subtract rel. offset from ASM_PC
FC0D F560    =1   2019              MOV     REL_OFFSET_HIGH,A
FC0F 704B    =1   2020              JNZ     CHANGE_ERROR                 ;Error if relative offset > 0FFH
FC11 7480    =1   2021              MOV     A,#80H
FC13 B56100  =1   2022              CJNE    A,REL_OFFSET_LOW,CHANGE_CASE_4F
             =1   2023      CHANGE_CASE_4F:
FC16 4044    =1   2024              JC      CHANGE_ERROR                 ;Error if relative offset is > 80H
FC18 7841    =1   2025              MOV     POINT0,#(WORKING_SPACE+1)
FC1A E561    =1   2026              MOV     A,REL_OFFSET_LOW
FC1C F4      =1   2027              CPL     A
FC1D 04      =1   2028              INC     A
FC1E F6      =1   2029              MOV     @POINT0,A                    ;Move REL_OFFSET_LOW into WORKING_SPACE
             =1   2030
             =1   2031      CHANGE_CASE_4_END:
FC1F 7840    =1   2032              MOV     POINT0,#WORKING_SPACE
FC21 A65F    =1   2033              MOV     @POINT0,INSTRUCTION          ;Move 8-bit inst into WORKING_SPACE (1)
FC23 02FCD4  =1   2034              JMP     CHANGE_END
             =1   2035      CHANGE_CASE_5:
FC26 B4056D  =1   2036              CJNE    A,#05H,CHANGE_CASE_6
FC29 754312  =1   2037              MOV     ERRNUM,#12H                  ;Adr out of range - 8 bit
FC2C 754D03  =1   2038              MOV     NUMBER_OF_BYTES,#03H         ;3 byte instruction
FC2F 716D    =1   2039              CALL    CHECK_AND_CHANGE_ASM_PC      ;Update ASM PC
FC31 7840    =1   2040              MOV     POINT0,#WORKING_SPACE
FC33 A65F    =1   2041              MOV     @POINT0,INSTRUCTION          ;Move instruction into WORKING_SPACE (0)
FC35 854A61  =1   2042              MOV     REL_OFFSET_LOW,VALLOW
FC38 854960  =1   2043              MOV     REL_OFFSET_HIGH,VALHGH       ;Move value into relative offset
FC3B E560    =1   2044              MOV     A,REL_OFFSET_HIGH
FC3D B54B03  =1   2045              CJNE    A,ASM_PC_HIGH,CHANGE_CASE_5A
FC40 02FC48  =1   2046              JMP     CHANGE_CASE_5AA
             =1   2047      CHANGE_CASE_5A:
FC43 402D    =1   2048              JC      BACKWARD_JUMP_CASE_5
FC45 02FC4F  =1   2049              JMP     FORWARD_JUMP_CASE_5
             =1   2050      CHANGE_CASE_5AA:
```

```
LOC  OBJ           LINE     SOURCE

FC48 E561       =1  2051             MOV      A,REL_OFFSET_LOW
FC4A B54C00     =1  2052             CJNE     A,ASM_PC_LOW,CHANGE_CASE_5C
                =1  2053    CHANGE_CASE_5C:
FC4D 4023       =1  2054             JC       BACKWARD_JUMP_CASE_5
                =1  2055    FORWARD_JUMP_CASE_5:
FC4F C3         =1  2056             CLR      C
FC50 E561       =1  2057             MOV      A,REL_OFFSET_LOW
FC52 954C       =1  2058             SUBB     A,ASM_PC_LOW
FC54 F561       =1  2059             MOV      REL_OFFSET_LOW,A
FC56 E560       =1  2060             MOV      A,REL_OFFSET_HIGH
FC58 954B       =1  2061             SUBB     A,ASM_PC_HIGH                ;Subtract ASM_PC from dest. addr
                =1  2062                                                   ;and place in relative offset
FC5A 6009       =1  2063             JZ       FJC_5_CONTINUE               ;Error if relative offset < OFFH
                =1  2064    CHANGE_ERROR:
FC5C 855D4B     =1  2065             MOV      ASM_PC_HIGH,OLD_ASM_PC_HIGH
FC5F 855E4C     =1  2066             MOV      ASM_PC_LOW,OLD_ASM_PC_LOW
FC62 02E05F     =1  2067             JMP      ERROR
                =1  2068    FJC_5_CONTINUE:
FC65 747F       =1  2069             MOV      A,#7FH
FC67 B56100     =1  2070             CJNE     A,REL_OFFSET_LOW,CHANGE_CASE_5D
                =1  2071    CHANGE_CASE_5D:
FC6A 40F0       =1  2072             JC       CHANGE_ERROR                 ;Error if relative offset < 07FH
FC6C 7842       =1  2073             MOV      POINTO,#(WORKING_SPACE+2)
FC6E A661       =1  2074             MOV      @POINTO,REL_OFFSET_LOW       ;Move REL_OFFSET_LOW into WORKING_SPACE
FC70 801D       =1  2075             SJMP     CHANGE_CASE_5_END
                =1  2076
                =1  2077    BACKWARD_JUMP_CASE_5:
FC72 C3         =1  2078             CLR      C
FC73 E54C       =1  2079             MOV      A,ASM_PC_LOW
FC75 9561       =1  2080             SUBB     A,REL_OFFSET_LOW
FC77 F561       =1  2081             MOV      REL_OFFSET_LOW,A
FC79 E54B       =1  2082             MOV      A,ASM_PC_HIGH
FC7B 9560       =1  2083             SUBB     A,REL_OFFSET_HIGH            ;Subtract relative offset from ASM_PC
FC7D F560       =1  2084             MOV      REL_OFFSET_HIGH,A            ;and store in relative offset
FC7F 70DB       =1  2085             JNZ      CHANGE_ERROR                 ;Error if relative offset > OFFH
FC81 7480       =1  2086             MOV      A,#80H
FC83 B56100     =1  2087             CJNE     A,REL_OFFSET_LOW,CHANGE_CASE_5F
                =1  2088    CHANGE_CASE_5F:
FC86 40D4       =1  2089             JC       CHANGE_ERROR                 ;Error if relative offset > 080H
FC88 7842       =1  2090             MOV      POINTO,#(WORKING_SPACE+2)
FC8A E561       =1  2091             MOV      A,REL_OFFSET_LOW
FC8C F4         =1  2092             CPL      A
FC8D 04         =1  2093             INC      A
FC8E F6         =1  2094             MOV      @POINTO,A                    ;Move REL_OFFSET_LOW into WORKING_SPACE
                =1  2095    CHANGE_CASE_5_END:
FC8F 7841       =1  2096             MOV      POINTO,#(WORKING_SPACE+1)
FC91 A662       =1  2097             MOV      @POINTO,TEMP_SEC             ;Move TEMP_LOW into WORKING_SPACE (1)
FC93 02FCD4     =1  2098             JMP      CHANGE_END
                =1  2099    CHANGE_CASE_6:
FC96 B40626     =1  2100             CJNE     A,#06H,CHANGE_CASE_7
FC99 754D02     =1  2101             MOV      NUMBER_OF_BYTES,#02H         ;2 byte instruction
FC9C 716D       =1  2102             CALL     CHECK_AND_CHANGE_ASM_PC
FC9E E549       =1  2103             MOV      A,VALHGH                     ;Move value into TEMP
FCA0 54F8       =1  2104             ANL      A,#0F8H                      ;Use 3 top bits of 11 to determine
FCA2 F547       =1  2105             MOV      TEMP_LOW,A                   ;which 2k page JMP or CALL it is
```

```
LOC  OBJ            LINE      SOURCE

FCA4 74F8       =1  2106             MOV     A,#0F8H
FCA6 554B       =1  2107             ANL     A,ASM_PC_HIGH
FCA8 754311     =1  2108             MOV     ERRNUM,#11H                   ;Adr out of range (11 bit)
FCAB B547AE     =1  2109             CJNE    A,TEMP_LOW,CHANGE_ERROR
FCAE 7840       =1  2110             MOV     POINT0,#WORKING_SPACE
FCB0 E549       =1  2111             MOV     A,VALHGH                      ;TEMP HIGH <= 07
FCB2 5407       =1  2112             ANL     A,#07H
FCB4 C4         =1  2113             SWAP    A
FCB5 23         =1  2114             RL      A                             ;TEMP_HIGH now rotated right 3X
FCB6 255F       =1  2115             ADD     A,INSTRUCTION
FCB8 F6         =1  2116             MOV     @POINT0,A                     ;Put result in WORKING_SPACE (0)
FCB9 08         =1  2117             INC     POINT0
FCBA A64A       =1  2118             MOV     @POINT0,VALLOW                ;TEMP_LOW stored in WORKING_SPACE (1)
FCBC 02FCD4     =1  2119             JMP     CHANGE_END                    ;truncates to 8 bits
                =1  2120      CHANGE_CASE_7:
FCBF 754310     =1  2121             MOV     ERRNUM,#10H                   ;Assembly syntax
FCC2 B40797     =1  2122             CJNE    A,#07H,CHANGE_ERROR           ;Error if orig NUMBER_OF_BYTES > 7
FCC5 754D03     =1  2123             MOV     NUMBER_OF_BYTES,#03H          ;3 byte instruction
FCC8 716D       =1  2124             CALL    CHECK_AND_CHANGE_ASM_PC
FCCA 7840       =1  2125             MOV     POINT0,#WORKING_SPACE
FCCC A65F       =1  2126             MOV     @POINT0,INSTRUCTION           ;Store instruction in WORKING_SPACE (0)
FCCE 08         =1  2127             INC     POINT0
FCCF A649       =1  2128             MOV     @POINT0,VALHGH                ;Store VALHGH in WORKING_SPACE (1)
FCD1 08         =1  2129             INC     POINT0
FCD2 A64A       =1  2130             MOV     @POINT0,VALLOW                ;Store VALLOW in WORKING_SPACE (2)
                =1  2131      CHANGE_END:
FCD4 754600     =1  2132             MOV     SELECT,#00H                   ;Select external ROM
FCD7 855E45     =1  2133             MOV     PNTLOW,OLD_ASM_PC_LOW
FCDA 855D44     =1  2134             MOV     PNTHGH,OLD_ASM_PC_HIGH        ;Load pointer for store
FCDD 855E4C     =1  2135             MOV     ASM_PC_LOW,OLD_ASM_PC_LOW
FCE0 855D4B     =1  2136             MOV     ASM_PC_HIGH,OLD_ASM_PC_HIGH
FCE3 7840       =1  2137             MOV     POINT0,#WORKING_SPACE
                =1  2138      CHANGE_END_LOOP:
FCE5 E6         =1  2139             MOV     A,@POINT0                     ;Parameter to be stored
FCE6 FA         =1  2140             MOV     PARAM1,A
FCE7 12E04D     =1  2141             CALL    STORE
FCEA 08         =1  2142             INC     POINT0
FCEB 0545       =1  2143             INC     PNTLOW
FCED E545       =1  2144             MOV     A,PNTLOW
FCEF 7002       =1  2145             JNZ     CHANGE_END_A
FCF1 0544       =1  2146             INC     PNTHGH
                =1  2147      CHANGE_END_A:
FCF3 D54DEF     =1  2148             DJNZ    NUMBER_OF_BYTES,CHANGE_END_LOOP
                =1  2149                                                   ;Store until NUMBER_OF_BYTES=0
FCF6 85454C     =1  2150             MOV     ASM_PC_LOW,PNTLOW
FCF9 85444B     =1  2151             MOV     ASM_PC_HIGH,PNTHGH
FCFC 22         =1  2152             RET                                   ;End of change routine
                    2153 +1   $EJECT
```

