SDK-51

MCS-51™ SYSTEM DESIGN KIT

MONITOR LISTING MANUAL

Manual Order No: 121590-001

SDK-51

MCS-51™ SYSTEM DESIGN KIT

MONITOR LISTING MANUAL

Manual Order No: 121590-001

Intel Corporation, 3065 Bowers Ave., Santa Clara CA 95051

REV.	REVISION HISTORY	PRINT DATE
-01	Original Issue	5/81

Additional copies of this manual or other Intel literature may be obtained from:

Literature Department
Intel Corporation
3065 Bowers Avenue
Santa Clara, CA 95051

The information in this document is subject to change without notice.

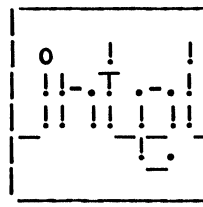
Intel Corporation makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Intel Corporation assumes no responsibility for any errors that may appear in this document. Intel Corporation makes no commitment to update nor to keep current the information contained in this document.

Intel Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in an Intel product. No other circuit patent licenses are implied.

The following are trademarks of Intel Corporation and its affiliates and may be used only to identify Intel products:

BXP
CREDIT
i
ICE
iCS
im
Insite
Intel
Intel
Intelevision
Intellec
iRMX
iSBC
iSBX
Library Manager
MCS
Megachassis
Micromap
Multibus
Multimodule
PROMPT
Promware
RMX/80
System 2000
UPI
μScope

and the combination of ICE, iCS, iRMX, iSBC, iSBX, MCS, or RMX and a numerical suffix.



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™ System Design Kit Assembly Manual, manual order number 121589.

SDK-51 MCS-51™ 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(5,18,81) WORKFILES(:F3
                       :,:F3:) EP DB SB


LOC  OBJ            LINE     SOURCE

                       1     $XREF
                       2     $nomacro
                       3     $TITLE('SDK-51 MONITOR CODE   INTEL PROPRIETARY VERS. #1.0')
                       4     ;*********************************************************************
                       5     ;
                       6     ;
                       7     ;       SDK-51 MONITOR  INTEL PROPRIETARY
                       8     ;       THIS SOFTWARE IS COPYRIGHTED UNDER INTEL PART NUMBER 162787-001
                       9     ;
                      10     ;       VERSION 1.0           5-18-81;
                      11     ;
                      12     ;
                      13     ;       NN  N  00000  TTTTT  EEEEE    !!
                      14     ;       NN  N  0   0    T    E        !!
                      15     ;       N N N  0   0    T    E        !!
                      16     ;       N N N  0   0    T    EEEE     !!
                      17     ;       N  NN  0   0    T    E        !!
                      18     ;       N  NN  0   0    T    E        !!
                      19     ;       N   N  00000    T    EEEEE    !!
                      20     ;
                      21     ;
                      22     ;*********************************************************************
                      23     ;
                      24     ;
                      25     ;       COPYRIGHT (C) 1981 INTEL CORPORATION.
                      26     ;
                      27     ;       ALL RIGHTS RESERVED.
                      28     ;
                      29     ;       NO PART OF THIS PROGRAM OR PUBLICATION MAY BE REPRODUCED,
                      30     ;       TRANSMITTED, TRANSCRIBED, STORED IN A RETRIEVAL SYSTEM, OR
                      31     ;       TRANSLATED INTO ANY LANGUAGE OR COMPUTER LANGUAGE, IN ANY
                      32     ;       FORM OR BY ANY MEANS, ELECTRONIC, MECHANICAL, MAGNETIC,
                      33     ;       OPTICAL, CHEMICAL, MANUAL OR OTHERWISE, WITHOUT THE PRIOR
                      34     ;       WRITTEN PERMISSION OF INTEL CORPORATION, 3065 BOWERS AVENUE,
                      35     ;       SANTA CLARA, CALIFORNIA 95051.
                      36     ;
                      37     ;
                      38     ;
                      39     ;*********************************************************************
                      40 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                       41    ;*****************************************************************
                       42    ;
                       43    ;               TABLE OF CONTENTS
                       44    ;
                       45    ;       PREFACE:  HOW TO USE THIS LISTING
                       46    ;
                       47    ;       This monitor and the assembler/disassembler are written
                       48    ;       in ASM51 code.  These listings may serve the user as
                       49    ;       both debug aids and as an example of how many of the unique
                       50    ;       ASM51 commands may be used in context.
                       51    ;
                       52    ;       In general, the organization on this monitor listing is as
                       53    ;       follows.  The POWER_ON routine is the 'cold start' location,
                       54    ;       that is, it does a hardware reset.  START is the main program
                       55    ;       which is the top of the idle loop.  It is also the 'warm start'
                       56    ;       location, that is it does software resets and initializations.
                       57    ;
                       58    ;       Upon receipt of a command from the user via the console, START
                       59    ;       determines which routine will handle each command and branches
                       60    ;       to it.  The command handler routines will always have a label
                       61    ;       with the suffix '_CMD'.
                       62    ;
                       63    ;       HEADER BLOCK INFORMATION:
                       64    ;
                       65    ;       At the beginning of each subroutine, on a new page, there will
                       66    ;       be a block containing the name of the routine.  The name may
                       67    ;       have an '(I)' or a '(U)' as a prefix.  The I indicates that
                       68    ;       the routine is internal only, the U indicates that the routine is
                       69    ;       only suitable for use by the user.
                       70    ;
                       71    ;       The abstract contains a brief description of what the function
                       72    ;       of that module is and highlights of any subtle cautions or user
                       73    ;       interface notes.  There will also be lists of inputs, outputs,
                       74    ;       error exits, variables modified and subroutines called.  The
                       75    ;       rules for these lists are strict.
                       76    ;
                       77    ;       Input lists contain only explicitly passed global or local variables.
                       78    ;       Information returned by any other procedure (i.e. passed parameters)
                       79    ;       that is called by the procedure whose block you are reading will not
                       80    ;       be included in the input list.
                       81    ;
                       82    ;       Output lists contain only variables altered by the procedure for
                       83    ;       the purpose of transmitting necessary information to another procedure.
                       84    ;
                       85    ;       The variables modified lists contains only local variables, registers
                       86    ;       or memory locations that are modified and not restored by the end on
                       87    ;       the routine.
                       88    ;
                       89    ;       The error exits will contain any error number that is locally
                       90    ;       generated.  There is the possibility that an error may be detected
                       91    ;       in a routine with no error exits noted if the error number was set
                       92    ;       in a previous routine and just 'falls through' because the error is
                       93    ;       still the same.
                       94    ;
                       95    ;       The subroutines called list will contain any other routine that is
```

```
LOC  OBJ            LINE     SOURCE

                      96     ;       directly called or jumped to by the procedure in question.
                      97     ;
                      98     ;       XREF:
                      99     ;
                     100     ;       At the back of the monitor listing and again at the back of the
                     101     ;       assembler/disassembler listing there is a table of cross references.
                     102     ;       Each variable name is listed in alphabetical order along with its
                     103     ;       type (that is in what type of memory does it reside, is it a label
                     104     ;       or a number), the address value it has and all of the line numbers
                     105     ;       where that variable name appears.  The line number with the '#'
                     106     ;       designation is the line where the variable is defined.
                     107     ;
                     108     ;
                     109     ;       CONTENTS:
                     110     ;
                     111     ;       This monitor listing contains one source file and five
                     112     ;       include files.  Each include file contains a number of functions,
                     113     ;       tables and subroutines which will each have their own header block
                     114     ;       and will begin on a new page.  The files are as follows:
                     115     ;
                     116     ;       SDKMON.SRC (SOURCE FILE)
                     117     ;
                     118     ;                       JUMP TABLE FOR USER ACCESSABLE ROUTINES
                     119     ;                       CONSTANTS
                     120     ;                       VARIABLES
                     121     ;                       FLAGS
                     122     ;                       TOKEN EQUATES
                     123     ;                       TOKEN TABLE
                     124     ;
                     125     ;                       POWER_ON
                     126     ;                       SIGN_ON
                     127     ;                       START
                     128     ;                       INIT_IO
                     129     ;                       (I)WAIT_FOR_USER
                     130     ;                       CHECK_EPROMS
                     131     ;
                     132     ;               COMMON.INC (INCLUDE FILE)
                     133     ;
                     134     ;                       CONSTANTS USED BY ALL MODULES
                     135     ;                       GLOBAL VARIABLES USED BY MORE THAT ONE MAIN MOD.
                     136     ;                       ARRAYS
                     137     ;                       VARIABLES
                     138     ;                       FLAGS
                     139     ;                       REGISTERS
                     140     ;                       JUMP TABLE ENTRY ADDRESSES FOR ALL MODULES
                     141     ;
                     142     ;               UTILIT.INC (INCLUDE FILE)
                     143     ;
                     144     ;                       (I)ERROR
                     145     ;                       (I)EOL_CHECK
                     146     ;                       INC_PNT/DEC_PNT/SWAP_POINTERS
                     147     ;                       SPACCO/(I)CO
                     148     ;                       ICI
                     149     ;                       ICSTS
                     150     ;                       (U)CSTS
```

```
LOC  OBJ            LINE     SOURCE

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

```
LOC  OBJ            LINE     SOURCE

                     206     ;                       HEXBIN
                     207     ;                       GET_TYPE
                     208     ;                       LOAD_HEX
                     209     ;                       STORE_HEX
                     210     ;                       LOAD_CMD
                     211     ;                       SAVE_CMD
                     212     ;                       DOWNLOAD_CMD
                     213     ;                       UPLOAD_CMD
                     214     ;
                     215     ;
                     216     ;*****************************************************************
                     217 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                     218 +1  $INCLUDE(:F1:COMMON.INC)
  E000          =1   219     BASE    EQU     0E000H
                =1   220     ;***** CONSTANTS USED BY ALL MODULES *****
                =1   221
  0001          =1   222     NUMBER_TOKE     EQU     01H             ;Constant (GETOKE,number token)
  0003          =1   223     BAR_TOKE        EQU     03H             ;Constant (GETOKE,slash (/) token)
  0006          =1   224     POUND_TOKE      EQU     06H     ;Constant
  0005          =1   225     PLUS_TOKE       EQU     05H
  0007          =1   226     EOL_TOKE        EQU     07H             ;Constant (GETOKE,end of line token)
  000A          =1   227     ATA_TOKE        EQU     0AH
  005E          =1   228     C_TOKE          EQU     05EH
  0080          =1   229     CBYTE_TOKE      EQU     080H
  00A1          =1   230     DPTR_TOKE       EQU     0A1H
  00D4          =1   231     ORG_TOKE        EQU     0D4H
  00A0          =1   232     PC_TOKE         EQU     0A0H
  0040          =1   233     REG             EQU     40H
  0010          =1   234     OFST            EQU     10H
  0018          =1   235     LINMAX          EQU     24
  0004          =1   236     TOKSIZ          EQU     4
  0080          =1   237     BLINK           EQU     80H             ;Set the blink bit in bytes to go to the UPI
  0000          =1   238     SELECT_CON      EQU     00H             ;Set up UPI for on-board console
                =1   239
                =1   240     ;***** GLOBAL VARIABLES USED BY MORE THAN ONE MAIN MODULE *****
  ----          =1   241             DSEG
  0024          =1   242             ORG     24H
                =1   243     ;***** ARRAYS *****
                =1   244
  0024          =1   245     LINBUF:                 DS      LINMAX          ;Input line buffer(24 chars)
  003C          =1   246     STRGBF:                 DS      TOKSIZ          ;Buffer for string
  0040          =1   247     WORKING_SPACE:          DS      3               ;Buffer for ASM/DASM
                =1   248
                =1   249     ;***** VARIABLES *****
                =1   250
  0043          =1   251     ERRNUM:                 DS      1
  0044          =1   252     PNTHGH:                 DS      1
  0045          =1   253     PNTLOW:                 DS      1
  0046          =1   254     SELECT:                 DS      1
  0047          =1   255     TEMP_LOW:               DS      1
  0048          =1   256     TOKSTR:                 DS      1
  0049          =1   257     VALHGH:                 DS      1
  004A          =1   258     VALLOW:                 DS      1
  004B          =1   259     ASM_PC_HIGH:            DS      1
  004C          =1   260     ASM_PC_LOW:             DS      1
  004D          =1   261     NUMBER_OF_BYTES:        DS      1
  004E          =1   262     OUR_CODE_HIGH:          DS      1
  004F          =1   263     OUR_CODE_LOW:           DS      1
  0050          =1   264     CHARIN:                 DS      1
  0051          =1   265     CHRCNT:                 DS      1
  0052          =1   266     LINE_START:             DS      1
  0053          =1   267     LINCNT:                 DS      1
  0054          =1   268     LNLGTH:                 DS      1
  0055          =1   269     STRGCT:                 DS      1
  0056          =1   270     TEMP1:                  DS      1
  0057          =1   271     PARTIT_LO_HIGH:         DS      1
  0058          =1   272     PARTIT_LO_LOW:          DS      1
```

```
LOC  OBJ            LINE     SOURCE

0059              =1   273     PARTIT_HI_HIGH:         DS      1
005A              =1   274     PARTIT_HI_LOW:          DS      1
                  =1   275
                  =1   276     ;***** FLAGS *****
----              =1   277             BSEG
0000              =1   278             ORG     0
                  =1   279
0000              =1   280     B_O_T:                  DBIT    1
0001              =1   281     LSTFLG:                 DBIT    1
----              =1   282             CSEG
                  =1   283     ;***** REGISTERS *****
  REG             =1   284     POINT0          EQU     R0              ;Register (addr pointer)
  REG             =1   285     POINT1          EQU     R1              ;Register (addr pointer)
  REG             =1   286     PARAM1          EQU     R2              ;Register (parameter passing media #1)
  REG             =1   287     PARAM2          EQU     R3              ;REGISTER (Parameter passing media #2)
  REG             =1   288     PARAM3          EQU     R4              ;REGISTER (Parameter passing media #3)
  REG             =1   289     PARAM4          EQU     R5
  REG             =1   290     PARAM5          EQU     R6
  REG             =1   291     PARAM6          EQU     R7
  REG             =1   292     COUNT           EQU     R7
  REG             =1   293     CHECKSUM        EQU     R6
  REG             =1   294     TEMP            EQU     R5
                  =1   295     ;***** END OF VARIABLE  EQUATES *****
                  =1   296     ;*****************************************************************
                  =1   297     ;               JUMP TABLE ENTRY ADDRESSES FOR ALL MODULES
                  =1   298     ;*****************************************************************
  E006            =1   299     CO                      EQU     6 + BASE
  E009            =1   300     CI                      EQU     9 + BASE
  E00C            =1   301     CSTS                    EQU     0CH + BASE
  E00F            =1   302     NEWLINE                 EQU     0FH + BASE
  E012            =1   303     TIME                    EQU     12H + BASE
  E015            =1   304     LSTBYT                  EQU     15H + BASE
  E018            =1   305     LSTWRD                  EQU     18H + BASE
  E01E            =1   306     PRINT_STRING            EQU     1EH + BASE
                  =1   307
  E04A            =1   308     FETCH                   EQU     4AH + BASE
  E04D            =1   309     STORE                   EQU     4DH + BASE
  E050            =1   310     GETNUM                  EQU     50H + BASE
  E053            =1   311     GETEOL                  EQU     53H + BASE
  E056            =1   312     GETOKE                  EQU     56H + BASE
  E059            =1   313     DISPLAY_TOKEN           EQU     59H + BASE
  E05C            =1   314     SAVE_AND_DISPLAY        EQU     5CH + BASE
  E05F            =1   315     ERROR                   EQU     5FH + BASE
  E062            =1   316     WAIT_FOR_USER           EQU     62H + BASE
  E065            =1   317     GET_PART                EQU     65H + BASE
  E068            =1   318     CONTINUATION_LINE       EQU     68H + BASE
  E06B            =1   319     GET_COMMA               EQU     6BH + BASE
  E06E            =1   320     EOL_CHECK               EQU     6EH + BASE
                       321 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                     322     ;*****************************************************************
E000                 323             ORG     BASE
                     324     ;
E000 02E267          325             JMP     POWER_ON        ; Initialize and start monitor.
                     326     ;
                     327     ;*****************************************************************
                     328     ;               JUMP TABLE FOR USER ACCESSABLE ROUTINES
                     329
                     330
E003 02ED94          331     BREAK:  LJMP    IBREAK                  ;Do not access this vector except through
                     332                                             ;normal SDK system interrupts,
                     333                                             ;breaks and keyclosures
                     334
E006 02E5CE          335             LJMP    ICO
E009 02E5FF          336             LJMP    UCI
E00C 02E5F9          337             LJMP    UCSTS
E00F 02E6FD          338             LJMP    INEWLINE
E012 02EA14          339             LJMP    ITIME
E015 02E7DF          340             LJMP    ILSTBYT
E018 02E7DA          341             LJMP    ILSTWRD
E01B 02EA0B          342             LJMP    IASCII_TO_HEX
E01E 02E9CD          343             LJMP    IPRINT_STRING
E021 02E267          344             LJMP    POWER_ON                ;The rest of the jump table reserved
E024 02E267          345             LJMP    POWER_ON                ;for future expansion.
E027 02E267          346             LJMP    POWER_ON
E02A 02E267          347             LJMP    POWER_ON
E02D 02E267          348             LJMP    POWER_ON
                     349
E030                 350             ORG     BASE+30H
                     351
                     352
E030 20284329        353     COPYRIGHT:      DB      ' (C) 1981 INTEL CORP. '
E034 20313938
E038 3120494E
E03C 54454C20
E040 434F5250
E044 2E20
E046 05              354     DATECODE:       DB 5H,16H,81H;
E047 16
E048 81
E049 05              355     STORED_CHECK_SUM:       DB      5H,18H,81H
E04A 18
E04B 81
E04C 02E651          356             LJMP    IFETCH
E04F 02E658          357             LJMP    ISTORE
E052 02E74F          358             LJMP    IGETNUM
E055 02E759          359             LJMP    IGETEOL
E058 02E8A0          360             LJMP    IGETOKE
E05B 02E9E0          361             LJMP    IDISPLAY_TOKEN
E05E 02E7C3          362             LJMP    ISAVE_AND_DISPLAY
E061 02E3CA          363             LJMP    IERROR
E064 02E396          364             LJMP    IWAIT_FOR_USER
E067 02E788          365             LJMP    IGET_PART
E06A 02E643          366             LJMP    ICONTINUATION_LINE
E06D 02E760          367             LJMP    IGET_COMMA
```

```
LOC  OBJ             LINE     SOURCE

E070 02E5A1           368             LJMP    IEOL_CHECK
                      369
                      370     ;***** CONSTANTS *****
                      371
  0004                372     EQUAL_TOKE              EQU     4       ;Constant (GETOKE,EQUAL TOKEN)
  0002                373     COMMA_TOKE              EQU     02H     ;Constant(Comma token)
  0008                374     BACKSP          EQU     08H     ;Constant (GETCHR,LITERAL 'BACK SPACE')
  000D                375     CR              EQU     0DH     ;Constant (NEWLIN,LITERAL 'CARRAGE RETURN')
  000A                376     LF              EQU     0AH     ;Constant (NEWLIN,LITERAL 'LINE FEED')
  0009                377     HORIZONTAL_TAB  EQU     09H     ;Constant (TAB KEY)
  007F                378     RBOUT           EQU     7FH     ;Constant (GETCHR,LITERAL 'DELETE')
  001B                379     ESC             EQU     1BH     ;Constant (EXECUT,LISTER 'ESCAPE')
  0007                380     STACK           EQU     07H
  0004                381     RESET_CMD       EQU     04H     ;UPI reset command
  0008                382     CLR_BRK_LATCHES EQU     08H
  0083                383     TOP_PORT        EQU     83H     ;UPI top port
  0003                384     GR_PORT         EQU     03H     ;UPI hardware GO register port
  0009                385     NO_BREAK        EQU     09H     ;Disables break logic
  0002                386     CASSETTE_READ   EQU     02H     ;UPI select cassette read mode
  0082                387     CASSETTE_WRITE  EQU     82H     ;UPI select cassette write mode
  0001                388     USART_MODE      EQU     01H     ;UPI serial port select for up/down load
  0001                389     SINGLE_BREAK    EQU     01H     ;Enables single step breaks.
  000D                390     DATA_BREAK      EQU     0DH     ;Enables data memory breaks
  000B                391     PROGRAM_BREAK   EQU     0BH     ;Enables program memory breaks
  A001                392     UPI_CONTROL     EQU     0A001H
  A000                393     UPI_DATA        EQU     0A000H
  B000                394     RAMOFF          EQU     0B000H  ;Constant (STORE,16-BIT INTERNAL RAM OFFSET)
  C000                395     BRKOFF          EQU     0C000H  ;Constant (STORE,16-BIT,BREAK RAM OFFSET)
  B800                396     RAMIO           EQU     0B800H  ;Constant (STORE,16-BIT INTERNAL RAM I/O OFFSET)
  0005                397     TIMER_HIGH      EQU     05H     ;Constant (ADDRESS OF 8155 TIMER HIGH BYTE)
  0040                398     CONTINUOUS_MODE EQU     40H     ;Constant (COMMAND MODE FOR TIMER)
  00C0                399     START_16_TIMER  EQU     0C0H    ;Constant (COMMAND TO LOAD AND START TIMER)
  00FF                400     MAXLOW          EQU     0FFH    ;Constant
  001F                401     MAXHGH          EQU     01FH    ;Constant
  00F1                402     UPI_DATA_IMAGE  EQU     0F1H    ;Software version of UPI input data.
  00F2                403     SAVE_SEL        EQU     0F2H    ;Used to store the token during emulation.
  00F3                404     ADDR_SAVE_HIGH  EQU     0F3H    ;Saves display address during emulation.
  00F4                405     ADDR_SAVE_LOW   EQU     0F4H
  00F5                406     DELAY           EQU     0F5H    ;Stores multi-step delay count.
  00F6                407     GR              EQU     0F6H    ;GO register
  00F7                408     BAUD_HIGH       EQU     0F7H
  00F8                409     BAUD_LOW        EQU     0F8H    ;Stores baud rate information.
  00F9                410     TOP_STORE       EQU     0F9H    ;Stores the user TOP value
  00FA                411     MON_FLAGS       EQU     0FAH    ;Stores monitor flags
  00FB                412     BREAK_STATUS    EQU     0FBH    ;Used to store the step flag during emulation.
  00FC                413     BAUDKEY         EQU     0FCH    ;Stores coded baud info in one byte
  00FB                414     NOT_STEP        EQU     0FBH    ;Stored in BREAK_STATUS to indicate not stepping
  00FE                415     SINGLESTEP      EQU     0FEH    ;Stored in BREAK_STATUS to indicate single step
  00FF                416     MULTISTEP       EQU     0FFH    ;Stored to indicate multiple single steps.
                      417
                      418
                      419     ;**********  VARIABLES **********
                      420
----                  421             DSEG
005B                  422             ORG     (PARTIT_HI_LOW+1)
```

```
LOC  OBJ            LINE     SOURCE

005B                 423     TOKSAV:         DS      1       ;DATA ADDR
005C                 424     DLYCNT:         DS      1       ;DATA ADDR
005D                 425     COUNTR:         DS      1
005E                 426     VPC_LOW:        DS      1
005F                 427     VPC_HIGH:       DS      1
0060                 428     CAUSE_IMAGE:    DS      1
0061                 429     PCNTHI:         DS      1
0062                 430     PCNTLO:         DS      1
0063                 431     LENGTH_HIGH:    DS      1
0064                 432     LENGTH_LOW:     DS      1
0065                 433     TYPE:           DS      1
                     434
                     435     ;***** FLAGS *****
                     436
----                 437             BSEG
0002                 438             ORG     (LSTFLG+1)
0002                 439     ANY_BR_FLAG:    DBIT    1
0003                 440     FIRST_FLAG:     DBIT    1
0004                 441     MAXNUM_FLAG:    DBIT    1
0005                 442     BINARY_FLG:     DBIT    1
----                 443             CSEG
                     444 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                     445     ;*****************************************************************
                     446     ;               TOKEN EQUATES
                     447     ;*****************************************************************
                     448
  005F               449     ATDPTR_TOKE     EQU     15+REG+OFST
  0052               450     ATRO_TOKE       EQU     2+REG+OFST
  0053               451     ATR1_TOKE       EQU     3+REG+OFST
  0051               452     A_TOKE          EQU     051H
  005C               453     AB_TOKE         EQU     12+REG+OFST
  0088               454     ABR_TOKE        EQU     088H
  0012               455     ACALL_TOKE      EQU     2+OFST
  0098               456     ACC_TOKE        EQU     098H
  0024               457     ADD_TOKE        EQU     20+OFST
  0023               458     ADDC_TOKE       EQU     19+OFST
  0013               459     AJMP_TOKE       EQU     3+OFST
  0021               460     ANL_TOKE        EQU     17+OFST
  00B0               461     ASM_TOKE        EQU     0B0H
  009B               462     B_TOKE          EQU     09BH
  00D0               463     BAUD_TOKE       EQU     0D0H
  0089               464     BR_TOKE         EQU     089H
  00D2               465     CAUSE_TOKE      EQU     0D2H
  0019               466     CJNE_TOKE       EQU     9+OFST
  002A               467     CLR_TOKE        EQU     26+OFST
  002B               468     CPL_TOKE        EQU     27+OFST
  002C               469     DA_TOKE         EQU     28+OFST
  00B8               470     DASM_TOKE       EQU     0B8H
  00D3               471     DATA_TOKE       EQU     0D3H
  0082               472     DBYTE_TOKE      EQU     082H
  0035               473     DEC_TOKE        EQU     37+OFST
  0031               474     DIV_TOKE        EQU     33+OFST
  0025               475     DJNZ_TOKE       EQU     21+OFST
  00E0               476     DOWNLOAD_TOKE   EQU     0E0H
  0008               477     FOREVER_TOKE    EQU     008H
  0009               478     FROM_TOKE       EQU     009H
  00C2               479     GO_TOKE         EQU     0C2H
  0037               480     INC_TOKE        EQU     39+OFST
  0027               481     JB_TOKE         EQU     23+OFST
  0028               482     JBC_TOKE        EQU     24+OFST
  0018               483     JC_TOKE         EQU     8+OFST
  0032               484     JMP_TOKE        EQU     34+OFST
  0026               485     JNB_TOKE        EQU     22+OFST
  0017               486     JNC_TOKE        EQU     7+OFST
  0015               487     JNZ_TOKE        EQU     5+OFST
  0016               488     JZ_TOKE         EQU     6+OFST
  0010               489     LCALL_TOKE      EQU     0+OFST
  00D7               490     LIST_TOKE       EQU     0D7H
  0011               491     LJMP_TOKE       EQU     1+OFST
  00E2               492     LOAD_TOKE       EQU     0E2H
  001F               493     MOV_TOKE        EQU     15+OFST
  001A               494     MOVC_TOKE       EQU     10+OFST
  001B               495     MOVX_TOKE       EQU     11+OFST
  0030               496     MUL_TOKE        EQU     32+OFST
  003B               497     NOP_TOKE        EQU     43+OFST
  000F               498     ON_TOKE         EQU     00FH
  000B               499     OR_TOKE         EQU     0BH
```

```
LOC  OBJ            LINE     SOURCE

  0022               500     ORL_TOKE        EQU     18+OFST
  002D               501     POP_TOKE        EQU     29+OFST
  00D5               502     PROGRAM_TOKE    EQU     0D5H
  0099               503     PSW_TOKE        EQU     099H
  002F               504     PUSH_TOKE       EQU     31+OFST
  0090               505     R0_TOKE         EQU     090H
  0091               506     R1_TOKE         EQU     091H
  0092               507     R2_TOKE         EQU     092H
  0093               508     R3_TOKE         EQU     093H
  0094               509     R4_TOKE         EQU     094H
  0095               510     R5_TOKE         EQU     095H
  0096               511     R6_TOKE         EQU     096H
  0097               512     R7_TOKE         EQU     097H
  0084               513     RBIT_TOKE       EQU     084H
  0000               514     RBS_TOKE        EQU     000
  0081               515     RBYTE_TOKE      EQU     081H
  000E               516     RESET_TOKE      EQU     00EH
  003A               517     RET_TOKE        EQU     42+OFST
  0039               518     RETI_TOKE       EQU     41+OFST
  0034               519     RL_TOKE         EQU     36+OFST
  0033               520     RLC_TOKE        EQU     35+OFST
  0038               521     RR_TOKE         EQU     40+OFST
  0036               522     RRC_TOKE        EQU     38+OFST
  00E3               523     SAVE_TOKE       EQU     0E3H
  0029               524     SETB_TOKE       EQU     25+OFST
  0014               525     SJMP_TOKE       EQU     4+OFST
  009A               526     SP_TOKE         EQU     09AH
  00C1               527     STEP_TOKE       EQU     0C1H
  001E               528     SUBB_TOKE       EQU     14+OFST
  002E               529     SWAP_TOKE       EQU     30+OFST
  000C               530     TILL_TOKE       EQU     00CH
  00A2               531     TM0_TOKE        EQU     0A2H
  00A3               532     TM1_TOKE        EQU     0A3H
  000D               533     T0_TOKE         EQU     00DH
  00D6               534     TOP_TOKE        EQU     0D6H
  00E1               535     UPLOAD_TOKE     EQU     0E1H
  0086               536     XBYTE_TOKE      EQU     086H
  001D               537     XCH_TOKE        EQU     13+OFST
  001C               538     XCHD_TOKE       EQU     12+OFST
  0020               539     XRL_TOKE        EQU     16+OFST
                     540     ;
                     541     ;
                     542     ;***** TOKEN TABLE *****
                     543     ;
                     544     TOKTBL:
E073 0A              545             DB      ATA_TOKE
E074 5F              546             DB      ATDPTR_TOKE
E075 52              547             DB      ATR0_TOKE
E076 53              548             DB      ATR1_TOKE
E077 51              549             DB      A_TOKE
E078 5C              550             DB      AB_TOKE
E079 88              551             DB      ABR_TOKE
E07A 12              552             DB      ACALL_TOKE
E07B 98              553             DB      ACC_TOKE
E07C 24              554             DB      ADD_TOKE
```

```
LOC  OBJ            LINE     SOURCE

E07D 23              555              DB      ADDC_TOKE
E07E 13              556              DB      AJMP_TOKE
E07F 21              557              DB      ANL_TOKE
E080 B0              558              DB      ASM_TOKE
E081 9B              559              DB      B_TOKE
E082 D0              560              DB      BAUD_TOKE
E083 89              561              DB      BR_TOKE
E084 5E              562              DB      C_TOKE
E085 D2              563              DB      CAUSE_TOKE
E086 80              564              DB      CBYTE_TOKE
E087 19              565              DB      CJNE_TOKE
E088 2A              566              DB      CLR_TOKE
E089 2B              567              DB      CPL_TOKE
E08A B8              568              DB      DASM_TOKE
E08B 2C              569              DB      DA_TOKE
E08C B8              570              DB      DASM_TOKE
E08D D3              571              DB      DATA_TOKE
E08E 82              572              DB      DBYTE_TOKE
E08F 35              573              DB      DEC_TOKE
E090 31              574              DB      DIV_TOKE
E091 25              575              DB      DJNZ_TOKE
E092 E0              576              DB      DOWNLOAD_TOKE
E093 A1              577              DB      DPTR_TOKE
E094 09              578              DB      FROM_TOKE
E095 08              579              DB      FOREVER_TOKE
E096 09              580              DB      FROM_TOKE
E097 C2              581              DB      GO_TOKE
E098 37              582              DB      INC_TOKE
E099 27              583              DB      JB_TOKE
E09A 28              584              DB      JBC_TOKE
E09B 18              585              DB      JC_TOKE
E09C 32              586              DB      JMP_TOKE
E09D 26              587              DB      JNB_TOKE
E09E 17              588              DB      JNC_TOKE
E09F 15              589              DB      JNZ_TOKE
E0A0 16              590              DB      JZ_TOKE
E0A1 10              591              DB      LCALL_TOKE
E0A2 D7              592              DB      LIST_TOKE
E0A3 11              593              DB      LJMP_TOKE
E0A4 E2              594              DB      LOAD_TOKE
E0A5 1F              595              DB      MOV_TOKE
E0A6 1A              596              DB      MOVC_TOKE
E0A7 1B              597              DB      MOVX_TOKE
E0A8 30              598              DB      MUL_TOKE
E0A9 3B              599              DB      NOP_TOKE
E0AA 0F              600              DB      ON_TOKE
E0AB 0B              601              DB      OR_TOKE
E0AC D4              602              DB      ORG_TOKE
E0AD 22              603              DB      ORL_TOKE
E0AE A0              604              DB      PC_TOKE
E0AF 2D              605              DB      POP_TOKE
E0B0 D5              606              DB      PROGRAM_TOKE
E0B1 99              607              DB      PSW_TOKE
E0B2 2F              608              DB      PUSH_TOKE
E0B3 90              609              DB      R0_TOKE
```

```
LOC  OBJ            LINE     SOURCE

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

```
LOC  OBJ            LINE     SOURCE

                     647
                     648
                     649     ;***** KEY WORD TABLE *****
                     650
E0D7 40412020        651     KEYTAB: DB      '@A  '
E0DB 40445054        652             DB      '@DPT'
E0DF 40523020        653             DB      '@RO '
E0E3 40523120        654             DB      '@R1 '
E0E7 41202020        655             DB      'A   '                  ;5
E0EB 41422020        656             DB      'AB  '
E0EF 41425220        657             DB      'ABR '
E0F3 4143414C        658             DB      'ACAL'
E0F7 41434320        659             DB      'ACC '
E0FB 41444420        660             DB      'ADD '                  ;10
E0FF 41444443        661             DB      'ADDC'
E103 414A4D50        662             DB      'AJMP'
E107 414E4C20        663             DB      'ANL '
E10B 41534D20        664             DB      'ASM '
E10F 42202020        665             DB      'B   '                  ;15
E113 42415544        666             DB      'BAUD'
E117 42522020        667             DB      'BR  '
E11B 43202020        668             DB      'C   '
E11F 43415553        669             DB      'CAUS'
E123 43425954        670             DB      'CBYT'                  ;20
E127 434A4E45        671             DB      'CJNE'
E12B 434C5220        672             DB      'CLR '
E12F 43504C20        673             DB      'CPL '
E133 44202020        674             DB      'D   '
E137 44412020        675             DB      'DA  '                  ;25
E13B 4441534D        676             DB      'DASM'
E13F 44415441        677             DB      'DATA'
E143 44425954        678             DB      'DBYT'
E147 44454320        679             DB      'DEC '
E14B 44495620        680             DB      'DIV '                  ;30
E14F 444A4E5A        681             DB      'DJNZ'
E153 444F574E        682             DB      'DOWN'
E157 44505452        683             DB      'DPTR'
E15B 46202020        684             DB      'F   '
E15F 464F5245        685             DB      'FORE'                  ;35
E163 46524F4D        686             DB      'FROM'
E167 474F2020        687             DB      'GO  '
E16B 494E4320        688             DB      'INC '
E16F 4A422020        689             DB      'JB  '
E173 4A424320        690             DB      'JBC '                  ;40
E177 4A432020        691             DB      'JC  '
E17B 4A4D5020        692             DB      'JMP '
E17F 4A4E4220        693             DB      'JNB '
E183 4A4E4320        694             DB      'JNC '
E187 4A4E5A20        695             DB      'JNZ '                  ;45
E18B 4A5A2020        696             DB      'JZ  '
E18F 4C43414C        697             DB      'LCAL'
E193 4C495354        698             DB      'LIST'
E197 4C4A4D50        699             DB      'LJMP'
E19B 4C4F4144        700             DB      'LOAD'                  ;50
E19F 4D4F5620        701             DB      'MOV '
```

```
LOC  OBJ            LINE     SOURCE

E1A3 4D4F5643         702             DB      'MOVC'
E1A7 4D4F5658         703             DB      'MOVX'
E1AB 4D554C20         704             DB      'MUL '
E1AF 4E4F5020         705             DB      'NOP '                  ;55
E1B3 4F4E2020         706             DB      'ON  '
E1B7 4F522020         707             DB      'OR  '
E1BB 4F524720         708             DB      'ORG '
E1BF 4F524C20         709             DB      'ORL '
E1C3 50432020         710             DB      'PC  '                  ;60
E1C7 504F5020         711             DB      'POP '
E1CB 50524F47         712             DB      'PROG'
E1CF 50535720         713             DB      'PSW '
E1D3 50555348         714             DB      'PUSH'
E1D7 52302020         715             DB      'R0  '                  ;65
E1DB 52312020         716             DB      'R1  '
E1DF 52322020         717             DB      'R2  '
E1E3 52332020         718             DB      'R3  '
E1E7 52342020         719             DB      'R4  '
E1EB 52352020         720             DB      'R5  '                  ;70
E1EF 52362020         721             DB      'R6  '
E1F3 52372020         722             DB      'R7  '
E1F7 52424954         723             DB      'RBIT'
E1FB 52425320         724             DB      'RBS '
E1FF 52425954         725             DB      'RBYT'                  ;75
E203 52455345         726             DB      'RESE'
E207 52455420         727             DB      'RET '
E20B 52455449         728             DB      'RETI'
E20F 524C2020         729             DB      'RL  '
E213 524C4320         730             DB      'RLC '                  ;80
E217 52522020         731             DB      'RR  '
E21B 52524320         732             DB      'RRC '
E21F 53415645         733             DB      'SAVE'
E223 53455442         734             DB      'SETB'
E227 534A4D50         735             DB      'SJMP'                  ;85
E22B 53502020         736             DB      'SP  '
E22F 53544550         737             DB      'STEP'
E233 53554242         738             DB      'SUBB'
E237 53574150         739             DB      'SWAP'
E23B 54202020         740             DB      'T   '                  ;90
E23F 54494C4C         741             DB      'TILL'
E243 544D3020         742             DB      'TM0 '
E247 544D3120         743             DB      'TM1 '
E24B 544F2020         744             DB      'TO  '
E24F 544F5020         745             DB      'TOP '                  ;95
E253 55504C4F         746             DB      'UPLO'
E257 58425954         747             DB      'XBYT'
E25B 58434820         748             DB      'XCH '
E25F 58434844         749             DB      'XCHD'
E263 58524C20         750             DB      'XRL '                  ;100
                      751     ;***** END OF KEYTAB *****
                      752 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                     753     ;*****************************************************************
                     754     ;
                     755     ;    NAME: POWER_ON
                     756     ;
                     757     ;    ABSTRACT: This routine initializes the breakpoint RAM, I/O
                     758     ;       channels, output buffer flag, TOP value, break status, user
                     759     ;       DPTR, B register and user PC.  It sets the baud rate to 2400
                     760     ;       and the GO condition to forever.  At the end, it jumps to
                     761     ;       BREAK which sets up the user area and jumps to SIGN_ON
                     762     ;       since the step flag has been cleared.
                     763     ;
                     764     ;    INPUTS: None
                     765     ;
                     766     ;    OUTPUTS: LSTFLG, GR, UPI_DATA_IMAGE, BAUDKEY, BAUD_HIGH, BAUD_LOW,
                     767     ;       ERRNUM, TOP_STORE, MON_FLAG, BREAK_STATUS, CAUSE_IMAGE, ASM_PC_LOW,
                     768     ;       ASM_PC_HIGH, DPTR, B, 2 stack locations, CHRCNT, LNLGTH, CHARIN,
                     769     ;       MAXNUM_FLG
                     770     ;
                     771     ;    VARIABLES MODIFIED: SP, LSTFLG, DPTR, A, PARAM1, DPL, ERRNUM,
                     772     ;       ASM_PC_HIGH, ASM_PC_LOW, CAUSE_IMAGE, DPH, B
                     773     ;
                     774     ;    ERROR EXITS: None
                     775     ;
                     776     ;    SUBROUTINES ACCESSED DIRECTLY: CHECK_EPROMS, INIT_IO, UPI_CMD,
                     777     ;       UPI_IN, UPI_OUT, SET_BAUD, BREAK
                     778     ;
                     779     ;*****************************************************************
                     780     POWER_ON:
E267 12ECAF          781             CALL    CLRBRK                          ;Clear breakpoint RAM and
                     782                                                     ;remove Monitor from over-
                     783                                                     ;laying user Config. Memory
E26A 758107          784             MOV     SP,#07H
E26D C201            785             CLR     LSTFLG
E26F 12E3A0          786             CALL    CHECK_EPROMS                    ;Verify integrity of Monitor code.
E272 12E36C          787             CALL    INIT_IO
E275 90A000          788             MOV     DPTR,#UPI_DATA
E278 E0              789             MOVX    A,@DPTR                         ;Initialize the IO channel and
E279 90B0F6          790             MOV     DPTR,#(RAMOFF+GR)               ;copy break enable image
E27C 7409            791             MOV     A,#NO_BREAK                     ;into hardware
E27E F0              792             MOVX    @DPTR,A
                     793                                                     ;Sets GO FOREVER as the power up
                     794                                                     ;Default condition
                     795                                                     ;Clear the users output buffer flag.
E27F 90B0F1          796             MOV     DPTR,#(RAMOFF+UPI_DATA_IMAGE)
E282 E4              797             CLR     A
E283 F0              798             MOVX    @DPTR,A
E284 7A83            799             MOV     PARAM1,#TOP_PORT                ;Initialize TOP port.
E286 12E60B          800             CALL    UPI_CMD
E289 12E632          801             CALL    UPI_IN                          ;Ignore current port value.
E28C 7A00            802             MOV     PARAM1,#00H                     ;Reselect the console.
E28E 12E61E          803             CALL    UPI_OUT
E291 7582FC          804             MOV     DPL,#BAUDKEY                    ;Set up the initial baud rate
E294 7404            805             MOV     A,#04H
E296 F0              806             MOVX    @DPTR,A
E297 12F1EA          807             CALL    SET_BAUD                        ;for 2400.
```

```
LOC  OBJ            LINE     SOURCE

E29A 90B0F7          808              MOV     DPTR,#(RAMOFF+BAUD_HIGH)
E29D 7424            809              MOV     A,#24H
E29F F0              810              MOVX    @DPTR,A
E2A0 E4              811              CLR     A
E2A1 F543            812              MOV     ERRNUM,A                ;Firmware checksum error
E2A3 A3              813              INC     DPTR                    ;Point to BAUD_LOW
E2A4 F0              814              MOVX    @DPTR,A
E2A5 A3              815              INC     DPTR                    ;Point to TOP_STORE and zero.
E2A6 F0              816              MOVX    @DPTR,A
E2A7 A3              817              INC     DPTR                    ;Point to MON_FLAGS and zero
E2A8 F0              818              MOVX    @DPTR,A
E2A9 A3              819              INC     DPTR                    ;Point to BREAK_STATUS
E2AA F0              820              MOVX    @DPTR,A                 ;Set it to the power on flag
E2AB F54B            821              MOV     ASM_PC_HIGH,A           ;Zero out the asm PC
E2AD F54C            822              MOV     ASM_PC_LOW,A
E2AF F560            823              MOV     CAUSE_IMAGE,A
E2B1 F582            824              MOV     DPL,A
E2B3 F583            825              MOV     DPH,A                   ;Clear DPTR and B so that
E2B5 F5D0            826              MOV     PSW,A                   ;break will report them correctly
E2B7 F5F0            827              MOV     B,A
E2B9 C0E0            828              PUSH    ACC                     ;Simulate the user PC in the stack
E2BB C0E0            829              PUSH    ACC
E2BD 0103            830              JMP     BREAK
                     831 +1   $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                     832     ;*********************************************************************
                     833     ;
                     834     ;    NAME: SIGN_ON
                     835     ;
                     836     ;    ABSTRACT: Puts sign on message on the display and waits for
                     837     ;       a character to be input.
                     838     ;
                     839     ;    INPUTS: None
                     840     ;
                     841     ;    OUTPUTS: None
                     842     ;
                     843     ;    VARIABLES MODIFIED: PARAM1, PARAM2
                     844     ;
                     845     ;    ERROR EXITS: None
                     846     ;
                     847     ;    SUBROUTINES ACCESSED DIRECTLY: IPRINT_STRING, IWAIT_FOR_USER
                     848     ;
                     849     ;*********************************************************************
                     850     SIGN_ON:
E2BF 7AE3            851             MOV     PARAM1,#HIGH(SIGN_ON_MSG)
E2C1 7B52            852             MOV     PARAM2,#LOW(SIGN_ON_MSG)
E2C3 12E9CD          853             CALL    IPRINT_STRING
E2C6 12E396          854             CALL    IWAIT_FOR_USER
                     855 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                     856     ;*****************************************************************
                     857     ;
                     858     ;    NAME: START
                     859     ;
                     860     ;    ABSTRACT: This routine initializes the stack and gets tokens
                     861     ;       until an EOL is encountered.  It then decodes the first token and
                     862     ;       branches to appropriate command routine.
                     863     ;
                     864     ;    INPUTS: None
                     865     ;
                     866     ;    OUTPUTS: LINE_START, SP, TOKSTR
                     867     ;
                     868     ;    VARIABLES MODIFIED: PARAM1, PARAM2, DPTR, A, SP, B,
                     869     ;
                     870     ;    ERROR EXITS: 02H (INVALID COMMAND)
                     871     ;
                     872     ;    SUBROUTINE  ACCESSED DIRECTLY: IGETOKE, INIT_IO, IERROR
                     873     ;
                     874     ;*****************************************************************
E2C9 758107          875     START:  MOV     SP,#STACK
E2CC 755200          876             MOV     LINE_START,#00H                 ;Default beginning of line
E2CF 12E36C          877             CALL    INIT_IO                         ;Reset UPI
E2D2 12E8A0          878             CALL    IGETOKE
E2D5 B40702          879             CJNE    A,#EOL_TOKE,DECODE_CALL         ;If EOL, branch to cmd routine
E2D8 80EF            880             JMP     START
                     881     DECODE_CALL:
E2DA 90E301          882             MOV     DPTR,#CMDTAB
E2DD 7A1B            883             MOV     PARAM1,#((SIGN_ON_MSG-CMDTAB)/3);Length of command table
E2DF 12E2E4          884             CALL    DECODE
E2E2 80E5            885             JMP     START
E2E4 E4              886     DECODE: CLR     A
E2E5 93              887             MOVC    A,@A+DPTR
E2E6 B5480D          888             CJNE    A,TOKSTR,NEXT_ENTRY             ;Check next entry if no match
E2E9 E4              889             CLR     A
E2EA A3              890             INC     DPTR
E2EB 93              891             MOVC    A,@A+DPTR                       ;Get high byte of cmd addr
E2EC F5F0            892             MOV     B,A
E2EE E4              893             CLR     A
E2EF A3              894             INC     DPTR
E2F0 93              895             MOVC    A,@A+DPTR                       ;Get low byte of cmd addr
E2F1 C0E0            896             PUSH    ACC
E2F3 C0F0            897             PUSH    B
E2F5 22              898             RET                                     ;'Return' to cmd addr
                     899     NEXT_ENTRY:
E2F6 A3              900             INC     DPTR
E2F7 A3              901             INC     DPTR
E2F8 A3              902             INC     DPTR                            ;Skip over 3 byte entries
E2F9 DAE9            903             DJNZ    PARAM1,DECODE                   ;Check for end of table
E2FB 754302          904             MOV     ERRNUM,#02H                     ;Invalid command
E2FE 02E3CA          905             JMP     IERROR
                     906     CMDTAB:
E301 88              907             DB      ABR_TOKE
E302 EB96            908             DW      BR_CMD
E304 98              909             DB      ACC_TOKE
E305 ECF8            910             DW      ACC_CMD
```