```
LOC  OBJ            LINE      SOURCE

                    2154 +1   $INCLUDE(:F1:SDKDSM.INC)
               =1   2155 +1   $EJECT
```

```
LOC  OBJ            LINE     SOURCE
                =1  2156     ;*****************************************************************
                =1  2157     ;
                =1  2158     ;     NAME: DISASSEMBLY_CMD
                =1  2159     ;
                =1  2160     ;     ABSTRACT: This routine gets a token and partition and displays
                =1  2161     ;        <address>=.  It then gets a byte of memory from code memory,
                =1  2162     ;        searches the hash table for a match to that byte and dissassembles
                =1  2163     ;        it if one is found.
                =1  2164     ;
                =1  2165     ;     INPUTS: None
                =1  2166     ;
                =1  2167     ;     OUTPUTS: None
                =1  2168     ;
                =1  2169     ;     VARIABLES MODIFIED: PARAM1, PARAM2, MEMORY_TRACE_ADDR_LOW,
                =1  2170     ;        MEMORY_TRACE_ADDR_HIGH, A, POINT1, PNTLOW, PNTHGH, SELECT,
                =1  2171     ;        TEMP_LOW, POINTO, PARTIT_LO_HIGH
                =1  2172     ;
                =1  2173     ;     ERROR EXITS: None
                =1  2174     ;
                =1  2175     ;     SUBROUTINES ACCESSED DIRECTLY: GETOKE, GET_PART, EOL_CHECK,
                =1  2176     ;        NEWLINE, LSTWRD, CO, FETCH, GET_HASH_VALUE, DISASSEMBLE,
                =1  2177     ;        CONTINUATION_LINE, WAIT_FOR_USER
                =1  2178     ;
                =1  2179     ;*****************************************************************
                =1  2180     DISASSEMBLY_CMD:
FCFD 12E056     =1  2181             CALL    GETOKE
FD00 12E065     =1  2182             CALL    GET_PART
FD03 12E06E     =1  2183             CALL    EOL_CHECK
                =1  2184     DS0:
FD06 12E00F     =1  2185             CALL    NEWLINE
FD09 AA57       =1  2186             MOV     PARAM1,PARTIT_LO_HIGH
FD0B AB58       =1  2187             MOV     PARAM2,PARTIT_LO_LOW
FD0D 12E018     =1  2188             CALL    LSTWRD
FD10 7A3D       =1  2189             MOV     PARAM1,#'='                   ;Display Adr = to console
FD12 12E006     =1  2190             CALL    CO
FD15 85586A     =1  2191             MOV     MEMORY_TRACE_ADDR_LOW,PARTIT_LO_LOW
FD18 855769     =1  2192             MOV     MEMORY_TRACE_ADDR_HIGH,PARTIT_LO_HIGH
FD1B 7900       =1  2193             MOV     POINT1,#00H
                =1  2194     DS4:
FD1D E9         =1  2195             MOV     A,POINT1
FD1E B40300     =1  2196             CJNE    A,#03H,DS1
                =1  2197     DS1:
FD21 501D       =1  2198             JNC     DS2
FD23 E558       =1  2199             MOV     A,PARTIT_LO_LOW
FD25 29         =1  2200             ADD     A,POINT1
FD26 F545       =1  2201             MOV     PNTLOW,A
FD28 855744     =1  2202             MOV     PNTHGH,PARTIT_LO_HIGH
FD2B 5002       =1  2203             JNC     DS3
FD2D 0544       =1  2204             INC     PNTHGH
FD2F 754600     =1  2205     DS3:    MOV     SELECT,#(CBYTE_TOKE AND 07H)
FD32 12E04A     =1  2206             CALL    FETCH                         ;Get a byte from code memory
FD35 F547       =1  2207             MOV     TEMP_LOW,A
FD37 7440       =1  2208             MOV     A,#WORKING_SPACE
FD39 29         =1  2209             ADD     A,POINT1
FD3A F8         =1  2210             MOV     POINTO,A
```

```
LOC  OBJ            LINE     SOURCE

FD3B A647       =1  2211                MOV     @POINT0,TEMP_LOW
FD3D 09         =1  2212                INC     POINT1
FD3E 80DD       =1  2213                JMP     DS4
FD40 12FD63     =1  2214     DS2:       CALL    GET_HASH_VALUE          ;Search hash table for match
FD43 12FF84     =1  2215                CALL    DISASSEMBLE
FD46 C558       =1  2216                XCH     A,PARTIT_LO_LOW
FD48 254D       =1  2217                ADD     A,NUMBER_OF_BYTES
FD4A C558       =1  2218                XCH     A,PARTIT_LO_LOW
FD4C 5002       =1  2219                JNC     DS5
FD4E 0557       =1  2220                INC     PARTIT_LO_HIGH
FD50 C3         =1  2221     DS5:       CLR     C
FD51 E55A       =1  2222                MOV     A,PARTIT_HI_LOW
FD53 9558       =1  2223                SUBB    A,PARTIT_LO_LOW         ;Subtract actual partition address low
                =1  2224                                                ;From ending address and carry borrow
FD55 E559       =1  2225                MOV     A,PARTIT_HI_HIGH
FD57 9557       =1  2226                SUBB    A,PARTIT_LO_HIGH        ;Subtract actual partition address high
                =1  2227                                                ;From ending address high
FD59 4005       =1  2228                JC      DSRET                   ;Exit if carry generated
FD5B 12E068     =1  2229                CALL    CONTINUATION_LINE
FD5E 80A6       =1  2230                JMP     DS0
FD60 02E062     =1  2231     DSRET:     JMP     WAIT_FOR_USER
                =1  2232 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                    =1       ;*****************************************************************
                    =1       ;
                    =1       ;    NAME: GET_HASH_VALUE
                    =1       ;
                    =1       ;    ABSTRACT: This routine takes the hash value in OUR_CODE and
                    =1       ;       divides it into one the 4 ordinals.  They are MNEMONIC_ORDINAL,
                    =1       ;       FIRST_OPER_ORDINAL, SECOND_OPER_ORDINAL and THIRD_OPER_ORDINAL.
                    =1       ;
                    =1       ;    INPUTS: WORKING_SPACE
                    =1       ;
                    =1       ;    OUTPUTS: MNEMONIC_ORDINAL, FIRST_OPER_ORDINAL, SECOND_OPER_ORDINAL,
                    =1       ;       THIRD_OPER_ORDINAL
                    =1       ;
                    =1       ;    VARIABLES MODIFIED: A, ERRNUM, DPTR, C, TEMP_LOW, OUR_CODE_LOW,
                    =1       ;       OUR_CODE_HIGH, DIVISOR, DIVIDEND_HIGH, DIVIDEND_LOW, PARAM5,
                    =1       ;       PARAM6, B, QUOTIENT_LOW, QUOTIENT_HIGH, MNEMONIC_ORDINAL,
                    =1       ;       NUMBER_OF_OPERANDS, FIRST_OPER_ORDINAL, SECOND_OPER_ORDINAL,
                    =1       ;       OPERAND_CHECK, NUMBER_OF_BYTES, THIRD_OPER_ORDINAL
                    =1       ;
                    =1       ;    ERROR EXITS: OFH (UNDEFINED OPCODE)
                    =1       ;
                    =1       ;    SUBROUTINES ACCESSED DIRECTLY: ERROR, START_DIVIDE, OPERAND_BYTE_CHECK
                    =1       ;
                    =1       ;*****************************************************************
                    =1       GET_HASH_VALUE:
FD63 E540           =1               MOV     A,WORKING_SPACE                      ;Memory containing opcode to be
FD65 B4A506         =1               CJNE    A,#UNDEFINED_OPCODE,HASH_CONTINUE    ;disassembled
FD68 75430F         =1               MOV     ERRNUM,#OFH                          ;Undefined opcode
FD6B 02E05F         =1               JMP     ERROR
                    =1       HASH_CONTINUE:
FD6E 90F5B3         =1               MOV     DPTR,#INSTRUCTION_CODE               ;Starting adr of hash tbl
FD71 C3             =1               CLR     C
FD72 33             =1               RLC     A                                    ;Multiply pointer by two
FD73 5002           =1               JNC     GHV_A1
FD75 0583           =1               INC     DPH                                  ;Increment DPH if rotate overflows
FD77 F547           =1       GHV_A1: MOV     TEMP_LOW,A
FD79 93             =1               MOVC    A,@A+DPTR
FD7A F54E           =1               MOV     OUR_CODE_HIGH,A
FD7C 0547           =1               INC     TEMP_LOW
FD7E E547           =1               MOV     A,TEMP_LOW
FD80 93             =1               MOVC    A,@A+DPTR
FD81 F54F           =1               MOV     OUR_CODE_LOW,A                       ;Ordinal of hashed value
FD83 75702C         =1               MOV     DIVISOR,#MNEMONIC_FACTOR
FD86 854E6E         =1               MOV     DIVIDEND_HIGH,OUR_CODE_HIGH
FD89 854F6F         =1               MOV     DIVIDEND_LOW,OUR_CODE_LOW
FD8C 31B4           =1               CALL    START_DIVIDE
FD8E AE72           =1               MOV     PARAM5,QUOTIENT_LOW
FD90 AF71           =1               MOV     PARAM6,QUOTIENT_HIGH
FD92 E572           =1               MOV     A,QUOTIENT_LOW
FD94 75F02C         =1               MOV     B,#MNEMONIC_FACTOR
FD97 A4             =1               MUL     AB
FD98 F572           =1               MOV     QUOTIENT_LOW,A
FD9A 85F071         =1               MOV     QUOTIENT_HIGH,B
FD9D EF             =1               MOV     A,PARAM6
FD9E 75F02C         =1               MOV     B,#MNEMONIC_FACTOR
```

```
LOC  OBJ            LINE     SOURCE

FDA1 A4        =1   2288             MUL     AB
FDA2 2571      =1   2289             ADD     A,QUOTIENT_HIGH
FDA4 F571      =1   2290             MOV     QUOTIENT_HIGH,A
FDA6 E54F      =1   2291             MOV     A,OUR_CODE_LOW
FDA8 C3        =1   2292             CLR     C
FDA9 9572      =1   2293             SUBB    A,QUOTIENT_LOW
FDAB F56D      =1   2294             MOV     MNEMONIC_ORDINAL,A                 ;Mnemonic ord
FDAD 8F4E      =1   2295             MOV     OUR_CODE_HIGH,PARAM6
FDAF 8E4F      =1   2296             MOV     OUR_CODE_LOW,PARAM5
FDB1 E54F      =1   2297             MOV     A,OUR_CODE_LOW
FDB3 700A      =1   2298             JNZ     GHV1
FDB5 E54E      =1   2299             MOV     A,OUR_CODE_HIGH
FDB7 7006      =1   2300             JNZ     GHV1
FDB9 756B00    =1   2301             MOV     NUMBER_OF_OPERANDS,#00H
FDBC 02FE17    =1   2302             JMP     GHV9
               =1   2303     GHV1:
 DBF 757018    =1   2304             MOV     DIVISOR,#OPERAND_FACTOR
FDC2 854E6E    =1   2305             MOV     DIVIDEND_HIGH,OUR_CODE_HIGH
FDC5 854F6F    =1   2306             MOV     DIVIDEND_LOW,OUR_CODE_LOW
FDC8 31B4      =1   2307             CALL    START_DIVIDE
FDCA AE72      =1   2308             MOV     PARAM5,QUOTIENT_LOW
FDCC AF71      =1   2309             MOV     PARAM6,QUOTIENT_HIGH
FDCE E572      =1   2310             MOV     A,QUOTIENT_LOW
FDD0 75F018    =1   2311             MOV     B,#OPERAND_FACTOR
FDD3 A4        =1   2312             MUL     AB
FDD4 F572      =1   2313             MOV     QUOTIENT_LOW,A
FDD6 85F071    =1   2314             MOV     QUOTIENT_HIGH,B
FDD9 EF        =1   2315             MOV     A,PARAM6
FDDA 75F018    =1   2316             MOV     B,#OPERAND_FACTOR
FDDD A4        =1   2317             MUL     AB
FDDE 2571      =1   2318             ADD     A,QUOTIENT_HIGH
FDE0 F571      =1   2319             MOV     QUOTIENT_HIGH,A
 DE2 E54F      =1   2320             MOV     A,OUR_CODE_LOW
 DE4 C3        =1   2321             CLR     C
FDE5 9572      =1   2322             SUBB    A,QUOTIENT_LOW
FDE7 F563      =1   2323             MOV     FIRST_OPER_ORDINAL,A               ;First operand ord
FDE9 B40F03    =1   2324             CJNE    A,#0FH,GHV2
FDEC 02FDF1    =1   2325             JMP     GHV2_2
               =1   2326     GHV2:
FDEF 5002      =1   2327             JNC     GHV3
               =1   2328     GHV2_2:
FDF1 1563      =1   2329             DEC     FIRST_OPER_ORDINAL
               =1   2330     GHV3:
FDF3 8F4E      =1   2331             MOV     OUR_CODE_HIGH,PARAM6
FDF5 8E4F      =1   2332             MOV     OUR_CODE_LOW,PARAM5
FDF7 E54F      =1   2333             MOV     A,OUR_CODE_LOW
FDF9 700A      =1   2334             JNZ     GHV5
FDFB E54E      =1   2335             MOV     A,OUR_CODE_HIGH
FDFD 7006      =1   2336             JNZ     GHV5
FDFF 756B01    =1   2337             MOV     NUMBER_OF_OPERANDS,#01H
 E02 02FE17    =1   2338             JMP     GHV9
               =1   2339     GHV5:
FE05 854F64    =1   2340             MOV     SECOND_OPER_ORDINAL,OUR_CODE_LOW   ;Second operand ord
FE08 E564      =1   2341             MOV     A,SECOND_OPER_ORDINAL
FE0A B40F03    =1   2342             CJNE    A,#0FH,GHV6
```

```
LOC  OBJ            LINE     SOURCE

FE0D 02FE12     =1  2343              JMP     GHV6_6
                =1  2344     GHV6:
FE10 5002       =1  2345              JNC     GHV7
                =1  2346     GHV6_6:
FE12 1564       =1  2347              DEC     SECOND_OPER_ORDINAL
                =1  2348     GHV7:
FE14 756B02     =1  2349              MOV     NUMBER_OF_OPERANDS,#02H
                =1  2350     GHV9:
FE17 E56D       =1  2351              MOV     A,MNEMONIC_ORDINAL
FE19 B40909     =1  2352              CJNE    A,#09H,GHV10
FE1C 754D02     =1  2353              MOV     NUMBER_OF_BYTES,#02H
FE1F 756516     =1  2354              MOV     THIRD_OPER_ORDINAL,#16H
FE22 02FE28     =1  2355              JMP     GHV11
                =1  2356     GHV10:
FE25 754D01     =1  2357              MOV     NUMBER_OF_BYTES,#01H
                =1  2358     GHV11:
FE28 90FE32     =1  2359              MOV     DPTR,#GHVTBL
FE2B E56B       =1  2360              MOV     A,NUMBER_OF_OPERANDS
FE2D 85636C     =1  2361              MOV     OPERAND_CHECK,FIRST_OPER_ORDINAL
FE30 23         =1  2362              RL      A
FE31 73         =1  2363              JMP     @A+DPTR
                =1  2364     GHVTBL:
FE32 22         =1  2365              RET                                    ;Entry 1 for GHVTBL
FE33 00         =1  2366              NOP
FE34 8006       =1  2367              SJMP    OPERAND_BYTE_CHECK             ;Entry 2 for GHVTBL
FE36 12FE3C     =1  2368              CALL    OPERAND_BYTE_CHECK             ;Entry 3 for GHVTBL
FE39 85646C     =1  2369              MOV     OPERAND_CHECK,SECOND_OPER_ORDINAL
                =1  2370
                =1  2371 +1  $EJECT
```

```
LOC  OBJ                   SOURCE
                    =1     ;***************************************************************
                    =1     ;
                    =1     ;     NAME: OPERAND_BYTE_CHECK
                    =1     ;
                    =1     ;     ABSTRACT: This routine is updating the number of bytes in the
                    =1     ;        opcode based on OPERAND_CHECK.
                    =1     ;
                    =1     ;        CAUTION: This routine is position sensitive.  It is entered from
                    =1     ;        the previous routine, GET_HASH_VALUE as 'in line' code.
                    =1     ;
                    =1     ;     INPUTS: OPERAND_CHECK
                    =1     ;
                    =1     ;     OUTPUTS: NUMBER_BYTES
                    =1     ;
                    =1     ;     VARIABLES MODIFIED: A, NUMBER_OF_BYTES
                    =1     ;
                    =1     ;     ERROR EXITS: None
                    =1     ;
                    =1     ;     SUBROUTINES ACCESSED DIRECTLY: None
                    =1     ;
                    =1     ;
                    =1     ;***************************************************************
                    =1     OPERAND_BYTE_CHECK:
FE3C E56C           =1             MOV     A,OPERAND_CHECK
FE3E B41000         =1             CJNE    A,#10H,OBC0
                    =1     OBC0:
FE41 400A           =1             JC      OBC1
FE43 B41603         =1             CJNE    A,#16H,OBC2
FE46 02FE4B         =1             JMP     OBC2_2
                    =1     OBC2:
FE49 5002           =1             JNC     OBC1
                    =1     OBC2_2:
FE4B 054D           =1             INC     NUMBER_OF_BYTES
                    =1     OBC1:
FE4D B41402         =1             CJNE    A,#14H,OBCRET
FE50 054D           =1             INC     NUMBER_OF_BYTES
FE52 22             =1     OBCRET: RET
                    =1  +1 $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1        ;******************************************************************
                =1        ;
                =1        ;     NAME: DISPLAY_OPERAND
                =1        ;
                =1        ;     ABSTRACT: This routine displays an operand of the disassembled
                =1        ;        opcode to the console.
                =1        ;
                =1        ;     INPUTS: NUNBER_OF_OPERANDS_PRINTED, FIRST_OPER_ORDINAL,
                =1        ;        SECOND_OPER_ORDINAL, THIRD_OPER_ORDINAL
                =1        ;
                =1        ;     OUTPUTS: NUMBER_OF_OPERANDS_PRINTED
                =1        ;
                =1        ;     VARIABLES MODIFIED: A, DPTR, CURRENT_OPERAND, C, PARAM1, POINTO,
                =1        ;        VALHGH, VALLOW, PARAM2, EXPRESSIONS_PRINTED, MEMORY_TRACE_ADDR_HIGH,
                =1        ;        TEMP_LOW, NO_OF_OPERANDS_PRINTED
                =1        ;
                =1        ;     ERROR EXITS: None
                =1        ;
                =1        ;     SUBROUTINES ACCESSED DIRECTLY: DISPLAY_TOKEN, LSTBYT, CO, LSTWRD,
                =1        ;        PRINT_STRING
                =1        ;
                =1        ;
                =1        ;******************************************************************
                =1        DISPLAY_OPERAND:
FE53 E567       =1                MOV     A,NO_OF_OPERANDS_PRINTED
FE55 14         =1                DEC     A
FE56 23         =1                RL      A
FE57 23         =1                RL      A
FE58 90FE5C     =1                MOV     DPTR,#DDTBL
FE5B 73         =1                JMP     @A+DPTR
FE5C E563       =1        DDTBL:  MOV     A,FIRST_OPER_ORDINAL
FE5E 8006       =1                SJMP    DD0
FE60 E564       =1                MOV     A,SECOND_OPER_ORDINAL
FE62 8002       =1                SJMP    DD0
FE64 E565       =1                MOV     A,THIRD_OPER_ORDINAL
FE66 F566       =1        DD0:    MOV     CURRENT_OPERAND,A
FE68 B40C05     =1                CJNE    A,#0CH,DD0_1
FE6B 74A1       =1                MOV     A,#0A1H
FE6D 02FE8E     =1                JMP     DD4_1
FE70 B40F03     =1        DD0_1:  CJNE    A,#0FH,DD1
FE73 02FE78     =1                JMP     DD1_1
FE76 501A       =1        DD1:    JNC     DD2
FE78 B40300     =1        DD1_1:  CJNE    A,#03H,DD3
FE7B 400E       =1        DD3:    JC      DD4
FE7D B40A03     =1                CJNE    A,#0AH,DD5
FE80 02FE85     =1                JMP     DD5_5
FE83 5006       =1        DD5:    JNC     DD4
FE85 C3         =1        DD5_5:  CLR     C
FE86 248D       =1                ADD     A,#8DH
FE88 02FE8E     =1                JMP     DD4_1
FE8B C3         =1        DD4:    CLR     C
FE8C 2451       =1                ADD     A,#(OFST+REG+1)
FE8E FA         =1        DD4_1:  MOV     PARAM1,A
FE8F 12E059     =1                CALL    DISPLAY_TOKEN
FE92 E566       =1        DD2:    MOV     A,CURRENT_OPERAND
```