```
LOC  OBJ            LINE     SOURCE

E307 B0              911              DB      ASM_TOKE
E308 F523            912              DW      ASMBASE                         ;Assemble command.
E30A 9B              913              DB      B_TOKE
E30B ED0A            914              DW      B_CMD
E30D D0              915              DB      BAUD_TOKE
E30E F1BE            916              DW      BAUD_CMD
E310 89              917              DB      BR_TOKE
E311 EB96            918              DW      BR_CMD
E313 D2              919              DB      CAUSE_TOKE
E314 F279            920              DW      CAUSE_CMD
E316 80              921              DB      CBYTE_TOKE
E317 EA2A            922              DW      MEMORY_CMD
E319 B8              923              DB      DASM_TOKE
E31A F526            924              DW      (ASMBASE + 3)                   ;Disassemble command.
E31C 82              925              DB      DBYTE_TOKE
E31D EA2A            926              DW      MEMORY_CMD
E31F E0              927              DB      DOWNLOAD_TOKE
E320 F4BA            928              DW      DOWNLOAD_CMD
E322 A1              929              DB      DPTR_TOKE
E323 ED4B            930              DW      DPTR_CMD
E325 C2              931              DB      GO_TOKE
E326 F0D0            932              DW      GO_CMD
E328 D7              933              DB      LIST_TOKE
E329 F18E            934              DW      LIST_CMD
E32B E2              935              DB      LOAD_TOKE
E32C F40E            936              DW      LOAD_CMD
E32E A0              937              DB      PC_TOKE
E32F ED2D            938              DW      PC_CMD
E331 99              939              DB      PSW_TOKE
E332 ECFE            940              DW      PSW_CMD
E334 84              941              DB      RBIT_TOKE
E335 EA2A            942              DW      MEMORY_CMD
E337 81              943              DB      RBYTE_TOKE
E338 EA2A            944              DW      MEMORY_CMD
E33A E3              945              DB      SAVE_TOKE
E33B F478            946              DW      SAVE_CMD
E33D 9A              947              DB      SP_TOKE
E33E ED04            948              DW      SP_CMD
E340 C1              949              DB      STEP_TOKE
E341 EF9F            950              DW      STEP_CMD
E343 A2              951              DB      TMO_TOKE
E344 ED54            952              DW      TMO_CMD
E346 A3              953              DB      TM1_TOKE
E347 ED5D            954              DW      TM1_CMD
E349 D6              955              DB      TOP_TOKE
E34A F239            956              DW      TOP_CMD
E34C E1              957              DB      UPLOAD_TOKE
E34D F4D0            958              DW      UPLOAD_CMD
E34F 86              959              DB      XBYTE_TOKE
E350 EA2A            960              DW      MEMORY_CMD
                     961      ;*****************************************************************
                     962      ;
                     963      SIGN_ON_MSG:
E352 1A              964              DB      26,CR,LF,('SDK-51 MONITOR VER. 1.0')
E353 0D
```

```
LOC  OBJ            LINE     SOURCE

E354 0A
E355 53444B2D
E359 3531204D
E35D 4F4E4954
E361 4F522056
E365 45522E20
E369 312E30
                     965 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                     966     ;*****************************************************************
                     967     ;
                     968     ;    NAME: INIT_IO
                     969     ;
                     970     ;    ABSTRACT: This routine initialized the UPI hardware ports
                     971     ;       and resets the line buffer.
                     972     ;
                     973     ;    INPUTS: None
                     974     ;
                     975     ;    OUTPUTS: CHRCNT, LNLGTH, CHARIN, MAXNUM_FLAG
                     976     ;
                     977     ;    VARIABLES MODIFIED: A, CHRCNT, CHARIN, PARAM1, PARAM2,
                     978     ;       LNLGTH, PSW
                     979     ;
                     980     ;    ERROR EXITS: None
                     981     ;
                     982     ;    SUBROUTINES ACCESSED DIRECTLY: UPI_CMD, ITIME
                     983     ;
                     984     ;*****************************************************************
E36C C204            985     INIT_IO:CLR     MAXNUM_FLAG
E36E E4              986             CLR     A
E36F F551            987             MOV     CHRCNT,A
E371 F554            988             MOV     LNLGTH,A
E373 755020          989             MOV     CHARIN,#' '
E376 7A04            990             MOV     PARAM1,#RESET_CMD
E378 12E60B          991             CALL    UPI_CMD
E37B 7A03            992             MOV     PARAM1,#GR_PORT
E37D 12E60B          993             CALL    UPI_CMD
E380 7A08            994             MOV     PARAM1,#CLR_BRK_LATCHES
E382 12E61E          995             CALL    UPI_OUT
E385 7A09            996             MOV     PARAM1,#NO_BREAK
E387 12E61E          997             CALL    UPI_OUT
E38A 7A00            998             MOV     PARAM1,#SELECT_CON
E38C 12E60B          999             CALL    UPI_CMD
E38F 7A00           1000             MOV     PARAM1,#00H
E391 7B70           1001             MOV     PARAM2,#70H             ;Delay approx. one UPI display scan (11.2ms)
E393 02EA14         1002             JMP     ITIME                   ;so the display won't flicker on reset.
                    1003 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                    1004     ;*********************************************************************
                    1005     ;
                    1006     ;    NAME: (I)WAIT_FOR_USER
                    1007     ;
                    1008     ;    ABSTRACT: Clears keyboard closures, waits for next keyboard
                    1009     ;       entry and then returns.  The entry causing the return is NOT
                    1010     ;       read, therefore, the UPI will not overwrite it until it is
                    1011     ;       read by some other procedure.
                    1012     ;
                    1013     ;    INPUTS: None
                    1014     ;
                    1015     ;    OUTPUTS: None
                    1016     ;
                    1017     ;    VARIABLES MODIFIED: DPTR, PARAM1, PARAM2
                    1018     ;
                    1019     ;    ERROR EXITS: None
                    1020     ;
                    1021     ;    SUBROUTINES ACCESSED DIRECTLY: ITIME, ICSTS
                    1022     ;
                    1023     ;*********************************************************************
                    1024     IWAIT_FOR_USER:
E396 90A000         1025             MOV     DPTR,#UPI_DATA
E399 E0             1026             MOVX    A,@DPTR                 ;Clear any keyboard closures
                    1027     IWAIT_FOR_USER_1:
E39A 12E5E8         1028             CALL    ICSTS
E39D 50FB           1029             JNC     IWAIT_FOR_USER_1
E39F 22             1030             RET
                    1031 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                    1032     ;*****************************************************************
                    1033     ;
                    1034     ;    NAME: CHECK_EPROMS
                    1035     ;
                    1036     ;    ABSTRACT: This routine calculates the checksum for both
                    1037     ;       EPROMS.  If not ok, print an error message and lock up
                    1038     ;       forever.
                    1039     ;
                    1040     ;    INPUTS: None
                    1041     ;
                    1042     ;    OUTPUTS: None
                    1043     ;
                    1044     ;    VARIABLES MODIFIED: DPTR, CHECK_SUM, PARAM1, PARAM2, A
                    1045     ;
                    1046     ;    ERROR EXITS: None
                    1047     ;
                    1048     ;    SUBROUTINES ACCESSED DIRECTLY: IPRINT_STRING, ILSTBYT, SPACCO
                    1049     ;
                    1050     ;*****************************************************************
                    1051     CHECK_EPROMS:
E3A0 90E000         1052             MOV     DPTR,#BASE                      ;Load dptr with beginning address
E3A3 7E00           1053             MOV     CHECKSUM,#00H                   ;Clear scratch pad
                    1054     CHECK_LOOP:
E3A5 E4             1055             CLR     A
E3A6 93             1056             MOVC    A,@A+DPTR                       ;Get code byte
E3A7 2E             1057             ADD     A,CHECKSUM                      ;Accumulate a running total
E3A8 FE             1058             MOV     CHECKSUM,A                      ;Save it
E3A9 A3             1059             INC     DPTR                            ;Point to next byte
E3AA E583           1060             MOV     A,DPH                           ;If address has not wrapped around,
E3AC 70F7           1061             JNZ     CHECK_LOOP                      ;continue adding
E3AE EE             1062             MOV     A,CHECKSUM                      ;else, check tally
E3AF 6018           1063             JZ      CHECK_OUT_OK                    ;If everthing adds up, return
E3B1 7AE4           1064             MOV     PARAM1,#HIGH(ERROR_MSG)
E3B3 7B0C           1065             MOV     PARAM2,#LOW(ERROR_MSG)
E3B5 12E9CD         1066             CALL    IPRINT_STRING
E3B8 7A00           1067             MOV     PARAM1,#00H                     ;Firmware checksum error
E3BA 12E7DF         1068             CALL    ILSTBYT
E3BD 12E5CC         1069             CALL    SPACCO
E3C0 7AE4           1070             MOV     PARAM1,#HIGH(ERROR_TABLE)
E3C2 7B13           1071             MOV     PARAM2,#LOW(ERROR_TABLE)
E3C4 12E9CD         1072             CALL    IPRINT_STRING
E3C7 80FE           1073             JMP     $                               ;and hang up here
                    1074     CHECK_OUT_OK:
E3C9 22             1075             RET
                    1076
                    1077 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                    1078 +1  $INCLUDE(:F1:UTILIT.INC)
                  =1 1079    ;*****************************************************************
                  =1 1080    ;
                  =1 1081    ;    NAME: (I)ERROR
                  =1 1082    ;
                  =1 1083    ;    ABSTRACT: This routine handles all error messages for the SDK-51
                  =1 1084    ;       except error 0.  These are not intended to be a standard format
                  =1 1085    ;       for any other SDK product. After printing an error message, it
                  =1 1086    ;       waits for any console entry and then starts fresh from START.
                  =1 1087    ;       To find the routine which generates a particular error number,
                  =1 1088    ;       check the cross reference listing (XREF) at the back of this
                  =1 1089    ;       document for all uses of the variable name ERRNUM.
                  =1 1090    ;
                  =1 1091    ;    INPUTS: ERRNUM, LSTFLG
                  =1 1092    ;
                  =1 1093    ;    OUTPUTS: None
                  =1 1094    ;
                  =1 1095    ;    VARIABLES MODIFIED: PARAM1, PARAM2, C, A, TEMP1
                  =1 1096    ;
                  =1 1097    ;    ERROR EXITS: None
                  =1 1098    ;
                  =1 1099    ;    SUBROUTINES ACCESSED DIRECTLY: ITIME, INIT_IO, UPI_CMD,
                  =1 1100    ;       IPRINT_STRING, ILSTBYT, SPACCO, IWAIT_FOR_USER
                  =1 1101    ;
                  =1 1102    ;
                  =1 1103    ;*****************************************************************
E3CA 7A07         =1 1104    IERROR: MOV     PARAM1,#07H
E3CC 7B00         =1 1105            MOV     PARAM2,#00H
E3CE 12EA14       =1 1106            CALL    ITIME                       ;Wait for the completion of any
E3D1 716C         =1 1107            CALL    INIT_IO                     ;list activity before emptying usart
E3D3 A201         =1 1108            MOV     C,LSTFLG                    ;about 180ms
E3D5 E4           =1 1109            CLR     A
E3D6 92E6         =1 1110            MOV     ACC.6,C
E3D8 FA           =1 1111            MOV     PARAM1,A
E3D9 12E60B       =1 1112            CALL    UPI_CMD                     ;Select console with list status
E3DC 7AE4         =1 1113            MOV     PARAM1,#HIGH(ERROR_MSG)
E3DE 7B0C         =1 1114            MOV     PARAM2,#LOW(ERROR_MSG)
E3E0 12E9CD       =1 1115            CALL    IPRINT_STRING
E3E3 AA43         =1 1116            MOV     PARAM1,ERRNUM
E3E5 12E7DF       =1 1117            CALL    ILSTBYT
E3E8 12E5CC       =1 1118            CALL    SPACCO
E3EB 755600       =1 1119            MOV     TEMP1,#00                   ;Table search counter
E3EE 90E413       =1 1120            MOV     DPTR,#ERROR_TABLE           ;Table entry
                  =1 1121    ERROR_TEST:
E3F1 E543         =1 1122            MOV     A,ERRNUM
E3F3 B5560B       =1 1123            CJNE    A,TEMP1,ERROR_BEGIN         ;Is it this entry?
E3F6 AA83         =1 1124            MOV     PARAM1,DPH
E3F8 AB82         =1 1125            MOV     PARAM2,DPL
E3FA 12E9CD       =1 1126            CALL    IPRINT_STRING
E3FD 7196         =1 1127            CALL    IWAIT_FOR_USER
E3FF 41C9         =1 1128            JMP     START                       ;Yes, print message
                  =1 1129    ERROR_BEGIN:
E401 E4           =1 1130            CLR     A
E402 93           =1 1131            MOVC    A,@A+DPTR                   ;No, get num of letters to skip
                  =1 1132    ERROR_LOOP:
```

```
LOC  OBJ            LINE     SOURCE

E403 A3          =1  1133             INC     DPTR
E404 D5E0FC      =1  1134             DJNZ    ACC,ERROR_LOOP
E407 A3          =1  1135             INC     DPTR                    ;Adjust addr of next table entry
E408 0556        =1  1136             INC     TEMP1                   ;Adjust table search counter
E40A 80E5        =1  1137             JMP     ERROR_TEST
                 =1  1138     ERROR_MSG:
E40C 06          =1  1139             DB      6,CR,LF,('ERR=')
E40D 0D
E40E 0A
E40F 4552523D
                 =1  1140     ERROR_TABLE:
E413 0A          =1  1141             DB      10,('PROM CKSUM')       ;Error #00
E414 50524F4D
E418 20434B53
E41C 554D
E41E 0C          =1  1142             DB      12,('INVALID WORD')     ;       01
E41F 494E5641
E423 4C494420
E427 574F5244
E42B 0F          =1  1143             DB      15,('INVALID COMMAND')  ;       02
E42C 494E5641
E430 4C494420
E434 434F4D4D
E438 414E44
E43B 0A          =1  1144             DB      10,('NUMBER REQ')       ;       03
E43C 4E554D42
E440 45522052
E444 4551
E446 0A          =1  1145             DB      10,('RETURN REQ')       ;       04
E447 52455455
E44B 524E2052
E44F 4551
E451 11          =1  1146             DB      17,('EQUAL OR RTRN REQ')        ;       05
E452 45515541
E456 4C204F52
E45A 20525452
E45E 4E205245
E462 51
E463 09          =1  1147             DB      09,('COMMA REQ')        ;       06
E464 434F4D4D
E468 41205245
E46C 51
E46D 0D          =1  1148             DB      13,('PARTITION ADR')    ;       07
E46E 50415254
E472 4954494F
E476 4E204144
E47A 52
E47B 0F          =1  1149             DB      15,('RESET OR ON REQ')  ;       08
E47C 52455345
E480 54204F52
E484 204F4E20
E488 524551
E48B 0F          =1  1150             DB      15,('DECIMAL NUM REQ')  ;       09
E48C 44454349
E490 4D414C20
```

```
LOC  OBJ            LINE     SOURCE

E494 4E554D20
E498 524551
E49B 10         =1  1151             DB      16,('ILLEGAL BAUD VAL')         ;       0A
E49C 494C4C45
E4A0 47414C20
E4A4 42415544
E4A8 2056414C
E4AC 10         =1  1152             DB      16,('BRK ENABL SYNTAX')         ;       0B
E4AD 42524B20
E4B1 454E4142
E4B5 4C205359
E4B9 4E544158
E4BD 10         =1  1153             DB      16,('NUM OR RESET REQ')         ;       0C
E4BE 4E554D20
E4C2 4F522052
E4C6 45534554
E4CA 20524551
E4CE 0B         =1  1154             DB      11,('TOP ) 7FFFH')              ;       0D
E4CF 544F5020
E4D3 29203746
E4D7 464648
E4DA 0C         =1  1155             DB      12,('DISPLAY ONLY')             ;       0E
E4DB 44495350
E4DF 4C415920
E4E3 4F4E4C59
E4E7 10         =1  1156             DB      16,('UNDEFINED OPCODE')         ;       0F
E4E8 554E4445
E4EC 46494E45
E4F0 44204F50
E4F4 434F4445
E4F8 0F         =1  1157             DB      15,('ASSEMBLY SYNTAX')          ;       10
E4F9 41535345
E4FD 4D424C59
E501 2053594E
E505 544158
E508 10         =1  1158             DB      16,('ADR OUT OF RANGE')         ;       11
E509 41445220
E50D 4F555420
E511 4F462052
E515 414E4745
E519 10         =1  1159             DB      16,('ADR OUT OF RANGE')         ;       12
E51A 41445220
E51E 4F555420
E522 4F462052
E526 414E4745
E52A 0F         =1  1160             DB      15,('ASM PC ) OFFFFH')          ;       13
E52B 41534D20
E52F 50432029
E533 20304646
E537 464648
E53A 0D         =1  1161             DB      13,('FILE RD OR WR')            ;       14
E53B 46494C45
E53F 20524420
E543 4F522057
E547 52
```

```
LOC  OBJ            LINE     SOURCE

E548 0C         =1  1162             DB      12,('MEMORY WRITE')           ;       15
E549 4D454D4F
E54D 52592057
E551 52495445
E555 10         =1  1163             DB      16,('EX ACROSS ADR 03')       ;       16
E556 45582041
E55A 43524F53
E55E 53204144
E562 52203033
E566 10         =1  1164             DB      16,('NO RAM AT ADR 03')       ;       17
E567 4E4F2052
E56B 414D2041
E56F 54204144
E573 52203033
E577 0E         =1  1165             DB      14,('CBYTE TYPE REQ')         ;       18
E578 43425954
E57C 45205459
E580 50452052
E584 4551
E586 0B         =1  1166             DB      11,('CHANGE ONLY')            ;       19
E587 4348414E
E58B 4745204F
E58F 4E4C59
E592 0E         =1  1167             DB      14,('CBY OR NUM REQ')         ;       1A
E593 43425920
E597 4F52204E
E59B 554D2052
E59F 4551
                =1  1168 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1  1169     ;*****************************************************************
                =1  1170     ;
                =1  1171     ;    NAME: (I)EOL_CHECK
                =1  1172     ;
                =1  1173     ;    ABSTRACT: This routine will check for a carraige return and error
                =1  1174     ;       if one is not found.  It returns to calling routine if one is.
                =1  1175     ;
                =1  1176     ;    INPUTS: A (byte to be checked)
                =1  1177     ;
                =1  1178     ;    OUTPUTS: None
                =1  1179     ;
                =1  1180     ;    VARIABLES MODIFIED: ERRNUM
                =1  1181     ;
                =1  1182     ;    ERROR EXITS: 04H (CARRAIGE RETURN EXPECTED)
                =1  1183     ;
                =1  1184     ;    SUBROUTINES ACCESSED DIRECTLY: IERROR
                =1  1185     ;
                =1  1186     ;
                =1  1187     ;*****************************************************************
                =1  1188     IEOL_CHECK:
E5A1 B40701     =1  1189             CJNE    A,#EOL_TOKE,EOL_ERROR
E5A4 22         =1  1190             RET
                =1  1191     EOL_ERROR:
E5A5 754304     =1  1192             MOV     ERRNUM,#04H                    ;Carriage return expected
E5A8 61CA       =1  1193             JMP     IERROR
                =1  1194 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                    =1  1195     ;*********************************************************************
                    =1  1196     ;
                    =1  1197     ;    NAME: INC_PNT/ DEC_PNT/ SWAP_POINTERS
                    =1  1198     ;
                    =1  1199     ;    ABSTRACT: These are general purpose 16 bit arithmetic
                    =1  1200     ;       routines which will increment, decrement or swap pointers.
                    =1  1201     ;
                    =1  1202     ;    INPUTS: PNTLOW, PNTHGH, PCNTLO, PCNTHI
                    =1  1203     ;
                    =1  1204     ;    OUTPUTS: PNTLOW, PNTHGH, PNCTLO, PCNTHI
                    =1  1205     ;
                    =1  1206     ;    VARIABLES MODIFIED: A, PNTLOW, PNTHGH, PCNTLO, PCNTHI
                    =1  1207     ;
                    =1  1208     ;    ERROR EXITS: None
                    =1  1209     ;
                    =1  1210     ;    SUBROUTINES ACCESSED DIRECTLY: None
                    =1  1211     ;
                    =1  1212     ;
                    =1  1213     ;
                    =1  1214     ;*********************************************************************
E5AA 0545           =1  1215     INC_PNT:INC     PNTLOW
E5AC E545           =1  1216             MOV     A,PNTLOW
E5AE 7002           =1  1217             JNZ     INC_HIGH
E5B0 0544           =1  1218             INC     PNTHGH
                    =1  1219     INC_HIGH:
E5B2 22             =1  1220             RET
                    =1  1221     ;*********************************************************************
E5B3 1545           =1  1222     DEC_PNT:DEC     PNTLOW
E5B5 E545           =1  1223             MOV     A,PNTLOW
E5B7 F4             =1  1224             CPL     A
E5B8 7002           =1  1225             JNZ     DEC_HIGH
E5BA 1544           =1  1226             DEC     PNTHGH
                    =1  1227     DEC_HIGH:
E5BC 22             =1  1228             RET
                    =1  1229     ;*********************************************************************
                    =1  1230     SWAP_POINTERS:
E5BD E545           =1  1231             MOV     A,PNTLOW
E5BF 856245         =1  1232             MOV     PNTLOW,PCNTLO
E5C2 F562           =1  1233             MOV     PCNTLO,A
E5C4 E544           =1  1234             MOV     A,PNTHGH
E5C6 856144         =1  1235             MOV     PNTHGH,PCNTHI
E5C9 F561           =1  1236             MOV     PCNTHI,A
E5CB 22             =1  1237             RET
                    =1  1238 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1  1239     ;*********************************************************************
                =1  1240     ;
                =1  1241     ;    NAME: SPACCO/ (I)CO
                =1  1242     ;
                =1  1243     ;    ABSTRACT: Outputs a space to the system console, falls through
                =1  1244     ;       to ICO then returns.
                =1  1245     ;
                =1  1246     ;    INPUTS: PARAM1 (ASCII character to be printed)
                =1  1247     ;
                =1  1248     ;    OUTPUTS: None
                =1  1249     ;
                =1  1250     ;    VARIABLES MODIFIED: PARAM1
                =1  1251     ;
                =1  1252     ;    ERROR EXITS: None
                =1  1253     ;
                =1  1254     ;    SUBROUTINES ACCESSED DIRECTLY: UPI_OUT
                =1  1255     ;
                =1  1256     ;
                =1  1257     ;*********************************************************************
E5CC 7A20       =1  1258     SPACCO: MOV     PARAM1,#' '
E5CE 02E61E     =1  1259     ICO:    JMP     UPI_OUT
                =1  1260 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1  1261     ;*********************************************************************
                =1  1262     ;
                =1  1263     ;    NAME: ICI
                =1  1264     ;
                =1  1265     ;    ABSTRACT: Inputs an ASCII character from the system console, clears
                =1  1266     ;       the parity bit and converts to upper case.  If there is no
                =1  1267     ;       user abort, it returns to caller.
                =1  1268     ;
                =1  1269     ;    INPUTS: None
                =1  1270     ;
                =1  1271     ;    OUTPUTS: A
                =1  1272     ;
                =1  1273     ;    VARIABLES MODIFIED: A
                =1  1274     ;
                =1  1275     ;    ERROR EXITS: None
                =1  1276     ;
                =1  1277     ;    SUBROUTINES ACCESSED DIRECTLY:  IUPI_IN
                =1  1278     ;
                =1  1279     ;*********************************************************************
                =1  1280
E5D1 12E632     =1  1281     ICI:    CALL    UPI_IN
E5D4 C2E7       =1  1282             CLR     ACC.7                   ;Clear parity bit
E5D6 B46100     =1  1283             CJNE    A,#'a',UPI_INA
                =1  1284     UPI_INA:
E5D9 4007       =1  1285             JC      UPI_INR
E5DB B47B00     =1  1286             CJNE    A,#('z'+1),UPI_INB
                =1  1287     UPI_INB:
E5DE 5002       =1  1288             JNC     UPI_INR
E5E0 C2E5       =1  1289             CLR     ACC.5                   ;Convert to upper case
                =1  1290     UPI_INR:
E5E2 B41B02     =1  1291             CJNE    A,#ESC,UPI_INE          ;Abort if its an ESC key
E5E5 41C9       =1  1292             JMP     START
E5E7 22         =1  1293     UPI_INE:        RET                     ;And return to the caller.
                =1  1294 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                    =1  1295     ;*****************************************************************
                    =1  1296     ;
                    =1  1297     ;    NAME: ICSTS
                    =1  1298     ;
                    =1  1299     ;    ABSTRACT: Returns carry=1 if there is a character waiting from
                    =1  1300     ;       the system console.  If no character is ready, carry will be
                    =1  1301     ;       cleared. CAUTION: this is not available for use except to the
                    =1  1302     ;       monitor itself.  See UCSTS for a general purpose version of
                    =1  1303     ;       this routine.
                    =1  1304     ;
                    =1  1305     ;    INPUTS: None
                    =1  1306     ;
                    =1  1307     ;    OUTPUTS: Carry bit (C)
                    =1  1308     ;
                    =1  1309     ;    VARIABLES MODIFIED: DPTR, A, C, 2 locations of the stack
                    =1  1310     ;
                    =1  1311     ;    ERROR EXITS: None
                    =1  1312     ;
                    =1  1313     ;    SUBROUTINES ACCESSED DIRECTLY: None
                    =1  1314     ;
                    =1  1315     ;
                    =1  1316     ;*****************************************************************
E5E8 C082           =1  1317     ICSTS:  PUSH    DPL
E5EA C083           =1  1318             PUSH    DPH
E5EC 90A001         =1  1319             MOV     DPTR,#UPI_CONTROL
E5EF E0             =1  1320     CSTS_1: MOVX    A,@DPTR
E5F0 20E2FC         =1  1321             JB      ACC.2,CSTS_1            ;Wait for status to be valid
E5F3 13             =1  1322             RRC     A                       ;Rotate UPI OBF into CARRY
E5F4 D083           =1  1323             POP     DPH
E5F6 D082           =1  1324             POP     DPL
E5F8 22             =1  1325             RET
                    =1  1326 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1  1327     ;*****************************************************************
                =1  1328     ;
                =1  1329     ;    NAME: (U)CSTS
                =1  1330     ;
                =1  1331     ;    ABSTRACT: This routine gets the console status bit from bit 7
                =1  1332     ;       of the accumulator into carry.  Carry = 1 if a character
                =1  1333     ;       is present.
                =1  1334     ;
                =1  1335     ;       Users writing application programs should use
                =1  1336     ;       this routine instead of ICSTS.  This reflects the buffered
                =1  1337     ;       version of the console port.
                =1  1338     ;
                =1  1339     ;    INPUTS: None
                =1  1340     ;
                =1  1341     ;    OUTPUTS: Carry bit (C)
                =1  1342     ;
                =1  1343     ;    VARIABLES MODIFIED: DPTR, A
                =1  1344     ;
                =1  1345     ;    ERROR EXITS: None
                =1  1346     ;
                =1  1347     ;    SUBROUTINES ACCESSED DIRECTLY: None
                =1  1348     ;
                =1  1349     ;
                =1  1350     ;*****************************************************************
                =1  1351
E5F9 90B0F1     =1  1352     UCSTS:  MOV     DPTR,#(RAMOFF+UPI_DATA_IMAGE)
E5FC E0         =1  1353             MOVX    A,@DPTR
E5FD 33         =1  1354             RLC     A
E5FE 22         =1  1355             RET
                =1  1356 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1  1357     ;*****************************************************************
                =1  1358     ;
                =1  1359     ;    NAME: (U)CI
                =1  1360     ;
                =1  1361     ;    ABSTRACT: This routine waits for the console status bit to
                =1  1362     ;       indicate that a character is ready (C=1), inputs it from
                =1  1363     ;       the console, clears the status bit and returns.
                =1  1364     ;
                =1  1365     ;       Users writing application programs should use
                =1  1366     ;       this routine instead of ICSTS.  This reflects the buffered
                =1  1367     ;       version of the console port.
                =1  1368     ;
                =1  1369     ;    INPUTS: None
                =1  1370     ;
                =1  1371     ;    OUTPUTS: UPI_DATA_IMAGE
                =1  1372     ;
                =1  1373     ;    VARIABLES MODIFIED: DPTR, A
                =1  1374     ;
                =1  1375     ;    ERROR EXITS: None
                =1  1376     ;
                =1  1377     ;    SUBROUTINES ACCESSED DIRECTLY: UCSTS
                =1  1378     ;
                =1  1379     ;
                =1  1380     ;*****************************************************************
E5FF B1F9       =1  1381     UCI:    CALL    UCSTS
E601 50FC       =1  1382             JNC     UCI
E603 90B0F1     =1  1383             MOV     DPTR,#(RAMOFF+UPI_DATA_IMAGE)
E606 E0         =1  1384             MOVX    A,@DPTR
E607 C2E7       =1  1385             CLR     ACC.7
E609 F0         =1  1386             MOVX    @DPTR,A
E60A 22         =1  1387             RET
                =1  1388 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1  1389     ;*********************************************************************
                =1  1390     ;
                =1  1391     ;    NAME: (I)UPI_CMD
                =1  1392     ;
                =1  1393     ;    ABSTRACT: Waits till the UPI is ready and then outputs a
                =1  1394     ;       command to it.
                =1  1395     ;
                =1  1396     ;    INPUTS: PARAM1=byte to be sent to UPI command port
                =1  1397     ;
                =1  1398     ;    OUTPUTS: None
                =1  1399     ;
                =1  1400     ;    VARIABLES MODIFIED: A, 2 locations in the stack
                =1  1401     ;
                =1  1402     ;    ERROR EXITS: None
                =1  1403     ;
                =1  1404     ;    SUBROUTINES ACCESSED DIRECTLY: None
                =1  1405     ;
                =1  1406     ;
                =1  1407     ;*********************************************************************
                =1  1408     UPI_CMD:
E60B C082       =1  1409             PUSH    DPL                     ;Save DPTR in the stack.
E60D C083       =1  1410             PUSH    DPH
E60F 90A001     =1  1411             MOV     DPTR,#UPI_CONTROL       ;Point to UPI control channel
                =1  1412     UPI_C_1:
E612 E0         =1  1413             MOVX    A,@DPTR                 ;And wait for valid status.
E613 5416       =1  1414             ANL     A,#16H
E615 70FB       =1  1415             JNZ     UPI_C_1
E617 EA         =1  1416             MOV     A,PARAM1                ;Then send out the command.
E618 F0         =1  1417             MOVX    @DPTR,A
E619 D083       =1  1418             POP     DPH                     ;Restore DPTR
E61B D082       =1  1419             POP     DPL
E61D 22         =1  1420             RET                             ;Return to caller.
                =1  1421 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1  1422     ;*********************************************************************
                =1  1423     ;
                =1  1424     ;    NAME: UPI_OUT
                =1  1425     ;
                =1  1426     ;    ABSTRACT: Waits until the UPI is ready and then outputs data to it.
                =1  1427     ;
                =1  1428     ;    INPUTS: PARAM1 = data to be sent to UPI
                =1  1429     ;
                =1  1430     ;    OUTPUTS: None
                =1  1431     ;
                =1  1432     ;    VARIABLES MODIFIED: A, 2 locations on the stack
                =1  1433     ;
                =1  1434     ;    ERROR EXITS: None
                =1  1435     ;
                =1  1436     ;    SUBROUTINES ACCESSED DIRECTLY: None
                =1  1437     ;
                =1  1438     ;
                =1  1439     ;*********************************************************************
E61E C082       =1  1440     UPI_OUT:PUSH    DPL                     ;Save DPTR in the stack.
E620 C083       =1  1441             PUSH    DPH
E622 90A001     =1  1442             MOV     DPTR,#UPI_CONTROL       ;Point to UPI control channel
E625 E0         =1  1443     UPI_0_1:MOVX    A,@DPTR                 ;and wait for valid status.
E626 5416       =1  1444             ANL     A,#16H
E628 70FB       =1  1445             JNZ     UPI_0_1
E62A A3         =1  1446             INC     DPTR                    ;Point to data port
E62B EA         =1  1447             MOV     A,PARAM1
E62C F0         =1  1448             MOVX    @DPTR,A
E62D D083       =1  1449             POP     DPH                     ;Restore DPTR
E62F D082       =1  1450             POP     DPL
E631 22         =1  1451             RET                             ;Return to caller.
                =1  1452 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1  1453     ;*****************************************************************
                =1  1454     ;
                =1  1455     ;    NAME: UPI_IN
                =1  1456     ;
                =1  1457     ;    ABSTRACT: Waits for a character from the UPI and returns it to
                =1  1458     ;       the caller in the accumulator.
                =1  1459     ;
                =1  1460     ;    INPUTS: None
                =1  1461     ;
                =1  1462     ;    OUTPUTS: A
                =1  1463     ;
                =1  1464     ;    VARIABLES MODIFIED: A, 2 locations of the stack
                =1  1465     ;
                =1  1466     ;    ERROR EXITS: None
                =1  1467     ;
                =1  1468     ;    SUBROUTINES ACCESSED DIRECTLY: ICSTS
                =1  1469     ;
                =1  1470     ;
                =1  1471     ;*****************************************************************
E632 B1E8       =1  1472     UPI_IN: CALL    ICSTS
E634 50FC       =1  1473             JNC     UPI_IN                  ;Wait for character
E636 C082       =1  1474             PUSH    DPL
E638 C083       =1  1475             PUSH    DPH
E63A 90A000     =1  1476             MOV     DPTR,#UPI_DATA          ;Point to UPI data port
E63D E0         =1  1477             MOVX    A,@DPTR                 ;Get byte
E63E D083       =1  1478             POP     DPH                     ;Restore DPTR
E640 D082       =1  1479             POP     DPL
E642 22         =1  1480             RET                             ;and return to the caller
                =1  1481 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1  1482     ;*********************************************************************
                =1  1483     ;
                =1  1484     ;    NAME: (I)CONTINUATION_LINE
                =1  1485     ;
                =1  1486     ;    ABSTRACT: This routine looks for LIST=ON.  If there is no user
                =1  1487     ;       abort, it gets a character and returns.  If LIST=RESET,
                =1  1488     ;       it outputs a blinking comma to the display, discards the
                =1  1489     ;       character, waits for the user to hit any key and returns.
                =1  1490     ;
                =1  1491     ;    INPUTS: LSTFLG
                =1  1492     ;
                =1  1493     ;    OUTPUTS: None
                =1  1494     ;
                =1  1495     ;    VARIABLES MODIFIED: PARAM1
                =1  1496     ;
                =1  1497     ;    ERROR EXITS: None
                =1  1498     ;
                =1  1499     ;    SUBROUTINES ACCESSED DIRECTLY: ICO, ICI, ICSTS
                =1  1500     ;
                =1  1501     ;
                =1  1502     ;*********************************************************************
                =1  1503     ICONTINUATION_LINE:
E643 200106     =1  1504          JB      LSTFLG,DONT_WAIT
E646 7AAC       =1  1505          MOV     PARAM1,#(',+BLINK)
E648 B1CE       =1  1506          CALL    ICO
                =1  1507     CHECK_ESC:
E64A 8085       =1  1508          JMP     ICI
                =1  1509     DONT_WAIT:
E64C B1E8       =1  1510          CALL    ICSTS
E64E 40FA       =1  1511          JC      CHECK_ESC
E650 22         =1  1512          RET
                =1  1513 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                    =1  1514     ;*****************************************************************
                    =1  1515     ;
                    =1  1516     ;    NAME: (I)FETCH/(I)STORE
                    =1  1517     ;
                    =1  1518     ;    ABSTRACT:
                    =1  1519     ;       This routine reads or writes one byte of data.  SELECT indicates
                    =1  1520     ;       the type of memory operation to be performed.  The following
                    =1  1521     ;       table lists the values of SELECT:
                    =1  1522     ;               0H) CBYTE - Program memory
                    =1  1523     ;               1H) RBYTE - Register memory
                    =1  1524     ;               2H) DBYTE - Internal data memory
                    =1  1525     ;               3H) Not used
                    =1  1526     ;               4H) RBIT - Bit memory
                    =1  1527     ;               5H) Not used
                    =1  1528     ;               6H) XBYTE - External data memory
                    =1  1529     ;       PNTLOW holds lower 8 bits of address
                    =1  1530     ;       PNTHGH Holds upper 8 bits of address and must be
                    =1  1531     ;               zeroed out if not used
                    =1  1532     ;       PARAM1 holds value to be stored, is only used by STORE
                    =1  1533     ;       A      holds the result of the fetch
                    =1  1534     ;       CBYTE does a read after write to verify byte value written,
                    =1  1535     ;               (i.e. detects writes to ROM).
                    =1  1536     ;
                    =1  1537     ;    INPUTS: SELECT, PARAM1, PNTLO
                    =1  1538     ;
                    =1  1539     ;    OUTPUTS: A, contents of memory being addressed
                    =1  1540     ;
                    =1  1541     ;    VARIABLES MODIFIED: A, PSW, DPTR, ERRNUM, TEMP1, B, C
                    =1  1542     ;
                    =1  1543     ;    ERROR EXITS: 12H (ADDRESS OUT OF RANGE )
                    =1  1544     ;                 15H (READ AFTER WRITE ERROR)
                    =1  1545     ;
                    =1  1546     ;    SUBROUTINES ACCESSED DIRECTLY: IERROR
                    =1  1547     ;
                    =1  1548     ;
                    =1  1549     ;*****************************************************************
E651 E546           =1  1550     IFETCH: MOV     A,SELECT                        ;Data value passed from calling routine
E653 C2D5           =1  1551             CLR     F0                              ;Zero = read memory
E655 02E65C         =1  1552             JMP     MEMORY
E658 E546           =1  1553     ISTORE: MOV     A,SELECT
E65A D2D5           =1  1554             SETB    F0                              ;One = write memory
E65C 854483         =1  1555     MEMORY: MOV     DPH, PNTHGH
E65F 854582         =1  1556             MOV     DPL, PNTLOW                     ;Put addr in data pointer
E662 B40012         =1  1557             CJNE    A,#(CBYTE_TOKE AND 07H),XBYTE
E665 30D50A         =1  1558             JNB     F0,C_READ                       ;Jump if not CBYTE
E668 EA             =1  1559             MOV     A,PARAM1
E669 F0             =1  1560             MOVX    @DPTR,A                         ;Program memory write
E66A E4             =1  1561             CLR     A
E66B 93             =1  1562             MOVC    A,@A+DPTR                       ;Program memory read after write
E66C 6A             =1  1563             XRL     A,PARAM1
E66D 7041           =1  1564             JNZ     FETERR                          ;Verify error if read doesn't match write
E66F 02E683         =1  1565             JMP     FETEND
E672 E4             =1  1566     C_READ: CLR     A
E673 93             =1  1567             MOVC    A,@A+DPTR                       ;Program memory read
E674 02E683         =1  1568             JMP     FETEND
```

```
LOC  OBJ            LINE     SOURCE

E677 B4060C     =1  1569     XBYTE:  CJNE    A,#(XBYTE_TOKE AND 07H),RBYTE   ;Check if external RAM was selected
E67A 20D504     =1  1570             JB      F0,XWRITE                       ;Jump to STORE if flag is set
E67D E0         =1  1571     XREAD:  MOVX    A,@DPTR                         ;Load EXT RAM into ACC
E67E 02E683     =1  1572             JMP     FETEND
E681 EA         =1  1573     XWRITE: MOV     A,PARAM1                        ;Load ACC with data to be output
E682 F0         =1  1574     X_WRT:  MOVX    @DPTR,A                         ;Output ACC to EXT RAM
E683 C2D5       =1  1575     FETEND: CLR     F0                              ;Clear flag
E685 22         =1  1576             RET
E686 90B000     =1  1577     RBYTE:  MOV     DPTR,#RAMOFF                    ;Load DPTR with base addr of 8155 RAM
E689 754312     =1  1578             MOV     ERRNUM,#12H                     ;Address out of range
E68C E544       =1  1579             MOV     A,PNTHGH
E68E 7023       =1  1580             JNZ     ERR                             ;Error if address is not 00XXH
E690 E546       =1  1581             MOV     A,SELECT
E692 B4010C     =1  1582             CJNE    A,#(RBYTE_TOKE AND 07H),DBYTE   ;Jump if not RBYTE
E695 E545       =1  1583             MOV     A,PNTLOW
E697 30E719     =1  1584             JNB     ACC.7,ERR                       ;Error if address is between 0 and 7FH
E69A F582       =1  1585             MOV     DPL,A
E69C 20D5E2     =1  1586             JB      F0,XWRITE                       ;Jump to STORE if flag is set
E69F 80DC       =1  1587             JMP     XREAD                           ;Exit from FETCH
E6A1 B40211     =1  1588     DBYTE:  CJNE    A,#(DBYTE_TOKE AND 07H),RBIT    ;Jump if RBIT is selected
E6A4 E545       =1  1589             MOV     A,PNTLOW                        ;Load ACC with low pointer
E6A6 20E70A     =1  1590             JB      ACC.7,ERR                       ;Error if addr if between 80H and FFH
E6A9 F582       =1  1591             MOV     DPL,A                           ;Load DPL with new low point value
E6AB 20D5D3     =1  1592             JB      F0,XWRITE                       ;Jump to STORE if flag is set
E6AE 80CD       =1  1593             JMP     XREAD                           ;Exit from FETCH
E6B0 754315     =1  1594     FETERR: MOV     ERRNUM,#15H                     ;Read after write did not match.
E6B3 61CA       =1  1595     ERR:    JMP     IERROR                          ;Exit from FETCH/STORE
E6B5 B404CB     =1  1596     RBIT:   CJNE    A,#(RBIT_TOKE AND 07H),FETEND   ;Check if selector is for direct bit
E6B8 E545       =1  1597             MOV     A,PNTLOW                        ;Load ACC with pointer
E6BA 54F8       =1  1598             ANL     A,#0F8H                         ;Mask off lower 3 bits
E6BC 20E705     =1  1599             JB      ACC.7,SPEFUN                    ;Jump to register bit array if over 7FH
E6BF 13         =1  1600             RRC     A
E6C0 03         =1  1601             RR      A
E6C1 03         =1  1602             RR      A                               ;Rotate ACC to obtain correct addr
E6C2 2420       =1  1603             ADD     A,#20H                          ;Add offset of internal bit registers
E6C4 2582       =1  1604     SPEFUN: ADD     A,DPL                           ;Add offset to pointer
E6C6 F582       =1  1605             MOV     DPL,A                           ;Load DPL with new addr
E6C8 20D513     =1  1606             JB      F0,BITSTR                       ;Jump to STORE if flag is set
E6CB E0         =1  1607             MOVX    A,@DPTR                         ;Move INT RAM simulator byte into ACC
E6CC 854556     =1  1608             MOV     TEMP1,PNTLOW                    ;Move pointer into TEMP1
E6CF 535607     =1  1609             ANL     TEMP1,#07H                      ;Mask lower 3 bits
E6D2 0556       =1  1610             INC     TEMP1                           ;For DJNZ
E6D4 D55604     =1  1611     BITLOP: DJNZ    TEMP1,BITROT                    ;Loop until PARAM1=0
E6D7 5401       =1  1612             ANL     A,#1                            ;Mask lowest bit
E6D9 80A8       =1  1613             JMP     FETEND                          ;Exit from FETCH
E6DB 03         =1  1614     BITROT: RR      A                               ;Rotate until PARAM1=0
E6DC 80F6       =1  1615             JMP     BITLOP
E6DE 854556     =1  1616     BITSTR: MOV     TEMP1,PNTLOW                    ;Load TEMP1 with pointer
E6E1 535607     =1  1617             ANL     TEMP1,#07H                      ;Mask lower 3 bits
E6E4 0556       =1  1618             INC     TEMP1                           ;For DJNZ
E6E6 E0         =1  1619             MOVX    A,@DPTR                         ;Load ACC with data in RAM simulator
E6E7 13         =1  1620     RHTROT: RRC     A
E6E8 D556FC     =1  1621             DJNZ    TEMP1,RHTROT                    ;Rotate until pointer reaches zero
E6EB 8AF0       =1  1622             MOV     B,PARAM1                        ;Move data to be output into B reg
E6ED A2F0       =1  1623             MOV     C,B.0                           ;Move into carry data to be output
```