```
LOC  OBJ           LINE      SOURCE

FE94 C3          =1  2465             CLR      C
FE95 9410        =1  2466             SUBB     A,#10H
FE97 B4000F      =1  2467             CJNE     A,#00H,DD_CASE_1             ;Byte expression 8-bits
                 =1  2468     DD_CASE_EXP8:                        ;Generalized byte expression display
FE9A 7440        =1  2469             MOV      A,#WORKING_SPACE
FE9C 2568        =1  2470             ADD      A,EXPRESSIONS_PRINTED
FE9E F8          =1  2471             MOV      POINT0,A
FE9F E6          =1  2472             MOV      A,@POINT0
FEA0 FA          =1  2473             MOV      PARAM1,A
FEA1 12E015      =1  2474             CALL     LSTBYT
FEA4 0568        =1  2475             INC      EXPRESSIONS_PRINTED
FEA6 02FF7C      =1  2476             JMP      DD_CASE_END
                 =1  2477     DD_CASE_1:
FEA9 B40102      =1  2478             CJNE     A,#01H,DD_CASE_2             ;Bit expression, 8-bits
FEAC 80EC        =1  2479             JMP      DD_CASE_EXP8
                 =1  2480     DD_CASE_2:
FEAE B40207      =1  2481             CJNE     A,#02H,DD_CASE_3             ;Immediate expression, 8-bits
FEB1 7A23        =1  2482             MOV      PARAM1,#'#'
FEB3 12E006      =1  2483             CALL     CO
FEB6 80E2        =1  2484             JMP      DD_CASE_EXP8
                 =1  2485     DD_CASE_3:
FEB8 B40307      =1  2486             CJNE     A,#03H,DD_CASE_4             ;Complimented byte expression, 8-bits
FEBB 7A2F        =1  2487             MOV      PARAM1,#'/'
FEBD 12E006      =1  2488             CALL     CO
FEC0 80D8        =1  2489             JMP      DD_CASE_EXP8
                 =1  2490     DD_CASE_4:
FEC2 B4043F      =1  2491             CJNE     A,#04H,DD_CASE_5             ;Expression, 16-bits
FEC5 7840        =1  2492             MOV      POINT0,#WORKING_SPACE
FEC7 08          =1  2493             INC      POINT0
FEC8 8649        =1  2494             MOV      VALHGH,@POINT0
FECA 08          =1  2495             INC      POINT0
FECB 864A        =1  2496             MOV      VALLOW,@POINT0
FECD E56D        =1  2497             MOV      A,MNEMONIC_ORDINAL
FECF B40F0F      =1  2498             CJNE     A,#0FH,DD_CASE_4_0
FED2 7A23        =1  2499             MOV      PARAM1,#'#'
FED4 12E006      =1  2500             CALL     CO
                 =1  2501     DD_CASE_EXP16:                               ;Generalized word expression display
FED7 AA49        =1  2502             MOV      PARAM1,VALHGH
FED9 AB4A        =1  2503             MOV      PARAM2,VALLOW
FEDB 12E018      =1  2504             CALL     LSTWRD
FEDE 02FF7C      =1  2505             JMP      DD_CASE_END
                 =1  2506     DD_CASE_4_0:
FEE1 E566        =1  2507             MOV      A,CURRENT_OPERAND
FEE3 B41403      =1  2508             CJNE     A,#14H,SS0
FEE6 02FEF2      =1  2509             JMP      SS3
FEE9 B41503      =1  2510     SS0:    CJNE     A,#21,SS1
FEEC 02FEF2      =1  2511             JMP      SS3
FEEF B4160A      =1  2512     SS1:    CJNE     A,#16H,SS2
FEF2 AA49        =1  2513     SS3:    MOV      PARAM1,VALHGH
FEF4 AB4A        =1  2514             MOV      PARAM2,VALLOW
FEF6 12E018      =1  2515             CALL     LSTWRD
FEF9 02FF7C      =1  2516             JMP      DD_CASE_END
FEFC AA4A        =1  2517     SS2:    MOV      PARAM1,VALLOW
FEFE 12E015      =1  2518             CALL     LSTBYT
FF01 02FF7C      =1  2519             JMP      DD_CASE_END
```

```
LOC  OBJ            LINE     SOURCE

               =1   2520     DD_CASE_5:
FF04 B4050E    =1   2521             CJNE    A,#05H,DD_CASE_6              ;Expression, 11-bits
FF07 7840      =1   2522             MOV     POINTO,#WORKING_SPACE
FF09 E6        =1   2523             MOV     A,@POINTO
FF0A 54E0      =1   2524             ANL     A,#0E0H
FF0C C4        =1   2525             SWAP    A
FF0D 03        =1   2526             RR      A
FF0E F549      =1   2527             MOV     VALHGH,A
FF10 08        =1   2528             INC     POINTO
FF11 864A      =1   2529             MOV     VALLOW,@POINTO
FF13 80C2      =1   2530             JMP     DD_CASE_EXP16
               =1   2531     DD_CASE_6:
FF15 B4063C    =1   2532             CJNE    A,#06H,DD_CASE_7              ;Relative offset
FF18 E56A      =1   2533             MOV     A,MEMORY_TRACE_ADDR_LOW
FF1A 254D      =1   2534             ADD     A,NUMBER_OF_BYTES
FF1C F56A      =1   2535             MOV     MEMORY_TRACE_ADDR_LOW,A
FF1E 5002      =1   2536             JNC     DD_CASE_6_0
FF20 0569      =1   2537             INC     MEMORY_TRACE_ADDR_HIGH
               =1   2538     DD_CASE_6_0:
FF22 7440      =1   2539             MOV     A,#WORKING_SPACE
FF24 2568      =1   2540             ADD     A,EXPRESSIONS_PRINTED
FF26 F8        =1   2541             MOV     POINTO,A
FF27 E6        =1   2542             MOV     A,@POINTO
FF28 B47F03    =1   2543             CJNE    A,#07FH,DD_CASE_6_1
FF2B 02FF45    =1   2544             JMP     DD_CASE_6_2
               =1   2545     DD_CASE_6_1:
FF2E 4015      =1   2546             JC      DD_CASE_6_2
FF30 F4        =1   2547             CPL     A
FF31 04        =1   2548             INC     A
FF32 F547      =1   2549             MOV     TEMP_LOW,A
FF34 E56A      =1   2550             MOV     A,MEMORY_TRACE_ADDR_LOW
FF36 C3        =1   2551             CLR     C
FF37 9547      =1   2552             SUBB    A,TEMP_LOW
FF39 F54A      =1   2553             MOV     VALLOW,A
FF3B E569      =1   2554             MOV     A,MEMORY_TRACE_ADDR_HIGH
FF3D 5001      =1   2555             JNC     DD_CASE_6_3
FF3F 14        =1   2556             DEC     A
               =1   2557     DD_CASE_6_3:
FF40 F549      =1   2558             MOV     VALHGH,A
FF42 02FF50    =1   2559             JMP     DD_CASE_6_5
               =1   2560     DD_CASE_6_2:
FF45 256A      =1   2561             ADD     A,MEMORY_TRACE_ADDR_LOW
FF47 F54A      =1   2562             MOV     VALLOW,A
FF49 E569      =1   2563             MOV     A,MEMORY_TRACE_ADDR_HIGH
FF4B 5001      =1   2564             JNC     DD_CASE_6_4
FF4D 04        =1   2565             INC     A
               =1   2566     DD_CASE_6_4:
FF4E F549      =1   2567             MOV     VALHGH,A
               =1   2568     DD_CASE_6_5:
FF50 0568      =1   2569             INC     EXPRESSIONS_PRINTED
FF52 8083      =1   2570             JMP     DD_CASE_EXP16
               =1   2571     DD_CASE_7:
FF54 B40712    =1   2572             CJNE    A,#07H,DD_CASE_8              ;Special case for @A+DPTR
FF57 7AFF      =1   2573             MOV     PARAM1,#HIGH DD_CASE_7_MSG
FF59 7B61      =1   2574             MOV     PARAM2,#LOW DD_CASE_7_MSG
```

```
LOC  OBJ            LINE      SOURCE

FF5B 12E01E      =1  2575              CALL    PRINT_STRING
FF5E 02FF7C      =1  2576              JMP     DD_CASE_END
                 =1  2577     DD_CASE_7_MSG:
FF61 07          =1  2578              DB      07,'@A+DPTR'
FF62 40412B44
FF66 505452
                 =1  2579     DD_CASE_8:
FF69 B40810      =1  2580              CJNE    A,#8,DD_CASE_END              ;Special case for @A+PC
FF6C 7AFF        =1  2581              MOV     PARAM1,#HIGH DD_CASE_8_MSG
FF6E 7B76        =1  2582              MOV     PARAM2,#LOW DD_CASE_8_MSG
FF70 12E01E      =1  2583              CALL    PRINT_STRING
FF73 02FF7C      =1  2584              JMP     DD_CASE_END
                 =1  2585     DD_CASE_8_MSG:
FF76 05          =1  2586              DB      05,'@A+PC'
FF77 40412B50
FF7B 43
                 =1  2587     DD_CASE_END:
FF7C 0567        =1  2588              INC     NO_OF_OPERANDS_PRINTED
FF7E 22          =1  2589              RET
                 =1  2590 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

               =1   2591     ;*****************************************************************
               =1   2592     ;
               =1   2593     ;    NAME: DISPLAY_COMMA
               =1   2594     ;
               =1   2595     ;    ABSTRACT: This routine displays a comma symbol to the console.
               =1   2596     ;
               =1   2597     ;    INPUTS: None
               =1   2598     ;
               =1   2599     ;    OUTPUTS: None
               =1   2600     ;
               =1   2601     ;    VARIABLES MODIFIED: PARAM1
               =1   2602     ;
               =1   2603     ;    ERROR EXITS: None
               =1   2604     ;
               =1   2605     ;    SUBROUTINES ACCESSED DIRECTLY: CO
               =1   2606     ;
               =1   2607     ;
               =1   2608     ;*****************************************************************
               =1   2609     DISPLAY_COMMA:
FF7F 7A2C      =1   2610            MOV     PARAM1,#','
FF81 02E006    =1   2611            JMP     CO
               =1   2612 +1  $EJECT
```

```
LOC  OBJ           LINE      SOURCE

                =1        ;*****************************************************************
                =1        ;
                =1        ;    NAME: DISASSEMBLE
                =1        ;
                =1        ;    ABSTRACT: This routine displays one disassembled opcode on the
                =1        ;       console.
                =1        ;
                =1        ;    INPUTS: MNEMONIC_ORDINAL
                =1        ;
                =1        ;    OUTPUTS: None
                =1        ;
                =1        ;    VARIABLES MODIFIED: A, PARAM1, DPTR, INSTRUCTION_VALUE,
                =1        ;       NO_OF_OPERANDS_PRINTED, EXPRESSIONS_PRINTED, C
                =1        ;
                =1        ;    ERROR EXITS: None
                =1        ;
                =1        ;    SUBROUTINES ACCESSED DIRECTLY: DISPLAY_TOKEN, CO, DISPLAY_OPERAND,
                =1        ;       DISPLAY_COMMA,
                =1        ;
                =1        ;*****************************************************************
                =1        DISASSEMBLE:
FF84 E56D       =1                MOV     A,MNEMONIC_ORDINAL
FF86 2410       =1                ADD     A,#OFST
FF88 FA         =1                MOV     PARAM1,A
FF89 12E059     =1                CALL    DISPLAY_TOKEN
FF8C 90F587     =1                MOV     DPTR,#MNEMONIC_TAB
FF8F E56D       =1                MOV     A,MNEMONIC_ORDINAL
FF91 93         =1                MOVC    A,@A+DPTR
FF92 F55B       =1                MOV     INSTRUCTION_VALUE,A
FF94 7A20       =1                MOV     PARAM1,#' '
FF96 12E006     =1                CALL    CO
FF99 756701     =1                MOV     NO_OF_OPERANDS_PRINTED,#1
FF9C 756801     =1                MOV     EXPRESSIONS_PRINTED,#1
 F9F E55B       =1                MOV     A,INSTRUCTION_VALUE
 FA1 C3         =1                CLR     C
FFA2 9407       =1                SUBB    A,#07H
FFA4 B40001     =1                CJNE    A,#00H,DISCASE_1
FFA7 22         =1                RET
                =1        DISCASE_1:
FFA8 B40102     =1                CJNE    A,#01H,DISCASE_2
FFAB C153       =1                JMP     DISPLAY_OPERAND
                =1        DISCASE_2:
FFAD B40212     =1                CJNE    A,#02H,DISCASE_3
FFB0 E540       =1                MOV     A,WORKING_SPACE
FFB2 B48507     =1                CJNE    A,#85H,DISCASE_2_1                ;Check for special case
FFB5 E541       =1                MOV     A,(WORKING_SPACE+1)               ;of MOV /,/ where operands
FFB7 854241     =1                MOV     (WORKING_SPACE+1),(WORKING_SPACE+2)     ;are in reverse order.
FFBA F542       =1                MOV     (WORKING_SPACE+2),A
                =1        DISCASE_2_1:
FFBC D153       =1                CALL    DISPLAY_OPERAND
FFBE F17F       =1                CALL    DISPLAY_COMMA
 FC0 C153       =1                JMP     DISPLAY_OPERAND
                =1        DISCASE_3:
FFC2 B40306     =1                CJNE    A,#03H,DISCASE_4
FFC5 D153       =1                CALL    DISPLAY_OPERAND
```