```
LOC  OBJ            LINE     SOURCE

E6EF 854556     =1  1624             MOV     TEMP1,PNTLOW                ;Load TEMP1 with pointer
E6F2 535607     =1  1625             ANL     TEMP1,#07H                  ;Mask lower 3 bits for counter
E6F5 0556       =1  1626             INC     TEMP1
E6F7 33         =1  1627     LFTROT: RLC     A
E6F8 D556FC     =1  1628             DJNZ    TEMP1,LFTROT                ;Rotate left until TEMP1 reaches zero
E6FB 8085       =1  1629             JMP     X_WRT
                =1  1630 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1  1631     ;*********************************************************************
                =1  1632     ;
                =1  1633     ;    NAME: (I)NEWLINE
                =1  1634     ;
                =1  1635     ;    ABSTRACT: Outputs a CR/LF to the console device.
                =1  1636     ;
                =1  1637     ;    INPUTS: None
                =1  1638     ;
                =1  1639     ;    OUTPUTS: None
                =1  1640     ;
                =1  1641     ;    VARIABLES MODIFIED: PARAM1
                =1  1642     ;
                =1  1643     ;    ERROR EXITS: None
                =1  1644     ;
                =1  1645     ;    SUBROUTINES ACCESSED DIRECTLY: ICO
                =1  1646     ;
                =1  1647     ;
                =1  1648     ;*********************************************************************
                =1  1649     INEWLINE:
E6FD 7A0D       =1  1650             MOV     PARAM1,#CR              ;Output a CR
E6FF B1CE       =1  1651             CALL    ICO
E701 7A0A       =1  1652             MOV     PARAM1,#LF              ;Output a LF
E703 B1CE       =1  1653             CALL    ICO
E705 22         =1  1654             RET
                =1  1655 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1  1656     ;*****************************************************************
                =1  1657     ;
                =1  1658     ;    NAME: AZTEST / NMTEST / HXTEST / ALFNUM
                =1  1659     ;
                =1  1660     ;    ABSTRACT: AZTEST will check to see if the input character is
                =1  1661     ;       an ASCII value between @ and Z.  Carry is set if it is.
                =1  1662     ;       NMTEST will check to see if the character was an ASCII number
                =1  1663     ;       between 0 and 9 and set carry if true.  HXTEST will look for the
                =1  1664     ;       ASCII representation of a hex value 0-9 and A-F and will set carry
                =1  1665     ;       if true.  ALFNUM will test for character to be alpha or numeric
                =1  1666     ;       and set carry if true.
                =1  1667     ;
                =1  1668     ;    INPUTS: PARAM1 (byte to be checked)
                =1  1669     ;
                =1  1670     ;    OUTPUTS: Carry bit (C)
                =1  1671     ;
                =1  1672     ;    VARIABLES MODIFIED: A, C
                =1  1673     ;
                =1  1674     ;    ERROR EXITS: None
                =1  1675     ;
                =1  1676     ;    SUBROUTINES ACCESSED DIRECTLY: None
                =1  1677     ;
                =1  1678     ;
                =1  1679     ;*****************************************************************
E706 EA         =1  1680     AZTEST: MOV     A,PARAM1        ;Move char to be tested into ACC
E707 B44002     =1  1681             CJNE    A,#'@',ZTEST    ;Carry will reset if char is <= '@'
E70A 8005       =1  1682             SJMP    CARSET          ;Set carry if equal to '@'
E70C 4003       =1  1683     ZTEST:  JC      CARSET          ;Reset carry if char is <= '@'
E70E B45A01     =1  1684             CJNE    A,#'Z',AZEND    ;Carry will set if char is <= 'Z'
E711 B3         =1  1685     CARSET: CPL     C               ;Set carry if equal to 'Z'
E712 22         =1  1686     AZEND:  RET                     ;Exit from AZTEST
                =1  1687     ;*****************************************************************
E713 EA         =1  1688     NMTEST:MOV      A,PARAM1        ;Move char into ACC
E714 C3         =1  1689             CLR     C
E715 9430       =1  1690             SUBB    A,#('0')        ;See if char is <= ASCII '0'
E717 B3         =1  1691             CPL     C
E718 5002       =1  1692             JNC     NUMEND          ;Carry left 0 if false
E71A 940A       =1  1693             SUBB    A,#('9'-'0'+1)  ;See if char is > ASCII '9'
E71C 22         =1  1694     NUMEND: RET                     ;Exit from NMTEST
                =1  1695     ;*****************************************************************
E71D F113       =1  1696     HXTEST: CALL NMTEST             ;See if char is between '0' and '9'
                =1  1697                                     ;Extra level of subroutine added
E71F 4008       =1  1698             JC      HEXEND          ;Jump if char between '0' and '9'
E721 EA         =1  1699             MOV     A,PARAM1        ;Move char into ACC
E722 9441       =1  1700             SUBB    A,#'A'          ;See if char is > 'A'
E724 B3         =1  1701             CPL     C
E725 5002       =1  1702             JNC     HEXEND          ;Carry left 0 if false
E727 9405       =1  1703             SUBB    A,#('F'-'A')    ;See if char is less than 'F'
E729 22         =1  1704     HEXEND: RET                     ;Exit from HXTEST
                =1  1705     ;*****************************************************************
E72A F106       =1  1706     ALFNUM: CALL    AZTEST          ;See if char is between '@' and 'Z'
                =1  1707                                     ;Add extra level of subroutine
E72C 4002       =1  1708             JC      ANEND           ;Carry set if true
E72E F113       =1  1709             CALL    NMTEST          ;See if char is between '0' and '9'
                =1  1710                                     ;Added extra level of subroutine
```

```
LOC  OBJ            LINE     SOURCE

E730 22         =1  1711     ANEND:  RET                     ;Exit from ALFNUM
                =1  1712 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1  1713     ;*********************************************************************
                =1  1714     ;
                =1  1715     ;    NAME: LSSEQL
                =1  1716     ;
                =1  1717     ;    ABSTRACT: This is a 16-bit 'less than' or 'equal' check.  The
                =1  1718     ;       carry bit is set to indicate true. If MAXNUM_FLAGS is
                =1  1719     ;       true, no check is made.
                =1  1720     ;
                =1  1721     ;    INPUTS: PARAM1 (high byte to be compared to)
                =1  1722     ;            PARAM2 (low byte to be compared to)
                =1  1723     ;            PARAM3 (high byte to be compared)
                =1  1724     ;            PARAM4 (low byte to be compared)
                =1  1725     ;
                =1  1726     ;    OUTPUTS: Carry bit (C)
                =1  1727     ;
                =1  1728     ;    VARIABLES MODIFIED: C, MAXNUM_FLAG, PARAM1
                =1  1729     ;
                =1  1730     ;    ERROR EXITS: None
                =1  1731     ;
                =1  1732     ;    SUBROUTINES ACCESSED DIRECTLY: None
                =1  1733     ;
                =1  1734     ;
                =1  1735     ;*********************************************************************
E731 200417     =1  1736     LSSEQL: JB      MAXNUM_FLAG,LAB1B
E734 BCFF05     =1  1737             CJNE    PARAM3,#0FFH,START_COMPARE
E737 BDFF02     =1  1738             CJNE    PARAM4,#0FFH,START_COMPARE
E73A D204       =1  1739             SETB    MAXNUM_FLAG
                =1  1740     START_COMPARE:
E73C C3         =1  1741             CLR     C
E73D EB         =1  1742             MOV     A,PARAM2        ;Move byte to be compared to into ACC
E73E 9D         =1  1743             SUBB    A,PARAM4        ;Subtract byte to be compared
E73F 5006       =1  1744             JNC     LAB1
E741 1A         =1  1745             DEC     PARAM1          ;Decrement upper byte if lower byte was smaller
E742 EA         =1  1746             MOV     A,PARAM1
E743 F4         =1  1747             CPL     A
E744 C3         =1  1748             CLR     C
E745 6003       =1  1749             JZ      LAB1A           ;Error if PARAM1 decremented to FF
E747 EA         =1  1750     LAB1:   MOV     A,PARAM1        ;Move upper byte to be compared,to into ACC
E748 9C         =1  1751             SUBB    A,PARAM3        ;Subtract upper byte to be compared
E749 B3         =1  1752             CPL     C               ;Set C if <= is true
E74A 22         =1  1753     LAB1A:  RET                     ;Exit from LSSEQL
E74B C204       =1  1754     LAB1B:  CLR     MAXNUM_FLAG
E74D C3         =1  1755             CLR     C
E74E 22         =1  1756             RET
                =1  1757 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1  1758     ;*********************************************************************
                =1  1759     ;
                =1  1760     ;    NAME: (I)GETNUM / (I)GETEOL / (I)GET_COMMA
                =1  1761     ;
                =1  1762     ;    ABSTRACT: These routines are general purpose token checks.
                =1  1763     ;       IGETNUM will get a token and error if it is not
                =1  1764     ;       a number token, it will return if it is.  IGETEOL will
                =1  1765     ;       look for an end-of-line token and error if it is not
                =1  1766     ;       found, it will return if it is.  IGET_COMMA will look for
                =1  1767     ;       a comma token and will error if one is not found and return
                =1  1768     ;       if it is.
                =1  1769     ;
                =1  1770     ;    INPUTS: None
                =1  1771     ;
                =1  1772     ;    OUTPUTS: None
                =1  1773     ;
                =1  1774     ;    VARIABLES MODIFIED: ERRNUM
                =1  1775     ;
                =1  1776     ;    ERROR EXITS: 03H (NUMBER EXPECTED)
                =1  1777     ;                 06H (COMMA REQUIRED)
                =1  1778     ;
                =1  1779     ;    SUBROUTINES ACCESSED DIRECTLY: IERROR, IGETOKE
                =1  1780     ;
                =1  1781     ;
                =1  1782     ;*********************************************************************
E74F 12E8A0     =1  1783     IGETNUM:CALL    IGETOKE
E752 754303     =1  1784             MOV     ERRNUM,#03H            ;Number expected
E755 B40106     =1  1785             CJNE    A,#NUMBER_TOKE,UTILIT_ERROR
E758 22         =1  1786             RET
                =1  1787     ;*********************************************************************
E759 12E8A0     =1  1788     IGETEOL:CALL    IGETOKE
E75C A1A1       =1  1789             JMP     IEOL_CHECK             ;Check for end of line token
                =1  1790     UTILIT_ERROR:
E75E 61CA       =1  1791             JMP     IERROR
                =1  1792     ;*********************************************************************
                =1  1793     IGET_COMMA:
E760 12E8A0     =1  1794             CALL    IGETOKE
E763 754306     =1  1795             MOV     ERRNUM,#06H            ;Comma required
E766 B402F5     =1  1796             CJNE    A,#COMMA_TOKE,UTILIT_ERROR
E769 22         =1  1797             RET
                =1  1798 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1  1799     ;*****************************************************************
                =1  1800     ;
                =1  1801     ;    NAME: ISIT_DISPLAY
                =1  1802     ;
                =1  1803     ;    ABSTRACT: This routine checks for an equal or an EOL token,
                =1  1804     ;       sends the command token to the display with an = sign and
                =1  1805     ;       sets carry if and equal sign is found.  Carry is cleared
                =1  1806     ;       if an EOL is found..  The value is filled in by another routine.
                =1  1807     ;
                =1  1808     ;    INPUTS: TOKSTR
                =1  1809     ;
                =1  1810     ;    OUTPUTS: Carry bit (C)
                =1  1811     ;
                =1  1812     ;    VARIABLES MODIFIED: C, TOKSAV, PARAM1
                =1  1813     ;
                =1  1814     ;    ERROR EXITS: 05H (EQUAL OR RETURN EXPECTED)
                =1  1815     ;
                =1  1816     ;    SUBROUTINES ACCESSED DIRECTLY: IGETOKE, INEWLINE, ICO, IERROR
                =1  1817     ;
                =1  1818     ;
                =1  1819     ;*****************************************************************
                =1  1820     ISIT_DISPLAY:
E76A C3         =1  1821             CLR     C
E76B 85485B     =1  1822             MOV     TOKSAV,TOKSTR
E76E 12E8A0     =1  1823             CALL    IGETOKE
E771 B4070D     =1  1824             CJNE    A,#EOL_TOKE,CHANGE_CHECK
E774 D1FD       =1  1825             CALL    INEWLINE
E776 AA5B       =1  1826             MOV     PARAM1,TOKSAV
E778 12E9E0     =1  1827             CALL    IDISPLAY_TOKEN
E77B 7A3D       =1  1828             MOV     PARAM1,#'='
E77D B1CE       =1  1829             CALL    ICO
E77F D3         =1  1830             SETB    C
E780 22         =1  1831             RET
                =1  1832     CHANGE_CHECK:
E781 754305     =1  1833             MOV     ERRNUM,#05H             ;Equal or return expected
E784 B404D7     =1  1834             CJNE    A,#EQUAL_TOKE,UTILIT_ERROR
E787 22         =1  1835             RET
                =1  1836 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1  1837     ;*****************************************************************
                =1  1838     ;
                =1  1839     ;    NAME: (I)GET_PART
                =1  1840     ;
                =1  1841     ;    ABSTRACT: This routine checks a token which is expected to be
                =1  1842     ;       a number, sets up the partition addresses and looks for
                =1  1843     ;       the upper partition limits from the user.  Carry will be set
                =1  1844     ;       if there is a partition or if there is an error condition.
                =1  1845     ;       The partition range, or length, will also be calculated.
                =1  1846     ;
                =1  1847     ;    INPUTS: TOKSTR, VALLOW, VALHGH
                =1  1848     ;
                =1  1849     ;    OUTPUTS: Carry bit (C)
                =1  1850     ;
                =1  1851     ;    VARIABLES MODIFIED: A, ERRNUM, PARTIT_HI_LOW, PARTIT_HI_HIGH,
                =1  1852     ;       PARTIT_LO_LOW, PARTIT_LO_HIGH, C, LENGTH_LOW, LENGTH_HIGH
                =1  1853     ;
                =1  1854     ;    ERROR EXITS: 07H (PARTITION ERROR, LOW ADDR > HIGH ADDR)
                =1  1855     ;
                =1  1856     ;    SUBROUTINES ACCESSED DIRECTLY: IGETOKE, IGETNUM, IERROR
                =1  1857     ;
                =1  1858     ;
                =1  1859     ;*****************************************************************
                =1  1860     IGET_PART:
E788 E548       =1  1861             MOV     A,TOKSTR
E78A 754303     =1  1862             MOV     ERRNUM,#03H             ;Number expected
E78D B401CE     =1  1863             CJNE    A,#NUMBER_TOKE,UTILIT_ERROR     ;Set EA and SA to the value of the number.
E790 854A5A     =1  1864             MOV     PARTIT_HI_LOW,VALLOW
E793 854959     =1  1865             MOV     PARTIT_HI_HIGH,VALHGH
E796 854A58     =1  1866             MOV     PARTIT_LO_LOW,VALLOW
E799 854957     =1  1867             MOV     PARTIT_LO_HIGH,VALHGH
E79C 12E8A0     =1  1868             CALL    IGETOKE                 ;Get the next token.
E79F B40D1F     =1  1869             CJNE    A,#TO_TOKE,PARTITION_E  ;else set EA to the ending address of
E7A2 F14F       =1  1870             CALL    IGETNUM                 ;the partition
E7A4 854A5A     =1  1871             MOV     PARTIT_HI_LOW,VALLOW
E7A7 854959     =1  1872             MOV     PARTIT_HI_HIGH,VALHGH
E7AA C3         =1  1873             CLR     C
E7AB E55A       =1  1874             MOV     A,PARTIT_HI_LOW
E7AD 9558       =1  1875             SUBB    A,PARTIT_LO_LOW
E7AF F564       =1  1876             MOV     LENGTH_LOW,A
E7B1 E559       =1  1877             MOV     A,PARTIT_HI_HIGH
E7B3 9557       =1  1878             SUBB    A,PARTIT_LO_HIGH
E7B5 F563       =1  1879             MOV     LENGTH_HIGH,A
E7B7 754307     =1  1880             MOV     ERRNUM,#07H             ;Partition error, low adr > high adr
E7BA 40A2       =1  1881             JC      UTILIT_ERROR
E7BC 12E8A0     =1  1882             CALL    IGETOKE                 ;and then read in the next token.
E7BF D3         =1  1883             SETB    C
E7C0 22         =1  1884             RET
                =1  1885     PARTITION_E:
E7C1 C3         =1  1886             CLR     C
E7C2 22         =1  1887             RET
                =1  1888 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1  1889     ;*****************************************************************
                =1  1890     ;
                =1  1891     ;    NAME: (I)SAVE_AND_DISPLAY
                =1  1892     ;
                =1  1893     ;    ABSTRACT: This routine will convert a hex byte into two ASCII
                =1  1894     ;       characters for display the next time PAINTER is called.
                =1  1895     ;       POINTO must be set before calling this routine to the character
                =1  1896     ;       position desired on the screen (ie LINBUF or LINBUF+n).  LNLGTH
                =1  1897     ;       and CHRCNT are not adjusted by this routine.
                =1  1898     ;
                =1  1899     ;    INPUTS: POINTO (the location in the line buffer desired), PARAM1
                =1  1900     ;       (the character to be displayed)
                =1  1901     ;
                =1  1902     ;    OUTPUTS: POINTO, 1 location in the line buffer
                =1  1903     ;
                =1  1904     ;    VARIABLES MODIFIED: POINTO, A, 1 location in the line buffer
                =1  1905     ;
                =1  1906     ;    ERROR EXITS: None
                =1  1907     ;
                =1  1908     ;    SUBROUTINES ACCESSED DIRECTLY: CONVHEX
                =1  1909     ;
                =1  1910     ;
                =1  1911     ;*****************************************************************
                =1  1912     ISAVE_AND_DISPLAY:
E7C3 EA         =1  1913             MOV     A,PARAM1
E7C4 C4         =1  1914             SWAP    A
E7C5 12E7D1     =1  1915             CALL    CONVHEX
E7C8 F6         =1  1916             MOV     @POINTO,A              ;ASCII of high byte in acc.
E7C9 08         =1  1917             INC     POINTO
E7CA EA         =1  1918             MOV     A,PARAM1
E7CB 12E7D1     =1  1919             CALL    CONVHEX
E7CE F6         =1  1920             MOV     @POINTO,A              ;ASCII of low byte in acc.
E7CF 08         =1  1921             INC     POINTO
E7D0 22         =1  1922             RET
                =1  1923 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1  1924     ;*********************************************************************
                =1  1925     ;
                =1  1926     ;    NAME: CONVHEX
                =1  1927     ;
                =1  1928     ;    ABSTRACT: Converts 4 bits to a hex character.
                =1  1929     ;
                =1  1930     ;    INPUTS: A (byte to be converted)
                =1  1931     ;
                =1  1932     ;    OUTPUTS: A
                =1  1933     ;
                =1  1934     ;    VARIABLES MODIFIED: A
                =1  1935     ;
                =1  1936     ;    ERROR EXITS: None
                =1  1937     ;
                =1  1938     ;    SUBROUTINES ACCESSED DIRECTLY: None
                =1  1939     ;
                =1  1940     ;
                =1  1941     ;*********************************************************************
                =1  1942     CONVHEX:
E7D1 540F       =1  1943             ANL     A,#0FH                  ;ASCII No. 90-99, aux.C=0
E7D3 2490       =1  1944             ADD     A,#90H                  ;9A-9F aux. C=1
E7D5 D4         =1  1945             DA      A
E7D6 3440       =1  1946             ADDC    A,#40H
E7D8 D4         =1  1947             DA      A
E7D9 22         =1  1948             RET
                =1  1949 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1  1950     ;*********************************************************************
                =1  1951     ;
                =1  1952     ;    NAME: (I)LSTWRD/ (I)LSTBYT
                =1  1953     ;
                =1  1954     ;    ABSTRACT: Outputs a word or a byte to the system console.
                =1  1955     ;
                =1  1956     ;    INPUTS: PARAM2 (low byte of a word), PARAM1 (high byte of a word
                =1  1957     ;       or the single byte in a byte display)
                =1  1958     ;
                =1  1959     ;    OUTPUTS: None
                =1  1960     ;
                =1  1961     ;    VARIABLES MODIFIED: A, PARAM1, PARAM3
                =1  1962     ;
                =1  1963     ;    ERROR EXITS: None
                =1  1964     ;
                =1  1965     ;    SUBROUTINES ACCESSED DIRECTLY: CONVHEX, ICO
                =1  1966     ;
                =1  1967     ;
                =1  1968     ;*********************************************************************
E7DA 12E7DF     =1  1969     ILSTWRD:CALL    ILSTBYT
E7DD EB         =1  1970             MOV     A,PARAM2
E7DE FA         =1  1971             MOV     PARAM1,A
                =1  1972     ;*********************************************************************
E7DF EA         =1  1973     ILSTBYT:MOV     A,PARAM1                ;Move byte into ACC
E7E0 FC         =1  1974             MOV     PARAM3,A
E7E1 C4         =1  1975             SWAP    A
E7E2 F1D1       =1  1976             CALL    CONVHEX
E7E4 FA         =1  1977             MOV     PARAM1,A
E7E5 B1CE       =1  1978             CALL    ICO                     ;Save lower 4 bits in lower 4 of PARAM3
E7E7 EC         =1  1979             MOV     A,PARAM3                ;Needed because reg to reg moves invalid
E7E8 F1D1       =1  1980             CALL    CONVHEX
E7EA FA         =1  1981             MOV     PARAM1,A
E7EB A1CE       =1  1982             JMP     ICO
                =1  1983 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1  1984     ;*****************************************************************
                =1  1985     ;
                =1  1986     ;    NAME: PAINTER
                =1  1987     ;
                =1  1988     ;    ABSTRACT: Repaints the contents of LINBUF to the display.
                =1  1989     ;
                =1  1990     ;    INPUTS: PARAM6 (contains line length, LNLGTH)
                =1  1991     ;
                =1  1992     ;    OUTPUTS: None
                =1  1993     ;
                =1  1994     ;    VARIABLES MODIFIED: A, PARAM1, POINT1, PARAM6
                =1  1995     ;
                =1  1996     ;    ERROR EXITS: None
                =1  1997     ;
                =1  1998     ;    SUBROUTINES ACCESSED DIRECTLY: UPI_OUT
                =1  1999     ;
                =1  2000     ;
                =1  2001     ;*****************************************************************
E7ED 7924       =1  2002     PAINTER:MOV     POINT1,#LINBUF
                =1  2003     REPAINT_2:
E7EF E7         =1  2004             MOV     A,@POINT1
E7F0 FA         =1  2005             MOV     PARAM1,A
E7F1 D11E       =1  2006             CALL    UPI_OUT
E7F3 09         =1  2007             INC     POINT1
E7F4 DFF9       =1  2008             DJNZ    PARAM6,REPAINT_2
E7F6 22         =1  2009             RET
                =1  2010 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                    =1  2011     ;*********************************************************************
                    =1  2012     ;
                    =1  2013     ;    NAME: GETCHR
                    =1  2014     ;
                    =1  2015     ;    ABSTRACT: This routine returns one character from the line
                    =1  2016     ;       buffer in CHARIN if a carraige return has bee received.
                    =1  2017     ;       If no °CR° is present, it gets characters from the UPI and
                    =1  2018     ;       fills the line buffer until a °CR° is encountered. It echos
                    =1  2019     ;       each character, as it is received, to the display. If LIST
                    =1  2020     ;       is on, it echoes the entire line to the serial port after a
                    =1  2021     ;       °CR° is encountered.
                    =1  2022     ;
                    =1  2023     ;    INPUTS: CHRCNT, LNLGTH, LSTFLG, LINE_START
                    =1  2024     ;
                    =1  2025     ;    OUTPUTS: CHARIN
                    =1  2026     ;
                    =1  2027     ;    VARIABLES MODIFIED: A, PARAM1, PARAM2, LNLGTH, CHRCNT, C, CHARIN
                    =1  2028     ;
                    =1  2029     ;    ERROR EXITS: None
                    =1  2030     ;
                    =1  2031     ;    SUBROUTINES ACCESSED DIRECTLY: ITIME, UPI_CMD, INEWLINE, PAINTER,
                    =1  2032     ;       UPI_OUT, ICI, ICO, SPACCO
                    =1  2033     ;
                    =1  2034     ;
                    =1  2035     ;*********************************************************************
E7F7 E551           =1  2036     GETCHR: MOV     A,CHRCNT                ;Move character counter into ACC
E7F9 B55444         =1  2037             CJNE    A,LNLGTH,OUTCHR         ;Compare ACC to line length and jump to
                    =1  2038                                             ;OUTCHR if not equal
E7FC 7A00           =1  2039             MOV     PARAM1,#SELECT_CON
E7FE 12E60B         =1  2040             CALL    UPI_CMD
E801 E552           =1  2041             MOV     A,LINE_START
E803 F554           =1  2042             MOV     LNLGTH,A                ;Clear character count and line length
E805 F551           =1  2043             MOV     CHRCNT,A
E807 2423           =1  2044             ADD     A,#(LINBUF-1)           ;Initialize R0 as pointer to line buffer
E809 F8             =1  2045             MOV     POINTO,A
E80A 12E6FD         =1  2046     CRWAIT: CALL    INEWLINE
E80D AF54           =1  2047             MOV     PARAM6,LNLGTH           ;Re-paint the alpha-numeric display.
E80F BF0003         =1  2048             CJNE    PARAM6,#00H,REPAINT
E812 02E818         =1  2049             JMP     REPAINT_1
E815 12E7ED         =1  2050     REPAINT:CALL    PAINTER
                    =1  2051     REPAINT_1:
E818 7AAD           =1  2052             MOV     PARAM1,#('-'+BLINK)
E81A 12E61E         =1  2053             CALL    UPI_OUT
E81D 12E5D1         =1  2054             CALL    ICI
E820 F550           =1  2055             MOV     CHARIN,A                ;Move input into character storage
E822 FA             =1  2056             MOV     PARAM1,A                ;Move CHARIN into R2
E823 BA0D25         =1  2057             CJNE    PARAM1,#CR,RUBOUT       ;Check for CR as input
E826 7424           =1  2058             MOV     A,#LINBUF
E828 2554           =1  2059             ADD     A,LNLGTH
E82A F8             =1  2060             MOV     POINTO,A                ;Load R0 to next char in line buffer
E82B 760D           =1  2061             MOV     @POINTO,#CR             ;Load CR into line buffer
E82D 0554           =1  2062             INC     LNLGTH
E82F E4             =1  2063             CLR     A
E830 A201           =1  2064             MOV     C,LSTFLG
E832 92E6           =1  2065             MOV     ACC.6,C
```

```
LOC  OBJ            LINE     SOURCE

E834 FA          =1  2066             MOV     PARAM1,A
E835 12E60B      =1  2067             CALL    UPI_CMD                 ;Turn list mode on if selected
E838 12E6FD      =1  2068             CALL    INEWLINE
E83B AF54        =1  2069             MOV     PARAM6,LNLGTH
E83D 12E7ED      =1  2070             CALL    PAINTER                 ;Echoes line a final time in list mode
E840 7424        =1  2071     OUTCHR: MOV     A,#LINBUF               ;Load A with base addr of storage array
E842 2551        =1  2072             ADD     A,CHRCNT                ;Add character count to ACC
E844 F8          =1  2073             MOV     POINTO,A                ;RO used as indirect pointer to char.
E845 E6          =1  2074             MOV     A,@POINTO               ;Return char to GETCHR call routine in ACC
E846 F550        =1  2075             MOV     CHARIN,A                ;Move character pointer to by RO
E848 0551        =1  2076             INC     CHRCNT                  ;Increment character counter
E84A 22          =1  2077             RET                             ;Exit from GETCHR
E84B BA7F18      =1  2078     RUBOUT: CJNE    PARAM1,#RBOUT,LEGALI    ;Check for rub out as input
E84E E554        =1  2079             MOV     A,LNLGTH                ;Move line length into ACC
E850 B55202      =1  2080             CJNE    A,LINE_START,DELET      ;Check if any characters were input yet
E853 80B5        =1  2081             JMP     CRWAIT                  ;CR wait loop
E855 7A08        =1  2082     DELET:  MOV     PARAM1,#BACKSP
E857 12E5CE      =1  2083             CALL    ICO                     ;Output back space
E85A 12E5CC      =1  2084             CALL    SPACCO                  ;Output space
E85D 7A08        =1  2085             MOV     PARAM1,#BACKSP
E85F 12E5CE      =1  2086             CALL    ICO                     ;Output back space
E862 1554        =1  2087             DEC     LNLGTH                  ;Decrement line length
E864 80A4        =1  2088             JMP     CRWAIT                  ;CR wait loop
E866 E554        =1  2089     LEGALI: MOV     A,LNLGTH
E868 B41702      =1  2090             CJNE    A,#LINMAX-1,TABKEY      ;Check that line does not exceed max
E86B 809D        =1  2091             JMP     CRWAIT                  ;CR wait loop
E86D BA091A      =1  2092     TABKEY: CJNE    PARAM1,#HORIZONTAL_TAB,INPUT
E870 7424        =1  2093             MOV     A,#LINBUF
E872 2554        =1  2094             ADD     A,LNLGTH
E874 F8          =1  2095             MOV     POINTO,A
E875 E554        =1  2096             MOV     A,LNLGTH
                 =1  2097     MORE_SPACE:
E877 04          =1  2098             INC     A
E878 F554        =1  2099             MOV     LNLGTH,A
E87A 7620        =1  2100             MOV     @POINTO,#' '
E87C 08          =1  2101             INC     POINTO
E87D B41702      =1  2102             CJNE    A,#LINMAX-1,MORE_CONT
E880 8088        =1  2103             JMP     CRWAIT
                 =1  2104     MORE_CONT:
E882 30E0F2      =1  2105             JNB     ACC.0,MORE_SPACE
E885 30E1EF      =1  2106             JNB     ACC.1,MORE_SPACE
E888 8080        =1  2107             JMP     CRWAIT
E88A E550        =1  2108     INPUT:  MOV     A,CHARIN
E88C 30E503      =1  2109             JNB     ACC.5,INPUTOK
E88F 30E600      =1  2110             JNB     ACC.6,INPUTOK
E892 7424        =1  2111     INPUTOK:        MOV A,#LINBUF           ;Load A with line buffer base addr
E894 2554        =1  2112             ADD     A,LNLGTH                ;Add line length to ACC
E896 F8          =1  2113             MOV     POINTO,A                ;POINTO used as pointer to array
E897 A650        =1  2114             MOV     @POINTO,CHARIN          ;Load input into storage array
E899 12E5CE      =1  2115             CALL    ICO                     ;Output input
E89C 0554        =1  2116             INC     LNLGTH                  ;Increment line length counter
E89E 010A        =1  2117             JMP     CRWAIT                  ;CR wait routine
                 =1  2118 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                    =1  2119     ;*****************************************************************
                    =1  2120     ;
                    =1  2121     ;    NAME: (I)GETOKE
                    =1  2122     ;
                    =1  2123     ;    ABSTRACT: This routine inputs characters, ignoring spaces, until
                    =1  2124     ;       string buffer is full (LNCNT).  If the characters are numbers
                    =1  2125     ;       the token type is designated °number° and its value goes into
                    =1  2126     ;       VALLOW and VALHGH.  It compares the input token to the keyword table
                    =1  2127     ;       and errors if not found.  If found, it checks the next keyword
                    =1  2128     ;       entry to see if the token is a valid abreviation.  Assembler
                    =1  2129     ;       operands that are not numbers will have the basic operand type
                    =1  2130     ;       flag set (B_O_T).
                    =1  2131     ;
                    =1  2132     ;    INPUTS: None
                    =1  2133     ;
                    =1  2134     ;    OUTPUTS: TOKSTR, B_O_T, A
                    =1  2135     ;
                    =1  2136     ;    VARIABLES MODIFIED: A, POINTO, LINCNT, @POINTO, PARAM1, TEMP1,
                    =1  2137     ;       ERRNUM, DPTR, TOKSTR, B_O_T
                    =1  2138     ;
                    =1  2139     ;    ERROR EXITS: 01H (INVALID WORD i.e. token)
                    =1  2140     ;
                    =1  2141     ;    SUBROUTINES ACCESSED DIRECTLY: IERROR, GETCHR, IGETOKE, AZTEST,
                    =1  2142     ;       NUMBER, ALFNUM, STRING_SPACE
                    =1  2143     ;
                    =1  2144     ;
                    =1  2145     ;*****************************************************************
E8A0 C200           =1  2146     IGETOKE:CLR     B_O_T
E8A2 E550           =1  2147             MOV     A,CHARIN                ;Move char into ACC
E8A4 B42005         =1  2148             CJNE    A,#' ',ALPHA            ;Loop on space inputs
E8A7 12E7F7         =1  2149             CALL    GETCHR                  ;Get new input
E8AA 80F4           =1  2150             SJMP    IGETOKE                 ;Space loop
E8AC 783C           =1  2151     ALPHA:  MOV     POINTO,#STRGBF
E8AE 755305         =1  2152             MOV     LINCNT,#TOKSIZ+1
E8B1 7420           =1  2153     SPFILL: MOV     A,#' '                  ;Load ACC with ASCII equiv of space
E8B3 F6             =1  2154             MOV     @POINTO,A               ;Fill buffer with spaces
E8B4 08             =1  2155             INC     POINTO                  ;Increment string buffer pointer
E8B5 D553F9         =1  2156             DJNZ    LINCNT,SPFILL           ;Loop until string buffer is filled
E8B8 755304         =1  2157             MOV     LINCNT,#TOKSIZ          ;Move length of string into R1
E8BB 783C           =1  2158             MOV     POINTO,#STRGBF          ;Move base addr of string buffer into RO
E8BD AA50           =1  2159             MOV     PARAM1,CHARIN           ;Move char into R2
E8BF 12E706         =1  2160             CALL    AZTEST                  ;See if char is a letter
E8C2 4003           =1  2161             JC      STRFIL
E8C4 02E917         =1  2162             JMP     NUMBER                  ;Jump to number if false
E8C7 12E72A         =1  2163     STRFIL: CALL    ALFNUM                  ;See if char is letter or number
E8CA 501B           =1  2164             JNC     STRTST                  ;Jump to filler routine if non-numerical
E8CC EA             =1  2165             MOV     A,PARAM1                ;Save char in string buffer
E8CD F6             =1  2166             MOV     @POINTO,A               ;Needed because reg to reg move invalid
E8CE 08             =1  2167             INC     POINTO                  ;Increment string buffer pointer
E8CF 8856           =1  2168             MOV     TEMP1,POINTO            ;Save pointer from GETCHAR
E8D1 12E7F7         =1  2169             CALL    GETCHR                  ;Get next input
E8D4 AA50           =1  2170             MOV     PARAM1,CHARIN           ;To pass param for ALFNUM
E8D6 A856           =1  2171             MOV     POINTO,TEMP1            ;Restore pointer for GETOKE
E8D8 D553EC         =1  2172             DJNZ    LINCNT,STRFIL           ;Get more char if line counter is not 0
E8DB 12E72A         =1  2173     SPWAIT: CALL    ALFNUM                  ;Check for alpha-numeric character
```

```
LOC  OBJ            LINE     SOURCE

E8DE 5007       =1  2174             JNC     STRTST                  ;Loop until space is input
E8E0 12E7F7     =1  2175             CALL    GETCHR                  ;Get next character
E8E3 AA50       =1  2176             MOV     PARAM1,CHARIN           ;Setup for ALFNUM
E8E5 80F4       =1  2177             SJMP    SPWAIT
E8E7 7A00       =1  2178     STRTST: MOV     PARAM1,#00H
E8E9 12E99B     =1  2179     STRTST1:CALL    STRING_SPACE            ;Compare STRGBF to the keyword table.
E8EC 7013       =1  2180             JNZ     GOOD_TOKE_FOUND
E8EE 400A       =1  2181             JC      CHECK_ABREV
E8F0 0A         =1  2182             INC     PARAM1
E8F1 BA65F5     =1  2183             CJNE    PARAM1,#(KEYTAB-TOKTBL+1),STRTST1
E8F4 754301     =1  2184     TOKERR: MOV     ERRNUM,#01H             ;Invalid word
E8F7 02E3CA     =1  2185             JMP     IERROR
                =1  2186     CHECK_ABREV:
E8FA 0A         =1  2187             INC     PARAM1
E8FB 12E99B     =1  2188             CALL    STRING_SPACE
E8FE 1A         =1  2189             DEC     PARAM1
E8FF 40F3       =1  2190             JC      TOKERR
                =1  2191     GOOD_TOKE_FOUND:
E901 EA         =1  2192             MOV     A,PARAM1
E902 90E072     =1  2193             MOV     DPTR,#(TOKTBL - 1)
E905 93         =1  2194             MOVC    A,@A+DPTR               ;Get token from table
E906 F548       =1  2195             MOV     TOKSTR,A                ;Put token in storage
E908 B44000     =1  2196             CJNE    A,#40H,GTO              ;Set basic operand type flag for
E90B 4007       =1  2197     GTO:    JC      NOTBOT                  ;Tokens that are assembler operands which
E90D B49800     =1  2198             CJNE    A,#98H,GT1              ;are not numbers.
E910 5002       =1  2199     GT1:    JNC     NOTBOT
E912 D200       =1  2200             SETB    B_O_T
E914 E548       =1  2201     NOTBOT: MOV     A,TOKSTR
E916 22         =1  2202             RET
                =1  2203 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                    =1  2204     ;*****************************************************************
                    =1  2205     ;
                    =1  2206     ;    NAME: NUMBER
                    =1  2207     ;
                    =1  2208     ;    ABSTRACT: This routine checks to see if a number of characters
                    =1  2209     ;       (1-24) is a valid hex number, converts it to a
                    =1  2210     ;       16 bit binary number and gives it a number token if
                    =1  2211     ;       is.  It ignores leading zeros and trailing 'Hs'.
                    =1  2212     ;
                    =1  2213     ;    INPUTS: A
                    =1  2214     ;
                    =1  2215     ;    OUTPUTS: TOKSTR, VALHGH, VALLOW
                    =1  2216     ;
                    =1  2217     ;    VARIABLES MODIFIED: VALLOW, VALHGH, PARAM2, A, B, TOKSTR
                    =1  2218     ;
                    =1  2219     ;    ERROR EXITS: None
                    =1  2220     ;
                    =1  2221     ;    SUBROUTINES ACCESSED DIRECTLY: NMTEST, HXTEST, GETCHR
                    =1  2222     ;
                    =1  2223     ;
                    =1  2224     ;*****************************************************************
E917 12E713         =1  2225     NUMBER: CALL    NMTEST
E91A 5045           =1  2226             JNC     SYMBOL                  ;Jump if char is not a number
E91C 754A00         =1  2227             MOV     VALLOW,#00H             ;Initialize value storage
E91F 754900         =1  2228             MOV     VALHGH,#00H
E922 12E71D         =1  2229     HEXSTR: CALL    HXTEST
E925 502C           =1  2230             JNC     HTEST                   ;Jump if char is not a hex character
E927 12E713         =1  2231             CALL    NMTEST                  ;Check for character=0 to 9
E92A 5023           =1  2232             JNC     HEXCHR                  ;Load A into PARAM2 for hex char
E92C 7B30           =1  2233             MOV     PARAM2,#'0'             ;Clear pointer
E92E E54A           =1  2234     RL4:    MOV     A,VALLOW
E930 75F010         =1  2235             MOV     B,#16                   ;To RL 4 places
E933 A4             =1  2236             MUL     AB
E934 F54A           =1  2237             MOV     VALLOW,A                ;ACC now holds VALLOW RL 4  places
E936 E550           =1  2238             MOV     A,CHARIN                ;Move last number entered into ACC
E938 9B             =1  2239             SUBB    A,PARAM2                ;Subtract ASCII equiv of 'A' or '0'
                    =1  2240                                             ;as appropriate for hex or decimal
E939 254A           =1  2241             ADD     A,VALLOW                ;Add number to rotated VALLOW
E93B F54A           =1  2242             MOV     VALLOW,A                ;Store new value in VALLOW
E93D AAF0           =1  2243             MOV     PARAM1,B                ;Store upper 4 bits from rotate
E93F 75F010         =1  2244             MOV     B,#10H
E942 E549           =1  2245             MOV     A,VALHGH                ;Move VALHGH into ACC
E944 A4             =1  2246             MUL     AB                      ;Rotate VALHGH 4 places to left
E945 2A             =1  2247             ADD     A,PARAM1                ;Add upper 4 bits from VALLOW
E946 F549           =1  2248             MOV     VALHGH,A                ;Store new value in VALHGH
E948 12E7F7         =1  2249             CALL    GETCHR                  ;Get next input
E94B AA50           =1  2250             MOV     PARAM1,CHARIN           ;Set up pass param for HXTEST
E94D 80D3           =1  2251             SJMP    HEXSTR                  ;Loop until non hex char entered
E94F 7B37           =1  2252     HEXCHR: MOV     PARAM2,#('A'-0AH)       ;Move ASCII equiv of 'A' into POINT1
E951 80DB           =1  2253             SJMP    RL4
E953 E550           =1  2254     HTEST:  MOV     A,CHARIN
E955 B44803         =1  2255             CJNE    A,#'H',NUMBER_FOUND             ;See if char is 'H' and ignore if so
E958 12E7F7         =1  2256             CALL    GETCHR
E95B 754801         =1  2257     NUMBER_FOUND:   MOV     TOKSTR,#NUMBER_TOKE             ;Load toke storage with number toke
                                 n
```

```
LOC  OBJ            LINE     SOURCE

E95E E548       =1  2258             MOV     A,TOKSTR                ;Load ACC with TOKEN
E960 22         =1  2259             RET
                =1  2260 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                    =1  2261     ;*********************************************************************
                    =1  2262     ;
                    =1  2263     ;    NAME: SYMBOL
                    =1  2264     ;
                    =1  2265     ;    ABSTRACT: This routine checks a token against the symbol
                    =1  2266     ;       table tokens (ie comma, equal sign, etc.), errors if
                    =1  2267     ;       there is no match and returns the token in ACC if it is
                    =1  2268     ;       found.
                    =1  2269     ;
                    =1  2270     ;    INPUTS: PARAM1
                    =1  2271     ;
                    =1  2272     ;    OUTPUTS: A, TOKSTR
                    =1  2273     ;
                    =1  2274     ;    VARIABLES MODIFIED: TOKSTR, A, DPTR, ERRNUM, CHARIN
                    =1  2275     ;
                    =1  2276     ;    ERROR EXITS: 01H (INVALID WORD)
                    =1  2277     ;
                    =1  2278     ;    SUBROUTINES ACCESSED DIRECTLY: IERROR, GETCHR
                    =1  2279     ;
                    =1  2280     ;
                    =1  2281     ;*********************************************************************
E961 8A48           =1  2282     SYMBOL: MOV     TOKSTR,PARAM1
E963 90E97B         =1  2283             MOV     DPTR,#SYMBOL_TBL
                    =1  2284     SYM_TBL_SRCH:
E966 E4             =1  2285             CLR     A
E967 93             =1  2286             MOVC    A,@A+DPTR
E968 754301         =1  2287             MOV     ERRNUM,#01H                    ;Invalid token (word)
E96B 601C           =1  2288             JZ      ERRSET
E96D B54807         =1  2289             CJNE    A,TOKSTR,NOT_MATCH_TBL
E970 A3             =1  2290             INC     DPTR
E971 E4             =1  2291             CLR     A
E972 93             =1  2292             MOVC    A,@A+DPTR
E973 F548           =1  2293             MOV     TOKSTR,A
E975 8015           =1  2294             SJMP    SYMEND
                    =1  2295     NOT_MATCH_TBL:
E977 A3             =1  2296             INC     DPTR
E978 A3             =1  2297             INC     DPTR
E979 80EB           =1  2298             SJMP    SYM_TBL_SRCH
                    =1  2299     SYMBOL_TBL:
E97B 2C             =1  2300             DB      ',',COMMA_TOKE
E97C 02
E97D 2F             =1  2301             DB      '/',BAR_TOKE
E97E 03
E97F 3D             =1  2302             DB      '=',EQUAL_TOKE
E980 04
E981 2B             =1  2303             DB      '+',PLUS_TOKE
E982 05
E983 23             =1  2304             DB      '#',POUND_TOKE
E984 06
E985 0D             =1  2305             DB      CR,EOL_TOKE
E986 07
E987 00             =1  2306             DB      0,0
E988 00
E989 02E3CA         =1  2307     ERRSET: JMP     IERROR
E98C BA0D06         =1  2308     SYMEND: CJNE    PARAM1,#CR,LAB10        ;See if last input was a 'CR'
```

```
LOC  OBJ            LINE     SOURCE

E98F 755020     =1  2309             MOV     CHARIN,#' '          ;Return a space to calling routine if 'CR'
E992 E548       =1  2310             MOV     A,TOKSTR             ;Load ACC with token
E994 22         =1  2311             RET                          ;Exit from GETOKE
E995 12E7F7     =1  2312     LAB10:  CALL    GETCHR               ;Get next character if 'CR' wasn't last char
E998 E548       =1  2313             MOV     A,TOKSTR             ;To return token in ACC
E99A 22         =1  2314             RET                          ;Exit from GETOKE
                =1  2315 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                    =1  2316     ;*****************************************************************
                    =1  2317     ;
                    =1  2318     ;    NAME: STRING_SPACE
                    =1  2319     ;
                    =1  2320     ;    ABSTRACT: This routine checks the contents of the string buffer
                    =1  2321     ;       against the keyword table for any match (ie a valid abreviation
                    =1  2322     ;       or an exact match) and returns to the calling routine.  There
                    =1  2323     ;       are 4 places in every keyword and this routine matches for
                    =1  2324     ;       spaces as well as characters. Carry and ACC are set
                    =1  2325     ;       if match is exact, carry is set and ACC is cleared if match is
                    =1  2326     ;       not exact (ie spaces do not match - could be an abrev.), both
                    =1  2327     ;       carry and ACC are cleared if there is no match at all.
                    =1  2328     ;
                    =1  2329     ;    INPUTS: STRGBF, PARAM1 (token ordinal in KEYTAB)
                    =1  2330     ;
                    =1  2331     ;    OUTPUTS: Carry bit (C), A
                    =1  2332     ;
                    =1  2333     ;    VARIABLES MODIFIED: C, A, POINTO, STRGCT, DPTR, B, TEMP1
                    =1  2334     ;
                    =1  2335     ;    ERROR EXITS: None
                    =1  2336     ;
                    =1  2337     ;    SUBROUTINES ACCESSED DIRECTLY: None
                    =1  2338     ;
                    =1  2339     ;
                    =1  2340     ;*****************************************************************
                    =1  2341     STRING_SPACE:
E99B 783C           =1  2342             MOV     POINTO,#STRGBF  ;Load RO with address of string buffer
E99D 755504         =1  2343             MOV     STRGCT,#TOKSIZ  ;Load counter with length of string
E9A0 90E0D3         =1  2344             MOV     DPTR,#(KEYTAB-4);Load DPTR with address of KEY TABLE
E9A3 75F004         =1  2345             MOV     B,#4
E9A6 EA             =1  2346             MOV     A,PARAM1        ;Load ACC with offset
E9A7 A4             =1  2347             MUL     AB              ;Multiply by 4 characters
E9A8 C3             =1  2348             CLR     C
E9A9 2582           =1  2349             ADD     A,DPL           ;Add offset to base
E9AB F582           =1  2350             MOV     DPL,A
E9AD E5F0           =1  2351             MOV     A,B
E9AF 3583           =1  2352             ADDC    A,DPH
E9B1 F583           =1  2353             MOV     DPH,A
E9B3 E4             =1  2354     S_S_1:  CLR     A
E9B4 93             =1  2355             MOVC    A,@A+DPTR
E9B5 F556           =1  2356             MOV     TEMP1,A
E9B7 E6             =1  2357             MOV     A,@POINTO
E9B8 B55609         =1  2358             CJNE    A,TEMP1,S_S_2
E9BB A3             =1  2359             INC     DPTR            ;Next key character
E9BC 08             =1  2360             INC     POINTO          ;Next string character
E9BD D555F3         =1  2361             DJNZ    STRGCT,S_S_1    ;Test the whole 4 char string
E9C0 D3             =1  2362             SETB    C               ;Match exactly including spaces
E9C1 E4             =1  2363             CLR     A
E9C2 F4             =1  2364             CPL     A
E9C3 22             =1  2365             RET
E9C4 B42003         =1  2366     S_S_2:  CJNE    A,#' ',S_S_3    ;Match but not exact (spaces)
E9C7 D3             =1  2367             SETB    C
E9C8 E4             =1  2368             CLR     A
E9C9 22             =1  2369             RET
E9CA C3             =1  2370     S_S_3:  CLR     C               ;No match at all
```

```
LOC  OBJ            LINE     SOURCE

E9CB E4          =1  2371             CLR     A
E9CC 22          =1  2372             RET
                 =1  2373 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1  2374     ;*****************************************************************
                =1  2375     ;
                =1  2376     ;    NAME: (I)PRINT_STRING
                =1  2377     ;
                =1  2378     ;    ABSTRACT: Prints a string from program memory.  At entry, PARAM1
                =1  2379     ;       and PARAM2 should point to the string.  The first element of
                =1  2380     ;       the string is the length (0-255), the rest of the elements are
                =1  2381     ;       output as ASCII characters.
                =1  2382     ;
                =1  2383     ;       WARNING: Calls to this routine may not be single-stepped through.
                =1  2384     ;
                =1  2385     ;    INPUTS: PARAM1(high byte), PARAM2(low byte)
                =1  2386     ;
                =1  2387     ;    OUTPUTS: None
                =1  2388     ;
                =1  2389     ;    VARIABLES MODIFIED: A, COUNT, DPTR, PARAM1
                =1  2390     ;
                =1  2391     ;    ERROR EXITS: None
                =1  2392     ;
                =1  2393     ;    SUBROUTINES ACCESSED DIRECTLY: ICO
                =1  2394     ;
                =1  2395     ;*****************************************************************
                =1  2396     IPRINT_STRING:
E9CD 8A83       =1  2397             MOV     DPH,PARAM1
E9CF 8B82       =1  2398             MOV     DPL,PARAM2
E9D1 E4         =1  2399             CLR     A               ;Counter:=string length.
E9D2 93         =1  2400             MOVC    A,@A+DPTR
E9D3 FF         =1  2401             MOV     COUNT,A
E9D4 6009       =1  2402             JZ      PRINT_STRING_E  ;Exit if a null string or
                =1  2403     PRINT_STRING_1:
E9D6 E4         =1  2404             CLR     A               ;else get the next element
E9D7 A3         =1  2405             INC     DPTR
E9D8 93         =1  2406             MOVC    A,@A+DPTR
E9D9 FA         =1  2407             MOV     PARAM1,A        ;and output it.
E9DA 12E5CE     =1  2408             CALL    ICO             ;Repeat loop until count=0.
E9DD DFF7       =1  2409             DJNZ    COUNT,PRINT_STRING_1
                =1  2410     PRINT_STRING_E:
E9DF 22         =1  2411             RET                     ;Then return to the caller.
                =1  2412 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1  2413     ;*****************************************************************
                =1  2414     ;
                =1  2415     ;    NAME: (I)DISPLAY_TOKEN
                =1  2416     ;
                =1  2417     ;    ABSTRACT: This routine displays an ASCII token using the token
                =1  2418     ;       value passed to it (PARAM1) to indicate which token to display.
                =1  2419     ;
                =1  2420     ;    INPUTS: PARAM1 (token to be displayed)
                =1  2421     ;
                =1  2422     ;    OUTPUTS: None
                =1  2423     ;
                =1  2424     ;    VARIABLES MODIFIED: PARAM2, DPTR, A, PARAM3, PARAM1
                =1  2425     ;
                =1  2426     ;    ERROR EXITS:  None
                =1  2427     ;
                =1  2428     ;    SUBROUTINES ACCESSED DIRECTLY: ICO
                =1  2429     ;
                =1  2430     ;*****************************************************************
                =1  2431     IDISPLAY_TOKEN:
E9E0 7B00       =1  2432             MOV     PARAM2,#00H
E9E2 C3         =1  2433             CLR     C
                =1  2434     DTO_0:
E9E3 90E073     =1  2435             MOV     DPTR,#TOKTBL
E9E6 EB         =1  2436             MOV     A,PARAM2
E9E7 93         =1  2437             MOVC    A,@A+DPTR
E9E8 B50203     =1  2438             CJNE    A,2,DTO                 ;2 is the direct addr of R2 which we call PARAM1
E9EB 02E9F2     =1  2439             JMP     DT1
                =1  2440     DTO:
E9EE 0B         =1  2441             INC     PARAM2
E9EF BB61F1     =1  2442             CJNE    PARAM2,#97,DTO_0
                =1  2443     DT1:
E9F2 90E0D7     =1  2444             MOV     DPTR,#KEYTAB
                =1  2445     DT_LOOP:
E9F5 A3         =1  2446             INC     DPTR
E9F6 A3         =1  2447             INC     DPTR
E9F7 A3         =1  2448             INC     DPTR
E9F8 A3         =1  2449             INC     DPTR
E9F9 DBFA       =1  2450             DJNZ    PARAM2,DT_LOOP
E9FB 7C04       =1  2451             MOV     PARAM3,#04H
E9FD E4         =1  2452     TOKLOP: CLR     A
E9FE 93         =1  2453             MOVC    A,@A+DPTR               ;Load ACC with first character of token
E9FF B42001     =1  2454             CJNE    A,#' ',TOK_WRITE
EA02 22         =1  2455             RET
                =1  2456     TOK_WRITE:
EA03 FA         =1  2457             MOV     PARAM1,A                ;To output character
EA04 12E5CE     =1  2458             CALL    ICO
EA07 A3         =1  2459             INC     DPTR
EA08 DCF3       =1  2460             DJNZ    PARAM3,TOKLOP           ;Loop if less than 4 characters output
EA0A 22         =1  2461             RET
                =1  2462     ;***** END OF DISPLAY_TOKEN *****
                =1  2463 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1  2464     ;*********************************************************************
                =1  2465     ;
                =1  2466     ;    NAME: ASCII_TO_HEX (PARAM1)
                =1  2467     ;
                =1  2468     ;    ABSTRACT: Assumes that PARAM1 is an ASCII character representing
                =1  2469     ;       a hexidecimal digit and converts it to binary.  The result
                =1  2470     ;       is returned in the lower four bits of the accumulator.  The
                =1  2471     ;       upper bits are cleared.
                =1  2472     ;
                =1  2473     ;    INPUTS: PARAM1 (ASCII character)
                =1  2474     ;
                =1  2475     ;    OUTPUTS: A
                =1  2476     ;
                =1  2477     ;    VARIABLES MODIFIED: A
                =1  2478     ;
                =1  2479     ;    ERROR EXITS: None
                =1  2480     ;
                =1  2481     ;    SUBROUTINES ACCESSED DIRECTLY: None
                =1  2482     ;
                =1  2483     ;*********************************************************************
                =1  2484     IASCII_TO_HEX:
EA0B EA         =1  2485             MOV     A,PARAM1                ;Put ASCII character into ACC
EA0C 30E602     =1  2486             JNB     ACC.6,HEX1              ;Jump to HEX1 if CHAR < 40H
EA0F 2409       =1  2487             ADD     A,#09H                  ;Add nine if CHAR > 3FH
EA11 540F       =1  2488     HEX1:   ANL     A,#0FH                  ;Mask lower 4 bits
EA13 22         =1  2489             RET
                =1  2490 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                    =1  2491     ;*********************************************************************
                    =1  2492     ;
                    =1  2493     ;    NAME: ITIME
                    =1  2494     ;
                    =1  2495     ;    ABSTRACT: TIME is a general purpose routine available through
                    =1  2496     ;       the jump table. Parameter 1 and 2 are the high and low bytes
                    =1  2497     ;       of a sixteen bit timer where each increment represents
                    =1  2498     ;       100 uS as in PLM.
                    =1  2499     ;       Time simply delays for the specified time and then returns.
                    =1  2500     ;
                    =1  2501     ;    INPUTS: PARAM1 (high byte), PARAM2 (low byte)
                    =1  2502     ;
                    =1  2503     ;    OUTPUTS: None
                    =1  2504     ;
                    =1  2505     ;    VARIABLES MODIFIED: A, DPTR, R5
                    =1  2506     ;
                    =1  2507     ;    ERROR EXITS: None
                    =1  2508     ;
                    =1  2509     ;    SUBROUTINES ACCESSED DIRECTLY: None
                    =1  2510     ;
                    =1  2511     ;
                    =1  2512     ;*********************************************************************
                    =1  2513
EA14 EA             =1  2514     ITIME:  MOV     A,PARAM1        ;Convert PARAM1 and PARAM2 into one 16-bit
EA15 F4             =1  2515             CPL     A               ;negative number in DPTR
EA16 F583           =1  2516             MOV     DPH,A
EA18 EB             =1  2517             MOV     A,PARAM2
EA19 F4             =1  2518             CPL     A
EA1A F582           =1  2519             MOV     DPL,A
EA1C A3             =1  2520             INC     DPTR
EA1D 7D2E           =1  2521     TIME1:  MOV     R5,#2EH         ;Setup and
EA1F DDFE           =1  2522             DJNZ    R5,$            ;Loop for 100 us
EA21 A3             =1  2523             INC     DPTR            ;Count out the 16-bit parameter
EA22 E582           =1  2524             MOV     A,DPL           ;Check DPTR for zero
EA24 4583           =1  2525             ORL     A,DPH
EA26 00             =1  2526             NOP
EA27 70F4           =1  2527             JNZ     TIME1
EA29 22             =1  2528             RET
                    =1  2529     ;*********************************************************************
                        2530 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                    2531 +1  $INCLUDE(:f1:DISCHA.INC)
                 =1 2532     ;*****************************************************************
                 =1 2533     ;
                 =1 2534     ;    NAME: MEMORY_CMD
                 =1 2535     ;
                 =1 2536     ;    ABSTRACT: This routine saves the kind of memory operation
                 =1 2537     ;       selected and checks for partitions and equal signs in order
                 =1 2538     ;       to dicide whether a fill, load, display or block move is
                 =1 2539     ;       requested.
                 =1 2540     ;
                 =1 2541     ;    INPUTS: TOKSTR
                 =1 2542     ;
                 =1 2543     ;    OUTPUTS: None
                 =1 2544     ;
                 =1 2545     ;    VARIABLES MODIFIED: A, TOKSAV, SELECT, PNTLOW, PNTHGH, B
                 =1 2546     ;
                 =1 2547     ;    ERROR EXITS: None
                 =1 2548     ;
                 =1 2549     ;    SUBROUTINES ACCESSED DIRECTLY: IGETOKE, IGET_PART, BMOVE,
                 =1 2550     ;       IEOL_CHECK, DISMEM, LODMEM, FILLMEM
                 =1 2551     ;
                 =1 2552     ;
                 =1 2553     ;*****************************************************************
EA2A E548        =1 2554     MEMORY_CMD:     MOV     A,TOKSTR
EA2C 5407        =1 2555             ANL     A,#07                   ;Last 3 bits of token determine selector
EA2E 85485B      =1 2556             MOV     TOKSAV,TOKSTR
EA31 F546        =1 2557             MOV     SELECT,A                ;Load selector
EA33 11A0        =1 2558             CALL    IGETOKE
EA35 12E788      =1 2559             CALL    IGET_PART               ;Partition? Returns 1 bit (C)=true if part.
EA38 855845      =1 2560             MOV     PNTLOW,PARTIT_LO_LOW
EA3B 855744      =1 2561             MOV     PNTHGH,PARTIT_LO_HIGH
EA3E 92F0        =1 2562             MOV     B.0,C
EA40 B4040B      =1 2563             CJNE    A,#EQUAL_TOKE,DIS_OR_ERR        ;Check for equal sign from GET_PART
EA43 30F00E      =1 2564             JNB     B.0,LODMEM              ;Single byte load (CBY addr = data)
EA46 11A0        =1 2565             CALL    IGETOKE
EA48 B48061      =1 2566             CJNE    A,#CBYTE_TOKE,FILLMEM   ;Block move (CBY addr TO addr =CBY addr)
EA4B 02EB27      =1 2567             JMP     BMOVE                   ;Fill mem. (CBY addr TO addr=data)
                 =1 2568     DIS_OR_ERR:
EA4E 12E5A1      =1 2569             CALL    IEOL_CHECK
EA51 02EAD1      =1 2570             JMP     DISMEM                  ;Display mem. (CBY addr TO addr-no equalsign)
                 =1 2571 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                    =1  2572     ;*****************************************************************
                    =1  2573     ;
                    =1  2574     ;    NAME: LODMEM
                    =1  2575     ;
                    =1  2576     ;    ABSTRACT: The pointer will be set to memory address upon entry.
                    =1  2577     ;       Parsing continues as long as new tokens are available on the
                    =1  2578     ;       command line.  Each new token either supplies a new value which
                    =1  2579     ;       goes into memory or a <CR> which terminates the command.  Commas
                    =1  2580     ;       are expected between any two numbers and at the end of a line
                    =1  2581     ;       when a continuation is desired.  When entry of data has gone
                    =1  2582     ;       beyond one line (a continuation line) the line buffer is filled
                    =1  2583     ;       with the message and address which tells the user what address
                    =1  2584     ;       is currently being modified.
                    =1  2585     ;
                    =1  2586     ;    INPUTS: SELECT, PNTHGH, PNTLOW
                    =1  2587     ;
                    =1  2588     ;    OUTPUTS: Memory which was supposed to be accessed by the command
                    =1  2589     ;       typed in at the console.
                    =1  2590     ;
                    =1  2591     ;    VARIABLES MODIFIED: PARAM1, A, POINTO, LINE_START
                    =1  2592     ;
                    =1  2593     ;    ERROR EXITS: None
                    =1  2594     ;
                    =1  2595     ;    SUBROUTINES ACCESSED DIRECTLY: IGETNUM, ISTORE, IGETOKE, INC_PNT,
                    =1  2596     ;       ISAVE_AND_DISPLAY, IEOL_CHECK, IERROR
                    =1  2597     ;
                    =1  2598     ;
                    =1  2599     ;*****************************************************************
EA54 12E74F         =1  2600     LODMEM: CALL    IGETNUM
EA57 AA4A           =1  2601     LDLOOP: MOV     PARAM1,VALLOW           ;Load PARAM1 with data to be output
EA59 12E658         =1  2602             CALL    ISTORE                  ;Output data into memory
EA5C 12E5AA         =1  2603             CALL    INC_PNT
EA5F 11A0           =1  2604             CALL    IGETOKE                 ;Get next token and character
EA61 B40242         =1  2605             CJNE    A,#COMMA_TOKE,EOLMEM    ;Jump to EOLMEM if token is not comma token
EA64 11A0           =1  2606             CALL    IGETOKE                 ;Get next token and character after comma
EA66 B40738         =1  2607             CJNE    A,#EOL_TOKE,NUMMEN      ;Check if CR was entered
EA69 7824           =1  2608             MOV     POINTO,#LINBUF
EA6B E546           =1  2609             MOV     A,SELECT                ;Choose first char, depending on type
EA6D 7652           =1  2610             MOV     @POINTO,#'R'            ;of memory access in progress
EA6F B40002         =1  2611             CJNE    A,#(CBYTE_TOKE AND 07H),B_LAB_1
EA72 7643           =1  2612             MOV     @POINTO,#'C'
EA74 B40202         =1  2613     B_LAB_1:CJNE    A,#(DBYTE_TOKE AND 07H),B_LAB_2
EA77 7644           =1  2614             MOV     @POINTO,#'D'
EA79 B40602         =1  2615     B_LAB_2:CJNE    A,#(XBYTE_TOKE AND 07H),B_LAB_3
EA7C 7658           =1  2616             MOV     @POINTO,#'X'
EA7E 08             =1  2617     B_LAB_3:INC     POINTO
EA7F 7642           =1  2618             MOV     @POINTO,#'B'
EA81 08             =1  2619             INC     POINTO
EA82 7659           =1  2620             MOV     @POINTO,#'Y'
EA84 B40402         =1  2621             CJNE    A,#(RBIT_TOKE AND 07H),T_LAB
EA87 7649           =1  2622             MOV     @POINTO,#'I'            ;Choose third char for bit or byte type
EA89 08             =1  2623     T_LAB:  INC     POINTO
EA8A 7654           =1  2624             MOV     @POINTO,#'T'
EA8C 08             =1  2625             INC     POINTO
EA8D 7620           =1  2626             MOV     @POINTO,#' '
```

```
LOC  OBJ            LINE     SOURCE

EA8F 08         =1  2627             INC     POINTO
EA90 AA44       =1  2628             MOV     PARAM1,PNTHGH
EA92 12E7C3     =1  2629             CALL    ISAVE_AND_DISPLAY
EA95 AA45       =1  2630             MOV     PARAM1,PNTLOW
EA97 12E7C3     =1  2631             CALL    ISAVE_AND_DISPLAY
EA9A 763D       =1  2632             MOV     @POINTO,#'='
EA9C 755210     =1  2633             MOV     LINE_START,#10H
EA9F 11A0       =1  2634             CALL    IGETOKE                 ;Get next token and character
EAA1 B40102     =1  2635     NUMMEN: CJNE    A,#NUMBER_TOKE,EOLMEM   ;Check that a number was last char entered
                =1  2636
EAA4 80B1       =1  2637             JMP     LDLOOP                  ;Loop until CR entered
EAA6 02E5A1     =1  2638     EOLMEM: JMP     IEOL_CHECK
EAA9 02E3CA     =1  2639     DISERR: JMP     IERROR
                =1  2640 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                    =1  2641     ;*****************************************************************
                    =1  2642     ;
                    =1  2643     ;    NAME: FILLMEM
                    =1  2644     ;
                    =1  2645     ;    ABSTRACT: This routine fills the memory selected with a single
                    =1  2646     ;       value from PNTLOW and PNTHGH up to the high end of the
                    =1  2647     ;       partition.
                    =1  2648     ;
                    =1  2649     ;    INPUTS: PNTLOW, PNTHGH, PARTIT_HI_LOW, PARTIT_HI_HIGH
                    =1  2650     ;
                    =1  2651     ;    OUTPUTS: Memory which was supposed to be accessed by the
                    =1  2652     ;       command typed in at the console.
                    =1  2653     ;
                    =1  2654     ;    VARIABLES MODIFIED: ERRNUM, A, TEMP_LOW, VALLOW, PARAM1, C
                    =1  2655     ;
                    =1  2656     ;    ERROR EXITS: 1AH (TOKEN MUST BE A NUMBER)
                    =1  2657     ;
                    =1  2658     ;    SUBROUTINES ACCESSED DIRECTLY: IGETEOL, ISTORE, INC_PNT
                    =1  2659     ;
                    =1  2660     ;
                    =1  2661     ;*****************************************************************
EAAC 75431A         =1  2662     FILLMEM:MOV     ERRNUM,#1AH             ;Token must be a number
EAAF B401F7         =1  2663             CJNE    A,#NUMBER_TOKE,DISERR
EAB2 854A47         =1  2664             MOV     TEMP_LOW,VALLOW
EAB5 12E759         =1  2665             CALL    IGETEOL
EAB8 85474A         =1  2666             MOV     VALLOW,TEMP_LOW
EABB AA4A           =1  2667     FILLOOP:MOV     PARAM1,VALLOW           ;Load PARAM1 with single byte data
EABD 12E658         =1  2668             CALL    ISTORE                  ;Output data into memory
EAC0 C3             =1  2669             CLR     C
EAC1 E545           =1  2670             MOV     A,PNTLOW
EAC3 955A           =1  2671             SUBB    A,PARTIT_HI_LOW         ;Subtract pointer from ending address
EAC5 E544           =1  2672             MOV     A,PNTHGH
EAC7 9559           =1  2673             SUBB    A,PARTIT_HI_HIGH        ;to see if partition is full yet
EAC9 5005           =1  2674             JNC     FILL1                   ;If not, continue filling
EACB 12E5AA         =1  2675             CALL    INC_PNT
EACE 80EB           =1  2676             JMP     FILLOOP
EAD0 22             =1  2677     FILL1:  RET
                    =1  2678 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                   =1  2679     ;*********************************************************************
                   =1  2680     ;
                   =1  2681     ;    NAME: DISMEM
                   =1  2682     ;
                   =1  2683     ;    ABSTRACT: This routine displays the data values of the selected
                   =1  2684     ;       memory partition to the console.
                   =1  2685     ;
                   =1  2686     ;    INPUTS: PNTLOW, PNTHGH, PARTIT_HI_LOW, PARTIT_HI_HIGH
                   =1  2687     ;
                   =1  2688     ;    OUTPUTS: None
                   =1  2689     ;
                   =1  2690     ;    VARIABLES MODIFIED: COUNTR, A, DPTR, PARAM1, PARAM2
                   =1  2691     ;
                   =1  2692     ;    ERROR EXITS: None
                   =1  2693     ;
                   =1  2694     ;    SUBROUTINES ACCESSED DIRECTLY: INEWLINE, IDISPLAY_TOKEN, SPACCO,
                   =1  2695     ;       ILSTWRD, ICO, IFETCH, ILSTBYT, IWAIT_FOR_USER, ICONTINUATION_LINE
                   =1  2696     ;
                   =1  2697     ;
                   =1  2698     ;*********************************************************************
EAD1 755D01        =1  2699     DISMEM: MOV     COUNTR,#1               ;Load counter with 1
EAD4 155D          =1  2700     DISLOP: DEC     COUNTR
EAD6 E55D          =1  2701             MOV     A,COUNTR
EAD8 701E          =1  2702             JNZ     DISFET                  ;Jump to DISFET if counter is not zero
EADA 12E6FD        =1  2703             CALL    INEWLINE
EADD E546          =1  2704             MOV     A,SELECT                ;Move selector into ACC
EADF 90EB20        =1  2705             MOV     DPTR,#LAB23             ;Load DPTR with base of table
EAE2 93            =1  2706             MOVC    A,@A+DPTR
EAE3 FA            =1  2707             MOV     PARAM1,A                ;Setup for DISPLAY_TOKEN
EAE4 31E0          =1  2708             CALL    IDISPLAY_TOKEN          ;Output token
EAE6 12E5CC        =1  2709             CALL    SPACCO                  ;Output space
EAE9 AB45          =1  2710             MOV     PARAM2,PNTLOW
EAEB AA44          =1  2711             MOV     PARAM1,PNTHGH           ;Set-up for ILSTWRD
EAED 12E7DA        =1  2712             CALL    ILSTWRD                 ;Output address
EAF0 7A3D          =1  2713             MOV     PARAM1,#'='
EAF2 12E5CE        =1  2714             CALL    ICO                     ;Output an equal sign
EAF5 755D04        =1  2715             MOV     COUNTR,#4               ;Load counter with 4
EAF8 12E651        =1  2716     DISFET: CALL    IFETCH                  ;to get memory location
EAFB FA            =1  2717             MOV     PARAM1,A                ;Set-up for ILSTBYT
EAFC 12E7DF        =1  2718             CALL    ILSTBYT
EAFF E545          =1  2719             MOV     A,PNTLOW
EB01 B55A08        =1  2720             CJNE    A,PARTIT_HI_LOW,COUNT1  ;See if PARTIT_LO_LOW=EALOW
EB04 E544          =1  2721             MOV     A,PNTHGH
EB06 B55903        =1  2722             CJNE    A,PARTIT_HI_HIGH,COUNT1 ;See if PARTIT_LO_HIGH=EAHGH
EB09 02E396        =1  2723             JMP     IWAIT_FOR_USER
EB0C E55D          =1  2724     COUNT1: MOV     A,COUNTR
EB0E B40108        =1  2725             CJNE    A,#1,NTLAST             ;See if counter = 1,
EB11 12E643        =1  2726             CALL    ICONTINUATION_LINE
EB14 12E5AA        =1  2727     NOWAIT: CALL    INC_PNT
EB17 80BB          =1  2728             JMP     DISLOP                  ;Loop until PNT is > EA
EB19 7A2C          =1  2729     NTLAST: MOV     PARAM1,#','
EB1B 12E5CE        =1  2730             CALL    ICO                     ;To output a comma
EB1E 80F4          =1  2731             JMP     NOWAIT
                   =1  2732
EB20 80            =1  2733     LAB23:  DB      CBYTE_TOKE
```

```
LOC  OBJ            LINE     SOURCE

EB21 81         =1  2734             DB      RBYTE_TOKE
EB22 82         =1  2735             DB      DBYTE_TOKE
EB23 00         =1  2736             DB      00
EB24 84         =1  2737             DB      RBIT_TOKE
EB25 00         =1  2738             DB      00
EB26 86         =1  2739             DB      XBYTE_TOKE
                =1  2740 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                    =1  2741     ;*****************************************************************
                    =1  2742     ;
                    =1  2743     ;    NAME: BMOVE
                    =1  2744     ;
                    =1  2745     ;    ABSTRACT: This routine will transfer CBYTE type memory from
                    =1  2746     ;       a specific location to another location in blocks of contiguous
                    =1  2747     ;       code.  It does not relocate addresses and it is possible
                    =1  2748     ;       to lose code by writing a block over the TOP address.  The
                    =1  2749     ;       pointer direction is changed depending on the direction of
                    =1  2750     ;       the move so that no change to the data will occur if the
                    =1  2751     ;       destination and source blocks overlap.
                    =1  2752     ;
                    =1  2753     ;    INPUTS: SELECT, PARTIT_HI_LOW, PARTIT_HI_HIGH, LENGTH_LOW,
                    =1  2754     ;       LENGTH_HIGH, PARTIT_LO_LOW, PARTIT_LO_HIGH
                    =1  2755     ;
                    =1  2756     ;    OUTPUTS: Memory which was supposed to be accessed by the
                    =1  2757     ;       command typed in at the console.
                    =1  2758     ;
                    =1  2759     ;    VARIABLES MODIFIED: A, ERRNUM, C, PCNTLO, PCNTHI, PNTLOW, C,
                    =1  2760     ;       PARAM1, PNTHGH
                    =1  2761     ;
                    =1  2762     ;    ERROR EXITS: 18H (CBYTE TYPE ONLY)
                    =1  2763     ;
                    =1  2764     ;    SUBROUTINES ACCESSED DIRECTLY: IGETNUM, SWAP_POINTERS, IFETCH,
                    =1  2765     ;       DEC_PNT, ISTORE
                    =1  2766     ;
                    =1  2767     ;
                    =1  2768     ;*****************************************************************
                    =1  2769
EB27 E546           =1  2770     BMOVE:  MOV     A,SELECT
EB29 754318         =1  2771             MOV     ERRNUM,#18H                        ;CBYTE type only
EB2C B40077         =1  2772             CJNE    A,#(CBYTE_TOKE AND 7),ERRMOD
EB2F 12E74F         =1  2773             CALL    IGETNUM
EB32 854A62         =1  2774             MOV     PCNTLO,VALLOW
EB35 854961         =1  2775             MOV     PCNTHI,VALHGH
EB38 C3             =1  2776             CLR     C
EB39 E545           =1  2777             MOV     A,PNTLOW
EB3B 9562           =1  2778             SUBB    A,PCNTLO
EB3D E544           =1  2779             MOV     A,PNTHGH
EB3F 9561           =1  2780             SUBB    A,PCNTHI
EB41 4032           =1  2781             JC      DOWN_MOVE
EB43 855A45         =1  2782             MOV     PNTLOW,PARTIT_HI_LOW
EB46 855944         =1  2783             MOV     PNTHGH,PARTIT_HI_HIGH
EB49 E562           =1  2784             MOV     A,PCNTLO
EB4B 2564           =1  2785             ADD     A,LENGTH_LOW
EB4D F562           =1  2786             MOV     PCNTLO,A
EB4F E561           =1  2787             MOV     A,PCNTHI
EB51 3563           =1  2788             ADDC    A,LENGTH_HIGH
EB53 F561           =1  2789             MOV     PCNTHI,A
EB55 12E5BD         =1  2790     UP_MOVE:CALL    SWAP_POINTERS
EB58 12E651         =1  2791             CALL    IFETCH
EB5B FA             =1  2792             MOV     PARAM1,A
EB5C 12E5B3         =1  2793             CALL    DEC_PNT
EB5F 12E5BD         =1  2794             CALL    SWAP_POINTERS
EB62 12E658         =1  2795             CALL    ISTORE
```

```
LOC  OBJ            LINE     SOURCE

EB65 C3         =1  2796             CLR     C
EB66 E558       =1  2797             MOV     A,PARTIT_LO_LOW
EB68 9545       =1  2798             SUBB    A,PNTLOW
EB6A E557       =1  2799             MOV     A,PARTIT_LO_HIGH
EB6C 9544       =1  2800             SUBB    A,PNTHGH
EB6E 5025       =1  2801             JNC     BEND
EB70 12E5B3     =1  2802             CALL    DEC_PNT
EB73 80E0       =1  2803             JMP     UP_MOVE
                =1  2804     DOWN_MOVE:
EB75 12E5BD     =1  2805             CALL    SWAP_POINTERS
EB78 12E651     =1  2806             CALL    IFETCH
EB7B FA         =1  2807             MOV     PARAM1,A
EB7C 12E5AA     =1  2808             CALL    INC_PNT
EB7F 12E5BD     =1  2809             CALL    SWAP_POINTERS
EB82 12E658     =1  2810             CALL    ISTORE
EB85 C3         =1  2811             CLR     C
EB86 E545       =1  2812             MOV     A,PNTLOW
EB88 955A       =1  2813             SUBB    A,PARTIT_HI_LOW
EB8A E544       =1  2814             MOV     A,PNTHGH
EB8C 9559       =1  2815             SUBB    A,PARTIT_HI_HIGH
EB8E 5005       =1  2816             JNC     BEND
EB90 12E5AA     =1  2817             CALL    INC_PNT
EB93 80E0       =1  2818             JMP     DOWN_MOVE
EB95 22         =1  2819     BEND:   RET
                =1  2820 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                    =1  2821     ;*****************************************************************
                    =1  2822     ;
                    =1  2823     ;    NAME: BR_CMD
                    =1  2824     ;
                    =1  2825     ;    ABSTRACT: This routine checks a token to see if it is a
                    =1  2826     ;       breakpoint display or change.  If it is change, it sets the pamameters
                    =1  2827     ;       of the range and clears or sets the breakpoints requested. (ABR is
                    =1  2828     ;       a change only command). If it is a display command, each breakpoint
                    =1  2829     ;       is output to the console. Reset is the default condition.
                    =1  2830     ;       If the token is BR, the entire breakpoint RAM is cleared and then
                    =1  2831     ;       breakpoints are added. If it is ABR, they are added without clearing
                    =1  2832     ;       RAM first.
                    =1  2833     ;
                    =1  2834     ;    INPUTS: TOKSTR
                    =1  2835     ;
                    =1  2836     ;    OUTPUTS: Bits within the breakpoint hardware register.
                    =1  2837     ;
                    =1  2838     ;    VARIABLES MODIFIED: TOKSAV, A, ERRNUM, PARAM1, PARAM2, PARAM3, PARAM4,
                    =1  2839     ;       LINE_START, POINTO, PNTLOW, PNTHGH, DPTR, VPC_LOW, VPC_HIGH,
                    =1  2840     ;       ANY_BR_FLAG, FIRST_FLAG
                    =1  2841     ;
                    =1  2842     ;    ERROR EXITS: 19H (DISPLAY ONLY COMMAND)
                    =1  2843     ;                 05H (EQUAL OR RETURN EXPECTED)
                    =1  2844     ;                 0CH (NUMBER OR RESET REQUIRED)
                    =1  2845     ;
                    =1  2846     ;    SUBROUTINES ACCESSED DIRECTLY: IGETOKE, IERROR, IGET_PART, IEOL_CHECK
                    =1  2847     ;       IGETEOL, LSSEQL, IDISPLAY_TOKEN, IWAIT_FOR_USER, INC_PNT,
                    =1  2848     ;       ICONTINUATION_LINE, ILSTWRD, SPACCO, INEWLINE, ICO, IERROR
                    =1  2849     ;       BRK_LINE_HDR, SETBRK, CLRBRK
                    =1  2850     ;
                    =1  2851     ;
                    =1  2852     ;*****************************************************************
EB96 85485B         =1  2853     BR_CMD: MOV     TOKSAV,TOKSTR           ;Save last token for comparison
EB99 11A0           =1  2854             CALL    IGETOKE                 ;Get next token
EB9B B4070B         =1  2855             CJNE    A,#EOL_TOKE,EQLMOD      ;Check if token is end of line
EB9E E55B           =1  2856             MOV     A,TOKSAV                ;Move last token into ACC
EBA0 B4884C         =1  2857             CJNE    A,#ABR_TOKE,LSTBRK      ;Jump to list mod if not ABR token
EBA3 754319         =1  2858             MOV     ERRNUM,#19H             ;ABR is not a displayable command
EBA6 02E3CA         =1  2859     ERRMOD: JMP     IERROR
EBA9 754305         =1  2860     EQLMOD: MOV     ERRNUM,#05H             ;Equal or return expected
EBAC B404F7         =1  2861             CJNE    A,#EQUAL_TOKE,ERRMOD    ;Error if '=' not entered here
EBAF 11A0           =1  2862             CALL    IGETOKE
EBB1 B4012F         =1  2863             CJNE    A,#NUMBER_TOKE,RSTMOD
EBB4 E55B           =1  2864             MOV     A,TOKSAV                ;Recall last token entered
EBB6 B48903         =1  2865             CJNE    A,#BR_TOKE,NUMMOD       ;Check if it was break token
EBB9 12ECAF         =1  2866             CALL    CLRBRK                  ;Clear breakpoints
EBBC 12E788         =1  2867     NUMMOD: CALL    IGET_PART
EBBF 12ECC0         =1  2868             CALL    SETBRK
EBC2 E548           =1  2869             MOV     A,TOKSTR                ;Recall present token
EBC4 B40219         =1  2870             CJNE    A,#COMMA_TOKE,ENDMOD    ;Check if comma was entered
EBC7 11A0           =1  2871             CALL    IGETOKE
EBC9 B407F0         =1  2872             CJNE    A,#EOL_TOKE,NUMMOD      ;Check for EOL
EBCC 755204         =1  2873             MOV     LINE_START,#04H
EBCF 7824           =1  2874             MOV     POINTO,#LINBUF
EBD1 7641           =1  2875             MOV     @POINTO,#'A'
```

```
LOC  OBJ            LINE     SOURCE

EBD3 08          =1  2876             INC     POINTO
EBD4 7642        =1  2877             MOV     @POINTO,#'B'
EBD6 08          =1  2878             INC     POINTO
EBD7 7652        =1  2879             MOV     @POINTO,#'R'
EBD9 08          =1  2880             INC     POINTO
EBDA 763D        =1  2881             MOV     @POINTO,#'='
EBDC 11A0        =1  2882             CALL    IGETOKE
EBDE 80DC        =1  2883             JMP     NUMMOD
EBE0 02E5A1      =1  2884     ENDMOD: JMP     IEOL_CHECK
EBE3 75430C      =1  2885     RSTMOD: MOV     ERRNUM,#0CH             ;Number or reset required
EBE6 B40EBD      =1  2886             CJNE    A,#RESET_TOKE,ERRMOD    ;Check for reset entered
EBE9 12ECAF      =1  2887             CALL    CLRBRK
EBEC 02E759      =1  2888             JMP     IGETEOL
                 =1  2889     ;*****************************************************************
EBEF E4          =1  2890     LSTBRK: CLR     A
EBF0 F545        =1  2891             MOV     PNTLOW,A                ;Clear low byte of break pointer
EBF2 F544        =1  2892             MOV     PNTHGH,A                ;Clear high byte of break pointer
EBF4 C202        =1  2893             CLR     ANY_BR_FLAG
EBF6 D203        =1  2894             SETB    FIRST_FLAG
EBF8 90C000      =1  2895     LAB2:   MOV     DPTR,#BRKOFF
EBFB 7A1F        =1  2896             MOV     PARAM1,#MAXHGH
EBFD 7BFF        =1  2897             MOV     PARAM2,#MAXLOW
EBFF AC44        =1  2898             MOV     PARAM3,PNTHGH
EC01 AD45        =1  2899             MOV     PARAM4,PNTLOW           ;Set up for LSSEQL test
EC03 12E731      =1  2900             CALL    LSSEQL                  ;Check that P??? <= MAX???
EC06 400D        =1  2901             JC      LAB5B                   ;Exit if greater than
EC08 200207      =1  2902             JB      ANY_BR_FLAG,BRKEND      ;If any breakpoints were displayed
                 =1  2903                                             ;don't display reset
EC0B 12ECA3      =1  2904             CALL    BRK_LINE_HDR
EC0E 7A0E        =1  2905             MOV     PARAM1,#RESET_TOKE
EC10 31E0        =1  2906             CALL    IDISPLAY_TOKEN
EC12 02E396      =1  2907     BRKEND: JMP     IWAIT_FOR_USER
EC15 E545        =1  2908     LAB5B:  MOV     A,PNTLOW                ;Load ACC with break pointer low addr
EC17 2582        =1  2909             ADD     A,DPL                   ;Add low addr of break to pointer
EC19 F582        =1  2910             MOV     DPL,A                   ;Put new low addr back into DPL
EC1B 5002        =1  2911             JNC     LAB5A
EC1D 0583        =1  2912             INC     DPH                     ;Increment DPH if DPL had a carry
EC1F E544        =1  2913     LAB5A:  MOV     A,PNTHGH
EC21 2583        =1  2914             ADD     A,DPH
EC23 F583        =1  2915             MOV     DPH,A
EC25 E0          =1  2916             MOVX    A,@DPTR                 ;Load ACC with external RAM memory
EC26 30E005      =1  2917             JNB     ACC.0,LAB3              ;Branch if break is set.
EC29 12E5AA      =1  2918             CALL    INC_PNT
EC2C 80CA        =1  2919             JMP     LAB2
                 =1  2920
EC2E 85455E      =1  2921     LAB3:   MOV     VPC_LOW,PNTLOW          ;Save break pointer low
EC31 85445F      =1  2922             MOV     VPC_HIGH,PNTHGH         ;Save break pointer high
EC34 D202        =1  2923             SETB    ANY_BR_FLAG
EC36 90C000      =1  2924     BK1LOP: MOV     DPTR,#BRKOFF
EC39 AC44        =1  2925             MOV     PARAM3,PNTHGH
EC3B AD45        =1  2926             MOV     PARAM4,PNTLOW           ;Set up for LSSEQL
EC3D 12E731      =1  2927             CALL    LSSEQL                  ;Check that P??? <= MAX???
EC40 5019        =1  2928             JNC     LSTOUT                  ;Jump to LSTOUT if greater than
EC42 E545        =1  2929             MOV     A,PNTLOW                ;Load ACC with low addr of break pointer
EC44 2582        =1  2930             ADD     A,DPL                   ;Add break RAM low addr offset to pointer low
```

```
LOC  OBJ            LINE     SOURCE

EC46 F582       =1  2931             MOV     DPL,A                   ;Put new addr back into DPL
EC48 5002       =1  2932             JNC     LAB6A
EC4A 0583       =1  2933             INC     DPH                     ;Increment DPH if DPL produced a carry
EC4C E544       =1  2934     LAB6A:  MOV     A,PNTHGH
EC4E 2583       =1  2935             ADD     A,DPH
EC50 F583       =1  2936             MOV     DPH,A
EC52 E0         =1  2937             MOVX    A,@DPTR                 ;Load ACC with data in break RAM
EC53 20E005     =1  2938             JB      ACC.0,LSTOUT            ;Branch if break is off.
EC56 12E5AA     =1  2939             CALL    INC_PNT
EC59 80DB       =1  2940             JMP     BK1LOP                  ;Loop until 0 occurs or end of break RAM
EC5B AC5F       =1  2941     LSTOUT: MOV     PARAM3,VPC_HIGH
EC5D AD5E       =1  2942             MOV     PARAM4,VPC_LOW          ;Set up for LSSEQL
EC5F 12E731     =1  2943             CALL    LSSEQL                  ;Check that SA??? <= MAX???
EC62 5094       =1  2944             JNC     LAB2                    ;Jump to BRK0 and exit if true
EC64 200303     =1  2945             JB      FIRST_FLAG,LB_10
EC67 12E643     =1  2946             CALL    ICONTINUATION_LINE
EC6A 12ECA3     =1  2947     LB_10:  CALL    BRK_LINE_HDR
EC6D C203       =1  2948             CLR     FIRST_FLAG
EC6F AA5F       =1  2949             MOV     PARAM1,VPC_HIGH
EC71 AB5E       =1  2950             MOV     PARAM2,VPC_LOW          ;Set up for ILSTWRD
EC73 12E7DA     =1  2951             CALL    ILSTWRD                 ;Output starting addr of one's in BRK RAM
EC76 055E       =1  2952             INC     VPC_LOW                 ;Increment starting address low
EC78 E55E       =1  2953             MOV     A,VPC_LOW
EC7A 7002       =1  2954             JNZ     LAB7                    ;Check for rollover
EC7C 055F       =1  2955             INC     VPC_HIGH                ;Increment starting addr high if true
EC7E E55F       =1  2956     LAB7:   MOV     A,VPC_HIGH
EC80 B54407     =1  2957             CJNE    A,PNTHGH,OUTOKE         ;Jump to OUTOKE if VPC_HIGH+1 <> PNTHGH
EC83 E55E       =1  2958             MOV     A,VPC_LOW
EC85 B54502     =1  2959             CJNE    A,PNTLOW,OUTOKE         ;Jump to OUTOKE if VPC_LOW+1 <> PNTLOW
EC88 61F8       =1  2960             JMP     LAB2                    ;Go process the end of line.
                =1  2961
EC8A 12E5CC     =1  2962     OUTOKE: CALL    SPACCO
EC8D 7A0D       =1  2963             MOV     PARAM1,#TO_TOKE
EC8F 31E0       =1  2964             CALL    IDISPLAY_TOKEN          ;Call display_token(to_toke).
EC91 12E5CC     =1  2965             CALL    SPACCO
EC94 E545       =1  2966             MOV     A,PNTLOW                ;Load ACC with break pointer low
EC96 14         =1  2967             DEC     A                       ;Decrement break pointer low
EC97 FB         =1  2968             MOV     PARAM2,A                ;Set up for ILSTWRD
EC98 F4         =1  2969             CPL     A
EC99 AA44       =1  2970             MOV     PARAM1,PNTHGH           ;Set up for ILSTWRD
EC9B 7001       =1  2971             JNZ     LAB8                    ;Check for roll over
EC9D 1A         =1  2972             DEC     PARAM1                  ;Decrement break pointer high if true
EC9E 12E7DA     =1  2973     LAB8:   CALL    ILSTWRD
                =1  2974     ;
ECA1 61F8       =1  2975             JMP     LAB2                    ;Continue the display of break RAM.
                =1  2976                                             ;as soon as the user hits a character
                =1  2977     BRK_LINE_HDR:
ECA3 12E6FD     =1  2978             CALL    INEWLINE
ECA6 7A89       =1  2979             MOV     PARAM1,#BR_TOKE
ECA8 31E0       =1  2980             CALL    IDISPLAY_TOKEN
ECAA 7A3D       =1  2981             MOV     PARAM1,#'='
ECAC 02E5CE     =1  2982             JMP     IC0
                =1  2983 +1  $eject
```