```
LOC  OBJ            LINE     SOURCE

FFC7 F17F      =1   2668              CALL     DISPLAY_COMMA
FFC9 C153      =1   2669              JMP      DISPLAY_OPERAND
               =1   2670     DISCASE_4:
FFCB B4040A    =1   2671              CJNE     A,#04H,DISCASE_5
FFCE D153      =1   2672              CALL     DISPLAY_OPERAND
FFD0 F17F      =1   2673              CALL     DISPLAY_COMMA
FFD2 D153      =1   2674              CALL     DISPLAY_OPERAND
FFD4 F17F      =1   2675              CALL     DISPLAY_COMMA
FFD6 C153      =1   2676              JMP      DISPLAY_OPERAND
               =1   2677     DISCASE_5:
FFD8 B40502    =1   2678              CJNE     A,#05H,DISCASE_6
FFDB C153      =1   2679              JMP      DISPLAY_OPERAND
               =1   2680     DISCASE_6:
FFDD B40606    =1   2681              CJNE     A,#06H,DISCASE_7
FFE0 D153      =1   2682              CALL     DISPLAY_OPERAND
FFE2 F17F      =1   2683              CALL     DISPLAY_COMMA
FFE4 C153      =1   2684              JMP      DISPLAY_OPERAND
               =1   2685     DISCASE_7:
FFE6 B40702    =1   2686              CJNE     A,#07H,DISCASE_8
FFE9 C153      =1   2687              JMP      DISPLAY_OPERAND
               =1   2688     DISCASE_8:
FFEB B40802    =1   2689              CJNE     A,#08H,DISCASE_END
FFEE D153      =1   2690              CALL     DISPLAY_OPERAND
               =1   2691     DISCASE_END:
FFF0 22        =1   2692              RET
                    2693     END
```

```
XREF SYMBOL TABLE LISTING
---- ------ ----- -------

NAME                           TYPE      VALUE AND REFERENCES

A_OP1 . . . . . . . . . . . .  N         002CH  363# 461 463 481 483 501 503 504 505 506 508 509 510 511 513 514 515 516
                                           521 523 524 525 526 528 529 530 531 533 534 535 536 543 544 545 546 548 549
                                           550 551 553 554 555 556 563 564 565 566 568 569 570 571 573 574 575 576 583
                                           584 585 586 588 589 590 591 593 594 595 596 603 621 641 643 644 645 646 648
                                           649 650 651 653 654 655 656 683 684 703 704 705 706 708 709 710 711 713 714
                                           715 716 723 725 726 738 740 741 743 744 745 746 748 749 750 751 753 754 755
                                           756 763
A_OP2 . . . . . . . . . . . .  N         0420H  387# 540 560 580 758 760 761 764 765 766 768 769 770 771 773 774 775 776
AB_OP1. . . . . . . . . . . .  N         0210H  374# 623 663
AMT0. . . . . . . . . . . . .  L CSEG    F98FH  1249 1255#
AMT1. . . . . . . . . . . . .  L CSEG    F992H  1256# 1269
AMT2. . . . . . . . . . . . .  L CSEG    F9ADH  1263 1267#
AMTERR. . . . . . . . . . . .  L CSEG    F9A7H  1218 1255 1265#
ASERR . . . . . . . . . . . .  L CSEG    F832H  834 836 849 886 893 901# 939 943 950 982
ASM_PC_HIGH . . . . . . . . .  L DSEG    004BH  82# 1251 1257 1877 1882 1946 1986 2002 2017 2045 2061 2065 2082 2107
                                           2136 2151
ASM_PC_LOW. . . . . . . . . .  L DSEG    004CH  83# 1252 1259 1874 1883 1947 1993 1999 2014 2052 2058 2066 2079 2135
                                           2150
ASMBASE . . . . . . . . . . .  N         F581H  35# 36
ASSEMBLY_CMD. . . . . . . . .  L CSEG    F977H  37 1246#
ATA_PLUS_DPTR_OP1 . . . . . .  N         03F4H  383# 601
ATA_PLUS_DPTR_OP2 . . . . . .  N         5EE0H  406# 641
ATA_PLUS_PC_OP2 . . . . . . .  N         6300H  407# 621
ATA_TOKE. . . . . . . . . . .  N         000AH  50# 832 943
ATDPTR_OP1. . . . . . . . . .  N         0294H  377# 758
ATDPTR_OP2. . . . . . . . . .  N         3DE0H  399# 738
ATR0_OP1. . . . . . . . . . .  N         0058H  364# 465 485 605 665 685 760 765
ATR0_OP2. . . . . . . . . . .  N         0840H  388# 505 525 545 565 585 625 645 705 725 740 745
ATR1_OP1. . . . . . . . . . .  N         0084H  365# 466 486 606 666 686 761 766
ATR1_OP2. . . . . . . . . . .  N         0C60H  389# 506 526 546 566 586 626 646 706 726 741 746
B . . . . . . . . . . . . . .  N DSEG    00F0H PREDEFINED  1154 1525 1529 1538 1540 1541 1545 1549 1578 1594 1611 1725
                                           1787 1798 1799 1803 1807 1820 1821 1825 1829 2282 2285 2287 2311 2314 2316
B_O_T . . . . . . . . . . . .  L BSEG    0000H  103# 840 890 898 939 982 1055
BACKWARD_JUMP_CASE_4. . . . .  L CSEG    FC02H  1989 1995 2012#
BACKWARD_JUMP_CASE_5. . . . .  L CSEG    FC72H  2048 2054 2077#
BAR_TOKE. . . . . . . . . . .  N         0003H  46# 908
BASE. . . . . . . . . . . . .  N         E000H  42# 122 123 124 125 126 127 128 129 131 132 133 134 135 136 137 138 139
                                           140 141 142 143
BIT_END . . . . . . . . . . .  N         001BH  451# 1208
BIT_EXP . . . . . . . . . . .  L BSEG    0002H  444# 1191 1210 1617 1699 1748
BIT_EXP8_OP1. . . . . . . . .  N         02ECH  379# 478 498 518 640 680 700 720
BIT_EXP8_OP2. . . . . . . . .  N         4620H  401# 600 620 660
BLINK . . . . . . . . . . . .  N         0080H  60#
BYTE_EXP8_OP1 . . . . . . . .  N         02C0H  378# 464 484 540 541 560 561 580 581 604 624 625 626 628 629 630 631 633
                                           634 635 636 698 718 724 764
BYTE_EXP8_OP2 . . . . . . . .  N         4200H  400# 504 524 544 564 584 624 644 665 666 668 669 670 671 673 674 675 676
                                           684 704 744
C_OP1 . . . . . . . . . . . .  N         0268H  376# 600 620 658 660 678 681 701 721
C_OP2 . . . . . . . . . . . .  N         39C0H  398# 640
C_TOKE. . . . . . . . . . . .  N         005EH  51# 1616
CALCULATE_INSTRUCTION_VALUE .  L CSEG    FA00H  803 846 851 889 996 1026 1058 1063 1096 1123 1477#
```

```
NAME                           TYPE     VALUE AND REFERENCES

CASE_3_MORE . . . . . . . . . . L CSEG   FBB9H   1968 1972#
CBYTE_TOKE. . . . . . . . . . . N        0080H   52# 2205
CHANGE_ASM_PC_1 . . . . . . . . L CSEG   FB7DH   1878 1881#
CHANGE_CASE_2 . . . . . . . . . L CSEG   FB96H   1949 1954#
CHANGE_CASE_3 . . . . . . . . . L CSEG   FBA5H   1955 1962#
CHANGE_CASE_4 . . . . . . . . . L CSEG   FBC3H   1963 1978#
CHANGE_CASE_4_END . . . . . . . L CSEG   FC1FH   2010 2031#
CHANGE_CASE_4A. . . . . . . . . L CSEG   FBDCH   1986 1988#
CHANGE_CASE_4AA . . . . . . . . L CSEG   FBE1H   1987 1991#
CHANGE_CASE_4C. . . . . . . . . L CSEG   FBE6H   1993 1994#
CHANGE_CASE_4D. . . . . . . . . L CSEG   FBFAH   2005 2006#
CHANGE_CASE_4F. . . . . . . . . L CSEG   FC16H   2022 2023#
CHANGE_CASE_5 . . . . . . . . . L CSEG   FC26H   1979 2035#
CHANGE_CASE_5_END . . . . . . . L CSEG   FC8FH   2075 2095#
CHANGE_CASE_5A. . . . . . . . . L CSEG   FC43H   2045 2047#
CHANGE_CASE_5AA . . . . . . . . L CSEG   FC48H   2046 2050#
CHANGE_CASE_5C. . . . . . . . . L CSEG   FC4DH   2052 2053#
CHANGE_CASE_5D. . . . . . . . . L CSEG   FC6AH   2070 2071#
CHANGE_CASE_5F. . . . . . . . . L CSEG   FC86H   2087 2088#
CHANGE_CASE_6 . . . . . . . . . L CSEG   FC96H   2036 2099#
CHANGE_CASE_7 . . . . . . . . . L CSEG   FCBFH   2100 2120#
CHANGE_END. . . . . . . . . . . L CSEG   FCD4H   1953 1961 1977 2034 2098 2119 2131#
CHANGE_END_A. . . . . . . . . . L CSEG   FCF3H   2145 2147#
CHANGE_END_LOOP . . . . . . . . L CSEG   FCE5H   2138# 2148
CHANGE_ERROR. . . . . . . . . . L CSEG   FC5CH   2003 2007 2020 2024 2064# 2072 2085 2089 2109 2122
CHANGE_TO_INSTRUCTION_OP. . . . L CSEG   FB82H   1220 1945#
CHARIN. . . . . . . . . . . . . L DSEG   0050H   87#
CHECK_AND_CHANGE_ASM_PC . . . . L CSEG   FB6DH   1872# 1950 1956 1964 1982 2039 2102 2124
CHECK_AND_INC_HASH_TAB. . . . . L CSEG   FA17H   1485 1492#
CHECK_AND_SET_EXP_FLAG. . . . . L CSEG   FAA8H   850 1061 1697#
CHECK_AND_SET_SECOND_EXP_FLAG   L CSEG   FAF0H   914 992 1745#
CHECK_EXP_FLAG. . . . . . . . . L CSEG   FAC9H   804 1719#
CHECK_HASH_TAB. . . . . . . . . L CSEG   FA18H   1489 1494#
CHECKSUM. . . . . . . . . . . . N REG    R6      116#
CHRCNT. . . . . . . . . . . . . L DSEG   0051H   88#
CI. . . . . . . . . . . . . . . N        E009H   123#
CO. . . . . . . . . . . . . . . N        E006H   122# 2190 2483 2488 2500 2611 2643
CONT_OUR_CODE . . . . . . . . . L CSEG   F95AH   1205# 1213
CONT_UPDATE_LSSTHN. . . . . . . L CSEG   FA34H   1527# 1536
CONTINUATION_LINE . . . . . . . N        E068H   141# 2229
COUNT . . . . . . . . . . . . . N REG    R7      115#
CSTS. . . . . . . . . . . . . . N        E00CH   124#
CURRENT_OPERAND . . . . . . . . L DSEG   0066H   425# 2445 2464 2507
DD_CASE_1 . . . . . . . . . . . L CSEG   FEA9H   2467 2477#
DD_CASE_2 . . . . . . . . . . . L CSEG   FEAEH   2478 2480#
DD_CASE_3 . . . . . . . . . . . L CSEG   FEB8H   2481 2485#
DD_CASE_4 . . . . . . . . . . . L CSEG   FEC2H   2486 2490#
DD_CASE_4_0 . . . . . . . . . . L CSEG   FEE1H   2498 2506#
DD_CASE_5 . . . . . . . . . . . L CSEG   FF04H   2491 2520#
DD_CASE_6 . . . . . . . . . . . L CSEG   FF15H   2521 2531#
DD_CASE_6_0 . . . . . . . . . . L CSEG   FF22H   2536 2538#
DD_CASE_6_1 . . . . . . . . . . L CSEG   FF2EH   2543 2545#
DD_CASE_6_2 . . . . . . . . . . L CSEG   FF45H   2544 2546 2560#
DD_CASE_6_3 . . . . . . . . . . L CSEG   FF40H   2555 2557#
DD_CASE_6_4 . . . . . . . . . . L CSEG   FF4EH   2564 2566#
```