```
LOC  OBJ            LINE     SOURCE

                =1  2984
                =1  2985     ;*****END OF LSTBRK*****
                =1  2986
ECAF 7AFF       =1  2987     CLRBRK: MOV     PARAM1,#MAXLOW          ;Load PARAM1 with size of break RAM,low 8 bits
ECB1 7B20       =1  2988             MOV     PARAM2,#(MAXHGH+1)      ;Load PARAM2 with size of break RAM+1,high bits
ECB3 90C000     =1  2989             MOV     DPTR,#BRKOFF            ;Load DPTR with break RAM offset
ECB6 7401       =1  2990             MOV     A,#01H                  ;To clear the break condition.
ECB8 F0         =1  2991     CLRLOP: MOVX    @DPTR,A                 ;Fill break RAM
ECB9 A3         =1  2992             INC     DPTR                    ;Increment pointer at break RAM
ECBA DAFC       =1  2993             DJNZ    PARAM1,CLRLOP           ;Repeat loop until PARAM1=0
ECBC F0         =1  2994             MOVX    @DPTR,A                 ;Once more for PARAM1=0
ECBD DBF9       =1  2995             DJNZ    PARAM2,CLRLOP           ;Continue loop until PARAM2=0
ECBF 22         =1  2996             RET                             ;Exit from CLRBRK
                =1  2997     ;*****END OF CLRBRK*****
                =1  2998
ECC0 C3         =1  2999     SETBRK: CLR     C
ECC1 E55A       =1  3000             MOV     A,PARTIT_HI_LOW         ;Load ACC with ending addr low
ECC3 9558       =1  3001             SUBB    A,PARTIT_LO_LOW         ;To obtain number of locations to set
ECC5 F582       =1  3002             MOV     DPL,A                   ;Save low number in PARAM4
ECC7 E559       =1  3003             MOV     A,PARTIT_HI_HIGH        ;Load ACC with ending addr high
ECC9 20E726     =1  3004             JB      ACC.7,BRKERR
ECCC 9557       =1  3005             SUBB    A,PARTIT_LO_HIGH        ;Subtract starting addr high from ending addr
ECCE F583       =1  3006             MOV     DPH,A                   ;Save high break count in PARAM3
ECD0 A3         =1  3007             INC     DPTR
ECD1 0583       =1  3008             INC     DPH
ECD3 AA83       =1  3009             MOV     PARAM1,DPH
ECD5 AB82       =1  3010             MOV     PARAM2,DPL
ECD7 90C000     =1  3011             MOV     DPTR,#BRKOFF
ECDA E557       =1  3012             MOV     A,PARTIT_LO_HIGH
ECDC 541F       =1  3013             ANL     A,#MAXHGH
ECDE FD         =1  3014             MOV     TEMP,A
ECDF E558       =1  3015             MOV     A,PARTIT_LO_LOW         ;Put starting addr low in ACC
ECE1 2582       =1  3016             ADD     A,DPL                   ;Add break offset low
ECE3 F582       =1  3017             MOV     DPL,A                   ;Put back into data pointer
ECE5 ED         =1  3018             MOV     A,TEMP                  ;Load ACC with starting addr high
ECE6 3583       =1  3019             ADDC    A,DPH                   ;Add break offset high
ECE8 F583       =1  3020             MOV     DPH,A                   ;Load DPH with starting addr high + offset
ECEA E4         =1  3021     OUT1BK: CLR     A                       ;To output 0'S
ECEB F0         =1  3022             MOVX    @DPTR,A                 ;Load break RAM
ECEC A3         =1  3023             INC     DPTR                    ;Increment break RAM pointer
ECED DBFB       =1  3024             DJNZ    PARAM2,OUT1BK           ;Loop until count low=0
ECEF DAF9       =1  3025             DJNZ    PARAM1,OUT1BK           ;Loop until PARAM3=0
ECF1 22         =1  3026             RET                             ;Exit from SETBRK
                =1  3027     ;*****END OF SETBRK*****
                =1  3028
ECF2 75430D     =1  3029     BRKERR: MOV     ERRNUM,#0DH             ;7 is the error number for
                =1  3030                                             ;break range low > range high
ECF5 02E3CA     =1  3031             JMP     IERROR                  ;Exit from break routine on error
                =1  3032
                =1  3033
                =1  3034 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1  3035     ;*********************************************************************
                =1  3036     ;
                =1  3037     ;    NAME: ACC_CMD/ PSW_CMD/ SP_CMD/ B_CMD
                =1  3038     ;
                =1  3039     ;    ABSTRACT: Displays or modifies the byte which is referenced
                =1  3040     ;       by the user register images passed to it.
                =1  3041     ;
                =1  3042     ;    INPUTS: None
                =1  3043     ;
                =1  3044     ;    OUTPUTS: Users version of the PC, DPTR, TMO, TM1
                =1  3045     ;
                =1  3046     ;    VARIABLES MODIFIED: PNTLOW, PNTHGH, SELECT, PARAM1
                =1  3047     ;
                =1  3048     ;    ERROR EXITS: None
                =1  3049     ;
                =1  3050     ;    SUBROUTINES ACCESSED DIRECTLY: ISIT_DISPLAY, IFETCH, ILSTBYT,
                =1  3051     ;       IWAIT_FOR_USER, ISTORE, KEY_BYTES
                =1  3052     ;
                =1  3053     ;
                =1  3054     ;*********************************************************************
                =1  3055     ACC_CMD:
ECF8 7545E0     =1  3056             MOV     PNTLOW,#ACC
ECFB 02ED10     =1  3057             JMP     KEY_BYTE
                =1  3058     ;*********************************************************************
                =1  3059     PSW_CMD:
ECFE 7545D0     =1  3060             MOV     PNTLOW,#PSW
ED01 02ED10     =1  3061             JMP     KEY_BYTE
                =1  3062     ;*********************************************************************
                =1  3063     SP_CMD:
ED04 754581     =1  3064             MOV     PNTLOW,#SP
ED07 02ED10     =1  3065             JMP     KEY_BYTE
                =1  3066     ;*********************************************************************
                =1  3067     B_CMD:
ED0A 7545F0     =1  3068             MOV     PNTLOW,#B
ED0D 02ED10     =1  3069             JMP     KEY_BYTE
                =1  3070     ;*********************************************************************
                =1  3071     KEY_BYTE:
ED10 12E76A     =1  3072             CALL    ISIT_DISPLAY
ED13 754400     =1  3073             MOV     PNTHGH,#00H
ED16 754601     =1  3074             MOV     SELECT,#(RBYTE_TOKE AND 07H)    ;Set-up for fetch
ED19 500A       =1  3075             JNC     CHANGE
ED1B 12E651     =1  3076             CALL    IFETCH
ED1E FA         =1  3077             MOV     PARAM1,A                        ;Call ILSTBYT (result) to display it
ED1F 12E7DF     =1  3078             CALL    ILSTBYT
ED22 02E396     =1  3079             JMP     IWAIT_FOR_USER
                =1  3080     CHANGE:
ED25 12E74F     =1  3081             CALL    IGETNUM                         ;Get the numeric parameter
ED28 AA4A       =1  3082             MOV     PARAM1,VALLOW
ED2A 02E658     =1  3083             JMP     ISTORE
                =1  3084 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1  3085     ;*****************************************************************
                =1  3086     ;
                =1  3087     ;    NAME: PC_CMD/ DPTR_CMD/ TM0_CMD/ TM1_CMD
                =1  3088     ;
                =1  3089     ;    ABSTRACT:  Decodes and exeutes those commands which display or alter
                =1  3090     ;       sixteen bit variables which have unique keywords to identify
                =1  3091     ;       them.
                =1  3092     ;
                =1  3093     ;    INPUTS: None
                =1  3094     ;
                =1  3095     ;    OUTPUTS: Users version of the PC, DPTR, TM0 and TM1
                =1  3096     ;
                =1  3097     ;    VARIABLES MODIFIED: PARAM1, PARAM2, PNTLOW, TEMP_LOW, PNTHGH, A
                =1  3098     ;
                =1  3099     ;    ERROR EXITS: None
                =1  3100     ;
                =1  3101     ;    SUBROUTINES ACCESSED DIRECTLY: ISIT_DISPLAY, READ_PC, ILSTWRD,
                =1  3102     ;       WRITE_PC, IFETCH, ISTORE, IGETEOL, IGETNUM, IWAIT_FOR_USER,
                =1  3103     ;       KEYWORD_DISPLAY
                =1  3104     ;
                =1  3105     ;
                =1  3106     ;*****************************************************************
                =1  3107     PC_CMD:
ED2D 12E76A     =1  3108             CALL    ISIT_DISPLAY
ED30 500C       =1  3109             JNC     PC_CHA
ED32 12EF58     =1  3110             CALL    READ_PC                 ;Get the user program counter.
ED35 FB         =1  3111             MOV     PARAM2,A                ;And set up parameters to display it.
ED36 AAF0       =1  3112             MOV     PARAM1,B
ED38 12E7DA     =1  3113             CALL    ILSTWRD
ED3B 02E396     =1  3114             JMP     IWAIT_FOR_USER
                =1  3115     PC_CHA:
ED3E 12E74F     =1  3116             CALL    IGETNUM
ED41 AA49       =1  3117             MOV     PARAM1,VALHGH
ED43 AB4A       =1  3118             MOV     PARAM2,VALLOW
ED45 12EF67     =1  3119             CALL    WRITE_PC
ED48 02E759     =1  3120             JMP     IGETEOL
                =1  3121     ;*****************************************************************
                =1  3122     DPTR_CMD:
ED4B 754583     =1  3123             MOV     PNTLOW,#DPH
ED4E 754782     =1  3124             MOV     TEMP_LOW,#DPL
ED51 02ED63     =1  3125             JMP     KEYWORD_DISPLAY
                =1  3126     ;*****************************************************************
                =1  3127     TM0_CMD:
ED54 75458C     =1  3128             MOV     PNTLOW,#TH0
ED57 75478A     =1  3129             MOV     TEMP_LOW,#TL0
ED5A 02ED63     =1  3130             JMP     KEYWORD_DISPLAY
                =1  3131     ;*****************************************************************
                =1  3132     TM1_CMD:
ED5D 75458D     =1  3133             MOV     PNTLOW,#TH1
ED60 75478B     =1  3134             MOV     TEMP_LOW,#TL1
                =1  3135     ;*****************************************************************
                =1  3136     KEYWORD_DISPLAY:
ED63 12E76A     =1  3137             CALL    ISIT_DISPLAY
ED66 754601     =1  3138             MOV     SELECT,#(RBYTE_TOKE AND 07H)
ED69 754400     =1  3139             MOV     PNTHGH,#0
```

```
LOC  OBJ            LINE     SOURCE

ED6C 5013       =1  3140             JNC     WCHANGE
ED6E 12E651     =1  3141             CALL    IFETCH
ED71 C547       =1  3142             XCH     A,TEMP_LOW
ED73 F545       =1  3143             MOV     PNTLOW,A
ED75 12E651     =1  3144             CALL    IFETCH
ED78 FB         =1  3145             MOV     PARAM2,A
ED79 AA47       =1  3146             MOV     PARAM1,TEMP_LOW
ED7B 12E7DA     =1  3147             CALL    ILSTWRD
ED7E 02E396     =1  3148             JMP     IWAIT_FOR_USER          ;Wait for CR then start the monitor.
                =1  3149     WCHANGE:
ED81 12E74F     =1  3150             CALL    IGETNUM                 ;If it is, get the data to be loaded.
ED84 AA49       =1  3151             MOV     PARAM1,VALHGH
ED86 12E658     =1  3152             CALL    ISTORE
ED89 854745     =1  3153             MOV     PNTLOW,TEMP_LOW
ED8C AA4A       =1  3154             MOV     PARAM1,VALLOW
ED8E 12E658     =1  3155             CALL    ISTORE
ED91 02E759     =1  3156             JMP     IGETEOL                 ;Process end of line and return to the
                =1  3157     ;*********************************************************************
                    3158 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                    3159 +1  $INCLUDE(:F1:XQT.INC)
                  =1 3160    ;*****************************************************************
                  =1 3161    ;
                  =1 3162    ;    NAME: (I)BREAK
                  =1 3163    ;
                  =1 3164    ;    ABSTRACT: Control is transferred to this point when a break
                  =1 3165    ;       interrupt occurs.  The current user status is saved in the
                  =1 3166    ;       page of external RAM starting at 'RAMOFF' and control then
                  =1 3167    ;       passes to one of the return routines, STEP return and RUN
                  =1 3168    ;       return.
                  =1 3169    ;
                  =1 3170    ;    INPUTS: BREAK_STATUS, MON_FLAGS
                  =1 3171    ;
                  =1 3172    ;    OUTPUTS: LINE_START, CAUSE_IMAGE, UPI_DATA_IMAGE, all the users
                  =1 3173    ;       RAM and register image area.
                  =1 3174    ;
                  =1 3175    ;    VARIABLES MODIFIED: DPTR, SP, A, IE, POINTO, CAUSE_IMAGE,
                  =1 3176    ;       ERRNUM, C, B, PARAM1, LINE_START, UPI_DATA_IMAGE
                  =1 3177    ;
                  =1 3178    ;    ERROR EXITS: 16H (EXECUTION OVER VECTOR AT LOCATION 3)
                  =1 3179    ;
                  =1 3180    ;    SUBROUTINES ACCESSED DIRECTLY: ICSTS, UPI_IN, WRITE_PC, READ_PC,
                  =1 3181    ;       INIT_IO, UPI_OUT, SET_BAUD, UPI_CMD, SIGN_ON, STEP51_RETURN,
                  =1 3182    ;       UNBREAK, RUN_USER_RETURN
                  =1 3183    ;
                  =1 3184    ;
                  =1 3185    ;*****************************************************************
ED94 C082         =1 3186    IBREAK: PUSH    DPL                     ;Save DPTR in the user stack.
ED96 C083         =1 3187            PUSH    DPH
ED98 90B0E0       =1 3188            MOV     DPTR,#(RAMOFF+ACC)
ED9B F0           =1 3189            MOVX    @DPTR,A                 ;Save user ACC.
ED9C 758283       =1 3190            MOV     DPL,#DPH
ED9F D0E0         =1 3191            POP     ACC
EDA1 F0           =1 3192            MOVX    @DPTR,A                 ;Move user DPH from the stack to save area.
EDA2 1582         =1 3193            DEC     DPL
EDA4 D0E0         =1 3194            POP     ACC
EDA6 F0           =1 3195            MOVX    @DPTR,A                 ;Move user DPL from the stack to save area.
EDA7 7582A8       =1 3196            MOV     DPL,#IE                 ;Save the special function registers.
EDAA E5A8         =1 3197            MOV     A,IE
EDAC F0           =1 3198            MOVX    @DPTR,A
EDAD 75A800       =1 3199            MOV     IE,#00H
EDB0 758288       =1 3200            MOV     DPL,#TCON
EDB3 E588         =1 3201            MOV     A,TCON
EDB5 F0           =1 3202            MOVX    @DPTR,A
EDB6 758800       =1 3203            MOV     TCON,#0
EDB9 7582F0       =1 3204            MOV     DPL,#B                  ;Start with 'B'.
EDBC E5F0         =1 3205            MOV     A,B
EDBE F0           =1 3206            MOVX    @DPTR,A
EDBF 7582B8       =1 3207            MOV     DPL,#IP
EDC2 E5B8         =1 3208            MOV     A,IP
EDC4 F0           =1 3209            MOVX    @DPTR,A
EDC5 758290       =1 3210            MOV     DPL,#P1
EDC8 E590         =1 3211            MOV     A,P1
EDCA F0           =1 3212            MOVX    @DPTR,A
EDCB 7582B0       =1 3213            MOV     DPL,#P3
```

```
LOC  OBJ            LINE     SOURCE

EDCE E5B0       =1  3214             MOV     A,P3
EDD0 F0         =1  3215             MOVX    @DPTR,A
EDD1 7582D0     =1  3216             MOV     DPL,#PSW
EDD4 E5D0       =1  3217             MOV     A,PSW
EDD6 F0         =1  3218             MOVX    @DPTR,A
EDD7 758298     =1  3219             MOV     DPL,#SCON
EDDA E598       =1  3220             MOV     A,SCON
EDDC F0         =1  3221             MOVX    @DPTR,A
EDDD 758281     =1  3222             MOV     DPL,#SP
EDE0 E581       =1  3223             MOV     A,SP
EDE2 14         =1  3224             DEC     A                       ;Compensate the SP for the break interrupt
EDE3 14         =1  3225             DEC     A
EDE4 F0         =1  3226             MOVX    @DPTR,A
EDE5 758107     =1  3227             MOV     SP,#STACK
EDE8 75828C     =1  3228             MOV     DPL,#TH0
EDEB E58C       =1  3229             MOV     A,TH0
EDED F0         =1  3230             MOVX    @DPTR,A
EDEE 75828D     =1  3231             MOV     DPL,#TH1
EDF1 E58D       =1  3232             MOV     A,TH1
EDF3 F0         =1  3233             MOVX    @DPTR,A
EDF4 75828A     =1  3234             MOV     DPL,#TL0
EDF7 E58A       =1  3235             MOV     A,TL0
EDF9 F0         =1  3236             MOVX    @DPTR,A
EDFA 75828B     =1  3237             MOV     DPL,#TL1
EDFD E58B       =1  3238             MOV     A,TL1
EDFF F0         =1  3239             MOVX    @DPTR,A
EE00 758289     =1  3240             MOV     DPL,#TMOD
EE03 E589       =1  3241             MOV     A,TMOD
EE05 F0         =1  3242             MOVX    @DPTR,A                 ;Save the user internal RAM.
EE06 758200     =1  3243             MOV     DPL,#0                  ;Set DPTR to start of save area.
EE09 75D000     =1  3244             MOV     PSW,#0                  ;Select register bank 0.
EE0C E8         =1  3245             MOV     A,R0                    ;Save users R0 (our POINT0)
EE0D F0         =1  3246             MOVX    @DPTR,A
EE0E 7801       =1  3247             MOV     POINT0,#01H             ;Then save user RAM.
                =1  3248     BRK_LOOP:
EE10 A3         =1  3249             INC     DPTR
EE11 E6         =1  3250             MOV     A,@POINT0
EE12 F0         =1  3251             MOVX    @DPTR,A
EE13 08         =1  3252             INC     POINT0
EE14 B880F9     =1  3253             CJNE    POINT0,#128,BRK_LOOP
EE17 90B0FA     =1  3254             MOV     DPTR,#(RAMOFF+MON_FLAGS)
EE1A E0         =1  3255             MOVX    A,@DPTR
EE1B F520       =1  3256             MOV     20H,A                   ;Move the monitor flags storage area to the
                =1  3257                                             ;first eight bit locations.
EE1D 7582FB     =1  3258             MOV     DPL,#BREAK_STATUS
EE20 E0         =1  3259             MOVX    A,@DPTR                 ;See if break was invoked by the power
EE21 6023       =1  3260             JZ      BREAK_CONTINUE          ;on and skip further checks if it was.
                =1  3261                                             ;If not continue.
EE23 90C000     =1  3262             MOV     DPTR,#BRKOFF            ;Find the cause of the break
EE26 E0         =1  3263             MOVX    A,@DPTR
EE27 F560       =1  3264             MOV     CAUSE_IMAGE,A
EE29 543C       =1  3265             ANL     A,#03CH
EE2B 7019       =1  3266             JNZ     BREAK_CONTINUE
EE2D 12E5E8     =1  3267             CALL    ICSTS                   ;No break set up-was it a keyboard entry?
EE30 4009       =1  3268             JC      BRKMORE
```

```
LOC  OBJ            LINE     SOURCE

EE32 754316     =1  3269             MOV     ERRNUM,#16H             ;Execution over vector at loc 3
EE35 756004     =1  3270             MOV     CAUSE_IMAGE,#4          ;Cause is guarded access.
EE38 02E3CA     =1  3271             JMP     IERROR
                =1  3272     BRKMORE:
EE3B 12E632     =1  3273             CALL    UPI_IN                  ;Else get the character
EE3E 547F       =1  3274             ANL     A,#7FH
EE40 B41B63     =1  3275             CJNE    A,#ESC,PRE_UNBREAK      ;Return to the user unless char is an ESCAPE.
EE43 756002     =1  3276             MOV     CAUSE_IMAGE,#2          ;Cause is user abort.
                =1  3277     BREAK_CONTINUE:                         ;The interrupt is due to a valid break.
                =1  3278                                             ;Determine which one and reenter the
                =1  3279                                             ;monitor at the appropriate point.
EE46 75A800     =1  3280             MOV     IE,#0                   ;Shut down all the interrupts while in the
EE49 758107     =1  3281             MOV     SP,#STACK               ;Set up the monitor stack pointer
EE4C E560       =1  3282             MOV     A,CAUSE_IMAGE
EE4E 20E409     =1  3283             JB      ACC.4,BRK3              ;Always adjust for data break
EE51 5428       =1  3284             ANL     A,#28H
EE53 6015       =1  3285             JZ      BRK4                    ;Bypass adjusting PC for any break
EE55 E560       =1  3286             MOV     A,CAUSE_IMAGE           ;except PROG or  STEP
EE57 30E610     =1  3287             JNB     ACC.6,BRK4              ;Check to see if NOP was forced on break.
                =1  3288                                             ;(i.e. PC is too big)
EE5A 12EF58     =1  3289     BRK3:   CALL    READ_PC
EE5D C3         =1  3290             CLR     C
EE5E 9401       =1  3291             SUBB    A,#1
EE60 5002       =1  3292             JNC     BRK5
EE62 15F0       =1  3293             DEC     B
EE64 FB         =1  3294     BRK5:   MOV     PARAM2,A
EE65 AAF0       =1  3295             MOV     PARAM1,B
EE67 12EF67     =1  3296             CALL    WRITE_PC
EE6A 12E36C     =1  3297     BRK4:   CALL    INIT_IO
EE6D 7A83       =1  3298             MOV     PARAM1,#TOP_PORT
EE6F 12E60B     =1  3299             CALL    UPI_CMD
EE72 7A00       =1  3300             MOV     PARAM1,#0
EE74 12E61E     =1  3301             CALL    UPI_OUT
EE77 12E632     =1  3302             CALL    UPI_IN                  ;Clear UPIOBF
EE7A 12E36C     =1  3303             CALL    INIT_IO
EE7D 12F1EA     =1  3304             CALL    SET_BAUD
EE80 A201       =1  3305             MOV     C,LSTFLG
EE82 755200     =1  3306             MOV     LINE_START,#0
EE85 E4         =1  3307             CLR     A
EE86 92E6       =1  3308             MOV     ACC.6,C
EE88 FA         =1  3309             MOV     PARAM1,A
EE89 12E60B     =1  3310             CALL    UPI_CMD
EE8C 90B0FB     =1  3311             MOV     DPTR,#(RAMOFF+BREAK_STATUS)
EE8F E0         =1  3312             MOVX    A,@DPTR
EE90 7003       =1  3313             JNZ     BRK1
EE92 02E2BF     =1  3314             JMP     SIGN_ON
EE95 E560       =1  3315     BRK1:   MOV     A,CAUSE_IMAGE
EE97 541E       =1  3316             ANL     A,#1EH
EE99 6003       =1  3317             JZ      BRK2                    ;Check for cause other than singlestep
EE9B 02F14F     =1  3318             JMP     RUN_USER_RETURN
EE9E E560       =1  3319     BRK2:   MOV     A,CAUSE_IMAGE
EEA0 30E503     =1  3320             JNB     ACC.5,PRE_UNBREAK       ;Reenter execution if not singlestep
EEA3 02F012     =1  3321             JMP     STEP51_RETURN           ;Return to the step command.
                =1  3322     PRE_UNBREAK:
EEA6 90B0F1     =1  3323             MOV     DPTR,#(RAMOFF+UPI_DATA_IMAGE)
```

```
LOC  OBJ            LINE     SOURCE

EEA9 D2E7       =1  3324             SETB    ACC.7
EEAB F0         =1  3325             MOVX    @DPTR,A                 ;escape
                =1  3326 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1  3327     ;*********************************************************************
                =1  3328     ;
                =1  3329     ;    NAME: UNBREAK
                =1  3330     ;
                =1  3331     ;    ABSTRACT: Restores the user status and starts execution of the
                =1  3332     ;       user program. CAUTION: This routine is position sensitive.
                =1  3333     ;       It is entered from BREAK as "in line" code.
                =1  3334     ;
                =1  3335     ;    INPUTS: All of the users registers and RAM images wil be used.,
                =1  3336     ;       TOP_STORE
                =1  3337     ;
                =1  3338     ;    OUTPUTS: MON_FLAGS
                =1  3339     ;
                =1  3340     ;    VARIABLES MODIFIED: A, DPTR, R0, B, PSW, SCON, SP, IP, TH0,
                =1  3341     ;       TH1, TMOD, TCON, IE, IE0, IT0, PX0
                =1  3342     ;
                =1  3343     ;    ERROR EXITS: None
                =1  3344     ;
                =1  3345     ;    SUBROUTINES ACCESSED DIRECTLY: UPI_CMD, UPI_OUT
                =1  3346     ;
                =1  3347     ;*********************************************************************
EEAC 7A01       =1  3348     UNBREAK:MOV     PARAM1,#USART_MODE
EEAE 12E60B     =1  3349             CALL    UPI_CMD
EEB1 7AFF       =1  3350             MOV     PARAM1,#0FFH
EEB3 12E61E     =1  3351             CALL    UPI_OUT
EEB6 12E61E     =1  3352             CALL    UPI_OUT                 ;Output nulls to clr usart b/f reset in break
EEB9 7A83       =1  3353             MOV     PARAM1,#TOP_PORT
EEBB 12E60B     =1  3354             CALL    UPI_CMD
EEBE 90B0F9     =1  3355             MOV     DPTR,#(RAMOFF+TOP_STORE)
EEC1 E0         =1  3356             MOVX    A,@DPTR
EEC2 FA         =1  3357             MOV     PARAM1,A
EEC3 12E61E     =1  3358             CALL    UPI_OUT
EEC6 12E632     =1  3359             CALL    UPI_IN                  ;Clear UPIOBF
EEC9 7A00       =1  3360             MOV     PARAM1,#SELECT_CON      ;Re-enable the console for I/O
EECB 12E60B     =1  3361             CALL    UPI_CMD                 ;then return
EECE E520       =1  3362             MOV     A,20H                   ;Save the MON_FLAGS during execution.
EED0 7582FA     =1  3363             MOV     DPL,#MON_FLAGS
EED3 F0         =1  3364             MOVX    @DPTR,A
EED4 787F       =1  3365             MOV     R0,#127                 ;First restore the internal RAM.
EED6 75827F     =1  3366             MOV     DPL,#127
                =1  3367     UNBRK_LOOP:
EED9 E0         =1  3368             MOVX    A,@DPTR
EEDA F6         =1  3369             MOV     @R0,A
EEDB 1582       =1  3370             DEC     DPL
EEDD D8FA       =1  3371             DJNZ    R0,UNBRK_LOOP
EEDF E0         =1  3372             MOVX    A,@DPTR
EEE0 F6         =1  3373             MOV     @R0,A
EEE1 7582F0     =1  3374             MOV     DPL,#B
EEE4 E0         =1  3375             MOVX    A,@DPTR
EEE5 F5F0       =1  3376             MOV     B,A
EEE7 758290     =1  3377             MOV     DPL,#P1
EEEA E0         =1  3378             MOVX    A,@DPTR
EEEB F590       =1  3379             MOV     P1,A
EEED 7582B0     =1  3380             MOV     DPL,#P3
EEF0 E0         =1  3381             MOVX    A,@DPTR
```

```
LOC  OBJ            LINE     SOURCE

EEF1 44C4       =1  3382             ORL     A,#0C4H
EEF3 F5B0       =1  3383             MOV     P3,A
EEF5 7582D0     =1  3384             MOV     DPL,#PSW
EEF8 E0         =1  3385             MOVX    A,@DPTR
EEF9 F5D0       =1  3386             MOV     PSW,A
EEFB 758298     =1  3387             MOV     DPL,#SCON
EEFE E0         =1  3388             MOVX    A,@DPTR
EEFF F598       =1  3389             MOV     SCON,A
EF01 758281     =1  3390             MOV     DPL,#SP
EF04 E0         =1  3391             MOVX    A,@DPTR
EF05 04         =1  3392             INC     A                       ;Allow for PC on STACK,
                =1  3393                                             ;RETI will POP PC and adjust SP
EF06 04         =1  3394             INC     A
EF07 F581       =1  3395             MOV     SP,A
EF09 7582B8     =1  3396             MOV     DPL,#IP
EF0C E0         =1  3397             MOVX    A,@DPTR
EF0D F5B8       =1  3398             MOV     IP,A
EF0F 75828C     =1  3399             MOV     DPL,#TH0
EF12 E0         =1  3400             MOVX    A,@DPTR
EF13 F58C       =1  3401             MOV     TH0,A
EF15 75828D     =1  3402             MOV     DPL,#TH1
EF18 E0         =1  3403             MOVX    A,@DPTR
EF19 F58D       =1  3404             MOV     TH1,A
EF1B 75828A     =1  3405             MOV     DPL,#TL0
EF1E E0         =1  3406             MOVX    A,@DPTR
EF1F F58A       =1  3407             MOV     TL0,A
EF21 75828B     =1  3408             MOV     DPL,#TL1
EF24 E0         =1  3409             MOVX    A,@DPTR
EF25 F58B       =1  3410             MOV     TL1,A
EF27 758289     =1  3411             MOV     DPL,#TMOD
EF2A E0         =1  3412             MOVX    A,@DPTR
EF2B F589       =1  3413             MOV     TMOD,A
EF2D 758288     =1  3414             MOV     DPL,#TCON
EF30 E0         =1  3415             MOVX    A,@DPTR
EF31 F588       =1  3416             MOV     TCON,A
EF33 7582A8     =1  3417             MOV     DPL,#IE
EF36 E0         =1  3418             MOVX    A,@DPTR
EF37 547E       =1  3419             ANL     A,#07EH                 ;Leave overall enable and external 0 off until
                =1  3420                                             ;interrupt mode is established.
EF39 F5A8       =1  3421             MOV     IE,A                    ;Set up IE.
EF3B 758282     =1  3422             MOV     DPL,#DPL
EF3E E0         =1  3423             MOVX    A,@DPTR
EF3F C0E0       =1  3424             PUSH    ACC                     ;Push user data pointer into the user stack.
EF41 0582       =1  3425             INC     DPL
EF43 E0         =1  3426             MOVX    A,@DPTR
EF44 C0E0       =1  3427             PUSH    ACC
EF46 7582E0     =1  3428             MOV     DPL,#ACC                ;Restore the user A register.
EF49 E0         =1  3429             MOVX    A,@DPTR
EF4A D083       =1  3430             POP     DPH
EF4C D082       =1  3431             POP     DPL                     ;Restore user data pointer.
EF4E C289       =1  3432             CLR     IE0                     ;Set up the break logic interrupts.
EF50 D288       =1  3433             SETB    IT0
EF52 D2B8       =1  3434             SETB    PX0
EF54 43A881     =1  3435             ORL     IE,#81H                 ;Edge mode, highest priority.
EF57 32         =1  3436             RETI                            ;'Return' to the user.
```

```
LOC  OBJ            LINE     SOURCE
                =1  3437 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                    =1  3438     ;*********************************************************************
                    =1  3439     ;
                    =1  3440     ;    NAME: READ_PC/WRITE_PC
                    =1  3441     ;
                    =1  3442     ;    ABSTRACT:
                    =1  3443     ;       READ_PC:  This routine returns a copy of the user program
                    =1  3444     ;       counter in A and B from the page of external RAM devoted to
                    =1  3445     ;       saving the user status.
                    =1  3446     ;
                    =1  3447     ;       WRITE_PC: this routine loads the user program counter
                    =1  3448     ;       with the parameter passed to it.
                    =1  3449     ;
                    =1  3450     ;    INPUTS: PARAM1 (high byte), PARAM2 (low byte)
                    =1  3451     ;
                    =1  3452     ;    OUTPUTS: ACC (low byte), B (high byte), users version of PC
                    =1  3453     ;
                    =1  3454     ;    VARIABLES MODIFIED: DPTR, A, B
                    =1  3455     ;
                    =1  3456     ;    ERROR EXITS: None
                    =1  3457     ;
                    =1  3458     ;    SUBROUTINES ACCESSED DIRECTLY: None
                    =1  3459     ;
                    =1  3460     ;
                    =1  3461     ;*********************************************************************
                    =1  3462     READ_PC:                        ;Set DPTR to point at the user PC in the
                    =1  3463                                     ;user stack.
EF58 90B081         =1  3464             MOV     DPTR,#(RAMOFF+SP)
EF5B E0             =1  3465             MOVX    A,@DPTR
EF5C F582           =1  3466             MOV     DPL,A
EF5E A3             =1  3467             INC     DPTR
EF5F E0             =1  3468             MOVX    A,@DPTR
EF60 F5F0           =1  3469             MOV     B,A             ;Load the user pc into B and A.
EF62 A3             =1  3470             INC     DPTR
EF63 E0             =1  3471             MOVX    A,@DPTR
EF64 C5F0           =1  3472             XCH     A,B
EF66 22             =1  3473             RET
                    =1  3474     WRITE_PC:                       ;Set the DPTR to point at the user PC in the
                    =1  3475                                     ;user stack.
EF67 90B081         =1  3476             MOV     DPTR,#(RAMOFF+SP)
EF6A E0             =1  3477             MOVX    A,@DPTR
EF6B F582           =1  3478             MOV     DPL,A
EF6D A3             =1  3479             INC     DPTR
EF6E EB             =1  3480             MOV     A,PARAM2        ;Write into the user PC.
EF6F F0             =1  3481             MOVX    @DPTR,A
EF70 A3             =1  3482             INC     DPTR
EF71 EA             =1  3483             MOV     A,PARAM1
EF72 F0             =1  3484             MOVX    @DPTR,A
EF73 22             =1  3485             RET
                    =1  3486 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                   =1  3487     ;*****************************************************************
                   =1  3488     ;
                   =1  3489     ;    NAME: CHECK_FROM
                   =1  3490     ;
                   =1  3491     ;    ABSTRACT: This routine gets a token and if it is a 'from', it
                   =1  3492     ;       will get the number and send it to the users PC.  It always
                   =1  3493     ;       leaves this routine with a 'fresh' token whether it finds a
                   =1  3494     ;       'from' or not.
                   =1  3495     ;
                   =1  3496     ;    INPUTS: None
                   =1  3497     ;
                   =1  3498     ;    OUTPUTS: TOKSTR
                   =1  3499     ;
                   =1  3500     ;    VARIABLES MODIFIED: PARAM1, PARAM2
                   =1  3501     ;
                   =1  3502     ;    ERROR EXITS: None
                   =1  3503     ;
                   =1  3504     ;    SUBROUTINES ACCESSED DIRECTLY: IGETOKE, IGETNUM, WRITE_PC
                   =1  3505     ;
                   =1  3506     ;
                   =1  3507     ;*****************************************************************
                   =1  3508     CHECK_FROM:
EF74 11A0          =1  3509             CALL    IGETOKE
EF76 B4090B        =1  3510             CJNE    A,#FROM_TOKE,NOTFRM
EF79 12E74F        =1  3511             CALL    IGETNUM
EF7C AA49          =1  3512             MOV     PARAM1,VALHGH
EF7E AB4A          =1  3513             MOV     PARAM2,VALLOW
EF80 F167          =1  3514             CALL    WRITE_PC
EF82 11A0          =1  3515             CALL    IGETOKE
EF84 22            =1  3516     NOTFRM: RET
                   =1  3517 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1  3518     ;*****************************************************************
                =1  3519     ;
                =1  3520     ;    NAME: BREAK_VECTOR
                =1  3521     ;
                =1  3522     ;    ABSTRACT: This routine writes location 03 as a break
                =1  3523     ;       vector, and verifies that it was able to write.  This vector
                =1  3524     ;       does a long call to a service routine for all level zero
                =1  3525     ;       interrupts.  Level zero interrupts include:
                =1  3526     ;               UPI interrupts (keyboard closures, USART buffer
                =1  3527     ;                    empty or full, cassette characters rec'd)
                =1  3528     ;               Hardware breakpoints (PROG, DATA, GUARDED ACCESS,
                =1  3529     ;                    SINGLESTEP)
                =1  3530     ;
                =1  3531     ;    INPUTS: None
                =1  3532     ;
                =1  3533     ;    OUTPUTS: Code memory locations 3, 4 and 5
                =1  3534     ;
                =1  3535     ;    VARIABLES MODIFIED: DPTR, A, ERRNUM
                =1  3536     ;
                =1  3537     ;    ERROR EXITS: 17H (NO RAM AT LOCATION 3)
                =1  3538     ;
                =1  3539     ;    SUBROUTINES ACCESSED DIRECTLY: IERROR
                =1  3540     ;
                =1  3541     ;
                =1  3542     ;*****************************************************************
                =1  3543     BREAK_VECTOR:
EF85 900003     =1  3544             MOV     DPTR,#0003H             ;Point to INTO vector address again
EF88 7402       =1  3545             MOV     A,#02H                  ;Store a "LCALL" instruction
EF8A F0         =1  3546             MOVX    @DPTR,A
EF8B 74E0       =1  3547             MOV     A,#HIGH(BREAK)          ;Store the high byte of address for "break"
EF8D A3         =1  3548             INC     DPTR
EF8E F0         =1  3549             MOVX    @DPTR,A
EF8F A3         =1  3550             INC     DPTR
EF90 7403       =1  3551             MOV     A,#LOW(BREAK)           ;Store low byte of "break" address
EF92 F0         =1  3552             MOVX    @DPTR,A
EF93 E4         =1  3553             CLR     A
EF94 93         =1  3554             MOVC    A,@A+DPTR               ;Verify that the write did go into RAM
EF95 B40301     =1  3555             CJNE    A,#LOW(BREAK),B_V_ERR   ;if not the same, go to error
EF98 22         =1  3556             RET
                =1  3557     ;*****************************************************************
                =1  3558     B_V_ERR:
EF99 754317     =1  3559             MOV     ERRNUM,#17H             ;No RAM at location 3
EF9C 02E3CA     =1  3560             JMP     IERROR
                =1  3561 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                    =1  3562     ;*****************************************************************
                    =1  3563     ;
                    =1  3564     ;    NAME: STEP_CMD
                    =1  3565     ;
                    =1  3566     ;    ABSTRACT: STEP executes one or more instructions at a user
                    =1  3567     ;       selectable rate, breaking after each instruction.
                    =1  3568     ;       The monitor displays the contents of the PC, ACC,
                    =1  3569     ;       DPTR, SP and, optionally, a specified bit or byte.
                    =1  3570     ;
                    =1  3571     ;    INPUTS: None
                    =1  3572     ;
                    =1  3573     ;    OUTPUTS: BREAL_STATUS
                    =1  3574     ;
                    =1  3575     ;    VARIABLES MODIFIED: A, TOKSAV, DPTR, ERRNUM, PARAM1, BREAK_STATUS
                    =1  3576     ;
                    =1  3577     ;    ERROR EXITS: 03H (NUMBER EXPECTED)
                    =1  3578     ;                 09H (DECIMAL NUMBER EXPECTED)
                    =1  3579     ;
                    =1  3580     ;    SUROUTINES ACCESSED DIRECTLY: CHECK_FROM, IGETOKE, IGETEOL,
                    =1  3581     ;       BREAK_VECTOR, UPI_CMD, UPI_OUT, UNBREAK, IEOL_CHECK, IERROR
                    =1  3582     ;
                    =1  3583     ;*****************************************************************
                    =1  3584     STEP_CMD:
EF9F F174           =1  3585             CALL    CHECK_FROM
EFA1 90B0F2         =1  3586             MOV     DPTR,#(RAMOFF+SAVE_SEL)
EFA4 E4             =1  3587             CLR     A
EFA5 F0             =1  3588             MOVX    @DPTR,A                 ;Clear SAVE_SEL to avoid unwanted display..
EFA6 E548           =1  3589             MOV     A,TOKSTR
EFA8 B4025D         =1  3590             CJNE    A,#COMMA_TOKE,STPEOL
EFAB 11A0           =1  3591             CALL    IGETOKE
EFAD 54F8           =1  3592             ANL     A,#0F8H                 ;Strip out the lower 3 bits
EFAF B4801D         =1  3593             CJNE    A,#80H,DCLAUSE          ;and skip to process the delay clause if
                    =1  3594                                             ;not a display memory token.
EFB2 85485B         =1  3595             MOV     TOKSAV,TOKSTR           ;Else proceed with display clause.
EFB5 12E74F         =1  3596             CALL    IGETNUM                 ;Save the address to be displayed in external
                    =1  3597                                             ;RAM.
EFB8 90B0F3         =1  3598             MOV     DPTR,#(RAMOFF+ADDR_SAVE_HIGH)
EFBB E549           =1  3599             MOV     A,VALHGH
EFBD F0             =1  3600             MOVX    @DPTR,A
EFBE A3             =1  3601             INC     DPTR
EFBF E54A           =1  3602             MOV     A,VALLOW
EFC1 F0             =1  3603             MOVX    @DPTR,A
EFC2 7582F2         =1  3604             MOV     DPL,#SAVE_SEL
EFC5 E55B           =1  3605             MOV     A,TOKSAV
EFC7 F0             =1  3606             MOVX    @DPTR,A                 ;Save token to be displayed after STEP
EFC8 11A0           =1  3607             CALL    IGETOKE
EFCA B4023B         =1  3608             CJNE    A,#COMMA_TOKE,STPEOL
EFCD 11A0           =1  3609             CALL    IGETOKE
EFCF E548           =1  3610     DCLAUSE:MOV     A,TOKSTR
EFD1 754303         =1  3611             MOV     ERRNUM,#03H             ;Number expected
EFD4 B40138         =1  3612             CJNE    A,#NUMBER_TOKE,EXERRO
EFD7 7409           =1  3613             MOV     A,#9
EFD9 B54A00         =1  3614             CJNE    A,VALLOW,LAB18
EFDC 754309         =1  3615     LAB18:  MOV     ERRNUM,#09H             ;Decimal number expected
EFDF 402E           =1  3616             JC      EXERRO                  ;Error unless number is less than 9.
```

```
LOC  OBJ            LINE     SOURCE

EFE1 E549       =1  3617             MOV     A,VALHGH
EFE3 702A       =1  3618             JNZ     EXERRO                  ;Upper bits must be zero also.
EFE5 90B0F5     =1  3619             MOV     DPTR,#(RAMOFF+DELAY)
EFE8 E54A       =1  3620             MOV     A,VALLOW
EFEA F0         =1  3621             MOVX    @DPTR,A
EFEB 12E759     =1  3622             CALL    IGETEOL                 ;Check that next entry is CR
                =1  3623     STPLOP:
EFEE 74FF       =1  3624             MOV     A,#MULTISTEP
                =1  3625     STEP51:
EFF0 90B0FB     =1  3626             MOV     DPTR,#(RAMOFF+BREAK_STATUS)
EFF3 F0         =1  3627             MOVX    @DPTR,A
EFF4 F185       =1  3628             CALL    BREAK_VECTOR
EFF6 7A03       =1  3629             MOV     PARAM1,#GR_PORT
EFF8 12E60B     =1  3630             CALL    UPI_CMD
EFFB 7A08       =1  3631             MOV     PARAM1,#CLR_BRK_LATCHES ;Clear all break latches
EFFD 12E61E     =1  3632             CALL    UPI_OUT
F000 7A01       =1  3633             MOV     PARAM1,#SINGLE_BREAK
F002 12E61E     =1  3634             CALL    UPI_OUT                 ;Send it to the UPI data channel
F005 02EEAC     =1  3635             JMP     UNBREAK
F008 12E5A1     =1  3636     STPEOL: CALL    IEOL_CHECK
F00B 74FE       =1  3637             MOV     A,#SINGLESTEP
F00D 80E1       =1  3638             JMP     STEP51
F00F 02E3CA     =1  3639     EXERRO: JMP     IERROR
                =1  3640 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1  3641     ;*****************************************************************
                =1  3642     ;
                =1  3643     ;    NAME: STEP51_RETURN
                =1  3644     ;
                =1  3645     ;    ABSTRACT: After the branch to UNBREAK in STEP_CMD, the user
                =1  3646     ;       execution has begun.  Exit from execution with the STEP_FLAG
                =1  3647     ;       set will result in a branch to STEP51_RETURN.
                =1  3648     ;
                =1  3649     ;    INPUTS: SAVE_SEL, BREAK_STATUS, DELAY, USER SP, ACC, DPTR,
                =1  3650     ;       ADDR_SAVE_HIGH, ADDR_SAVE_LOW
                =1  3651     ;
                =1  3652     ;    OUTPUTS: None
                =1  3653     ;
                =1  3654     ;    VARIABLES MODIFIED: PARAM1, PARAM2, ERRNUM, CAUSE_IMAGE, DPTR,
                =1  3655     ;
                =1  3656     ;    ERROR EXITS: 16H (EXECUTION ACROSS LOCATION 3)
                =1  3657     ;
                =1  3658     ;    SUBROUTINES ACCESSED DIRECTLY: INEWLINE, READ_PC, ICO, ILSTWRD,
                =1  3659     ;       SPACCO, ILSTBYT, IFETCH, ITIME, ICSTS, UPI_IN, ICI,
                =1  3660     ;       IWAIT_FOR_USER, IERROR
                =1  3661     ;
                =1  3662     ;
                =1  3663     ;*****************************************************************
                =1  3664     STEP51_RETURN:
F012 12E6FD     =1  3665             CALL    INEWLINE                ;Output a CR-LF.
F015 12EF58     =1  3666             CALL    READ_PC                 ;Output the contents of the user PC to the
F018 AAF0       =1  3667             MOV     PARAM1,B                ;console.
F01A FB         =1  3668             MOV     PARAM2,A
F01B BAE00C     =1  3669             CJNE    PARAM1,#0E0H,NOT_STEP_THREE
F01E BB0309     =1  3670             CJNE    PARAM2,#3,NOT_STEP_THREE
F021 754316     =1  3671             MOV     ERRNUM,#16H             ;Adr 3 executed
F024 756004     =1  3672             MOV     CAUSE_IMAGE,#4          ;Cause is guarded access to loc 3
F027 02E3CA     =1  3673             JMP     IERROR
                =1  3674     NOT_STEP_THREE:
F02A 7A50       =1  3675             MOV     PARAM1,#'P'             ;Output PC label
F02C 12E5CE     =1  3676             CALL    ICO
F02F AAF0       =1  3677             MOV     PARAM1,B                ;Restore PC value to register for display.
F031 12E7DA     =1  3678             CALL    ILSTWRD                 ;Output address
F034 12E5CC     =1  3679             CALL    SPACCO                  ;Output space
F037 7A41       =1  3680             MOV     PARAM1,#'A'             ;Output user accumulator label
F039 12E5CE     =1  3681             CALL    ICO
F03C 90B0E0     =1  3682             MOV     DPTR,#(RAMOFF+ACC)
F03F E0         =1  3683             MOVX    A,@DPTR
F040 FA         =1  3684             MOV     PARAM1,A                ;Call ILSTBYT(user ACC).
F041 12E7DF     =1  3685             CALL    ILSTBYT
F044 12E5CC     =1  3686             CALL    SPACCO
F047 7A44       =1  3687             MOV     PARAM1,#'D'
F049 12E5CE     =1  3688             CALL    ICO                     ;Output DPTR label
F04C 90B082     =1  3689             MOV     DPTR,#(RAMOFF+DPL)
F04F E0         =1  3690             MOVX    A,@DPTR                 ;Displays the low and high byte of DPTR
F050 FB         =1  3691             MOV     PARAM2,A
F051 A3         =1  3692             INC     DPTR
F052 E0         =1  3693             MOVX    A,@DPTR
F053 FA         =1  3694             MOV     PARAM1,A
F054 12E7DA     =1  3695             CALL    ILSTWRD
```