```
NAME                              TYPE     VALUE AND REFERENCES

DD_CASE_6_5 . . . . . . . . . .   L CSEG   FF50H  2559 2568#
DD_CASE_7 . . . . . . . . . . .   L CSEG   FF54H  2532 2571#
DD_CASE_7_MSG . . . . . . . . .   L CSEG   FF61H  2573 2574 2577#
DD_CASE_8 . . . . . . . . . . .   L CSEG   FF69H  2572 2579#
DD_CASE_8_MSG . . . . . . . . .   L CSEG   FF76H  2581 2582 2585#
DD_CASE_END . . . . . . . . . .   L CSEG   FF7CH  2476 2505 2516 2519 2576 2580 2584 2587#
DD_CASE_EXP16 . . . . . . . . .   L CSEG   FED7H  2501# 2530 2570
DD_CASE_EXP8. . . . . . . . . .   L CSEG   FE9AH  2468# 2479 2484 2489
DD0 . . . . . . . . . . . . . .   L CSEG   FE66H  2441 2443 2445#
DD0_1 . . . . . . . . . . . . .   L CSEG   FE70H  2446 2449#
DD1 . . . . . . . . . . . . . .   L CSEG   FE76H  2449 2451#
DD1_1 . . . . . . . . . . . . .   L CSEG   FE78H  2450 2452#
DD2 . . . . . . . . . . . . . .   L CSEG   FE92H  2451 2464#
DD3 . . . . . . . . . . . . . .   L CSEG   FE7BH  2452 2453#
DD4 . . . . . . . . . . . . . .   L CSEG   FE8BH  2453 2456 2460#
DD4_1 . . . . . . . . . . . . .   L CSEG   FE8EH  2448 2459 2462#
DD5 . . . . . . . . . . . . . .   L CSEG   FE83H  2454 2456#
DD5_5 . . . . . . . . . . . . .   L CSEG   FE85H  2455 2457#
DDTBL . . . . . . . . . . . . .   L CSEG   FE5CH  2438 2440#
DISASSEMBLE . . . . . . . . . .   L CSEG   FF84H  2215 2633#
DISASSEMBLY_CMD . . . . . . . .   L CSEG   FCFDH  38 2180#
DISCASE_1 . . . . . . . . . . .   L CSEG   FFA8H  2649 2651#
DISCASE_2 . . . . . . . . . . .   L CSEG   FFADH  2652 2654#
DISCASE_2_1 . . . . . . . . . .   L CSEG   FFBCH  2657 2661#
DISCASE_3 . . . . . . . . . . .   L CSEG   FFC2H  2655 2665#
DISCASE_4 . . . . . . . . . . .   L CSEG   FFCBH  2666 2670#
DISCASE_5 . . . . . . . . . . .   L CSEG   FFD8H  2671 2677#
DISCASE_6 . . . . . . . . . . .   L CSEG   FFDDH  2678 2680#
DISCASE_7 . . . . . . . . . . .   L CSEG   FFE6H  2681 2685#
DISCASE_8 . . . . . . . . . . .   L CSEG   FFEBH  2686 2688#
DISCASE_END . . . . . . . . . .   L CSEG   FFF0H  2689 2691#
DISPLAY_COMMA . . . . . . . . .   L CSEG   FF7FH  2609# 2663 2668 2673 2675 2683
DISPLAY_OPERAND . . . . . . . .   L CSEG   FE53H  2433# 2653 2662 2664 2667 2669 2672 2674 2676 2679 2682 2684 2687 2690
DISPLAY_TOKEN . . . . . . . . .   N        E059H  136# 2463 2637
DIVIDE_1. . . . . . . . . . . .   L CSEG   F9BCH  1402# 1417 1429
DIVIDE_2. . . . . . . . . . . .   L CSEG   F9BDH  1404# 1451
DIVIDE_3. . . . . . . . . . . .   L CSEG   F9CFH  1413 1415#
DIVIDE_4. . . . . . . . . . . .   L CSEG   F9D8H  1421 1423#
DIVIDEND_HIGH . . . . . . . . .   L DSEG   006EH  433# 1405 1410 1426 1428 1448 1450 2276 2305
DIVIDEND_LOW. . . . . . . . . .   L DSEG   006FH  434# 1444 1447 2277 2306
DIVISOR . . . . . . . . . . . .   L DSEG   0070H  435# 1398 1408 1427 2275 2304
DPH . . . . . . . . . . . . . .   N DSEG   0083H PREDEFINED  1495 2267
DPL . . . . . . . . . . . . . .   N DSEG   0082H PREDEFINED  1497
DPTR_OP1. . . . . . . . . . . .   N        023CH  375# 638 661
DPTR_TOKE . . . . . . . . . . .   N        00A1H  53# 836 843 881 946
DS0 . . . . . . . . . . . . . .   L CSEG   FD06H  2184# 2230
DS1 . . . . . . . . . . . . . .   L CSEG   FD21H  2196 2197#
DS2 . . . . . . . . . . . . . .   L CSEG   FD40H  2198 2214#
DS3 . . . . . . . . . . . . . .   L CSEG   FD2FH  2203 2205#
DS4 . . . . . . . . . . . . . .   L CSEG   FD1DH  2194# 2213
DS5 . . . . . . . . . . . . . .   L CSEG   FD50H  2219 2221#
DSRET . . . . . . . . . . . . .   L CSEG   FD60H  2228 2231#
END_FIRST_OPERAND . . . . . . .   L CSEG   FAA7H  1616 1618#
END_SECOND_OPERAND. . . . . . .   L CSEG   FB6CH  1835 1837 1839#
END_SELECT_INSTRUCTION_TAIL . .   L CSEG   F968H  1209 1211 1214#
```

```
NAME                           TYPE      VALUE AND REFERENCES

EOL_CHECK . . . . . . . . . .  N         E06EH  143# 2183
EOL_TOKE. . . . . . . . . . .  N         0007H  49# 1255 1263
ERRNUM. . . . . . . . . . . .  L DSEG    0043H  74# 901 912 990 1149 1265 1499 1879 1980 2037 2108 2121 2260
ERROR . . . . . . . . . . . .  N         E05FH  138# 902 999 1266 1500 1880 2067 2261
EXP_FLAG_TABLE. . . . . . . .  L CSEG    FAD8H  1726 1729#
EXP11_OP1 . . . . . . . . . .  N         039CH  381# 459 479 499 519 539 559 579 599 619 639 659 679 699 719 739 759
EXP16_OP1 . . . . . . . . . .  N         0370H  380# 460 480
EXP16_OP2 . . . . . . . . . .  N         5280H  404# 638
EXPRESSIONS_PRINTED . . . . .  L DSEG    0068H  427# 2470 2475 2540 2569 2645
FETCH . . . . . . . . . . . .  N         E04AH  131# 2206
FIRST_EXP . . . . . . . . . .  L BSEG    0003H  445# 1192 1703 1716 1721
FIRST_NOT_REGISTER. . . . . .  L CSEG    FA88H  1582 1584 1598#
FIRST_OPER_ORDINAL. . . . . .  L DSEG    0063H  422# 2323 2329 2361 2440
FJC_5_CONTINUE. . . . . . . .  L CSEG    FC65H  2063 2068#
FORWARD_JUMP_CASE_4 . . . . .  L CSEG    FBE8H  1990 1996#
FORWARD_JUMP_CASE_5 . . . . .  L CSEG    FC4FH  2049 2055#
GE_FI_OP_1. . . . . . . . . .  L CSEG    FA7FH  1591 1593#
GE_FI_OP_2. . . . . . . . . .  L CSEG    FA9AH  1608 1610#
GET_COMMA . . . . . . . . . .  N         E06BH  142# 884 896 941 984 994 1066
GET_FIRST_OPERAND . . . . . .  L CSEG    FA63H  841 891 940 983 1056 1577#
GET_HASH_VALUE. . . . . . . .  L CSEG    FD63H  2214 2257#
GET_PART. . . . . . . . . . .  N         E065H  140# 2182
GET_SECOND_OPERAND. . . . . .  L CSEG    FB0DH  899 1786#
GETEOL. . . . . . . . . . . .  N         E053H  134# 1268
GETNUM. . . . . . . . . . . .  N         E050H  133# 887 906 910 988 995 1024 1067 1092 1121 1250
GETOKE. . . . . . . . . . . .  N         E056H  135# 831 833 835 880 885 897 938 942 944 945 981 985 1054 1248 1254 1262
                                           2181
GHV_A1. . . . . . . . . . . .  L CSEG    FD77H  2266 2268#
GHV1. . . . . . . . . . . . .  L CSEG    FDBFH  2298 2300 2303#
GHV10 . . . . . . . . . . . .  L CSEG    FE25H  2352 2356#
GHV11 . . . . . . . . . . . .  L CSEG    FE28H  2355 2358#
GHV2. . . . . . . . . . . . .  L CSEG    FDEFH  2324 2326#
GHV2_2. . . . . . . . . . . .  L CSEG    FDF1H  2325 2328#
GHV3. . . . . . . . . . . . .  L CSEG    FDF3H  2327 2330#
GHV5. . . . . . . . . . . . .  L CSEG    FE05H  2334 2336 2339#
GHV6. . . . . . . . . . . . .  L CSEG    FE10H  2342 2344#
GHV6_6. . . . . . . . . . . .  L CSEG    FE12H  2343 2346#
GHV7. . . . . . . . . . . . .  L CSEG    FE14H  2345 2348#
GHV9. . . . . . . . . . . . .  L CSEG    FE17H  2302 2338 2350#
GHVTBL. . . . . . . . . . . .  L CSEG    FE32H  2359 2364#
HASH_CONTINUE . . . . . . . .  L CSEG    FD6EH  2259 2262#
INST_VALUE_LOOP . . . . . . .  L CSEG    FA06H  1480# 1496 1498
INSTRUCTION . . . . . . . . .  L DSEG    005FH  418# 1479 1483 1490 1952 1958 1966 1967 2033 2041 2115 2126
INSTRUCTION_CODE. . . . . . .  L CSEG    F5B3H  456# 1478 1496 1498 2263
INSTRUCTION_VALUE . . . . . .  L DSEG    005BH  414# 1151 1202 2641 2646
JT00. . . . . . . . . . . . .  L CSEG    F8E1H  1055 1060#
JTRET . . . . . . . . . . . .  L CSEG    F8F0H  1059 1065#
JUMP_ABSOLUTE_OPERAND . . . .  L CSEG    F8F9H  1091# 1165
JUMP_END. . . . . . . . . . .  N         0016H  450# 1203
JUMP_LONG_OPERAND . . . . . .  L CSEG    F908H  1120# 1166
JUMP_OPERAND. . . . . . . . .  L CSEG    F8C6H  1023# 1163
JUMP_TWO_OPERANDS . . . . . .  L CSEG    F8CFH  1053# 1164
LINBUF. . . . . . . . . . . .  L DSEG    0024H  68# 1256
LINCNT. . . . . . . . . . . .  L DSEG    0053H  90#
LINE_START. . . . . . . . . .  L DSEG    0052H  89# 1247
```