```
LOC  OBJ            LINE     SOURCE

F057 12E5CC     =1  3696             CALL    SPACCO
F05A 7A53       =1  3697             MOV     PARAM1,#'S'             ;Output the SP label
F05C 12E5CE     =1  3698             CALL    ICO
F05F 90B081     =1  3699             MOV     DPTR,#(RAMOFF+SP)
F062 E0         =1  3700             MOVX    A,@DPTR
F063 FA         =1  3701             MOV     PARAM1,A
F064 12E7DF     =1  3702             CALL    ILSTBYT                 ;Output the value of SP
F067 90B0F2     =1  3703             MOV     DPTR,#(RAMOFF+SAVE_SEL)
F06A E0         =1  3704             MOVX    A,@DPTR                 ;Get the select code saved in memory.
F06B F55B       =1  3705             MOV     TOKSAV,A
F06D 6022       =1  3706             JZ      STEP51_EXIT             ;Exit if no optional display.
F06F 12E5CC     =1  3707             CALL    SPACCO                  ;Output space
F072 7A28       =1  3708             MOV     PARAM1,#'('
F074 12E5CE     =1  3709             CALL    ICO                     ;Output left parentheses
F077 E55B       =1  3710             MOV     A,TOKSAV                ;Move saved token into ACC
F079 5407       =1  3711             ANL     A,#07H                  ;Mask lower 3 bits
F07B F546       =1  3712             MOV     SELECT,A                ;Move lower 3 bits into selector for FETCH
F07D A3         =1  3713             INC     DPTR                    ;Fetch the saved address.
F07E E0         =1  3714             MOVX    A,@DPTR
F07F F544       =1  3715             MOV     PNTHGH,A
F081 A3         =1  3716             INC     DPTR
F082 E0         =1  3717             MOVX    A,@DPTR
F083 F545       =1  3718             MOV     PNTLOW,A                ;Fetch the memory byte the user wants
                =1  3719                                             ;displayed.
F085 12E651     =1  3720             CALL    IFETCH
F088 FA         =1  3721             MOV     PARAM1,A                ;And display it.
F089 12E7DF     =1  3722             CALL    ILSTBYT
F08C 7A29       =1  3723             MOV     PARAM1,#')'             ;Output right parentheses
F08E 12E5CE     =1  3724             CALL    ICO
                =1  3725     STEP51_EXIT:
F091 90B0FB     =1  3726             MOV     DPTR,#(RAMOFF+BREAK_STATUS)
F094 E0         =1  3727             MOVX    A,@DPTR
F095 B4FF2F     =1  3728             CJNE    A,#MULTISTEP,SSRET
F098 90B0F5     =1  3729             MOV     DPTR,#(RAMOFF+DELAY)
F09B E0         =1  3730             MOVX    A,@DPTR                 ;Execute multiple single steps
F09C F55C       =1  3731             MOV     DLYCNT,A
F09E E55C       =1  3732     STPDLY: MOV     A,DLYCNT
F0A0 600B       =1  3733      JZ     DLY_THRU
F0A2 155C       =1  3734             DEC     DLYCNT
F0A4 7A13       =1  3735             MOV     PARAM1,#13H
F0A6 7B88       =1  3736             MOV     PARAM2,#88H
F0A8 12EA14     =1  3737             CALL    ITIME                   ;Delay for about 1/2 second per DLYCNT
F0AB 80F1       =1  3738             JMP     STPDLY                  ;Loop until delay count = 0
                =1  3739     DLY_THRU:
F0AD 7A00       =1  3740             MOV     PARAM1,#00H
F0AF 7BA5       =1  3741             MOV     PARAM2,#0A5H
F0B1 12EA14     =1  3742             CALL    ITIME                   ;Delays 16ms
F0B4 12E5E8     =1  3743             CALL    ICSTS
F0B7 4003       =1  3744             JC      STEP_STOP               ;No carry means no input pending
                =1  3745     STPLOP_REACH:
F0B9 02EFEE     =1  3746             JMP     STPLOP
                =1  3747     STEP_STOP:
F0BC 12E632     =1  3748             CALL    UPI_IN
F0BF B41BF7     =1  3749             CJNE    A,#ESC,STPLOP_REACH
F0C2 12E5D1     =1  3750             CALL    ICI                     ;First esc stops step,2nd will exit.
```

```
LOC  OBJ            LINE     SOURCE

F0C5 80F2       =1  3751             JMP     STPLOP_REACH           ;Any key after 1st esc resumes step
                =1  3752     ;*********************************************************************
F0C7 12E396     =1  3753     SSRET:  CALL    IWAIT_FOR_USER
F0CA 02E2C9     =1  3754             JMP     START
F0CD 02E3CA     =1  3755     EXERR1: JMP     IERROR
                =1  3756
                =1  3757 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1  3758     ;*****************************************************************
                =1  3759     ;
                =1  3760     ;    NAME: GO_CMD
                =1  3761     ;
                =1  3762     ;    ABSTRACT: This routine sets up conditions for entering user execution.
                =1  3763     ;       It looks for partition information and breakpoints and saves
                =1  3764     ;       an image of break enable hardware in software.
                =1  3765     ;
                =1  3766     ;    INPUTS: GR
                =1  3767     ;
                =1  3768     ;    OUTPUTS: GR, BREAK_STATUS
                =1  3769     ;
                =1  3770     ;    VARIABLES MODIFIED: A, ERRNUM, DPTR, PARAM1, PARAM2, GR
                =1  3771     ;
                =1  3772     ;    ERROR EXITS: OBH (BREAK ENABLE SYNTAX)
                =1  3773     ;
                =1  3774     ;    SUBROUTINES ACCESSED DIRECTLY: CHECK_FROM, IGETEOL, IGETOKE,
                =1  3775     ;       IEOL_CHECK, BREAK_VECTOR, UPI_CMD, UPI_OUT, UNBREAK, IPRINT_STRING,
                =1  3776     ;       READ_PC, ILSTWRD, IWAIT_FOR_USER
                =1  3777     ;
                =1  3778     ;*****************************************************************
                =1  3779     GO_CMD:
FOD0 12EF74     =1  3780             CALL    CHECK_FROM
FOD3 6407       =1  3781             XRL     A,#EOL_TOKE
FOD5 6053       =1  3782             JZ      RUN_USER                ;If have the end of line token go to user
                =1  3783                                             ;emulation.
                =1  3784                                             ;If not then find out what kind of emulation
                =1  3785                                             ;is required.
                =1  3786
FOD7 E548       =1  3787             MOV     A,TOKSTR                ; First restore the token.
FOD9 B4080C     =1  3788             CJNE    A,#FOREVER_TOKE,NOTFOR
                =1  3789                                             ;See if token is FOREVER token
FODC 12E759     =1  3790             CALL    IGETEOL                 ;Wait for CR after FOREVER
FODF 90B0F6     =1  3791             MOV     DPTR,#(RAMOFF+GR)       ;Copy break enable image into hrdwr
FOE2 7409       =1  3792             MOV     A,#NO_BREAK
FOE4 F0         =1  3793             MOVX    @DPTR,A
FOE5 02F12A     =1  3794             JMP     RUN_USER                ;Jump to GO loop
FOE8 75430B     =1  3795     NOTFOR: MOV     ERRNUM,#OBH             ;BRK enable syntax
FOEB B40CDF     =1  3796             CJNE    A,#TILL_TOKE,EXERR1     ;Jump to error routine if not TIL token
FOEE 12E8A0     =1  3797             CALL    IGETOKE
FOF1 B4D30C     =1  3798             CJNE    A,#DATA_TOKE,NOTDAT     ;See if next input was data break
FOF4 12E759     =1  3799             CALL    IGETEOL                 ;Make sure next input is CR
FOF7 90B0F6     =1  3800             MOV     DPTR,#(RAMOFF+GR)       ;Copy break enable image into hrdwr
FOFA 740D       =1  3801             MOV     A,#DATA_BREAK
FOFC F0         =1  3802             MOVX    @DPTR,A
FOFD 02F12A     =1  3803             JMP     RUN_USER                ;Jump to GO loop
F100 75430B     =1  3804     NOTDAT: MOV     ERRNUM,#OBH             ;BRK enable syntax
F103 B4D5C7     =1  3805             CJNE    A,#PROGRAM_TOKE,EXERR1  ;If program break not entered by now,
                =1  3806                                             ;Then error
F106 12E8A0     =1  3807             CALL    IGETOKE
F109 B40B15     =1  3808             CJNE    A,#OR_TOKE,PGMBRK       ;See if OR was entered in break sequence
F10C 12E8A0     =1  3809             CALL    IGETOKE
F10F 75430B     =1  3810             MOV     ERRNUM,#OBH             ;BRK enable syntax
F112 B4D3B8     =1  3811             CJNE    A,#DATA_TOKE,EXERR1     ;Make sure data token was entered next
F115 12E759     =1  3812             CALL    IGETEOL                 ;Make sure CR was entered last
```

```
LOC  OBJ            LINE     SOURCE

F118 90B0F6     =1  3813             MOV     DPTR,#(RAMOFF+GR)       ;Copy break enable image into sftwr
F11B 740F       =1  3814             MOV     A,#(DATA_BREAK OR PROGRAM_BREAK)
F11D F0         =1  3815             MOVX    @DPTR,A
F11E 02F12A     =1  3816             JMP     RUN_USER
F121 12E5A1     =1  3817     PGMBRK: CALL    IEOL_CHECK
F124 90B0F6     =1  3818             MOV     DPTR,#(RAMOFF+GR)       ;Copy break enable image into sftwr
F127 740B       =1  3819             MOV     A,#PROGRAM_BREAK
F129 F0         =1  3820             MOVX    @DPTR,A
                =1  3821     RUN_USER:
F12A 90B0FB     =1  3822             MOV     DPTR,#(RAMOFF+BREAK_STATUS)
F12D 74FB       =1  3823             MOV     A,#NOT_STEP
F12F F0         =1  3824             MOVX    @DPTR,A                 ;Clear the step flag to show a 'run' condition
F130 12EF85     =1  3825             CALL    BREAK_VECTOR
F133 7AF1       =1  3826             MOV     PARAM1,#HIGH(XEQT_MSG)
F135 7B65       =1  3827             MOV     PARAM2,#LOW(XEQT_MSG)
F137 12E9CD     =1  3828             CALL    IPRINT_STRING
F13A 7A03       =1  3829             MOV     PARAM1,#GR_PORT
F13C 12E60B     =1  3830             CALL    UPI_CMD
F13F 7A08       =1  3831             MOV     PARAM1,#CLR_BRK_LATCHES ;Clear all break latches
F141 12E61E     =1  3832             CALL    UPI_OUT
F144 90B0F6     =1  3833             MOV     DPTR,#(RAMOFF+GR)       ;Copy break enable image into hrdwr
F147 E0         =1  3834             MOVX    A,@DPTR
F148 FA         =1  3835             MOV     PARAM1,A
F149 12E61E     =1  3836             CALL    UPI_OUT                 ;Send it to the UPI data channel
F14C 02EEAC     =1  3837             JMP     UNBREAK
                =1  3838     ;*********************************************************************
                =1  3839     RUN_USER_RETURN:
F14F 7AF1       =1  3840             MOV     PARAM1,#HIGH(BREAK_MSG)
F151 7B77       =1  3841             MOV     PARAM2,#LOW(BREAK_MSG)
F153 12E9CD     =1  3842             CALL    IPRINT_STRING
F156 12EF58     =1  3843             CALL    READ_PC
F159 AAF0       =1  3844             MOV     PARAM1,B                ;Display the user PC
F15B FB         =1  3845             MOV     PARAM2,A
F15C 12E7DA     =1  3846             CALL    ILSTWRD
F15F 12E396     =1  3847             CALL    IWAIT_FOR_USER          ;And goto the monitor.
F162 02E2C9     =1  3848             JMP     START
                =1  3849     ;*********************************************************************
                =1  3850     XEQT_MSG:
F165 11         =1  3851             DB      17,CR,LF,('EXECUTION BEGUN')
F166 0D
F167 0A
F168 45584543
F16C 5554494F
F170 4E204245
F174 47554E
                =1  3852     BREAK_MSG:
F177 16         =1  3853             DB      22,CR,LF,('EXECUTION HALTED PC=')
F178 0D
F179 0A
F17A 45584543
F17E 5554494F
F182 4E204841
F186 4C544544
F18A 2050433D
                3854 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                    3855 +1  $INCLUDE(:F1:MONFUN.INC)
                  =1  3856     ;*****************************************************************
                  =1  3857     ;
                  =1  3858     ;    NAME: LIST_CMD
                  =1  3859     ;
                  =1  3860     ;    ABSTRACT: This routine gets the 'keyword =' message and sets
                  =1  3861     ;       up the LSTFLG to display tokens to the console and an auxilary
                  =1  3862     ;       terminal.  Anytime display is called for. It will also terminate
                  =1  3863     ;       any ISIS files with a control Z. List is on when LSTFLG = 1.
                  =1  3864     ;
                  =1  3865     ;    INPUTS: LSTFLG
                  =1  3866     ;
                  =1  3867     ;    OUTPUTS: LSTFLG
                  =1  3868     ;
                  =1  3869     ;    VARIABLES MODIFIED: LSTFLG, PARAM1, ERRNUM
                  =1  3870     ;
                  =1  3871     ;    ERROR EXITS: 08H (RESET OR ON REQUIRED)
                  =1  3872     ;
                  =1  3873     ;    SUBROUTINES ACCESSED DIRECTLY: ISIT_DISPLAY, IGETOKE,
                  =1  3874     ;       IDISPLAY_TOKEN, ICO, UPI_CMD, INEWLINE, IWAIT_FOR_USER
                  =1  3875     ;
                  =1  3876     ;
                  =1  3877     ;*****************************************************************
                  =1  3878     LIST_CMD:
F18E 12E76A       =1  3879             CALL    ISIT_DISPLAY                ;Sets up 'keyword =' msg
F191 401E         =1  3880             JC      DISPLAY_LIST                ;C=1 if display only
F193 12E8A0       =1  3881             CALL    IGETOKE
F196 B40F03       =1  3882             CJNE    A,#ON_TOKE,LIST_2           ;List turned on, no display
F199 D201         =1  3883             SETB    LSTFLG
F19B 22           =1  3884             RET
F19C 754308       =1  3885     LIST_2: MOV     ERRNUM,#08H                 ;Reset or on required
F19F B40E71       =1  3886             CJNE    A,#RESET_TOKE,STATE_ERR     ;List turned off, no display
F1A2 12E6FD       =1  3887             CALL    INEWLINE                    ;Send a CR,LF for MDS
F1A5 7A01         =1  3888             MOV     PARAM1,#USART_MODE
F1A7 12E60B       =1  3889             CALL    UPI_CMD
F1AA C201         =1  3890             CLR     LSTFLG
F1AC 7A1A         =1  3891             MOV     PARAM1,#1AH
F1AE 02E5CE       =1  3892             JMP     ICO                         ;Send cntrl-Z to close MDS file
                  =1  3893     DISPLAY_LIST:
F1B1 7A0F         =1  3894             MOV     PARAM1,#ON_TOKE             ;Display 'on' set up
F1B3 200102       =1  3895             JB      LSTFLG,LIST_1
F1B6 7A0E         =1  3896             MOV     PARAM1,#RESET_TOKE          ;Display 'reset' set up
F1B8 12E9E0       =1  3897     LIST_1: CALL    IDISPLAY_TOKEN
F1BB 02E396       =1  3898             JMP     IWAIT_FOR_USER
                  =1  3899 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1  3900     ;*********************************************************************
                =1  3901     ;
                =1  3902     ;    NAME: BAUD_CMD/ SET_BAUD
                =1  3903     ;
                =1  3904     ;    ABSTRACT: This routine will allow the user to display the
                =1  3905     ;       baud rate or change the baud rate to any legal value between
                =1  3906     ;       110 and 9600.  Default on power up is 2400.
                =1  3907     ;
                =1  3908     ;    INPUTS: BAUD_HIGH, BAUD_LOW
                =1  3909     ;
                =1  3910     ;    OUTPUTS: BAUD_HIGH, BAUD_LOW, BAUDKEY
                =1  3911     ;
                =1  3912     ;    VARIABLES MODIFIED: DPTR, ERRNUM, A, B, BAUD_HIGH, BAUD_LOW, BAUDKEY
                =1  3913     ;
                =1  3914     ;    ERROR EXITS: OAH (ILLEGAL BAUD VALUE)
                =1  3915     ;
                =1  3916     ;    SUBROUTINES ACCESSED DIRECTLY: ISIT_DISPLAY, IGETNUM, IERROR,
                =1  3917     ;       ILSTWRD, IWAIT_FOR_USER
                =1  3918     ;
                =1  3919     ;
                =1  3920     ;*********************************************************************
                =1  3921     BAUD_CMD:
F1BE 12E76A     =1  3922             CALL    ISIT_DISPLAY
F1C1 4068       =1  3923             JC      BAUD_DISPLAY
F1C3 12E74F     =1  3924             CALL    IGETNUM
F1C6 90F216     =1  3925             MOV     DPTR,#BAUD_RATE         ;Check table for a valid baud rate request.
F1C9 7800       =1  3926             MOV     POINTO,#00H
                =1  3927     BS_LOOP:
F1CB E8         =1  3928             MOV     A,POINTO
F1CC 93         =1  3929             MOVC    A,@A+DPTR
F1CD B5493F     =1  3930             CJNE    A,VALHGH,BS_2
F1D0 E54A       =1  3931             MOV     A,VALLOW
F1D2 B80038     =1  3932             CJNE    POINTO,#00H,BM_1
                =1  3933                                             ;If POINTO=0, the lower 2 digits better be
                =1  3934                                             ;10 because the baud rate is 110.
F1D5 75430A     =1  3935             MOV     ERRNUM,#0AH             ;Illegal baud value
F1D8 B41038     =1  3936             CJNE    A,#10H,STATE_ERR
                =1  3937     PRE_SET_BAUD:
F1DB 90B0F7     =1  3938             MOV     DPTR,#(RAMOFF+BAUD_HIGH)
F1DE E549       =1  3939             MOV     A,VALHGH
F1E0 F0         =1  3940             MOVX    @DPTR,A
F1E1 A3         =1  3941             INC     DPTR
F1E2 E54A       =1  3942             MOV     A,VALLOW
F1E4 F0         =1  3943             MOVX    @DPTR,A
F1E5 7582FC     =1  3944             MOV     DPL,#BAUDKEY
F1E8 E8         =1  3945             MOV     A,POINTO
F1E9 F0         =1  3946             MOVX    @DPTR,A
                =1  3947     ;*********************************************************************
                =1  3948     SET_BAUD:
F1EA 90B0FC     =1  3949             MOV     DPTR,#(RAMOFF+BAUDKEY)
F1ED E0         =1  3950             MOVX    A,@DPTR
F1EE 23         =1  3951             RL      A
F1EF F5F0       =1  3952             MOV     B,A
F1F1 90F21D     =1  3953             MOV     DPTR,#TIMER_PRESET
F1F4 93         =1  3954             MOVC    A,@A+DPTR
```

```
LOC  OBJ            LINE     SOURCE

F1F5 C5F0       =1  3955             XCH     A,B
F1F7 A3         =1  3956             INC     DPTR
F1F8 93         =1  3957             MOVC    A,@A+DPTR
F1F9 C5F0       =1  3958             XCH     A,B                     ;Store the timer preset value.
F1FB 90B805     =1  3959             MOV     DPTR,#(RAMIO+TIMER_HIGH)
F1FE 4440       =1  3960             ORL     A,#CONTINUOUS_MODE
F200 F0         =1  3961             MOVX    @DPTR,A
F201 1582       =1  3962             DEC     DPL
F203 E5F0       =1  3963             MOV     A,B
F205 F0         =1  3964             MOVX    @DPTR,A
F206 90B800     =1  3965             MOV     DPTR,#RAMIO             ;Start - load timer
F209 74C0       =1  3966             MOV     A,#START_16_TIMER
F20B F0         =1  3967             MOVX    @DPTR,A
F20C 22         =1  3968             RET
F20D 60CC       =1  3969     BM_1:   JZ      PRE_SET_BAUD            ;Else the lower 2 digits better be 0
                =1  3970                                             ;because all the other rates end in 0.
F20F 08         =1  3971     BS_2:   INC     POINTO
F210 B496B8     =1  3972             CJNE    A,#HIGH(9600H),BS_LOOP
                =1  3973     STATE_ERR:
F213 02E3CA     =1  3974             JMP     IERROR
                =1  3975     BAUD_RATE:
F216 01         =1  3976             DB      HIGH(110H)
F217 03         =1  3977             DB      HIGH(300H)
F218 06         =1  3978             DB      HIGH(600H)
F219 12         =1  3979             DB      HIGH(1200H)
F21A 24         =1  3980             DB      HIGH(2400H)
F21B 48         =1  3981             DB      HIGH(4800H)
F21C 96         =1  3982             DB      HIGH(9600H)
                =1  3983     TIMER_PRESET:
F21D 0470       =1  3984             DW      1136
F21F 01A1       =1  3985             DW      0417
F221 00D0       =1  3986             DW      0208
F223 0068       =1  3987             DW      0104
F225 0034       =1  3988             DW      0052
F227 001A       =1  3989             DW      0026
F229 000D       =1  3990             DW      0013
                =1  3991     ;*****************************************************************
                =1  3992     BAUD_DISPLAY:
F22B 90B0F7     =1  3993             MOV     DPTR,#(RAMOFF+BAUD_HIGH)
F22E E0         =1  3994             MOVX    A,@DPTR
F22F FA         =1  3995             MOV     PARAM1,A
F230 A3         =1  3996             INC     DPTR
F231 E0         =1  3997             MOVX    A,@DPTR
F232 FB         =1  3998             MOV     PARAM2,A
F233 12E7DA     =1  3999             CALL    ILSTWRD
F236 02E396     =1  4000             JMP     IWAIT_FOR_USER
                =1  4001 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                    =1  4002     ;*****************************************************************
                    =1  4003     ;
                    =1  4004     ;    NAME: TOP_CMD
                    =1  4005     ;
                    =1  4006     ;    ABSTRACT: This routine will set the top of memory to a value
                    =1  4007     ;       requested by the user.  It will error for values > 7FFFH.
                    =1  4008     ;       It will also list the current TOP value to the console upon
                    =1  4009     ;       request.
                    =1  4010     ;
                    =1  4011     ;    INPUTS: TOP_STORE
                    =1  4012     ;
                    =1  4013     ;    OUTPUTS: TOP_STORE
                    =1  4014     ;
                    =1  4015     ;    VARIABLES MODIFIED: DPTR, A, B, PARAM1, ERRNUM
                    =1  4016     ;
                    =1  4017     ;    ERROR EXITS: ODH (TOP VALUE > 7FFFH)
                    =1  4018     ;
                    =1  4019     ;    SUBROUTINES ACCESSED DIRECTLY: ISIT_DISPLAY, IGETNUM, ILSTBYT,
                    =1  4020     ;       IWAIT_FOR_USER
                    =1  4021     ;
                    =1  4022     ;
                    =1  4023     ;*****************************************************************
F239 12E76A         =1  4024     TOP_CMD:CALL    ISIT_DISPLAY
F23C 90B0F9         =1  4025             MOV     DPTR,#(RAMOFF+TOP_STORE)
F23F 401A           =1  4026             JC      TOP_DISPLAY
F241 12E74F         =1  4027             CALL    IGETNUM
F244 E549           =1  4028             MOV     A,VALHGH                ;Do not allow top > 32k
F246 75430D         =1  4029             MOV     ERRNUM,#0DH             ;Top value > 7FFFH
F249 20E7C7         =1  4030             JB      ACC.7,STATE_ERR
F24C F5F0           =1  4031             MOV     B,A                     ;Check for the special case of 0000H
                    =1  4032                                             ;otherwise the display should end
F24E 454A           =1  4033             ORL     A,VALLOW                ;with an FFH
F250 6002           =1  4034             JZ      ST_1
F252 05F0           =1  4035             INC     B
                    =1  4036     ST_1:
F254 E5F0           =1  4037             MOV     A,B
F256 90B0F9         =1  4038             MOV     DPTR,#(RAMOFF+TOP_STORE)
F259 F0             =1  4039             MOVX    @DPTR,A
F25A 22             =1  4040             RET
                    =1  4041     ;*****************************************************************
                    =1  4042     TOP_DISPLAY:
F25B E0             =1  4043             MOVX    A,@DPTR                 ;Call listbyte(top).
F25C 6001           =1  4044             JZ      TOP_LIST_2
F25E 14             =1  4045             DEC     A
                    =1  4046     TOP_LIST_2:
F25F FA             =1  4047             MOV     PARAM1,A
F260 12E7DF         =1  4048             CALL    ILSTBYT
F263 90B0F9         =1  4049             MOV     DPTR,#(RAMOFF+TOP_STORE)
F266 E0             =1  4050             MOVX    A,@DPTR
F267 6008           =1  4051             JZ      TOP_LIST_0
F269 7AFF           =1  4052             MOV     PARAM1,#0FFH
F26B 12E7DF         =1  4053             CALL    ILSTBYT
F26E 02F276         =1  4054             JMP     TOP_LIST_1
                    =1  4055     TOP_LIST_0:
F271 7A00           =1  4056             MOV     PARAM1,#00H
```

```
LOC  OBJ            LINE     SOURCE

F273 12E7DF     =1  4057             CALL    ILSTBYT
                =1  4058     TOP_LIST_1:
F276 02E396     =1  4059             JMP     IWAIT_FOR_USER
                =1  4060 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1  4061     ;*****************************************************************
                =1  4062     ;
                =1  4063     ;    NAME: CAUSE_CMD
                =1  4064     ;
                =1  4065     ;    ABSTRACT: This routine will display the reason detected
                =1  4066     ;       for a break execution.  It is a display-only function.
                =1  4067     ;       The cause is determined and stored during BREAK.
                =1  4068     ;
                =1  4069     ;    INPUTS: CAUSE_IMAGE
                =1  4070     ;
                =1  4071     ;    OUTPUTS: None
                =1  4072     ;
                =1  4073     ;    VARIABLES MODIFIED: A, DPTR, COUNT, PARAM1, PARAM2, ERRNUM
                =1  4074     ;
                =1  4075     ;    ERROR EXITS: OEH (DISPLAY ONLY)
                =1  4076     ;
                =1  4077     ;    SUBROUTINES ACCESSED DIRECTLY: ISIT_DISPLAY, IPRINT_STRING,
                =1  4078     ;       IWAIT_FOR_USER
                =1  4079     ;
                =1  4080     ;
                =1  4081     ;*****************************************************************
                =1  4082     CAUSE_CMD:
F279 12E76A     =1  4083             CALL    ISIT_DISPLAY
F27C 75430E     =1  4084             MOV     ERRNUM,#0EH             ;Display only
F27F 5092       =1  4085             JNC     STATE_ERR
F281 E560       =1  4086             MOV     A,CAUSE_IMAGE
F283 90F29D     =1  4087             MOV     DPTR,#CAUSE_TAB
F286 7F05       =1  4088             MOV     COUNT,#5                ;Output the appropriate message.
                =1  4089     CL_LOOP:
F288 13         =1  4090             RRC     A                       ;Isolate the bit which indicates the
                =1  4091                                             ;cause of the break.
F289 20E004     =1  4092             JB      ACC.0,CL_0
F28C A3         =1  4093             INC     DPTR
F28D A3         =1  4094             INC     DPTR
F28E DFF8       =1  4095             DJNZ    COUNT,CL_LOOP
                =1  4096     CL_0:
F290 E4         =1  4097             CLR     A
F291 93         =1  4098             MOVC    A,@A+DPTR
F292 FA         =1  4099             MOV     PARAM1,A
F293 E4         =1  4100             CLR     A
F294 A3         =1  4101             INC     DPTR
F295 93         =1  4102             MOVC    A,@A+DPTR
F296 FB         =1  4103             MOV     PARAM2,A
F297 12E9CD     =1  4104             CALL    IPRINT_STRING
F29A 02E396     =1  4105             JMP     IWAIT_FOR_USER
                =1  4106     CAUSE_TAB:
F29D F2A9       =1  4107             DW      USER_MSG
F29F F2B4       =1  4108             DW      GUARD_MSG
F2A1 F2C3       =1  4109             DW      PROG_MSG
F2A3 F2D1       =1  4110             DW      DATA_MSG
F2A5 F2DC       =1  4111             DW      SINGLE_STEP_MSG
F2A7 F2E8       =1  4112             DW      NOBRK_MSG
                =1  4113     USER_MSG:
F2A9 0A         =1  4114             DB      10,('USER ABORT')
F2AA 55534552
```

```
LOC  OBJ            LINE     SOURCE

F2AE 2041424F
F2B2 5254
                =1  4115     GUARD_MSG:
F2B4 0E         =1  4116             DB      14,('GUARDED ACCESS')
F2B5 47554152
F2B9 44454420
F2BD 41434345
F2C1 5353
                =1  4117     PROG_MSG:
F2C3 0D         =1  4118             DB      13,('PROGRAM BREAK')
F2C4 50524F47
F2C8 52414D20
F2CC 42524541
F2D0 4B
                =1  4119     DATA_MSG:
F2D1 0A         =1  4120             DB      10,('DATA BREAK')
F2D2 44415441
F2D6 20425245
F2DA 414B
                =1  4121     SINGLE_STEP_MSG:
F2DC 0B         =1  4122             DB      11,('SINGLE STEP')
F2DD 53494E47
F2E1 4C452053
F2E5 544550
                =1  4123     NOBRK_MSG:
F2E8 0B         =1  4124             DB      11,('WHAT BREAK?')
F2E9 57484154
F2ED 20425245
F2F1 414B3F
                =1  4125 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1  4126     ;*****************************************************************
                =1  4127     ;
                =1  4128     ;    NAME: SEND_BYTE
                =1  4129     ;
                =1  4130     ;    ABSTRACT: This routine outputs one byte, in either hex or
                =1  4131     ;       binary depending on the setting of the binary flag, to
                =1  4132     ;       the USART. A new checksum is calculated and returned.
                =1  4133     ;
                =1  4134     ;    INPUTS: CHECKSUM, A
                =1  4135     ;
                =1  4136     ;    OUTPUTS: CHECKSUM
                =1  4137     ;
                =1  4138     ;    VARIABLES MODIFIED: A, PARAM1
                =1  4139     ;
                =1  4140     ;    ERROR EXITS: None
                =1  4141     ;
                =1  4142     ;    SUBROUTINES ACCESSED DIRECTLY: ICO, ILSTBYT
                =1  4143     ;
                =1  4144     ;
                =1  4145     ;*****************************************************************
                =1  4146     SEND_BYTE:
F2F4 CE         =1  4147             XCH     A,CHECKSUM
F2F5 2E         =1  4148             ADD     A,CHECKSUM
F2F6 CE         =1  4149             XCH     A,CHECKSUM
F2F7 FA         =1  4150             MOV     PARAM1,A
F2F8 200503     =1  4151             JB      BINARY_FLG,SEND_BINARY
F2FB 02E7DF     =1  4152             JMP     ILSTBYT
                =1  4153     SEND_BINARY:
F2FE 02E5CE     =1  4154             JMP     ICO
                =1  4155 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1  4156     ;*****************************************************************
                =1  4157     ;
                =1  4158     ;    NAME: HEXBIN
                =1  4159     ;
                =1  4160     ;    ABSTRACT: Reads two characters from the input device and
                =1  4161     ;       converts them to binary.  If the binary flag is set, then
                =1  4162     ;       one binary character is input.  This value is added to the
                =1  4163     ;       checksum byte and also returned to the calling routine.
                =1  4164     ;
                =1  4165     ;    INPUTS: BINARY_FLG, CHECKSUM
                =1  4166     ;
                =1  4167     ;    OUTPUTS: CHECKSUM
                =1  4168     ;
                =1  4169     ;    VARIABLES MODIFIED: PARAM1, A, TEMP
                =1  4170     ;
                =1  4171     ;    ERROR EXITS: None
                =1  4172     ;
                =1  4173     ;    SUBROUTINES ACCESSED DIRECTLY: UPI_IN, IASCII_TO_HEX, ICI
                =1  4174     ;
                =1  4175     ;
                =1  4176     ;*****************************************************************
F301 12E632     =1  4177     HEXBIN: CALL    UPI_IN
F304 20050E     =1  4178             JB      BINARY_FLG,BINARY_LOAD
F307 FA         =1  4179             MOV     PARAM1,A
F308 12EA0B     =1  4180             CALL    IASCII_TO_HEX
F30B C4         =1  4181             SWAP    A                       ;Move the digit to the upper nibble.
F30C FD         =1  4182             MOV     TEMP,A                  ;Then save it in a temporary location.
F30D 12E5D1     =1  4183             CALL    ICI
F310 FA         =1  4184             MOV     PARAM1,A
F311 12EA0B     =1  4185             CALL    IASCII_TO_HEX
F314 4D         =1  4186             ORL     A,TEMP                  ;Then combine with previous digit.
                =1  4187     BINARY_LOAD:
F315 CE         =1  4188             XCH     A,CHECKSUM              ;Before returning the binary value
F316 2E         =1  4189             ADD     A,CHECKSUM              ;include it in checksum calculation.
F317 CE         =1  4190             XCH     A,CHECKSUM
F318 22         =1  4191             RET
                =1  4192 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1  4193     ;*********************************************************************
                =1  4194     ;
                =1  4195     ;    NAME: GET_TYPE
                =1  4196     ;
                =1  4197     ;    ABSTRACT: This routine looks for a colon from the cassette or
                =1  4198     ;       auxilary termimal input, gets the byte count, address and
                =1  4199     ;       file-type information contained in the header and does a checksum.
                =1  4200     ;
                =1  4201     ;    INPUTS: None
                =1  4202     ;
                =1  4203     ;    OUTPUTS: TYPE, PNTLOW, PNTHGH, COUNT, CHECKSUM
                =1  4204     ;
                =1  4205     ;    VARIABLES MODIFIED: A, CHECKSUM, COUNT, PNTHGH, PNTLOW, TYPE
                =1  4206     ;
                =1  4207     ;    ERROR EXITS: None
                =1  4208     ;
                =1  4209     ;    SUBROUTINES ACCESSED DIRECTLY: ICI, HEXBIN
                =1  4210     ;
                =1  4211     ;
                =1  4212     ;*********************************************************************
                =1  4213     GET_TYPE:
F319 12E632     =1  4214             CALL    UPI_IN                  ;Scan for a colon.
F31C 547F       =1  4215             ANL     A,#7FH
F31E B43AF8     =1  4216             CJNE    A,#':',GET_TYPE
F321 E4         =1  4217             CLR     A
F322 FE         =1  4218             MOV     CHECKSUM,A
F323 7101       =1  4219             CALL    HEXBIN                  ;Load the byte count from
F325 FF         =1  4220             MOV     COUNT,A                 ;the next two characters of the record.
F326 7101       =1  4221             CALL    HEXBIN                  ;Load the load address
F328 F544       =1  4222             MOV     PNTHGH,A
F32A 7101       =1  4223             CALL    HEXBIN
F32C F545       =1  4224             MOV     PNTLOW,A
F32E 7101       =1  4225             CALL    HEXBIN                  ;Load the record type.
F330 F565       =1  4226             MOV     TYPE,A
F332 22         =1  4227             RET
                =1  4228 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1  4229     ;*********************************************************************
                =1  4230     ;
                =1  4231     ;    NAME: LOAD_HEX
                =1  4232     ;
                =1  4233     ;    ABSTRACT: Loads audio cassette data files (type 0) until EOF
                =1  4234     ;       is encountered.  Calculates a checksum, passes label (addr), writes
                =1  4235     ;       user PC, converts hex data to binary and returns.
                =1  4236     ;
                =1  4237     ;    INPUTS: None
                =1  4238     ;
                =1  4239     ;    OUTPUTS: Code memory locations addressed in the file being loaded.
                =1  4240     ;
                =1  4241     ;    VARIABLES MODIFIED: A, PARAM1, SELECT, PNTLOW, PNTHGH, ERRNUM
                =1  4242     ;
                =1  4243     ;    ERROR EXITS: None
                =1  4244     ;
                =1  4245     ;    SUBROUTINES ACCESSED DIRECTLY: GET_TYPE, HEXBIN, INIT_IO,
                =1  4246     ;       ISTORE, WRITE_PC, ITIME
                =1  4247     ;
                =1  4248     ;
                =1  4249     ;*********************************************************************
                =1  4250     LOAD_HEX:
F333 7119       =1  4251             CALL    GET_TYPE
F335 7019       =1  4252             JNZ     LH_7                    ;If type is not zero (data record)
                =1  4253                                             ;then quit loading records.
F337 EF         =1  4254     LH_4:   MOV     A,COUNT                 ;Load memory until the count gets
F338 600E       =1  4255             JZ      LH_6                    ;to zero, COUNT=length read from file
F33A 7101       =1  4256             CALL    HEXBIN
F33C FA         =1  4257             MOV     PARAM1,A
F33D 754600     =1  4258             MOV     SELECT,#0
F340 12E658     =1  4259             CALL    ISTORE                  ;Increment the load address.
F343 12E5AA     =1  4260             CALL    INC_PNT                 ;Repeat the load loop until zero.
F346 DFEF       =1  4261             DJNZ    COUNT,LH_4
F348 7101       =1  4262     LH_6:   CALL    HEXBIN                  ;The end of the record has been reached
F34A EE         =1  4263             MOV     A,CHECKSUM              ;so check the checksum field.
F34B 60E6       =1  4264             JZ      LOAD_HEX                ;Recall CHECKSUM from HEXBIN
F34D 02F408     =1  4265     LH_8:   JMP     LH_ERROR
F350 B401FA     =1  4266     LH_7:   CJNE    A,#1,LH_8               ;Look for EOF (type 1)
F353 7101       =1  4267             CALL    HEXBIN
F355 EE         =1  4268             MOV     A,CHECKSUM
F356 70F5       =1  4269             JNZ     LH_8
F358 12E36C     =1  4270             CALL    INIT_IO
F35B AB45       =1  4271             MOV     PARAM2,PNTLOW           ;Write addr (label) to user PC
F35D AA44       =1  4272             MOV     PARAM1,PNTHGH
F35F 12EF67     =1  4273             CALL    WRITE_PC
F362 7A07       =1  4274             MOV     PARAM1,#07H
F364 7B00       =1  4275             MOV     PARAM2,#00H
F366 12EA14     =1  4276             CALL    ITIME                   ;Wait for 2 char times at 110 baud
F369 90A000     =1  4277             MOV     DPTR,#UPI_DATA          ;So no other chars get into the
F36C E0         =1  4278             MOVX    A,@DPTR                 ;Command buffer.  Flush output
F36D 22         =1  4279             RET                             ;buffer flag.
                =1  4280 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1  4281     ;*****************************************************************
                =1  4282     ;
                =1  4283     ;    NAME: STORE_HEX
                =1  4284     ;
                =1  4285     ;    ABSTRACT: This routine writes hex bytes on the cassette or to
                =1  4286     ;       the USART from memory.  It outputs all record marks and header
                =1  4287     ;       information and calculates a checksum.
                =1  4288     ;
                =1  4289     ;    INPUTS: BINARY_FLG, PARTIT_LO_LOW, PARTIT_LO_HIGH, PARTIT_HI_LOW,
                =1  4290     ;       PARTIT_HI_HIGH, Memory contents within the partition bounds.
                =1  4291     ;
                =1  4292     ;    OUTPUTS: None
                =1  4293     ;
                =1  4294     ;    VARIABLES MODIFIED: PARAM1, PARAM1, C, A, COUNT, TEMP, CHECKSUM,
                =1  4295     ;       SELECT, PNTHGH, PNTLOW, PARTIT_LO_LOW, PARTIT_LO_HIGH,
                =1  4296     ;       ERRNUM
                =1  4297     ;
                =1  4298     ;    ERROR EXITS: 14H (FILE READ/WRITE ERROR)
                =1  4299     ;
                =1  4300     ;    SUBROUTINES ACCESSED DIRECTLY: INEWLINE, ITIME, SEND_BYTE,
                =1  4301     ;       IFETCH, READ_PC, UPI_CMD, ICO, IERROR
                =1  4302     ;
                =1  4303     ;
                =1  4304     ;*****************************************************************
                =1  4305     STORE_HEX:
F36E 200511     =1  4306             JB      BINARY_FLG,SH_6
F371 7A01       =1  4307             MOV     PARAM1,#01H             ;Delay 40 milliseconds.
F373 7B90       =1  4308             MOV     PARAM2,#90H
F375 12EA14     =1  4309             CALL    ITIME
F378 12E6FD     =1  4310             CALL    INEWLINE                ;Start sending record.
F37B 7A13       =1  4311             MOV     PARAM1,#13H
F37D 7B88       =1  4312             MOV     PARAM2,#88H             ;Delay 1/2 sec.
F37F 12EA14     =1  4313             CALL    ITIME
F382 7A3A       =1  4314     SH_6:   MOV     PARAM1,#':'             ;Output the record mark.
F384 12E5CE     =1  4315             CALL    ICO
F387 C3         =1  4316             CLR     C                       ;Output hex records while sa<=ea.
F388 E55A       =1  4317             MOV     A,PARTIT_HI_LOW
F38A 9558       =1  4318             SUBB    A,PARTIT_LO_LOW
F38C FF         =1  4319             MOV     COUNT,A                 ;(Save difference for later use).
F38D E559       =1  4320             MOV     A,PARTIT_HI_HIGH
F38F 9557       =1  4321             SUBB    A,PARTIT_LO_HIGH
F391 FD         =1  4322             MOV     TEMP,A                  ;Set count to 16 or the number of bytes
F392 403E       =1  4323             JC      SH_5                    ;left-whichever is less.
F394 ED         =1  4324             MOV     A,TEMP
F395 6002       =1  4325             JZ      SH_1
F397 7F0F       =1  4326             MOV     COUNT,#0FH
F399 EF         =1  4327     SH_1:   MOV     A,COUNT
F39A 54F0       =1  4328             ANL     A,#0F0H
F39C 6002       =1  4329             JZ      SH_2
F39E 7F0F       =1  4330             MOV     COUNT,#0FH
F3A0 0F         =1  4331     SH_2:   INC     COUNT
F3A1 E4         =1  4332             CLR     A
F3A2 FE         =1  4333             MOV     CHECKSUM,A
F3A3 EF         =1  4334             MOV     A,COUNT
F3A4 51F4       =1  4335             CALL    SEND_BYTE
```

```
LOC  OBJ            LINE     SOURCE

F3A6 E557       =1  4336             MOV     A,PARTIT_LO_HIGH
F3A8 51F4       =1  4337             CALL    SEND_BYTE
F3AA E558       =1  4338             MOV     A,PARTIT_LO_LOW
F3AC 51F4       =1  4339             CALL    SEND_BYTE
F3AE E4         =1  4340             CLR     A
F3AF 51F4       =1  4341             CALL    SEND_BYTE
                =1  4342     SH_3:                                   ;Now go into a loop to output the data.
F3B1 754600     =1  4343             MOV     SELECT,#00H             ;Call fetch(0,sa).
F3B4 855744     =1  4344             MOV     PNTHGH,PARTIT_LO_HIGH
F3B7 855845     =1  4345             MOV     PNTLOW,PARTIT_LO_LOW
F3BA 12E651     =1  4346             CALL    IFETCH
F3BD 51F4       =1  4347             CALL    SEND_BYTE
F3BF E558       =1  4348             MOV     A,PARTIT_LO_LOW         ;Increment the address
F3C1 2401       =1  4349             ADD     A,#01H
F3C3 F558       =1  4350             MOV     PARTIT_LO_LOW,A
F3C5 5002       =1  4351             JNC     SH_4
F3C7 0557       =1  4352             INC     PARTIT_LO_HIGH
F3C9 DFE6       =1  4353     SH_4:   DJNZ    COUNT,SH_3              ;Decrement count and loop till zero.
F3CB EE         =1  4354             MOV     A,CHECKSUM              ;Once done output the negation of the
F3CC F4         =1  4355             CPL     A                       ;checksum.
F3CD 04         =1  4356             INC     A
F3CE 51F4       =1  4357             CALL    SEND_BYTE               ;Then go output another record
F3D0 809C       =1  4358             JMP     STORE_HEX
F3D2 E4         =1  4359     SH_5:   CLR     A
F3D3 FE         =1  4360             MOV     CHECKSUM,A
F3D4 51F4       =1  4361             CALL    SEND_BYTE
F3D6 12EF58     =1  4362             CALL    READ_PC
F3D9 C5F0       =1  4363             XCH     A,B
F3DB 51F4       =1  4364             CALL    SEND_BYTE
F3DD E5F0       =1  4365             MOV     A,B
F3DF 51F4       =1  4366             CALL    SEND_BYTE
F3E1 E4         =1  4367             CLR     A
F3E2 04         =1  4368             INC     A
F3E3 51F4       =1  4369             CALL    SEND_BYTE
F3E5 EE         =1  4370             MOV     A,CHECKSUM
F3E6 F4         =1  4371             CPL     A
F3E7 04         =1  4372             INC     A
F3E8 51F4       =1  4373             CALL    SEND_BYTE
F3EA 7A01       =1  4374             MOV     PARAM1,#1
F3EC 7B90       =1  4375             MOV     PARAM2,#90H
F3EE 12EA14     =1  4376             CALL    ITIME
F3F1 12E6FD     =1  4377             CALL    INEWLINE
F3F4 20050A     =1  4378             JB      BINARY_FLG,SH_7         ;Skip control-Z if cassette operation.
F3F7 7A01       =1  4379             MOV     PARAM1,#USART_MODE      ;Select USART mode.
F3F9 12E60B     =1  4380             CALL    UPI_CMD
F3FC 7A1A       =1  4381             MOV     PARAM1,#1AH             ;Insert control Z to close MDS file
F3FE 12E5CE     =1  4382             CALL    ICO
F401 7A13       =1  4383     SH_7:   MOV     PARAM1,#13H
F403 7B88       =1  4384             MOV     PARAM2,#88H
F405 02EA14     =1  4385             JMP     ITIME                   ;Delay 1/2 sec to catch cntrl Z in list mode
                =1  4386     LH_ERROR:
F408 754314     =1  4387             MOV     ERRNUM,#14H             ;File read/write error
F40B 12E3CA     =1  4388             CALL    IERROR
                =1  4389 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                    =1  4390     ;*****************************************************************
                    =1  4391     ;
                    =1  4392     ;    NAME: LOAD_CMD
                    =1  4393     ;
                    =1  4394     ;    ABSTRACT: This routine calls the routine LOAD_HEX which
                    =1  4395     ;       reads data files from the audio cassette in binary. It sets
                    =1  4396     ;       up the user messages and does checksums.
                    =1  4397     ;
                    =1  4398     ;    INPUTS: None
                    =1  4399     ;
                    =1  4400     ;    OUTPUTS: Code memory locations referenced by the file being loaded.
                    =1  4401     ;
                    =1  4402     ;    VARIABLES MODIFIED: PCNHTI, PCNTLO, BINARY_FLG, PARAM1, A,
                    =1  4403     ;       PARAM2
                    =1  4404     ;
                    =1  4405     ;    ERROR EXITS: None
                    =1  4406     ;
                    =1  4407     ;    SUBROUTINES ACCESSED DIRECTLY: IGETOKE, IPRINT_STRING,
                    =1  4408     ;       ICI, UPI_CMD, GET_TYPE, HEXBIN, LOAD_HEX, INIT_IO, ILSTWRD,
                    =1  4409     ;       IWAIT_FOR_USER
                    =1  4410     ;
                    =1  4411     ;
                    =1  4412     ;*****************************************************************
                    =1  4413     LOAD_CMD:
F40E 12E8A0         =1  4414             CALL    IGETOKE                         ;Have a valid LOAD cmd
F411 854961         =1  4415             MOV     PCNTHI,VALHGH                   ;Save addr (label) field
F414 854A62         =1  4416             MOV     PCNTLO,VALLOW
F417 7AF4           =1  4417             MOV     PARAM1,#HIGH CASS_MSG           ;Set up °start cassette° msg
F419 7BEE           =1  4418             MOV     PARAM2,#LOW CASS_MSG
F41B 12E9CD         =1  4419             CALL    IPRINT_STRING
F41E 12E5D1         =1  4420             CALL    ICI                             ;Holds msg on display long enough to be see
                                    n
F421 D205           =1  4421             SETB    BINARY_FLG                      ;Indicates a binary file
F423 7A02           =1  4422             MOV     PARAM1,#CASSETTE_READ
F425 12E60B         =1  4423             CALL    UPI_CMD                         ;Select cassette mode
F428 E548           =1  4424             MOV     A,TOKSTR                        ;Restore original token
F42A B4012E         =1  4425             CJNE    A,#NUMBER_TOKE,FILE_DISPLAY     ;If not a number, need to get next
                    =1  4426     LOAD_LOOP:                                      ;Get number off cass and display it (direct
                                    ory)
F42D 7119           =1  4427             CALL    GET_TYPE                        ;0=data file, 1=EOF, 2=file label record
F42F B402FB         =1  4428             CJNE    A,#2,LOAD_LOOP                  ;Is it the beginning of a file?
F432 E561           =1  4429             MOV     A,PCNTHI                        ;Yes, get the label (addr)
F434 B544F6         =1  4430             CJNE    A,PNTHGH,LOAD_LOOP
F437 E562           =1  4431             MOV     A,PCNTLO
F439 B545F1         =1  4432             CJNE    A,PNTLOW,LOAD_LOOP
F43C 7101           =1  4433             CALL    HEXBIN                          ;Convert to hex, calculate checksum
F43E EE             =1  4434             MOV     A,CHECKSUM
F43F 70C7           =1  4435             JNZ     LH_ERROR                        ;Checksum error
F441 7133           =1  4436             CALL    LOAD_HEX                        ;Read the data file from cassette
F443 12E36C         =1  4437             CALL    INIT_IO
F446 90A000         =1  4438             MOV     DPTR,#UPI_DATA
F449 E0             =1  4439             MOVX    A,@DPTR                         ;Go back to console mode, clear OBF status
                                    bit
F44A 7AF4           =1  4440             MOV     PARAM1,#HIGH FILE_FOUND         ;Set up °File loaded° msg
F44C 7BFF           =1  4441             MOV     PARAM2,#LOW FILE_FOUND
```