```
NAME                            TYPE      VALUE AND REFERENCES

LINMAX. . . . . . . . . . . . . N         0018H  58# 68
LNLGTH. . . . . . . . . . . . . L DSEG    0054H  91#
LSTBYT. . . . . . . . . . . . . N         E015H  127# 2474 2518
LSTFLG. . . . . . . . . . . . . L BSEG    0001H  104# 443
LSTWRD. . . . . . . . . . . . . N         E018H  128# 2188 2504 2515
MEMORY_TRACE_ADDR_HIGH. . . . . L DSEG    0069H  428# 2192 2537 2554 2563
MEMORY_TRACE_ADDR_LOW . . . . . L DSEG    006AH  429# 2191 2533 2535 2550 2561
MF00. . . . . . . . . . . . . . L CSEG    F7D3H  832 840#
MF01. . . . . . . . . . . . . . L CSEG    F7DBH  840 843#
MF02. . . . . . . . . . . . . . L CSEG    F7EBH  843 849#
MFT00 . . . . . . . . . . . . . L CSEG    F81BH  890 893#
MI0 . . . . . . . . . . . . . . L CSEG    F970H  1216 1218#
MI1 . . . . . . . . . . . . . . L CSEG    F972H  1217 1219#
MIT_JMP_TBL . . . . . . . . . . L CSEG    F921H  1150 1157#
MNE_ACALL . . . . . . . . . . . N         0002H  317# 479 519 559 599 639 679 719 759
MNE_ADD . . . . . . . . . . . . N         0014H  335# 503 504 505 506 508 509 510 511 513 514 515 516
MNE_ADDC. . . . . . . . . . . . N         0013H  334# 523 524 525 526 528 529 530 531 533 534 535 536
MNE_AJMP. . . . . . . . . . . . N         0003H  318# 459 499 539 579 619 659 699 739
MNE_ANL . . . . . . . . . . . . N         0011H  332# 560 561 563 564 565 566 568 569 570 571 573 574 575 576 620 678
MNE_CJNE. . . . . . . . . . . . N         0009H  324# 683 684 685 686 688 689 690 691 693 694 695 696
MNE_CLR . . . . . . . . . . . . N         001AH  341# 700 701 743
MNE_CPL . . . . . . . . . . . . N         001BH  342# 680 681 763
MNE_DA. . . . . . . . . . . . . N         001CH  343# 723
MNE_DEC . . . . . . . . . . . . N         0025H  352# 483 484 485 486 488 489 490 491 493 494 495 496
MNE_DIV . . . . . . . . . . . . N         0021H  348# 623
MNE_DJNZ. . . . . . . . . . . . N         0015H  336# 724 728 729 730 731 733 734 735 736
MNE_INC . . . . . . . . . . . . N         0027H  354# 463 464 465 466 468 469 470 471 473 474 475 476 661
MNE_JB. . . . . . . . . . . . . N         0017H  338# 498
MNE_JBC . . . . . . . . . . . . N         0018H  339# 478
MNE_JC. . . . . . . . . . . . . N         0008H  323# 538
MNE_JMP . . . . . . . . . . . . N         0022H  349# 601
MNE_JNB . . . . . . . . . . . . N         0016H  337# 518
MNE_JNC . . . . . . . . . . . . N         0007H  322# 558
MNE_JNZ . . . . . . . . . . . . N         0005H  320# 598
MNE_JZ. . . . . . . . . . . . . N         0006H  321# 578
MNE_LCALL . . . . . . . . . . . N         0000H  315# 480
MNE_LJMP. . . . . . . . . . . . N         0001H  316# 460
MNE_MOV . . . . . . . . . . . . N         000FH  330# 603 604 605 606 608 609 610 611 613 614 615 616 624 625 626 628 629
                                                   630 631 633 634 635 636 638 640 660 665 666 668 669 670 671 673 674 675 676
                                                   744 745 746 748 749 750 751 753 754 755 756 764 765 766 768 769 770 771 773
                                                   774 775 776
MNE_MOVC. . . . . . . . . . . . N         000AH  325# 621 641
MNE_MOVX. . . . . . . . . . . . N         000BH  326# 738 740 741 758 760 761
MNE_MUL . . . . . . . . . . . . N         0020H  347# 663
MNE_NOP . . . . . . . . . . . . N         002BH  358# 458
MNE_ORL . . . . . . . . . . . . N         0012H  333# 540 541 543 544 545 546 548 549 550 551 553 554 555 556 600 658
MNE_POP . . . . . . . . . . . . N         001DH  344# 718
MNE_PUSH. . . . . . . . . . . . N         001FH  346# 698
MNE_RET . . . . . . . . . . . . N         002AH  357# 500
MNE_RETI. . . . . . . . . . . . N         0029H  356# 520
MNE_RL. . . . . . . . . . . . . N         0024H  351# 501
MNE_RLC . . . . . . . . . . . . N         0023H  350# 521
MNE_RR. . . . . . . . . . . . . N         0028H  355# 461
MNE_RRC . . . . . . . . . . . . N         0026H  353# 481
MNE_SETB. . . . . . . . . . . . N         0019H  340# 720 721
```

```
NAME                             TYPE      VALUE AND REFERENCES

MNE_SJMP. . . . . . . . . . . .  N         0004H  319# 618
MNE_SUBB. . . . . . . . . . . .  N         000EH  329# 643 644 645 646 648 649 650 651 653 654 655 656
MNE_SWAP. . . . . . . . . . . .  N         001EH  345# 703
MNE_UNDEF . . . . . . . . . . .  N         FFFFH  263# 664
MNE_XCH . . . . . . . . . . . .  N         000DH  328# 704 705 706 708 709 710 711 713 714 715 716
MNE_XCHD. . . . . . . . . . . .  N         000CH  327# 725 726
MNE_XRL . . . . . . . . . . . .  N         0010H  331# 580 581 583 584 585 586 588 589 590 591 593 594 595 596
MNEMONIC_FACTOR . . . . . . . .  N         002CH  452# 1524 1537 1578 1787 2275 2282 2287
MNEMONIC_FIRST_OPERAND. . . . .  L CSEG    F7B9H  830# 1159
MNEMONIC_INSTR_LIST_TAIL. . . .  L CSEG    F93CH  1190# 1267
MNEMONIC_INSTRUCTION_TAIL . . .  L CSEG    F911H  1148# 1219
MNEMONIC_ORDINAL. . . . . . . .  L DSEG    006DH  432# 2294 2351 2497 2634 2639
MNEMONIC_SECOND_OPERAND_TAIL.    L CSEG    F7B3H  802# 900 907 911 915
MNEMONIC_TAB. . . . . . . . . .  L CSEG    F587H  265# 1200 2638
MNEMONIC_TWO_OPERANDS . . . . .  L CSEG    F7F4H  879# 1160
MOVC_OPERANDS . . . . . . . . .  L CSEG    F85EH  937# 1161
MS00. . . . . . . . . . . . . .  L CSEG    F838H  898 903#
MS01. . . . . . . . . . . . . .  L CSEG    F846H  904 908#
MS02. . . . . . . . . . . . . .  L CSEG    F852H  908 912#
MT0 . . . . . . . . . . . . . .  L CSEG    F882H  946 950#
MT00. . . . . . . . . . . . . .  L CSEG    F812H  881 890#
MT01. . . . . . . . . . . . . .  L CSEG    F824H  892 896#
NEWLINE . . . . . . . . . . . .  N         E00FH  125# 1253 2185
NO_BIT_8. . . . . . . . . . . .  L CSEG    FAB0H  1699 1701#
NO_OF_OPERANDS_PRINTED. . . . .  L DSEG    0067H  426# 2434 2588 2644
NUMBER_OF_BYTES . . . . . . . .  L DSEG    004DH  84# 847 997 1064 1095 1194 1704 1710 1730 1732 1734 1736 1763 1873 1948
                                                    1981 2038 2101 2123 2148 2217 2353 2357 2404 2407 2534
NUMBER_OF_OPERANDS. . . . . . .  L DSEG    006BH  430# 2301 2337 2349 2360
NUMBER_TOKE . . . . . . . . . .  N         0001H  45# 849 893 913 991 1060
OBC0. . . . . . . . . . . . . .  L CSEG    FE41H  2396 2397#
OBC1. . . . . . . . . . . . . .  L CSEG    FE4DH  2398 2402 2405#
OBC2. . . . . . . . . . . . . .  L CSEG    FE49H  2399 2401#
OBC2_2. . . . . . . . . . . . .  L CSEG    FE4BH  2400 2403#
OBCRET. . . . . . . . . . . . .  L CSEG    FE52H  2406 2408#
OFST. . . . . . . . . . . . . .  N         0010H  57# 1198 1599 1813 2461 2635
OLD_ASM_PC_HIGH . . . . . . . .  L DSEG    005DH  416# 1946 2065 2134 2136
OLD_ASM_PC_LOW. . . . . . . . .  L DSEG    005EH  417# 1947 2066 2133 2135
ONE_BYTE_TAIL . . . . . . . . .  L CSEG    F7B3H  801# 839 842 949 953 1158
OPERAND_BYTE_CHECK. . . . . . .  L CSEG    FE3CH  2367 2368 2394#
OPERAND_C . . . . . . . . . . .  L CSEG    FB5FH  1811 1833#
OPERAND_CHECK . . . . . . . . .  L DSEG    006CH  431# 2361 2369 2395
OPERAND_FACTOR. . . . . . . . .  N         0018H  453# 1541 1549 1799 1807 1821 1829 2304 2311 2316
ORDINAL . . . . . . . . . . . .  L DSEG    005CH  415# 837 844 882 947 951 1093 1525 1538 1698 1700 1708 1714 1740 1751
                                                    1755 1761
ORG_TOKE. . . . . . . . . . . .  N         00D4H  54# 1249
OUR_CODE_HIGH . . . . . . . . .  L DSEG    004EH  85# 1195 1485 1522 1529 1532 1546 1547 1551 1554 1592 1595 1596 1609
                                                    1612 1613 1804 1805 1809 1810 1826 1827 1831 1832 1834 2270 2276 2295 2299
                                                    2305 2331 2335
OUR_CODE_LOW. . . . . . . . . .  L DSEG    004FH  86# 1199 1204 1207 1215 1489 1526 1530 1533 1543 1544 1589 1590 1606
                                                    1607 1801 1802 1823 1824 1836 1838 2274 2277 2291 2296 2297 2306 2320 2332
                                                    2333 2340
OUR_GTRTHN. . . . . . . . . . .  L CSEG    F966H  1204 1212#
PARAM1. . . . . . . . . . . . .  N REG     R2     109# 1257 1259 1875 1883 2140 2186 2189 2462 2473 2482 2487 2499 2502
                                                    2513 2517 2573 2581 2610 2636 2642
PARAM2. . . . . . . . . . . . .  N REG     R3     110# 2187 2503 2514 2574 2582
```

```
NAME                            TYPE      VALUE AND REFERENCES

PARAM3. . . . . . . . . . . . . N REG     R4      111#
PARAM4. . . . . . . . . . . . . N REG     R5      112# 1401 1414 1422 1432 1440 1442
PARAM5. . . . . . . . . . . . . N REG     R6      113# 1400 1411 1416 1419 1424 1433 1437 1439 2279 2296 2308 2332
PARAM6. . . . . . . . . . . . . N REG     R7      114# 1399 1431 1540 1548 1601 1604 1798 1806 1815 1818 1820 1828 2280
                                            2286 2295 2309 2315 2331
PARTIT_HI_HIGH. . . . . . . . . L DSEG    0059H   96# 2225
PARTIT_HI_LOW . . . . . . . . . L DSEG    005AH   97# 413 2222
PARTIT_LO_HIGH. . . . . . . . . L DSEG    0057H   94# 2186 2192 2202 2220 2226
PARTIT_LO_LOW . . . . . . . . . L DSEG    0058H   95# 2187 2191 2199 2216 2218 2223
PC_TOKE . . . . . . . . . . . . N         00A0H   55# 950
PLUS_TOKE . . . . . . . . . . . N         0005H   48# 834
PNTHGH. . . . . . . . . . . . . L DSEG    0044H   75# 2134 2146 2151 2202 2204
PNTLOW. . . . . . . . . . . . . L DSEG    0045H   76# 2133 2143 2144 2150 2201
POINT0. . . . . . . . . . . . . N REG     R0      107# 1256 1261 1951 1952 1957 1958 1959 1960 1965 1966 1973 1974 1975
                                            1976 2008 2009 2025 2029 2032 2033 2040 2041 2073 2074 2090 2094 2096 2097
                                            2110 2116 2117 2118 2125 2126 2127 2128 2129 2130 2137 2139 2142 2210 2211
                                            2471 2472 2492 2493 2494 2495 2496 2522 2523 2528 2529 2541 2542
POINT1. . . . . . . . . . . . . N REG     R1      108# 2193 2195 2200 2209 2212
POUND_EXP_OP2 . . . . . . . . . N         4A40H   402# 503 523 541 543 561 563 581 583 603 604 605 606 608 609 610 611 613
                                            614 615 616 643 683 685 686 688 689 690 691 693 694 695 696
POUND_TOKE. . . . . . . . . . . N         0006H   47# 886 904 986
PRINT_STRING. . . . . . . . . . N         E01EH   129# 2575 2583
QUOTIENT_HIGH . . . . . . . . . L DSEG    0071H   436# 1432 2280 2285 2289 2290 2309 2314 2318 2319
QUOTIENT_LOW. . . . . . . . . . L DSEG    0072H   437# 1433 2279 2281 2284 2293 2308 2310 2313 2322
R0_OP1. . . . . . . . . . . . . N         00B0H   366# 468 488 608 668 688 728 768
R0_OP2. . . . . . . . . . . . . N         1080H   390# 508 528 548 568 588 628 648 708 748
R1_OP1. . . . . . . . . . . . . N         00DCH   367# 469 489 609 669 689 729 769
R1_OP2. . . . . . . . . . . . . N         14A0H   391# 509 529 549 569 589 629 649 709 749
R2_OP1. . . . . . . . . . . . . N         0108H   368# 470 490 610 670 690 730 770
R2_OP2. . . . . . . . . . . . . N         18C0H   392# 510 530 550 570 590 630 650 710 750
R3_OP1. . . . . . . . . . . . . N         0134H   369# 471 491 611 671 691 731 771
R3_OP2. . . . . . . . . . . . . N         1CE0H   393# 511 531 551 571 591 631 651 711 751
R4_OP1. . . . . . . . . . . . . N         0160H   370# 473 493 613 673 693 733 773
R4_OP2. . . . . . . . . . . . . N         2100H   394# 513 533 553 573 593 633 653 713 753
R5_OP1. . . . . . . . . . . . . N         018CH   371# 474 494 614 674 694 734 774
R5_OP2. . . . . . . . . . . . . N         2520H   395# 514 534 554 574 594 634 654 714 754
R6_OP1. . . . . . . . . . . . . N         01B8H   372# 475 495 615 675 695 735 775
R6_OP2. . . . . . . . . . . . . N         2940H   396# 515 535 555 575 595 635 655 715 755
R7_OP1. . . . . . . . . . . . . N         01E4H   373# 476 496 616 676 696 736 776
R7_OP2. . . . . . . . . . . . . N         2D60H   397# 516 536 556 576 596 636 656 716 756
REG . . . . . . . . . . . . . . N         0040H   56# 1600 1814 2461
REL_OFFSET_HIGH . . . . . . . . L DSEG    0060H   419# 1984 1985 2001 2018 2019 2043 2044 2060 2083 2084
REL_OFFSET_LOW. . . . . . . . . L DSEG    0061H   420# 1983 1992 1998 2000 2005 2009 2015 2016 2022 2026 2042 2051 2057
                                            2059 2070 2074 2080 2081 2087 2091
REL8_OP1. . . . . . . . . . . . N         03C8H   382# 538 558 578 598 618
REL8_OP2. . . . . . . . . . . . N         5AC0H   405# 478 498 518 724 728 729 730 731 733 734 735 736
ROTATE. . . . . . . . . . . . . L CSEG    F9E4H   1407 1409 1431#
ROTATE_CONTINUE . . . . . . . . L CSEG    F9EBH   1431 1435#
SAVE_AND_DISPLAY. . . . . . . . N         E05CH   137# 1258 1260
SECOND_EXP. . . . . . . . . . . L BSEG    0004H   446# 1193 1723 1742 1746 1757
SECOND_NO_BIT_8 . . . . . . . . L CSEG    FAF8H   1748 1750#
SECOND_NOT_REGISTER . . . . . . L CSEG    FB3CH   1791 1793 1812#
SECOND_OPER_ORDINAL . . . . . . L DSEG    0064H   423# 2340 2341 2347 2369 2442
SELECT. . . . . . . . . . . . . L DSEG    0046H   77# 2132 2205
SELECT_CON. . . . . . . . . . . N         0000H   61#
```

```
NAME                             TYPE      VALUE AND REFERENCES

SET_BIT_EXP . . . . . . . . . .  L CSEG    FAA0H   1597 1614#
SET_EXP_16_FLAG . . . . . . . .  L CSEG    FAB8H   888 1122 1707#
SET_EXP_FLAG. . . . . . . . . .  L CSEG    FAC1H   894 1713#
SET_POUND_EXP_FLAG. . . . . . .  L CSEG    FAE8H   905 987 1739#
SET_REL_FLAG. . . . . . . . . .  L CSEG    FB04H   1025 1057 1062 1760#
SET_SLASH_EXP_FLAG. . . . . . .  L CSEG    FAFCH   909 1754#
SLASH_EXP_OP2 . . . . . . . . .  N         4E60H   403# 658 678
SS0 . . . . . . . . . . . . . .  L CSEG    FEE9H   2508 2510#
SS1 . . . . . . . . . . . . . .  L CSEG    FEEFH   2510 2512#
SS2 . . . . . . . . . . . . . .  L CSEG    FEFCH   2512 2517#
SS3 . . . . . . . . . . . . . .  L CSEG    FEF2H   2509 2511 2513#
START_DIVIDE. . . . . . . . . .  L CSEG    F9B4H   1397# 2278 2307
ST01. . . . . . . . . . . . . .  L CSEG    F8A9H   986 990#
STORE . . . . . . . . . . . . .  N         E04DH   132# 2141
STORET. . . . . . . . . . . . .  L CSEG    F8B2H   989 993#
STRGBF. . . . . . . . . . . . .  L DSEG    003CH   69#
STRGCT. . . . . . . . . . . . .  L DSEG    0055H   92#
SUBTRACT_WITH_C . . . . . . . .  L CSEG    F9D2H   1406 1418# 1430
TEMP. . . . . . . . . . . . . .  N REG     R5      117#
TEMP_LOW. . . . . . . . . . . .  L DSEG    0047H   78# 2105 2109 2207 2211 2268 2271 2272 2549 2552
TEMP_SEC. . . . . . . . . . . .  L DSEG    0062H   421# 895 993 1065 1969 1971 1974 2097
TEMPT . . . . . . . . . . . . .  L DSEG    0056H   93#
THIRD_OPER_ORDINAL. . . . . . .  L DSEG    0065H   424# 2354 2444
THREE_OPERANDS. . . . . . . . .  L CSEG    F88EH   980# 1162
TIME. . . . . . . . . . . . . .  N         E012H   126#
TOERR . . . . . . . . . . . . .  L CSEG    F8C2H   913 991 999# 1060
TOKSIZ. . . . . . . . . . . . .  N         0004H   59# 69
TOKSTR. . . . . . . . . . . . .  L DSEG    0048H   79# 903 1197 1579 1585 1602 1615 1788 1794 1816
UL0 . . . . . . . . . . . . . .  L CSEG    FA43H   1523 1537#
UL1 . . . . . . . . . . . . . .  L CSEG    FA3EH   1531 1533#
UNDEFINED_OPCODE. . . . . . . .  N         00A5H   454# 2259
UPDATE_LSSTHN . . . . . . . . .  L CSEG    FA41H   1526 1535#
UPDATE_OUR_CODE . . . . . . . .  L CSEG    FA28H   838 845 883 948 952 1094 1521# 1702 1709 1715 1741 1752 1756 1762
VALHGH. . . . . . . . . . . . .  L DSEG    0049H   80# 1251 1984 2043 2103 2111 2128 2494 2502 2513 2527 2558 2567
VALLOW. . . . . . . . . . . . .  L DSEG    004AH   81# 895 993 1065 1252 1960 1970 1976 1983 2042 2118 2130 2496 2503 2514
                                                     2517 2529 2553 2562
WAIT_FOR_USER . . . . . . . . .  N         E062H   139# 2231
WORKING_SPACE . . . . . . . . .  L DSEG    0040H   70# 1951 1957 1965 1973 2008 2025 2032 2040 2073 2090 2096 2110 2125
                                                     2137 2208 2258 2469 2492 2522 2539 2656 2658 2659 2659 2660

ASSEMBLY COMPLETE, NO ERRORS FOUND
```

INTEL CORPORATION, 3065 Bowers Avenue, Santa Clara, California 95051 (408) **987-8080**

Printed in U.S.A.