```
LOC  OBJ            LINE     SOURCE

F44E 12E9CD     =1  4442             CALL    IPRINT_STRING
F451 AA61       =1  4443             MOV     PARAM1,PCNTHI              ;Set up file number for display
F453 AB62       =1  4444             MOV     PARAM2,PCNTLO
F455 12E7DA     =1  4445             CALL    ILSTWRD
F458 02E396     =1  4446             JMP     IWAIT_FOR_USER             ;Holds msg on display a short time
                =1  4447     FILE_DISPLAY:
F45B 7119       =1  4448             CALL    GET_TYPE                   ;Get here by saying LOAD <CR>
F45D B402FB     =1  4449             CJNE    A,#2,FILE_DISPLAY          ;Ask for directory, cant load w/o file #
F460 12E36C     =1  4450             CALL    INIT_IO
F463 90A000     =1  4451             MOV     DPTR,#UPI_DATA
F466 E0         =1  4452             MOVX    A,@DPTR                    ;Go back to console mode, clr OBF status bi
                                     t
F467 7AF5       =1  4453             MOV     PARAM1,#HIGH NUM_FOUND     ;Sets up °first file found° msg
F469 7B0F       =1  4454             MOV     PARAM2,#LOW NUM_FOUND
F46B 12E9CD     =1  4455             CALL    IPRINT_STRING
F46E AA44       =1  4456             MOV     PARAM1,PNTHGH              ;Set up file number (addr) for display
F470 AB45       =1  4457             MOV     PARAM2,PNTLOW
F472 12E7DA     =1  4458             CALL    ILSTWRD
F475 02E396     =1  4459             JMP     IWAIT_FOR_USER             ;Holds msg on display a short time
                =1  4460 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1  4461     ;*****************************************************************
                =1  4462     ;
                =1  4463     ;    NAME: SAVE_CMD
                =1  4464     ;
                =1  4465     ;    ABSTRACT: This routine writes data in a user specified partition
                =1  4466     ;       to the audio cassette in binary using STORE_HEX which provides
                =1  4467     ;       address, type and checksum for each record. This procedure
                =1  4468     ;       takes care of all UPI set up.
                =1  4469     ;
                =1  4470     ;    INPUTS: Code memory within the partition
                =1  4471     ;
                =1  4472     ;    OUTPUTS: None
                =1  4473     ;
                =1  4474     ;    VARIABLES MODIFIED: PCNTHI, PCNTLO, PARAM1, PARAM2, BINARY_FLG
                =1  4475     ;       A, CHECKSUM
                =1  4476
                =1  4477     ;    ERROR EXITS: None
                =1  4478     ;
                =1  4479     ;    SUBROUTINES ACCESSED DIRECTLY: IGETNUM, IGETOKE, IGET_PART, IPRINT_STRING,
                =1  4480     ;       ICI, UPI_CMD, ICO, SEND_BYTE, IGET_COMMA, IEOL_CHECK, STORE_HEX
                =1  4481     ;
                =1  4482     ;
                =1  4483     ;*****************************************************************
                =1  4484     SAVE_CMD:
F478 12E74F     =1  4485             CALL    IGETNUM
F47B 854961     =1  4486             MOV     PCNTHI,VALHGH
F47E 854A62     =1  4487             MOV     PCNTLO,VALLOW
F481 12E760     =1  4488             CALL    IGET_COMMA
F484 12E8A0     =1  4489             CALL    IGETOKE
F487 12E788     =1  4490             CALL    IGET_PART
F48A 12E5A1     =1  4491             CALL    IEOL_CHECK
F48D 7AF4       =1  4492             MOV     PARAM1,#HIGH CASS_MSG
F48F 7BEE       =1  4493             MOV     PARAM2,#LOW CASS_MSG
F491 12E9CD     =1  4494             CALL    IPRINT_STRING
F494 12E5D1     =1  4495             CALL    ICI
F497 D2O5       =1  4496             SETB    BINARY_FLG
F499 7A82       =1  4497             MOV     PARAM1,#CASSETTE_WRITE
F49B 12E60B     =1  4498             CALL    UPI_CMD                 ;Select cassette mode
F49E 7A3A       =1  4499             MOV     PARAM1,#':'
F4A0 12E5CE     =1  4500             CALL    ICO
F4A3 E4         =1  4501             CLR     A
F4A4 FE         =1  4502             MOV     CHECKSUM,A
F4A5 51F4       =1  4503             CALL    SEND_BYTE
F4A7 E561       =1  4504             MOV     A,PCNTHI
F4A9 51F4       =1  4505             CALL    SEND_BYTE
F4AB E562       =1  4506             MOV     A,PCNTLO
F4AD 51F4       =1  4507             CALL    SEND_BYTE
F4AF 7402       =1  4508             MOV     A,#2
F4B1 51F4       =1  4509             CALL    SEND_BYTE
F4B3 EE         =1  4510             MOV     A,CHECKSUM
F4B4 F4         =1  4511             CPL     A
F4B5 04         =1  4512             INC     A
F4B6 51F4       =1  4513             CALL    SEND_BYTE
F4B8 616E       =1  4514             JMP     STORE_HEX
                =1  4515 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1  4516     ;*****************************************************************
                =1  4517     ;
                =1  4518     ;    NAME: DOWNLOAD_CMD
                =1  4519     ;
                =1  4520     ;    ABSTRACT: This routine temporarily turns off the list mode,
                =1  4521     ;       selects the console, configures the UPI and loads hex files
                =1  4522     ;       from the auxilary terminal into memory.
                =1  4523     ;
                =1  4524     ;    INPUTS: None
                =1  4525     ;
                =1  4526     ;    OUTPUTS: Code memory location specified in the file being loaded.
                =1  4527     ;
                =1  4528     ;    VARIABLES MODIFIED: PARAM1, PARAM2, BINARY_FLG
                =1  4529     ;
                =1  4530     ;    ERROR EXITS: None
                =1  4531     ;
                =1  4532     ;    SUBROUTINES ACCESSED DIRECTLY: IPRINT_STRING, UPI_CMD,
                =1  4533     ;       LOAD_HEX
                =1  4534     ;
                =1  4535     ;
                =1  4536     ;*****************************************************************
                =1  4537     DOWNLOAD_CMD:
F4BA C205       =1  4538             CLR     BINARY_FLG              ;Set "LIST = RESET"
F4BC 7A00       =1  4539             MOV     PARAM1,#SELECT_CON
F4BE 12E60B     =1  4540             CALL    UPI_CMD
F4C1 7AF4       =1  4541             MOV     PARAM1,#HIGH LOAD_MSG
F4C3 7BE4       =1  4542             MOV     PARAM2,#LOW LOAD_MSG
F4C5 12E9CD     =1  4543             CALL    IPRINT_STRING           ;Print loading msg
F4C8 7A01       =1  4544             MOV     PARAM1,#USART_MODE
F4CA 12E60B     =1  4545             CALL    UPI_CMD                 ;Select USART mode
F4CD 7133       =1  4546             CALL    LOAD_HEX
F4CF 22         =1  4547             RET
                =1  4548 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1  4549     ;*****************************************************************
                =1  4550     ;
                =1  4551     ;    NAME: UPLOAD_CMD
                =1  4552     ;
                =1  4553     ;    ABSTRACT: This routine gets a token and partition, turns off
                =1  4554     ;       list mode and outputs hex files to the console through the
                =1  4555     ;       UPI.
                =1  4556     ;
                =1  4557     ;    INPUTS: Code memory locations specified by the partition typed
                =1  4558     ;       by the user.
                =1  4559     ;
                =1  4560     ;    OUTPUTS: None
                =1  4561     ;
                =1  4562     ;    VARIABLES MODIFIED: PARAM1, BINARY_FLG, LSTFLG
                =1  4563     ;
                =1  4564     ;    ERROR EXITS: None
                =1  4565     ;
                =1  4566     ;    SUBROUTINES ACCESSED DIRECTLY: IGET_PART, IGETOKE,
                =1  4567     ;       UPI_CMD, STORE_HEX, IEOL_CHECK
                =1  4568     ;
                =1  4569     ;
                =1  4570     ;*****************************************************************
                =1  4571     UPLOAD_CMD:
F4D0 12E8A0     =1  4572             CALL    IGETOKE
F4D3 12E788     =1  4573             CALL    IGET_PART
F4D6 12E5A1     =1  4574             CALL    IEOL_CHECK
F4D9 C205       =1  4575             CLR     BINARY_FLG
F4DB C201       =1  4576             CLR     LSTFLG                  ;Set 'LIST = RESET'
F4DD 7A40       =1  4577             MOV     PARAM1,#40H             ;Select Keybd/Disply with list on.
F4DF 12E60B     =1  4578             CALL    UPI_CMD
F4E2 616E       =1  4579             JMP     STORE_HEX
                =1  4580     ;*****************************************************************
F4E4 09         =1  4581     LOAD_MSG:       DB      9,CR,LF,('LOADING')
F4E5 0D
F4E6 0A
F4E7 4C4F4144
F4EB 494E47
F4EE 10         =1  4582     CASS_MSG:       DB      16,CR,LF,('START CASSETTE')
F4EF 0D
F4F0 0A
F4F1 53544152
F4F5 54204341
F4F9 53534554
F4FD 5445
F4FF 0F         =1  4583     FILE_FOUND:     DB      15,CR,LF,('LOADED FILE  ')
F500 0D
F501 0A
F502 4C4F4144
F506 45442046
F50A 494C4520
F50E 20
F50F 13         =1  4584     NUM_FOUND:      DB      19,('FIRST FILE FOUND = ')
F510 46495253
F514 54204649
F518 4C452046
```

```
LOC  OBJ            LINE     SOURCE

F51C 4F554E44
F520 203D20
                    4585     ASMBASE:
                    4586     END
```

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

```
NAME                TYPE     VALUE AND REFERENCES

A_TOKE. . . . . N        0051H  452# 549
AB_TOKE . . . . . N        005CH  453# 550
ABR_TOKE. . . . . N        0088H  454# 551 907 2857
ACALL_TOKE. . . . . N        0012H  455# 552
ACC . . . . . N DSEG   00E0H PREDEFINED  828 829 896 1110 1134 1282 1289 1321 1385 1584 1590 1599 2065 2105 2106
                         2109 2110 2486 2917 2938 3004 3056 3188 3191 3194 3283 3287 3308 3320 3324 3424 3427
                         3428 3682 4030 4092
ACC_CMD . . . . . L CSEG   ECF8H  910 3055#
ACC_TOKE. . . . . N        0098H  456# 553 909
ADD_TOKE. . . . . N        0024H  457# 554
ADDC_TOKE . . . . . N        0023H  458# 555
ADDR_SAVE_HIGH. . . N        00F3H  404# 3598
ADDR_SAVE_LOW . . . N        00F4H  405#
AJMP_TOKE . . . . . N        0013H  459# 556
ALFNUM. . . . . L CSEG   E72AH  1706# 2163 2173
ALPHA . . . . . L CSEG   E8ACH  2148 2151#
ANEND . . . . . L CSEG   E730H  1708 1711#
ANL_TOKE. . . . . N        0021H  460# 557
ANY_BR_FLAG . . . . L BSEG   0002H  439# 2893 2902 2923
ASM_PC_HIGH . . . . L DSEG   004BH  259# 821
ASM_PC_LOW. . . . . L DSEG   004CH  260# 822
ASM_TOKE. . . . . N        00B0H  461# 558 911
ASMBASE . . . . . L CSEG   F523H  912 924 4585#
ATA_TOKE. . . . . N        000AH  227# 545
ATDPTR_TOKE . . . . N        005FH  449# 546
ATR0_TOKE . . . . . N        0052H  450# 547
ATR1_TOKE . . . . . N        0053H  451# 548
AZEND . . . . . L CSEG   E712H  1684 1686#
AZTEST. . . . . L CSEG   E706H  1680# 1706 2160
B . . . . . N DSEG   00F0H PREDEFINED  827 892 897 1622 1623 2235 2243 2244 2345 2351 2562 2564 3068 3112 3204
                         3205 3293 3295 3374 3376 3469 3472 3667 3677 3844 3952 3955 3958 3963 4031 4035 4037
                         4363 4365
B_CMD . . . . . L CSEG   EDOAH  914 3067#
B_LAB_1 . . . . . L CSEG   EA74H  2611 2613#
B_LAB_2 . . . . . L CSEG   EA79H  2613 2615#
B_LAB_3 . . . . . L CSEG   EA7EH  2615 2617#
B_O_T . . . . . L BSEG   0000H  280# 2146 2200
B_TOKE. . . . . N        009BH  462# 559 913
B_V_ERR . . . . . L CSEG   EF99H  3555 3558#
BACKSP. . . . . N        0008H  374# 2082 2085
BAR_TOKE. . . . . N        0003H  223# 2301
BASE. . . . . N        E000H  219# 299 300 301 302 303 304 305 306 308 309 310 311 312 313 314 315 316 317 318
                         319 320 323 350 1052
BAUD_CMD. . . . . L CSEG   F1BEH  916 3921#
BAUD_DISPLAY. . . . L CSEG   F22BH  3923 3992#
BAUD_HIGH . . . . . N        00F7H  408# 808 3938 3993
BAUD_LOW. . . . . N        00F8H  409#
BAUD_RATE . . . . . L CSEG   F216H  3925 3975#
BAUD_TOKE . . . . . N        00D0H  463# 560 915
BAUDKEY . . . . . N        00FCH  413# 804 3944 3949
BEND. . . . . L CSEG   EB95H  2801 2816 2819#
```

```
NAME                TYPE     VALUE AND REFERENCES

BINARY_FLG. . . . . L BSEG   0005H  442# 4151 4178 4306 4378 4421 4496 4538 4575
BINARY_LOAD . . . . L CSEG   F315H  4178 4187#
BITLOP. . . . . L CSEG   E6D4H  1611# 1615
BITROT. . . . . L CSEG   E6DBH  1611 1614#
BITSTR. . . . . L CSEG   E6DEH  1606 1616#
BK1LOP. . . . . L CSEG   EC36H  2924# 2940
BLINK . . . . . N        0080H  237# 1505 2052
BM_1. . . . . L CSEG   F20DH  3932 3969#
BMOVE . . . . . L CSEG   EB27H  2567 2770#
BR_CMD. . . . . L CSEG   EB96H  908 918 2853#
BR_TOKE . . . . . N        0089H  464# 561 917 2865 2979
BREAK . . . . . L CSEG   E003H  331# 830 3547 3551 3555
BREAK_CONTINUE. . . L CSEG   EE46H  3260 3266 3277#
BREAK_MSG . . . . . L CSEG   F177H  3840 3841 3852#
BREAK_STATUS. . . . N        00FBH  412# 3258 3311 3626 3726 3822
BREAK_VECTOR. . . . L CSEG   EF85H  3543# 3628 3825
BRK_LINE_HDR. . . . L CSEG   ECA3H  2904 2947 2977#
BRK_LOOP. . . . . L CSEG   EE10H  3248# 3253
BRK1. . . . . L CSEG   EE95H  3313 3315#
BRK2. . . . . L CSEG   EE9EH  3317 3319#
BRK3. . . . . L CSEG   EE5AH  3283 3289#
BRK4. . . . . L CSEG   EE6AH  3285 3287 3297#
BRK5. . . . . L CSEG   EE64H  3292 3294#
BRKEND. . . . . L CSEG   EC12H  2902 2907#
BRKERR. . . . . L CSEG   ECF2H  3004 3029#
BRKMORE . . . . . L CSEG   EE3BH  3268 3272#
BRKOFF. . . . . N        C000H  395# 2895 2924 2989 3011 3262
BS_2. . . . . L CSEG   F20FH  3930 3971#
BS_LOOP . . . . . L CSEG   F1CBH  3927# 3972
C_READ. . . . . L CSEG   E672H  1558 1566#
C_TOKE. . . . . N        005EH  228# 562
CARSET. . . . . L CSEG   E711H  1682 1683 1685#
CASS_MSG. . . . . L CSEG   F4EEH  4417 4418 4492 4493 4582#
CASSETTE_READ . . . N        0002H  386# 4422
CASSETTE_WRITE. . . N        0082H  387# 4497
CAUSE_CMD . . . . . L CSEG   F279H  920 4082#
CAUSE_IMAGE . . . . L DSEG   0060H  428# 823 3264 3270 3276 3282 3286 3315 3319 3672 4086
CAUSE_TAB . . . . . L CSEG   F29DH  4087 4106#
CAUSE_TOKE. . . . . N        00D2H  465# 563 919
CBYTE_TOKE. . . . . N        0080H  229# 564 921 1557 2566 2611 2733 2772
CHANGE. . . . . L CSEG   ED25H  3075 3080#
CHANGE_CHECK. . . . L CSEG   E781H  1824 1832#
CHARIN. . . . . L DSEG   0050H  264# 989 2055 2075 2108 2114 2147 2159 2170 2176 2238 2250 2254 2309
CHECK_ABREV . . . . L CSEG   E8FAH  2181 2186#
CHECK_EPROMS. . . . L CSEG   E3A0H  786 1051#
CHECK_ESC . . . . . L CSEG   E64AH  1507# 1511
CHECK_FROM. . . . . L CSEG   EF74H  3508# 3585 3780
CHECK_LOOP. . . . . L CSEG   E3A5H  1054# 1061
CHECK_OUT_OK. . . . L CSEG   E3C9H  1063 1074#
CHECKSUM. . . . . N REG    R6     293# 1053 1057 1058 1062 4147 4148 4149 4188 4189 4190 4218 4263 4268 4333 4354
                                    4360 4370 4434 4502 4510
CHRCNT. . . . . L DSEG   0051H  265# 987 2036 2043 2072 2076
CI. . . . . N        E009H  300#
CJNE_TOKE . . . . . N        0019H  466# 565
CL_LOOP . . . . . L CSEG   F288H  4089# 4095
```

```
NAME                TYPE     VALUE AND REFERENCES

CL_0. . . . . L CSEG   F290H  4092 4096#
CLR_BRK_LATCHES . . N        0008H  382# 994 3631 3831
CLR_TOKE. . . . . N        002AH  467# 566
CLRBRK. . . . . L CSEG   ECAFH  781 2866 2887 2987#
CLRLOP. . . . . L CSEG   ECB8H  2991# 2993 2995
CMDTAB. . . . . L CSEG   E301H  882 883 906#
CO. . . . . N        E006H  299#
COMMA_TOKE. . . . . N        0002H  373# 1796 2300 2605 2870 3590 3608
CONTINUATION_LINE . N        E068H  318#
CONTINUOUS_MODE . . N        0040H  398# 3960
CONVHEX . . . . . L CSEG   E7D1H  1915 1919 1942# 1976 1980
COPYRIGHT . . . . . L CSEG   E030H  353#
COUNT . . . . . N REG    R7     292# 2401 2409 4088 4095 4220 4254 4261 4319 4326 4327 4330 4331 4334 4353
COUNT1. . . . . L CSEG   EB0CH  2720 2722 2724#
COUNTR. . . . . L DSEG   005DH  425# 2699 2700 2701 2715 2724
CPL_TOKE. . . . . N        002BH  468# 567
CR. . . . . N        000DH  375# 964 1139 1650 2057 2061 2305 2308 3851 3853 4581 4582 4583
CRWAIT. . . . . L CSEG   E80AH  2046# 2081 2088 2091 2103 2107 2117
CSTS. . . . . N        E00CH  301#
CSTS_1. . . . . L CSEG   E5EFH  1320# 1321
DA_TOKE . . . . . N        002CH  469# 569
DASM_TOKE . . . . . N        00B8H  470# 568 570 923
DATA_BREAK. . . . . N        000DH  390# 3801 3814
DATA_MSG. . . . . L CSEG   F2D1H  4110 4119#
DATA_TOKE . . . . . N        00D3H  471# 571 3798 3811
DATECODE. . . . . L CSEG   E046H  354#
DBYTE . . . . . L CSEG   E6A1H  1582 1588#
DBYTE_TOKE. . . . . N        0082H  472# 572 925 1588 2613 2735
DCLAUSE . . . . . L CSEG   EFCFH  3593 3610#
DEC_HIGH. . . . . L CSEG   E5BCH  1225 1227#
DEC_PNT . . . . . L CSEG   E5B3H  1222# 2793 2802
DEC_TOKE. . . . . N        0035H  473# 573
DECODE. . . . . L CSEG   E2E4H  884 886# 903
DECODE_CALL . . . . L CSEG   E2DAH  879 881#
DELAY . . . . . N        00F5H  406# 3619 3729
DELET . . . . . L CSEG   E855H  2080 2082#
DIS_OR_ERR. . . . . L CSEG   EA4EH  2563 2568#
DISERR. . . . . L CSEG   EAA9H  2639# 2663
DISFET. . . . . L CSEG   EAF8H  2702 2716#
DISLOP. . . . . L CSEG   EAD4H  2700# 2728
DISMEM. . . . . L CSEG   EAD1H  2570 2699#
DISPLAY_LIST. . . . L CSEG   F1B1H  3880 3893#
DISPLAY_TOKEN . . . N        E059H  313#
DIV_TOKE. . . . . N        0031H  474# 574
DJNZ_TOKE . . . . . N        0025H  475# 575
DLY_THRU. . . . . L CSEG   F0ADH  3733 3739#
DLYCNT. . . . . L DSEG   005CH  424# 3731 3732 3734
DONT_WAIT . . . . . L CSEG   E64CH  1504 1509#
DOWN_MOVE . . . . . L CSEG   EB75H  2781 2804# 2818
DOWNLOAD_CMD. . . . L CSEG   F4BAH  928 4537#
DOWNLOAD_TOKE . . . N        00E0H  476# 576 927
DPH . . . . . N DSEG   0083H PREDEFINED  825 1060 1124 1318 1323 1410 1418 1441 1449 1475 1478 1555 2352 2353
                                2397 2516 2525 2912 2914 2915 2933 2935 2936 3006 3008 3009 3019 3020 3123 3187 3190
                                3430
DPL . . . . . N DSEG   0082H PREDEFINED  804 824 1125 1317 1324 1409 1419 1440 1450 1474 1479 1556 1585 1591 1604
```

```
NAME                TYPE     VALUE AND REFERENCES

                               1605 2349 2350 2398 2519 2524 2909 2910 2930 2931 3002 3010 3016 3017 3124 3186 3190
                               3193 3196 3200 3204 3207 3210 3213 3216 3219 3222 3228 3231 3234 3237 3240 3243 3258
                               3363 3366 3370 3374 3377 3380 3384 3387 3390 3396 3399 3402 3405 3408 3411 3414 3417
                               3422 3422 3425 3428 3431 3466 3478 3604 3689 3944 3962
DPTR_CMD. . . . . L CSEG   ED4BH  930 3122#
DPTR_TOKE . . . . . N      00A1H  230# 577 929
DT_LOOP . . . . . L CSEG   E9F5H  2445# 2450
DT0 . . . . . L CSEG   E9EEH  2438 2440#
DT0_0 . . . . . L CSEG   E9E3H  2434# 2442
DT1. . . . . L CSEG   E9F2H  2439 2443#
ENDMOD. . . . . L CSEG   EBE0H  2870 2884#
EOL_CHECK . . . . . N      E06EH  320#
EOL_ERROR . . . . . L CSEG   E5A5H  1189 1191#
EOL_TOKE. . . . . N      0007H  226# 879 1189 1824 2305 2607 2855 2872 3781
EOLMEM. . . . . L CSEG   EAA6H  2605 2635 2638#
EQLMOD. . . . . L CSEG   EBA9H  2855 2860#
EQUAL_TOKE. . . . . N      0004H  372# 1834 2302 2563 2861
ERR . . . . . L CSEG   E6B3H  1580 1584 1590 1595#
ERRMOD. . . . . L CSEG   EBA6H  2772 2859# 2861 2886
ERRNUM. . . . . L DSEG   0043H  251# 812 904 1116 1122 1192 1578 1594 1784 1795 1833 1862 1880 2184 2287 2662 2771
                           2858 2860 2885 3029 3269 3559 3611 3615 3671 3795 3804 3810 3885 3935 4029 4084 4387
ERROR . . . . . N      E05FH  315#
ERROR_BEGIN . . . . L CSEG   E401H  1123 1129#
ERROR_LOOP. . . . . L CSEG   E403H  1132# 1134
ERROR_MSG . . . . . L CSEG   E40CH  1064 1065 1113 1114 1138#
ERROR_TABLE . . . . L CSEG   E413H  1070 1071 1120 1140#
ERROR_TEST. . . . . L CSEG   E3F1H  1121# 1137
ERRSET. . . . . L CSEG   E989H  2288 2307#
ESC . . . . . N      001BH  379# 1291 3275 3749
EXERR0. . . . . L CSEG   F00FH  3612 3616 3618 3639#
EXERR1. . . . . L CSEG   F0CDH  3755# 3796 3805 3811
F0. . . . . N BSEG   00D5H PREDEFINED  1551 1554 1558 1570 1575 1586 1592 1606
FETCH . . . . . N      E04AH  308#
FETEND. . . . . L CSEG   E683H  1565 1568 1572 1575# 1596 1613
FETERR. . . . . L CSEG   E6B0H  1564 1594#
FILE_DISPLAY. . . . L CSEG   F45BH  4425 4447# 4449
FILE_FOUND. . . . . L CSEG   F4FFH  4440 4441 4583#
FILL1 . . . . . L CSEG   EAD0H  2674 2677#
FILLMEM . . . . . L CSEG   EAACH  2566 2662#
FILLOOP . . . . . L CSEG   EABBH  2667# 2676
FIRST_FLAG. . . . . L BSEG   0003H  440# 2894 2945 2948
FOREVER_TOKE. . . . N      0008H  477# 579 3788
FROM_TOKE . . . . . N      0009H  478# 578 580 3510
GET_COMMA . . . . . N      E06BH  319#
GET_PART. . . . . N      E065H  317#
GET_TYPE. . . . . L CSEG   F319H  4213# 4216 4251 4427 4448
GETCHR. . . . . L CSEG   E7F7H  2036# 2149 2169 2175 2249 2256 2312
GETEOL. . . . . N      E053H  311#
GETNUM. . . . . N      E050H  310#
GETOKE. . . . . N      E056H  312#
GO_CMD. . . . . L CSEG   F0D0H  932 3779#
GO_TOKE . . . . . N      00C2H  479# 581 931
GOOD_TOKE_FOUND . . L CSEG   E901H  2180 2191#
GR. . . . . N      00F6H  407# 790 3791 3800 3813 3818 3833
GR_PORT . . . . . N      0003H  384# 992 3629 3829
```

```
NAME                TYPE     VALUE AND REFERENCES

GTO . . . . . L CSEG   E90BH  2196 2197#
GT1 . . . . . L CSEG   E910H  2198 2199#
GUARD_MSG . . . . . L CSEG   F2B4H  4108 4115#
HEX1. . . . . L CSEG   EA11H  2486 2488#
HEXBIN. . . . . L CSEG   F301H  4177# 4219 4221 4223 4225 4256 4262 4267 4433
HEXCHR. . . . . L CSEG   E94FH  2232 2252#
HEXEND. . . . . L CSEG   E729H  1698 1702 1704#
HEXSTR. . . . . L CSEG   E922H  2229# 2251
HORIZONTAL_TAB. . . N        0009H  377# 2092
HTEST . . . . . L CSEG   E953H  2230 2254#
HXTEST. . . . . L CSEG   E71DH  1696# 2229
IASCII_TO_HEX . . . L CSEG   EA0BH  342 2484# 4180 4185
IBREAK. . . . . L CSEG   ED94H  331 3186#
ICI . . . . . L CSEG   E5D1H  1281# 1508 2054 3750 4183 4420 4495
ICO . . . . . L CSEG   E5CEH  335 1259# 1506 1651 1653 1829 1978 1982 2083 2086 2115 2408 2458 2714 2730 2982
                       3676 3681 3688 3698 3709 3724 3892 4154 4315 4382 4500
ICONTINUATION_LINE. L CSEG   E643H  366 1503# 2726 2946
ICSTS . . . . . L CSEG   E5E8H  1028 1317# 1472 1510 3267 3743
IDISPLAY_TOKEN. . . L CSEG   E9E0H  361 1827 2431# 2708 2906 2964 2980 3897
IE. . . . . N DSEG   00A8H PREDEFINED  3196 3197 3199 3280 3417 3421 3435
IEO . . . . . N BSEG   0089H PREDEFINED  3432
IEOL_CHECK. . . . . L CSEG   E5A1H  368 1188# 1789 2569 2638 2884 3636 3817 4491 4574
IERROR. . . . . L CSEG   E3CAH  363 905 1104# 1193 1595 1791 2185 2307 2639 2859 3031 3271 3560 3639 3673 3755 3974
                       4388
IFETCH. . . . . L CSEG   E651H  356 1550# 2716 2791 2806 3076 3141 3144 3720 4346
IGET_COMMA. . . . . L CSEG   E760H  367 1793# 4488
IGET_PART . . . . . L CSEG   E788H  365 1860# 2559 2867 4490 4573
IGETEOL . . . . . L CSEG   E759H  359 1788# 2665 2888 3120 3156 3622 3790 3799 3812
IGETNUM . . . . . L CSEG   E74FH  358 1783# 1870 2600 2773 3081 3116 3150 3511 3596 3924 4027 4485
IGETOKE . . . . . L CSEG   E8A0H  360 878 1783 1788 1794 1823 1868 1882 2146# 2150 2558 2565 2604 2606 2634 2854 2862
                       2871 2882 3509 3515 3591 3607 3609 3797 3807 3809 3881 4414 4489 4572
ILSTBYT . . . . . L CSEG   E7DFH  340 1068 1117 1969 1973# 2718 3078 3685 3702 3722 4048 4053 4057 4152
ILSTWRD . . . . . L CSEG   E7DAH  341 1969# 2712 2951 2973 3113 3147 3678 3695 3846 3999 4445 4458
INC_HIGH. . . . . L CSEG   E5B2H  1217 1219#
INC_PNT . . . . . L CSEG   E5AAH  1215# 2603 2675 2727 2808 2817 2918 2939 4260
INC_TOKE. . . . . N        0037H  480# 582
INEWLINE. . . . . L CSEG   E6FDH  338 1649# 1825 2046 2068 2703 2978 3665 3887 4310 4377
INIT_IO . . . . . L CSEG   E36CH  787 877 985# 1107 3297 3303 4270 4437 4450
INPUT . . . . . L CSEG   E88AH  2092 2108#
INPUTOK . . . . . L CSEG   E892H  2109 2110 2111#
IP. . . . . N DSEG   00B8H PREDEFINED  3207 3208 3396 3398
IPRINT_STRING . . . L CSEG   E9CDH  343 853 1066 1072 1115 1126 2396# 3828 3842 4104 4419 4442 4455 4494 4543
ISAVE_AND_DISPLAY . L CSEG   E7C3H  362 1912# 2629 2631
ISIT_DISPLAY. . . . L CSEG   E76AH  1820# 3072 3108 3137 3879 3922 4024 4083
ISTORE. . . . . L CSEG   E658H  357 1553# 2602 2668 2795 2810 3083 3152 3155 4259
ITO . . . . . N BSEG   0088H PREDEFINED  3433
ITIME . . . . . L CSEG   EA14H  339 1002 1106 2514# 3737 3742 4276 4309 4313 4376 4385
IWAIT_FOR_USER. . . L CSEG   E396H  364 854 1024# 1127 2723 2907 3079 3114 3148 3753 3847 3898 4000 4059 4105 4446 4459
IWAIT_FOR_USER_1. . L CSEG   E39AH  1027# 1029
JB_TOKE . . . . . N        0027H  481# 583
JBC_TOKE. . . . . N        0028H  482# 584
JC_TOKE . . . . . N        0018H  483# 585
JMP_TOKE. . . . . N        0032H  484# 586
JNB_TOKE. . . . . N        0026H  485# 587
JNC_TOKE. . . . . N        0017H  486# 588
```

```
NAME                TYPE     VALUE AND REFERENCES

JNZ_TOKE. . . . . N        0015H  487# 589
JZ_TOKE . . . . . N        0016H  488# 590
KEY_BYTE. . . . . L CSEG   ED10H  3057 3061 3065 3069 3071#
KEYTAB. . . . . . L CSEG   E0D7H  651# 2183 2344 2444
KEYWORD_DISPLAY . . L CSEG   ED63H  3125 3130 3136#
LAB1. . . . . . . L CSEG   E747H  1744 1750#
LAB10 . . . . . . L CSEG   E995H  2308 2312#
LAB18 . . . . . . L CSEG   EFDCH  3614 3615#
LAB1A . . . . . . L CSEG   E74AH  1749 1753#
LAB1B . . . . . . L CSEG   E74BH  1736 1754#
LAB2. . . . . . . L CSEG   EBF8H  2895# 2919 2944 2960 2975
LAB23 . . . . . . L CSEG   EB20H  2705 2733#
LAB3. . . . . . . L CSEG   EC2EH  2917 2921#
LAB5A . . . . . . L CSEG   EC1FH  2911 2913#
LAB5B . . . . . . L CSEG   EC15H  2901 2908#
LAB6A . . . . . . L CSEG   EC4CH  2932 2934#
LAB7. . . . . . . L CSEG   EC7EH  2954 2956#
LAB8. . . . . . . L CSEG   EC9EH  2971 2973#
LB_10 . . . . . . L CSEG   EC6AH  2945 2947#
LCALL_TOKE. . . . . N        0010H  489# 591
LDLOOP. . . . . . L CSEG   EA57H  2601# 2637
LEGALI. . . . . . L CSEG   E866H  2078 2089#
LENGTH_HIGH . . . . L DSEG   0063H  431# 1879 2788
LENGTH_LOW. . . . . L DSEG   0064H  432# 1876 2785
LF. . . . . . . . N        000AH  376# 964 1139 1652 3851 3853 4581 4582 4583
LFTROT. . . . . . L CSEG   E6F7H  1627# 1628
LH_4. . . . . . . L CSEG   F337H  4254# 4261
LH_6. . . . . . . L CSEG   F348H  4255 4262#
LH_7. . . . . . . L CSEG   F350H  4252 4266#
LH_8. . . . . . . L CSEG   F34DH  4265# 4266 4269
LH_ERROR. . . . . L CSEG   F408H  4265 4386# 4435
LINBUF. . . . . . L DSEG   0024H  245# 2002 2044 2058 2071 2093 2111 2608 2874
LINCNT. . . . . . L DSEG   0053H  267# 2152 2156 2157 2172
LINE_START. . . . . L DSEG   0052H  266# 876 2041 2080 2633 2873 3306
LINMAX. . . . . . N        0018H  235# 245 2090 2102
LIST_1. . . . . . L CSEG   F1B8H  3895 3897#
LIST_2. . . . . . L CSEG   F19CH  3882 3885#
LIST_CMD. . . . . L CSEG   F18EH  934 3878#
LIST_TOKE . . . . . N        00D7H  490# 592 933
LJMP_TOKE . . . . . N        0011H  491# 593
LNLGTH. . . . . . L DSEG   0054H  268# 988 2037 2042 2047 2059 2062 2069 2079 2087 2089 2094 2096 2099 2112 2116
LOAD_CMD. . . . . L CSEG   F40EH  936 4413#
LOAD_HEX. . . . . L CSEG   F333H  4250# 4264 4436 4546
LOAD_LOOP . . . . . L CSEG   F42DH  4426# 4428 4430 4432
LOAD_MSG. . . . . L CSEG   F4E4H  4541 4542 4581#
LOAD_TOKE . . . . . N        00E2H  492# 594 935
LODMEM. . . . . . L CSEG   EA54H  2564 2600#
LSSEQL. . . . . . L CSEG   E731H  1736# 2900 2927 2943
LSTBRK. . . . . . L CSEG   EBEFH  2857 2890#
LSTBYT. . . . . . N        E015H  304#
LSTFLG. . . . . . L BSEG   0001H  281# 438 785 1108 1504 2064 3305 3883 3890 3895 4576
LSTOUT. . . . . . L CSEG   EC5BH  2928 2938 2941#
LSTWRD. . . . . . N        E018H  305#
MAXHGH. . . . . . N        001FH  401# 2896 2988 3013
MAXLOW. . . . . . N        00FFH  400# 2897 2987
```

```
NAME                TYPE     VALUE AND REFERENCES

MAXNUM_FLAG . . . . L BSEG   0004H  441# 985 1736 1739 1754
MEMORY. . . . . . . L CSEG   E65CH  1552 1555#
MEMORY_CMD. . . . . L CSEG   EA2AH  922 926 942 944 960 2554#
MON_FLAGS . . . . . N        00FAH  411# 3254 3363
MORE_CONT . . . . . L CSEG   E882H  2102 2104#
MORE_SPACE. . . . . L CSEG   E877H  2097# 2105 2106
MOV_TOKE. . . . . . N        001FH  493# 595
MOVC_TOKE . . . . . N        001AH  494# 596
MOVX_TOKE . . . . . N        001BH  495# 597
MUL_TOKE. . . . . . N        0030H  496# 598
MULTISTEP . . . . . N        00FFH  416# 3624 3728
NEWLINE . . . . . . N        E00FH  302#
NEXT_ENTRY. . . . . L CSEG   E2F6H  888 899#
NMTEST. . . . . . . L CSEG   E713H  1688# 1696 1709 2225 2231
NO_BREAK. . . . . . N        0009H  385# 791 996 3792
NOBRK_MSG . . . . . L CSEG   F2E8H  4112 4123#
NOP_TOKE. . . . . . N        003BH  497# 599
NOT_MATCH_TBL . . . L CSEG   E977H  2289 2295#
NOT_STEP. . . . . . N        00FBH  414# 3823
NOT_STEP_THREE. . . L CSEG   F02AH  3669 3670 3674#
NOTBOT. . . . . . . L CSEG   E914H  2197 2199 2201#
NOTDAT. . . . . . . L CSEG   F100H  3798 3804#
NOTFOR. . . . . . . L CSEG   F0E8H  3788 3795#
NOTFRM. . . . . . . L CSEG   EF84H  3510 3516#
NOWAIT. . . . . . . L CSEG   EB14H  2727# 2731
NTLAST. . . . . . . L CSEG   EB19H  2725 2729#
NUM_FOUND . . . . . L CSEG   F50FH  4453 4454 4584#
NUMBER. . . . . . . L CSEG   E917H  2162 2225#
NUMBER_FOUND. . . . L CSEG   E95BH  2255 2257#
NUMBER_OF_BYTES . . L DSEG   004DH  261#
NUMBER_TOKE . . . . N        0001H  222# 1785 1863 2257 2635 2663 2863 3612 4425
NUMEND. . . . . . . L CSEG   E71CH  1692 1694#
NUMMEN. . . . . . . L CSEG   EAA1H  2607 2635#
NUMMOD. . . . . . . L CSEG   EBBCH  2865 2867# 2872 2883
OFST. . . . . . . . N        0010H  234# 449 450 451 453 455 457 458 459 460 466 467 468 469 473 474 475 480 481 482
                                      483 484 485 486 487 488 489 491 493 494 495 496 497 500 501 504 517 518 519 520 521 522
                                      524 525 528 529 537 538 539
ON_TOKE . . . . . . N        000FH  498# 600 3882 3894
OR_TOKE . . . . . . N        000BH  499# 601 3808
ORG_TOKE. . . . . . N        00D4H  231# 602
ORL_TOKE. . . . . . N        0022H  500# 603
OUR_CODE_HIGH . . . L DSEG   004EH  262#
OUR_CODE_LOW. . . . L DSEG   004FH  263#
OUTIBK. . . . . . . L CSEG   ECEAH  3021# 3024 3025
OUTCHR. . . . . . . L CSEG   E840H  2037 2071#
OUTOKE. . . . . . . L CSEG   EC8AH  2957 2959 2962#
P1. . . . . . . . . N DSEG   0090H PREDEFINED  3210 3211 3377 3379
P3. . . . . . . . . N DSEG   00B0H PREDEFINED  3213 3214 3380 3383
PAINTER . . . . . . L CSEG   E7EDH  2002# 2050 2070
PARAM1. . . . . . . N REG    R2     286# 799 802 851 883 903 990 992 994 996 998 1000 1064 1067 1070 1104 1111 1113
                                      1116 1124 1258 1416 1447 1505 1559 1563 1573 1622 1650 1652 1680 1688 1699 1745 1746
                                      1750 1826 1828 1913 1918 1971 1973 1977 1981 2005 2039 2052 2056 2057 2066 2078 2082
                                      2085 2092 2159 2165 2170 2176 2178 2182 2183 2187 2189 2192 2243 2247 2250 2282 2308
                                      2346 2397 2407 2457 2485 2514 2601 2628 2630 2667 2707 2711 2713 2717 2729 2792 2807
                                      2896 2905 2949 2963 2970 2972 2979 2981 2987 2993 3009 3025 3077 3082 3112 3117 3146
```

```
NAME                TYPE     VALUE AND REFERENCES

                               3151 3154 3295 3298 3300 3309 3348 3350 3353 3357 3360 3483 3512 3629 3631 3633 3667
                               3669 3675 3677 3680 3684 3687 3694 3697 3701 3708 3721 3723 3735 3740 3826 3829 3831
                               3835 3840 3844 3888 3891 3894 3896 3995 4047 4052 4056 4099 4150 4179 4184 4257 4272
                               4274 4307 4311 4314 4374 4379 4381 4383 4417 4422 4440 4443 4453 4456 4492 4497 4499
                               4539 4541 4544 4577
PARAM2. . . . . N REG    R3     287# 852 1001 1065 1071 1105 1114 1125 1742 1970 2233 2239 2252 2398 2432 2436 2441
                               2442 2450 2517 2710 2897 2950 2968 2988 2995 3010 3024 3111 3118 3145 3294 3480 3513
                               3668 3670 3691 3736 3741 3827 3841 3845 3998 4103 4271 4275 4308 4312 4375 4384 4418
                               4441 4444 4454 4457 4493 4542
PARAM3. . . . . N REG    R4     288# 1737 1751 1974 1979 2451 2460 2898 2925 2941
PARAM4. . . . . N REG    R5     289# 1738 1743 2899 2926 2942
PARAM5. . . . . N REG    R6     290#
PARAM6. . . . . N REG    R7     291# 2008 2047 2048 2069
PARTIT_HI_HIGH. . . L DSEG   0059H  273# 1865 1872 1877 2673 2722 2783 2815 3003 4320
PARTIT_HI_LOW . . . L DSEG   005AH  274# 422 1864 1871 1874 2671 2720 2782 2813 3000 4317
PARTIT_LO_HIGH. . . L DSEG   0057H  271# 1867 1878 2561 2799 3005 3012 4321 4336 4344 4352
PARTIT_LO_LOW . . . L DSEG   0058H  272# 1866 1875 2560 2797 3001 3015 4318 4338 4345 4348 4350
PARTITION_E . . . . L CSEG   E7C1H  1869 1885#
PC_CHA. . . . . L CSEG   ED3EH  3109 3115#
PC_CMD. . . . . L CSEG   ED2DH  938 3107#
PC_TOKE . . . . . N      00A0H  232# 604 937
PCNTHI. . . . . L DSEG   0061H  429# 1235 1236 2775 2780 2787 2789 4415 4429 4443 4486 4504
PCNTLO. . . . . L DSEG   0062H  430# 1232 1233 2774 2778 2784 2786 4416 4431 4444 4487 4506
PGMBRK. . . . . L CSEG   F121H  3808 3817#
PLUS_TOKE . . . . . N      0005H  225# 2303
PNTHGH. . . . . L DSEG   0044H  252# 1218 1226 1234 1235 1555 1579 2561 2628 2672 2711 2721 2779 2783 2800 2814
                               2892 2898 2913 2922 2925 2934 2957 2970 3073 3139 3715 4222 4272 4344 4430 4456
PNTLOW. . . . . L DSEG   0045H  253# 1215 1216 1222 1223 1231 1232 1556 1583 1589 1597 1608 1616 1624 2560 2630
                               2670 2710 2719 2777 2782 2798 2812 2891 2899 2908 2921 2926 2929 2959 2966 3056 3060
                               3064 3068 3123 3128 3133 3143 3153 3718 4224 4271 4345 4432 4457
POINTO. . . . . N REG    R0     284# 1916 1917 1920 1921 2045 2060 2061 2073 2074 2095 2100 2101 2113 2114 2151
                               2154 2155 2158 2166 2167 2168 2171 2342 2357 2360 2608 2610 2612 2614 2616 2617 2618
                               2619 2620 2622 2623 2624 2625 2626 2627 2632 2874 2875 2876 2877 2878 2879 2880 2881
                               3247 3250 3252 3253 3926 3928 3932 3945 3971
POINT1. . . . . N REG    R1     285# 2002 2004 2007
POP_TOKE. . . . . N      002DH  501# 605
POUND_TOKE. . . . . N      0006H  224# 2304
POWER_ON. . . . . L CSEG   E267H  325 344 345 346 347 348 780#
PRE_SET_BAUD. . . . L CSEG   F1DBH  3937# 3969
PRE_UNBREAK . . . . L CSEG   EEA6H  3275 3320 3322#
PRINT_STRING. . . . N      E01EH  306#
PRINT_STRING_1. . . L CSEG   E9D6H  2403# 2409
PRINT_STRING_E. . . L CSEG   E9DFH  2402 2410#
PROG_MSG. . . . . L CSEG   F2C3H  4109 4117#
PROGRAM_BREAK . . . N      000BH  391# 3814 3819
PROGRAM_TOKE. . . . N      00D5H  502# 606 3805
PSW . . . . . N DSEG   00D0H PREDEFINED  826 3060 3216 3217 3244 3384 3386
PSW_CMD . . . . . L CSEG   ECFEH  940 3059#
PSW_TOKE. . . . . N      0099H  503# 607 939
PUSH_TOKE . . . . . N      002FH  504# 608
PX0 . . . . . N BSEG   00B8H PREDEFINED  3434
R0_TOKE . . . . . N      0090H  505# 609
R1_TOKE . . . . . N      0091H  506# 610
R2_TOKE . . . . . N      0092H  507# 611
R3_TOKE . . . . . N      0093H  508# 612
```

```
NAME                TYPE     VALUE AND REFERENCES

R4_TOKE . . . . . N        0094H  509# 613
R5_TOKE . . . . . N        0095H  510# 614
R6_TOKE . . . . . N        0096H  511# 615
R7_TOKE . . . . . N        0097H  512# 616
RAMIO . . . . . N        B800H  396# 3959 3965
RAMOFF. . . . . N        B000H  394# 790 796 808 1352 1383 1577 3188 3254 3311 3323 3355 3464 3476 3586 3598 3619
                                  3626 3682 3689 3699 3703 3726 3729 3791 3800 3813 3818 3822 3833 3938 3949 3993 4025
                                  4038 4049
RBIT. . . . . L CSEG   E6B5H  1588 1596#
RBIT_TOKE . . . . . N        0084H  513# 617 941 1596 2621 2737
RBOUT . . . . . N        007FH  378# 2078
RBS_TOKE. . . . . N        0000H  514# 618
RBYTE . . . . . L CSEG   E686H  1569 1577#
RBYTE_TOKE. . . . . N        0081H  515# 619 943 1582 2734 3074 3138
READ_PC . . . . . L CSEG   EF58H  3110 3289 3462# 3666 3843 4362
REG . . . . . N        0040H  233# 449 450 451 453
REPAINT . . . . . L CSEG   E815H  2048 2050#
REPAINT_1 . . . . . L CSEG   E818H  2049 2051#
REPAINT_2 . . . . . L CSEG   E7EFH  2003# 2008
RESET_CMD . . . . . N        0004H  381# 990
RESET_TOKE. . . . . N        000EH  516# 620 2886 2905 3886 3896
RET_TOKE. . . . . N        003AH  517# 621
RETI_TOKE . . . . . N        0039H  518# 622
RHTROT. . . . . L CSEG   E6E7H  1620# 1621
RL_TOKE . . . . . N        0034H  519# 623
RL4 . . . . . L CSEG   E92EH  2234# 2253
RLC_TOKE. . . . . N        0033H  520# 624
RR_TOKE . . . . . N        0038H  521# 625
RRC_TOKE. . . . . N        0036H  522# 626
RSTMOD. . . . . L CSEG   EBE3H  2863 2885#
RUBOUT. . . . . L CSEG   E84BH  2057 2078#
RUN_USER. . . . . L CSEG   F12AH  3782 3794 3803 3816 3821#
RUN_USER_RETURN . . L CSEG   F14FH  3318 3839#
S_S_1 . . . . . L CSEG   E9B3H  2354# 2361
S_S_2 . . . . . L CSEG   E9C4H  2358 2366#
S_S_3 . . . . . L CSEG   E9CAH  2366 2370#
SAVE_AND_DISPLAY. . N        E05CH  314#
SAVE_CMD. . . . . L CSEG   F478H  946 4484#
SAVE_SEL. . . . . N        00F2H  403# 3586 3604 3703
SAVE_TOKE . . . . . N        00E3H  523# 627 945
SCON. . . . . N DSEG   0098H PREDEFINED  3219 3220 3387 3389
SELECT. . . . . L DSEG   0046H  254# 1550 1553 1581 2557 2609 2704 2770 3074 3138 3712 4258 4343
SELECT_CON. . . . . N        0000H  238# 998 2039 3360 4539
SEND_BINARY . . . . L CSEG   F2FEH  4151 4153#
SEND_BYTE . . . . . L CSEG   F2F4H  4146# 4335 4337 4339 4341 4347 4357 4361 4364 4366 4369 4373 4503 4505 4507 4509
                                  4513
SET_BAUD. . . . . L CSEG   F1EAH  807 3304 3948#
SETB_TOKE . . . . . N        0029H  524# 628
SETBRK. . . . . L CSEG   ECC0H  2868 2999#
SH_1. . . . . L CSEG   F399H  4325 4327#
SH_2. . . . . L CSEG   F3A0H  4329 4331#
SH_3. . . . . L CSEG   F3B1H  4342# 4353
SH_4. . . . . L CSEG   F3C9H  4351 4353#
SH_5. . . . . L CSEG   F3D2H  4323 4359#
SH_6. . . . . L CSEG   F382H  4306 4314#
```

```
NAME                TYPE     VALUE AND REFERENCES

SH_7. . . . . L CSEG   F401H  4378 4383#
SIGN_ON . . . . . L CSEG   E2BFH  850# 3314
SIGN_ON_MSG . . . . L CSEG   E352H  851 852 883 963#
SINGLE_BREAK. . . . N        0001H  389# 3633
SINGLE_STEP_MSG . . L CSEG   F2DCH  4111 4121#
SINGLESTEP. . . . . N        00FEH  415# 3637
SJMP_TOKE . . . . . N        0014H  525# 629
SP. . . . . N DSEG   0081H PREDEFINED  784 875 3064 3222 3223 3227 3281 3390 3395 3464 3476 3699
SP_CMD. . . . . L CSEG   ED04H  948 3063#
SP_TOKE . . . . . N        009AH  526# 630 947
SPACCO. . . . . L CSEG   E5CCH  1069 1118 1258# 2084 2709 2962 2965 3679 3686 3696 3707
SPEFUN. . . . . L CSEG   E6C4H  1599 1604#
SPFILL. . . . . L CSEG   E8B1H  2153# 2156
SPWAIT. . . . . L CSEG   E8DBH  2173# 2177
SSRET . . . . . L CSEG   F0C7H  3728 3753#
ST_1. . . . . L CSEG   F254H  4034 4036#
STACK . . . . . N        0007H  380# 875 3227 3281
START . . . . . L CSEG   E2C9H  875# 880 885 1128 1292 3754 3848
START_16_TIMER. . . N        00C0H  399# 3966
START_COMPARE . . . L CSEG   E73CH  1737 1738 1740#
STATE_ERR . . . . . L CSEG   F213H  3886 3936 3973# 4030 4085
STEP_CMD. . . . . L CSEG   EF9FH  950 3584#
STEP_STOP . . . . . L CSEG   F0BCH  3744 3747#
STEP_TOKE . . . . . N        00C1H  527# 631 949
STEP51. . . . . L CSEG   EFF0H  3625# 3638
STEP51_EXIT . . . . L CSEG   F091H  3706 3725#
STEP51_RETURN . . . L CSEG   F012H  3321 3664#
STORE . . . . . N        E04DH  309#
STORE_HEX . . . . . L CSEG   F36EH  4305# 4358 4514 4579
STORED_CHECK_SUM. . L CSEG   E049H  355#
STPDLY. . . . . L CSEG   F09EH  3732# 3738
STPEOL. . . . . L CSEG   F008H  3590 3608 3636#
STPLOP. . . . . L CSEG   EFEEH  3623# 3746
STPLOP_REACH. . . . L CSEG   F0B9H  3745# 3749 3751
STRFIL. . . . . L CSEG   E8C7H  2161 2163# 2172
STRGBF. . . . . L DSEG   003CH  246# 2151 2158 2342
STRGCT. . . . . L DSEG   0055H  269# 2343 2361
STRING_SPACE. . . . L CSEG   E99BH  2179 2188 2341#
STRTST. . . . . L CSEG   E8E7H  2164 2174 2178#
STRTST1 . . . . . L CSEG   E8E9H  2179# 2183
SUBB_TOKE . . . . . N        001EH  528# 632
SWAP_POINTERS . . . L CSEG   E5BDH  1230# 2790 2794 2805 2809
SWAP_TOKE . . . . . N        002EH  529# 633
SYM_TBL_SRCH. . . . L CSEG   E966H  2284# 2298
SYMBOL. . . . . L CSEG   E961H  2226 2282#
SYMBOL_TBL. . . . . L CSEG   E97BH  2283 2299#
SYMEND. . . . . L CSEG   E98CH  2294 2308#
T_LAB . . . . . L CSEG   EA89H  2621 2623#
TABKEY. . . . . L CSEG   E86DH  2090 2092#
TCON. . . . . N DSEG   0088H PREDEFINED  3200 3201 3203 3414 3416
TEMP. . . . . N REG    R5     294# 3014 3018 4182 4186 4322 4324
TEMP_LOW. . . . . L DSEG   0047H  255# 2664 2666 3124 3129 3134 3142 3146 3153
TEMP1 . . . . . L DSEG   0056H  270# 1119 1123 1136 1608 1609 1610 1611 1616 1617 1618 1621 1624 1625 1626 1628
                                  2168 2171 2356 2358
TH0 . . . . . N DSEG   008CH PREDEFINED  3128 3228 3229 3399 3401
```

```
NAME                TYPE     VALUE AND REFERENCES

TH1 . . . . . N DSEG   008DH PREDEFINED  3133 3231 3232 3402 3404
TILL_TOKE . . . . . N        000CH  530# 634 635 3796
TIME. . . . . N        E012H  303#
TIME1 . . . . . L CSEG   EA1DH  2521# 2527
TIMER_HIGH. . . . . N        0005H  397# 3959
TIMER_PRESET. . . . L CSEG   F21DH  3953 3983#
TL0 . . . . . N DSEG   008AH PREDEFINED  3129 3234 3235 3405 3407
TL1 . . . . . N DSEG   008BH PREDEFINED  3134 3237 3238 3408 3410
TM0_CMD . . . . . L CSEG   ED54H  952 3127#
TM0_TOKE. . . . . N        00A2H  531# 636 951
TM1_CMD . . . . . L CSEG   ED5DH  954 3132#
TM1_TOKE. . . . . N        00A3H  532# 637 953
TMOD. . . . . N DSEG   0089H PREDEFINED  3240 3241 3411 3413
TO_TOKE . . . . . N        000DH  533# 638 1869 2963
TOK_WRITE . . . . . L CSEG   EA03H  2454 2456#
TOKERR. . . . . L CSEG   E8F4H  2184# 2190
TOKLOP. . . . . L CSEG   E9FDH  2452# 2460
TOKSAV. . . . . L DSEG   005BH  423# 1822 1826 2556 2853 2856 2864 3595 3605 3705 3710
TOKSIZ. . . . . N        0004H  236# 246 2152 2157 2343
TOKSTR. . . . . L DSEG   0048H  256# 888 1822 1861 2195 2201 2257 2258 2282 2289 2293 2310 2313 2554 2556 2853 2869
                                  3589 3595 3610 3787 4424
TOKTBL. . . . . L CSEG   E073H  544# 2183 2193 2435
TOP_CMD . . . . . L CSEG   F239H  956 4024#
TOP_DISPLAY . . . . L CSEG   F25BH  4026 4042#
TOP_LIST_0. . . . . L CSEG   F271H  4051 4055#
TOP_LIST_1. . . . . L CSEG   F276H  4054 4058#
TOP_LIST_2. . . . . L CSEG   F25FH  4044 4046#
TOP_PORT. . . . . N        0083H  383# 799 3298 3353
TOP_STORE . . . . . N        00F9H  410# 3355 4025 4038 4049
TOP_TOKE. . . . . N        00D6H  534# 639 955
TYPE. . . . . L DSEG   0065H  433# 4226
UCI . . . . . L CSEG   E5FFH  336 1381# 1382
UCSTS . . . . . L CSEG   E5F9H  337 1352# 1381
UNBREAK . . . . . L CSEG   EEACH  3348# 3635 3837
UNBRK_LOOP. . . . . L CSEG   EED9H  3367# 3371
UP_MOVE . . . . . L CSEG   EB55H  2790# 2803
UPI_C_1 . . . . . L CSEG   E612H  1412# 1415
UPI_CMD . . . . . L CSEG   E60BH  800 991 993 999 1112 1408# 2040 2067 3299 3310 3349 3354 3361 3630 3830 3889 4380
                                  4423 4498 4540 4545 4578
UPI_CONTROL . . . . N        A001H  392# 1319 1411 1442
UPI_DATA. . . . . N        A000H  393# 788 1025 1476 4277 4438 4451
UPI_DATA_IMAGE. . . N        00F1H  402# 796 1352 1383 3323
UPI_IN. . . . . L CSEG   E632H  801 1281 1472# 1473 3273 3302 3359 3748 4177 4214
UPI_INA . . . . . L CSEG   E5D9H  1283 1284#
UPI_INB . . . . . L CSEG   E5DEH  1286 1287#
UPI_INE . . . . . L CSEG   E5E7H  1291 1293#
UPI_INR . . . . . L CSEG   E5E2H  1285 1288 1290#
UPI_O_1 . . . . . L CSEG   E625H  1443# 1445
UPI_OUT . . . . . L CSEG   E61EH  803 995 997 1259 1440# 2006 2053 3301 3351 3352 3358 3632 3634 3832 3836
UPLOAD_CMD. . . . . L CSEG   F4D0H  958 4571#
UPLOAD_TOKE . . . . N        00E1H  535# 640 957
USART_MODE. . . . . N        0001H  388# 3348 3888 4379 4544
USER_MSG. . . . . L CSEG   F2A9H  4107 4113#
UTILIT_ERROR. . . . L CSEG   E75EH  1785 1790# 1796 1834 1863 1881
VALHGH. . . . . L DSEG   0049H  257# 1865 1867 1872 2228 2245 2248 2775 3117 3151 3512 3599 3617 3930 3939 4028
```

```
NAME                TYPE     VALUE AND REFERENCES

                               4415 4486
VALLOW. . . . . . . L DSEG   004AH  258# 1864 1866 1871 2227 2234 2237 2241 2242 2601 2664 2666 2667 2774 3082 3118
                               3154 3513 3602 3614 3620 3931 3942 4033 4416 4487
VPC_HIGH. . . . . . L DSEG   005FH  427# 2922 2941 2949 2955 2956
VPC_LOW . . . . . . L DSEG   005EH  426# 2921 2942 2950 2952 2953 2958
WAIT_FOR_USER . . . N        E062H  316#
WCHANGE . . . . . . L CSEG   ED81H  3140 3149#
WORKING_SPACE . . . L DSEG   0040H  247#
WRITE_PC. . . . . . L CSEG   EF67H  3119 3296 3474# 3514 4273
X_WRT . . . . . . . L CSEG   E682H  1574# 1629
XBYTE . . . . . . . L CSEG   E677H  1557 1569#
XBYTE_TOKE. . . . . N        0086H  536# 641 959 1569 2615 2739
XCH_TOKE. . . . . . N        001DH  537# 642
XCHD_TOKE . . . . . N        001CH  538# 643
XEQT_MSG. . . . . . L CSEG   F165H  3826 3827 3850#
XREAD . . . . . . . L CSEG   E67DH  1571# 1587 1593
XRL_TOKE. . . . . . N        0020H  539# 644
XWRITE. . . . . . . L CSEG   E681H  1570 1573# 1586 1592
ZTEST . . . . . . . L CSEG   E70CH  1681 1683#


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(5,18,81) WORKFILES(:F3
                       :,:F3:) EP DB SB


LOC  OBJ            LINE     SOURCE

                       1     $NOMACRO
                       2     $XREF
                       3     $TITLE('SDK-51 ASSEMBLER/DISASSEMBLER INTEL PROPRIETARY VERS. #1.0')
                       4     ;*********************************************************************
                       5
                       6
                       7     ;       SDK-51 MONITOR  INTEL PROPRIETARY
                       8
                       9     ;       VERSION 1.0           5-18-81;
                      10
                      11
                      12     ;       NN  N  OOOOO  TTTTT  EEEEE    !!
                      13     ;       NN  N  O   O    T    E        !!
                      14     ;       N N N  O   O    T    E        !!
                      15     ;       N N N  O   O    T    EEEE     !!
                      16     ;       N  NN  O   O    T    E        !!
                      17     ;       N  NN  O   O    T    E
                      18     ;       N   N  OOOOO    T    EEEEE    !!
                      19
                      20
                      21     ;*********************************************************************
                      22
                      23
                      24     ;       COPYRIGHT (C) 1981 INTEL CORPORATION.
                      25
                      26     ;       ALL RIGHTS RESERVED.
                      27
                      28     ;       NO PART OF THIS PROGRAM OR PUBLICATION MAY BE REPRODUCED,
                      29     ;       TRANSMITTED, TRANSCRIBED, STORED IN A RETRIEVAL SYSTEM, OR
                      30     ;       TRANSLATED INTO ANY LANGUAGE OR COMPUTER LANGUAGE, IN ANY
                      31     ;       FORM OR BY ANY MEANS, ELECTRONIC, MECHANICAL, MAGNETIC,
                      32     ;       OPTICAL, CHEMICAL, MANUAL OR OTHERWISE, WITHOUT THE PRIOR
                      33     ;       WRITTEN PERMISSION OF INTEL CORPORATION, 3065 BOWERS AVENUE,
                      34     ;       SANTA CLARA, CALIFORNIA 95051.
                      35
                      36
                      37
                      38     ;*********************************************************************
  F523                39     ASMBASE EQU     0F523H
F523                  40     ORG ASMBASE
F523 02F916           41             LJMP    ASSEMBLY_CMD
F526 02FC9C           42             LJMP    DISASSEMBLY_CMD
                      43     ;INCLUDE FOR COMMON.INC
                      44 +1  $NOLIST
                     149 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                     150     ;*****************************************************************
                     151     ;
                     152     ;       TABLE OF CONTENTS:
                     153     ;
                     154     ;       This listing contains a source file and 3 include files.
                     155     ;       Each include file contains a number of subroutines.  Each
                     156     ;       subroutine listed has its own 'header' block and begins on
                     157     ;       a new page.
                     158     ;       The files are as follows:
                     159     ;
                     160     ;       SDKADM.SRC (SOURCE FILE)
                     161     ;
                     162     ;                       MNEMONIC_TAB
                     163     ;                       TEMPORARY VARIABLES
                     164     ;                       FLAG ADDRESSES
                     165     ;                       CONSTANTS
                     166     ;                       INSTRUCTION_CODE
                     167     ;
                     168     ;                       ONE_BYTE_TAIL
                     169     ;                       MNEMONIC_FIRST_OPERAND
                     170     ;                       MNEMONIC_TWO_OPERANDS
                     171     ;                       MOVC_OPERANDS
                     172     ;                       THREE_OPERANDS
                     173     ;                       JUMP_OPERAND
                     174     ;                       JUMP_TWO_OPERANDS
                     175     ;                       JUMP_ABSOLUTE_OPERAND
                     176     ;                       JUMP_LONG_OPERAND
                     177     ;                       MNEMONIC_INSTRUCTION_TAIL
                     178     ;                       MNEMONIC_INSTR_LIST_TAIL
                     179     ;                       ASSEMBLY_CMD
                     180     ;
                     181     ;               ASM.INC (INCLUDE FILE)
                     182     ;
                     183     ;                       START_DIVIDE
                     184     ;                       CALCULATE_INSTRUCTION_VALUE
                     185     ;                       UPDATE_OUR_CODE
                     186     ;                       GET_FIRST_OPERAND
                     187     ;                       CHECK_AND_SET_EXP_FLAG/SET_EXP_16_FLAG/SET_EXP_FLAG/CHECK_EXP_FLAG
                     188     ;                               SET_POUND_FLAG/CHECK_AND_SET_SECOND_EXP_FLAG/SET_SLASH_EXP_
                         FLAG
                     189     ;                               SET_REL_FLAG/GET_SECOND_EXP
                     190     ;
                     191     ;               ASMA.INC (INCLUDE FILE)
                     192     ;
                     193     ;                       CHECK_AND_CHANGE_ASM_PC
                     194     ;                       CHANGE_TO_INSTRUCTION_OP
                     195     ;
                     196     ;               SDKDSM.INC (INCLUDE FILE)
                     197     ;
                     198     ;                       DISASSEMBLY_CMD
                     199     ;                       GET_HASH_VALUE
                     200     ;                       OPERAND_BYTE_CHECK
                     201     ;                       DISPLAY_OPERAND
                     202     ;                       DISPLAY_COMMA
                     203     ;                       DISASSEMBLE
```

```
LOC  OBJ            LINE     SOURCE

                     204     ;
                     205     ;*****************************************************************
                     206 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                     207     ;*****************************************************************************
                     208     ; *                                                                           *
                     209     ; * THIS MODULE CONTAINS THE TABLES USED TO IMPLEMENT ASSEMBLY AND            *
                     210     ; * DISASSEMBLY:                                                              *
                     211     ; *                                                                           *
                     212     ; *   INSTRUCTION$CODE - A table of 256 address entries, one per opcode.      *
                     213     ; *     Each entry codes up for its opcode the mnemonic, first operand and    *
                     214     ; *     second operand.  Specifically, the entry equals                       *
                     215     ; *       M + F*MNEMONIC$FACTOR + S*MNEMONIC$FACTOR*OPERAND$FACTOR            *
                     216     ; *     WHERE                                                                 *
                     217     ; *       M is the ordinal of the mnemonic in MNEMONIC$TAB,                   *
                     218     ; *       F is 0 if there are no operands; otherwise F is one more than the   *
                     219     ; *         ordinal of the first operand in the OPERAND$TAB, and              *
                     220     ; *       S is 0 if there is no second operand; otherwise S is one more than  *
                     221     ; *         the ordinal of the second operand in the OPERAND$TAB.             *
                     222     ; *     The entry OFFFFH in this table indicates the opcode is undefined.     *
                     223     ; *                                                                           *
                     224     ; *   MNEMONIC$TAB - A symbol table listing all the mnemonics (operands       *
                     225     ; *     not included).  The value associated with each is the instruction     *
                     226     ; *     format, a number between 7 and 15 corresponding to the instruction    *
                     227     ; *     tail in the grammar appropriate to the mnemonic.  The instruction     *
                     228     ; *     format is also needed to disassemble the instruction.  The formats    *
                     229     ; *     are:                                                                  *
                     230     ; *       7 - No operands (e.g. RETI)                                         *
                     231     ; *       8 - One operand (e.g. CLR A)                                        *
                     232     ; *       9 - Two operands (e.g. ADD A,R0)                                    *
                     233     ; *      10 - MOVC - Two operands (e.g. MOVC A,@A + DPTR)                     *
                     234     ; *      11 - CJNE - Three operands (e.g. CJNE @R0,#56H,42H)                  *
                     235     ; *      12 - JUMP - Relative - One operand (e.g. JC 44H)                     *
                     236     ; *      13 - JUMP - Relative - Two operands (e.g. JNB 5H,45H)                *
                     237     ; *      14 - Absolute CALL and JUMP (e.g. ACALL 341H)                        *
                     238     ; *      15 - Long CALL and JUMP (e.g. LJMP 4536H)                            *
                     239     ; *                                                                           *
                     240     ; *     The first mnemonics in this table are long call and jump(15), next    *
                     241     ; *     are the absolute call and jump instructions(14), then jump-relative   *
                     242     ; *     one-operand instructions(13), the CJNE three operand instructions     *
                     243     ; *     11), the MOVC instructions(10), the two operand instructions(9),      *
                     244     ; *     the jump-relative one-operand instructions(12), the one operand       *
                     245     ; *     instructions(8), and the no operand instructions(7).  The jump-       *
                     246     ; *     relative one-operand instructions are in between the two operand      *
                     247     ; *     instructions and the one operand instructions because in the action   *
                     248     ; *     SELECT$INSTRUCTION$TAIL it has to be determined if the mnemonic is    *
                     249     ; *     JNB, JB, JBC, SETB, CLR, or CPL since these six instructions, if they*
                     250     ; *     have an expression, have a bit expression so BIT$EXP must be set.     *
                     251     ; *                                                                           *
                     252     ; *   OPERAND$TAB - A symbol table listing the operands.  No value is         *
                     253     ; *     associated with them.  Only the ordinal in the table is important.    *
                     254     ; *                                                                           *
                     255     ; *****************************************************************************
                     256
                     257     ;
                     258     ;
                     259     ;       DECLARE
                     260     ;       UNDEF LIT 'OFFFFH';
                     261     ;
```

```
LOC  OBJ            LINE     SOURCE

                     262     ;  DECLARE
                     263     ;    MNEMONIC$TAB$HEAD TABLE PUBLIC DATA(
                     264     ;      .MNEMONIC$TAB+0EDH,
                     265     ;      OFFFFH - 0EDH),
                     266     ;
  FFFF               267     MNE_UNDEF       EQU     OFFFFH
                     268
                     269     MNEMONIC_TAB:           ;(*) BYTE PUBLIC DATA(
                     270
F529 0F              271             DB      0FH     ; LCALL
F52A 0F              272             DB      0FH     ; LJMP
F52B 0E              273             DB      0EH     ; ACALL
F52C 0E              274             DB      0EH     ; AJMP
F52D 0C              275             DB      0CH     ; SJMP
F52E 0C              276             DB      0CH     ; JNZ
F52F 0C              277             DB      0CH     ; JZ
F530 0C              278             DB      0CH     ; JNC
F531 0C              279             DB      0CH     ; JC
F532 0B              280             DB      0BH     ; CJNE
F533 0A              281             DB      0AH     ; MOVC
F534 09              282             DB      09H     ; MOVX
F535 09              283             DB      09H     ; XCHD
F536 09              284             DB      09H     ; XCH
F537 09              285             DB      09H     ; SUBB
F538 09              286             DB      09H     ; MOV
F539 09              287             DB      09H     ; XRL
F53A 09              288             DB      09H     ; ANL
F53B 09              289             DB      09H     ; ORL
F53C 09              290             DB      09H     ; ADDC
F53D 09              291             DB      09H     ; ADD
F53E 0D              292             DB      0DH     ; DJNZ
F53F 0D              293             DB      0DH     ; JNB
F540 0D              294             DB      0DH     ; JB
F541 0D              295             DB      0DH     ; JBC
F542 08              296             DB      08H     ; SETB
F543 08              297             DB      08H     ; CLR
F544 08              298             DB      08H     ; CPL
F545 08              299             DB      08H     ; DA
F546 08              300             DB      08H     ; POP
F547 08              301             DB      08H     ; SWAP
F548 08              302             DB      08H     ; PUSH
F549 08              303             DB      08H     ; MUL
F54A 08              304             DB      08H     ; DIV
F54B 08              305             DB      08H     ; JMP(@A+DPTR)
F54C 08              306             DB      08H     ; RLC
F54D 08              307             DB      08H     ; RL
F54E 08              308             DB      08H     ; DEC
F54F 08              309             DB      08H     ; RRC
F550 08              310             DB      08H     ; INC
F551 08              311             DB      08H     ; RR
F552 07              312             DB      07H     ; RETI
F553 07              313             DB      07H     ; RET
F554 07              314             DB      07H     ; NOP
                     315
                     316     ;  DECLARE   ; ORDINALS OF MNEMONICS IN MNEMONIC$TAB
```

```
LOC  OBJ            LINE     SOURCE

                     317
                     318
  0000               319     MNE_LCALL       EQU     00
  0001               320     MNE_LJMP        EQU     01
  0002               321     MNE_ACALL       EQU     02
  0003               322     MNE_AJMP        EQU     03
  0004               323     MNE_SJMP        EQU     04
  0005               324     MNE_JNZ         EQU     05
  0006               325     MNE_JZ          EQU     06
  0007               326     MNE_JNC         EQU     07
  0008               327     MNE_JC          EQU     08
  0009               328     MNE_CJNE        EQU     09
  000A               329     MNE_MOVC        EQU     10
  000B               330     MNE_MOVX        EQU     11
  000C               331     MNE_XCHD        EQU     12
  000D               332     MNE_XCH         EQU     13
  000E               333     MNE_SUBB        EQU     14
  000F               334     MNE_MOV         EQU     15
  0010               335     MNE_XRL         EQU     16
  0011               336     MNE_ANL         EQU     17
  0012               337     MNE_ORL         EQU     18
  0013               338     MNE_ADDC        EQU     19
  0014               339     MNE_ADD         EQU     20
  0015               340     MNE_DJNZ        EQU     21
  0016               341     MNE_JNB         EQU     22
  0017               342     MNE_JB          EQU     23
  0018               343     MNE_JBC         EQU     24
  0019               344     MNE_SETB        EQU     25
  001A               345     MNE_CLR         EQU     26
  001B               346     MNE_CPL         EQU     27
  001C               347     MNE_DA          EQU     28
  001D               348     MNE_POP         EQU     29
  001E               349     MNE_SWAP        EQU     30
  001F               350     MNE_PUSH        EQU     31
  0020               351     MNE_MUL         EQU     32
  0021               352     MNE_DIV         EQU     33
  0022               353     MNE_JMP         EQU     34
  0023               354     MNE_RLC         EQU     35
  0024               355     MNE_RL          EQU     36
  0025               356     MNE_DEC         EQU     37
  0026               357     MNE_RRC         EQU     38
  0027               358     MNE_INC         EQU     39
  0028               359     MNE_RR          EQU     40
  0029               360     MNE_RETI        EQU     41
  002A               361     MNE_RET         EQU     42
  002B               362     MNE_NOP         EQU     43;
                     363     ;*********************************************************************
                     364     ; DECLARE   ; MNEMONIC_FACTOR (I.E. 44) TIMES ORDINAL+1 OF FIRST OPERANDS IN
                     365     ;             OPERAND_TAB.
                     366
  002C               367     A_OP1           EQU     0044
  0058               368     ATR0_OP1        EQU     0088
  0084               369     ATR1_OP1        EQU     0132
  00B0               370     R0_OP1          EQU     0176
  00DC               371     R1_OP1          EQU     0220
```

```
LOC  OBJ            LINE     SOURCE

  0108               372     R2_OP1           EQU     0264
  0134               373     R3_OP1           EQU     0308
  0160               374     R4_OP1           EQU     0352
  018C               375     R5_OP1           EQU     0396
  01B8               376     R6_OP1           EQU     0440
  01E4               377     R7_OP1           EQU     0484
  0210               378     AB_OP1           EQU     0528
  023C               379     DPTR_OP1         EQU     0572
  0268               380     C_OP1            EQU     0616
  0294               381     ATDPTR_OP1       EQU     660
  02C0               382     BYTE_EXP8_OP1    EQU     0704
  02EC               383     BIT_EXP8_OP1     EQU     0748
  0370               384     EXP16_OP1        EQU     0880
  039C               385     EXP11_OP1        EQU     0924
  03C8               386     REL8_OP1         EQU     0968
  03F4               387     ATA_PLUS_DPTR_OP1        EQU     1012;
                     388     ;DECLARE;OPERAND_FACTOR*MNEMONIC_FACTOR(I.E.1056)TIMESORDINALSOF
                     389     ;SECONDOPERANDSINOPERAND_TAB
                     390
  0420               391     A_OP2            EQU     01056
  0840               392     ATR0_OP2         EQU     02112
  0C60               393     ATR1_OP2         EQU     03168
  1080               394     R0_OP2           EQU     04224
  14A0               395     R1_OP2           EQU     05280
  18C0               396     R2_OP2           EQU     06336
  1CE0               397     R3_OP2           EQU     07392
  2100               398     R4_OP2           EQU     08448
  2520               399     R5_OP2           EQU     09504
  2940               400     R6_OP2           EQU     10560
  2D60               401     R7_OP2           EQU     11616
  39C0               402     C_OP2            EQU     14784
  3DE0               403     ATDPTR_OP2       EQU     15840
  4200               404     BYTE_EXP8_OP2    EQU     16896
  4620               405     BIT_EXP8_OP2     EQU     17952
  4A40               406     POUND_EXP_OP2    EQU     19008
  4E60               407     SLASH_EXP_OP2    EQU     20064
  5280               408     EXP16_OP2        EQU     21120
  5AC0               409     REL8_OP2         EQU     23232
  5EE0               410     ATA_PLUS_DPTR_OP2        EQU     24288
  6300               411     ATA_PLUS_PC_OP2 EQU      25344;
                     412 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                     413     ;********** TEMPORARY VARIABLES **********
                     414     ;***** DATA ADDRESSES *****
                     415
----                 416             DSEG
005B                 417             ORG     (PARTIT_HI_LOW+1)
005B                 418     INSTRUCTION_VALUE:      DS      1
005C                 419     ORDINAL:                DS      1
005D                 420     OLD_ASM_PC_HIGH:        DS      1
005E                 421     OLD_ASM_PC_LOW:         DS      1
005F                 422     INSTRUCTION:            DS      1
0060                 423     REL_OFFSET_HIGH:        DS      1
0061                 424     REL_OFFSET_LOW:         DS      1
0062                 425     TEMP_SEC:               DS      1
0063                 426     FIRST_OPER_ORDINAL:     DS      1
0064                 427     SECOND_OPER_ORDINAL:    DS      1
0065                 428     THIRD_OPER_ORDINAL:     DS      1
0066                 429     CURRENT_OPERAND:        DS      1
0067                 430     NO_OF_OPERANDS_PRINTED: DS      1
0068                 431     EXPRESSIONS_PRINTED:    DS      1
0069                 432     MEMORY_TRACE_ADDR_HIGH: DS      1
006A                 433     MEMORY_TRACE_ADDR_LOW:  DS      1
006B                 434     NUMBER_OF_OPERANDS:     DS      1
006C                 435     OPERAND_CHECK:          DS      1
006D                 436     MNEMONIC_ORDINAL:       DS      1
006E                 437     DIVIDEND_HIGH:  DS      1
006F                 438     DIVIDEND_LOW:   DS      1
0070                 439     DIVISOR:        DS      1
0071                 440     QUOTIENT_HIGH:  DS      1
0072                 441     QUOTIENT_LOW:   DS      1
                     442
                     443
                     444     ;***** FLAG ADDRESSES *****
                     445
----                 446             BSEG
0002                 447             ORG     (LSTFLG+1)
0002                 448     BIT_EXP:                DBIT    1
0003                 449     FIRST_EXP:              DBIT    1
0004                 450     SECOND_EXP:             DBIT    1
----                 451             CSEG
                     452
                     453     ;***** CONSTANTS *****
  0016               454     JUMP_END                EQU     22
  001B               455     BIT_END                 EQU     27
  002C               456     MNEMONIC_FACTOR         EQU     44
  0018               457     OPERAND_FACTOR          EQU     24
  00A5               458     UNDEFINED_OPCODE        EQU     0A5H
                     459
                     460     INSTRUCTION_CODE:                               ;Hash Table
                     461     ;00
F555 002B            462             DW      MNE_NOP
F557 039F            463             DW      MNE_AJMP+EXP11_OP1
F559 0371            464             DW      MNE_LJMP+EXP16_OP1
F55B 0054            465             DW      MNE_RR+A_OP1
                     466     ;04
F55D 0053            467             DW      MNE_INC+A_OP1
```

```
LOC  OBJ            LINE     SOURCE

F55F 02E7            468            DW      MNE_INC+BYTE_EXP8_OP1
F561 007F            469            DW      MNE_INC+ATR0_OP1
F563 00AB            470            DW      MNE_INC+ATR1_OP1
                     471     ;08
F565 00D7            472            DW      MNE_INC+R0_OP1
F567 0103            473            DW      MNE_INC+R1_OP1
F569 012F            474            DW      MNE_INC+R2_OP1
F56B 015B            475            DW      MNE_INC+R3_OP1
                     476     ;0C
F56D 0187            477            DW      MNE_INC+R4_OP1
F56F 01B3            478            DW      MNE_INC+R5_OP1
F571 01DF            479            DW      MNE_INC+R6_OP1
F573 020B            480            DW      MNE_INC+R7_OP1
                     481     ;10
F575 5DC4            482            DW      MNE_JBC+BIT_EXP8_OP1+REL8_OP2
F577 039E            483            DW      MNE_ACALL+EXP11_OP1
F579 0370            484            DW      MNE_LCALL+EXP16_OP1
F57B 0052            485            DW      MNE_RRC+A_OP1
                     486     ;14
F57D 0051            487            DW      MNE_DEC+A_OP1
F57F 02E5            488            DW      MNE_DEC+BYTE_EXP8_OP1
F581 007D            489            DW      MNE_DEC+ATR0_OP1
F583 00A9            490            DW      MNE_DEC+ATR1_OP1
                     491     ;18
F585 00D5            492            DW      MNE_DEC+R0_OP1
F587 0101            493            DW      MNE_DEC+R1_OP1
F589 012D            494            DW      MNE_DEC+R2_OP1
F58B 0159            495            DW      MNE_DEC+R3_OP1
                     496     ;1C
F58D 0185            497            DW      MNE_DEC+R4_OP1
F58F 01B1            498            DW      MNE_DEC+R5_OP1
F591 01DD            499            DW      MNE_DEC+R6_OP1
F593 0209            500            DW      MNE_DEC+R7_OP1
                     501     ;20
F595 5DC3            502            DW      MNE_JB+BIT_EXP8_OP1+REL8_OP2
F597 039F            503            DW      MNE_AJMP+EXP11_OP1
F599 002A            504            DW      MNE_RET
F59B 0050            505            DW      MNE_RL+A_OP1
                     506     ;24
F59D 4A80            507            DW      MNE_ADD+A_OP1+POUND_EXP_OP2
F59F 4240            508            DW      MNE_ADD+A_OP1+BYTE_EXP8_OP2
F5A1 0880            509            DW      MNE_ADD+A_OP1+ATR0_OP2
F5A3 0CA0            510            DW      MNE_ADD+A_OP1+ATR1_OP2
                     511     ;28
F5A5 10C0            512            DW      MNE_ADD+A_OP1+R0_OP2
F5A7 14E0            513            DW      MNE_ADD+A_OP1+R1_OP2
F5A9 1900            514            DW      MNE_ADD+A_OP1+R2_OP2
F5AB 1D20            515            DW      MNE_ADD+A_OP1+R3_OP2
                     516     ;2C
F5AD 2140            517            DW      MNE_ADD+A_OP1+R4_OP2
F5AF 2560            518            DW      MNE_ADD+A_OP1+R5_OP2
F5B1 2980            519            DW      MNE_ADD+A_OP1+R6_OP2
F5B3 2DA0            520            DW      MNE_ADD+A_OP1+R7_OP2
                     521     ;30
F5B5 5DC2            522            DW      MNE_JNB+BIT_EXP8_OP1+REL8_OP2
```

```
LOC  OBJ            LINE     SOURCE

F5B7 039E            523            DW      MNE_ACALL+EXP11_OP1
F5B9 0029            524            DW      MNE_RETI
F5BB 004F            525            DW      MNE_RLC+A_OP1
                     526     ;34
F5BD 4A7F            527            DW      MNE_ADDC+A_OP1+POUND_EXP_OP2
F5BF 423F            528            DW      MNE_ADDC+A_OP1+BYTE_EXP8_OP2
F5C1 087F            529            DW      MNE_ADDC+A_OP1+ATR0_OP2
F5C3 0C9F            530            DW      MNE_ADDC+A_OP1+ATR1_OP2
                     531     ;38
F5C5 10BF            532            DW      MNE_ADDC+A_OP1+R0_OP2
F5C7 14DF            533            DW      MNE_ADDC+A_OP1+R1_OP2
F5C9 18FF            534            DW      MNE_ADDC+A_OP1+R2_OP2
F5CB 1D1F            535            DW      MNE_ADDC+A_OP1+R3_OP2
                     536     ;3C
F5CD 213F            537            DW      MNE_ADDC+A_OP1+R4_OP2
F5CF 255F            538            DW      MNE_ADDC+A_OP1+R5_OP2
F5D1 297F            539            DW      MNE_ADDC+A_OP1+R6_OP2
F5D3 2D9F            540            DW      MNE_ADDC+A_OP1+R7_OP2
                     541     ;40
F5D5 03D0            542            DW      MNE_JC+REL8_OP1
F5D7 039F            543            DW      MNE_AJMP+EXP11_OP1
F5D9 06F2            544            DW      MNE_ORL+BYTE_EXP8_OP1+A_OP2
F5DB 4D12            545            DW      MNE_ORL+BYTE_EXP8_OP1+POUND_EXP_OP2
                     546     ;44
F5DD 4A7E            547            DW      MNE_ORL+A_OP1+POUND_EXP_OP2
F5DF 423E            548            DW      MNE_ORL+A_OP1+BYTE_EXP8_OP2
F5E1 087E            549            DW      MNE_ORL+A_OP1+ATR0_OP2
F5E3 0C9E            550            DW      MNE_ORL+A_OP1+ATR1_OP2
                     551     ;48
F5E5 10BE            552            DW      MNE_ORL+A_OP1+R0_OP2
F5E7 14DE            553            DW      MNE_ORL+A_OP1+R1_OP2
F5E9 18FE            554            DW      MNE_ORL+A_OP1+R2_OP2
F5EB 1D1E            555            DW      MNE_ORL+A_OP1+R3_OP2
                     556     ;4C
F5ED 213E            557            DW      MNE_ORL+A_OP1+R4_OP2
F5EF 255E            558            DW      MNE_ORL+A_OP1+R5_OP2
F5F1 297E            559            DW      MNE_ORL+A_OP1+R6_OP2
F5F3 2D9E            560            DW      MNE_ORL+A_OP1+R7_OP2
                     561     ;50
F5F5 03CF            562            DW      MNE_JNC+REL8_OP1
F5F7 039E            563            DW      MNE_ACALL+EXP11_OP1
F5F9 06F1            564            DW      MNE_ANL+BYTE_EXP8_OP1+A_OP2
F5FB 4D11            565            DW      MNE_ANL+BYTE_EXP8_OP1+POUND_EXP_OP2
                     566     ;54
F5FD 4A7D            567            DW      MNE_ANL+A_OP1+POUND_EXP_OP2
F5FF 423D            568            DW      MNE_ANL+A_OP1+BYTE_EXP8_OP2
F601 087D            569            DW      MNE_ANL+A_OP1+ATR0_OP2
F603 0C9D            570            DW      MNE_ANL+A_OP1+ATR1_OP2
                     571     ;58
F605 10BD            572            DW      MNE_ANL+A_OP1+R0_OP2
F607 14DD            573            DW      MNE_ANL+A_OP1+R1_OP2
F609 18FD            574            DW      MNE_ANL+A_OP1+R2_OP2
F60B 1D1D            575            DW      MNE_ANL+A_OP1+R3_OP2
                     576     ;5C
F60D 213D            577            DW      MNE_ANL+A_OP1+R4_OP2
```

```
LOC  OBJ            LINE     SOURCE

F60F 255D            578              DW      MNE_ANL+A_OP1+R5_OP2
F611 297D            579              DW      MNE_ANL+A_OP1+R6_OP2
F613 2D9D            580              DW      MNE_ANL+A_OP1+R7_OP2
                     581      ;60
F615 03CE            582              DW      MNE_JZ+REL8_OP1
F617 039F            583              DW      MNE_AJMP+EXP11_OP1
F619 06F0            584              DW      MNE_XRL+BYTE_EXP8_OP1+A_OP2
F61B 4D10            585              DW      MNE_XRL+BYTE_EXP8_OP1+POUND_EXP_OP2
                     586      ;64
F61D 4A7C            587              DW      MNE_XRL+A_OP1+POUND_EXP_OP2
F61F 423C            588              DW      MNE_XRL+A_OP1+BYTE_EXP8_OP2
F621 087C            589              DW      MNE_XRL+A_OP1+ATR0_OP2
F623 0C9C            590              DW      MNE_XRL+A_OP1+ATR1_OP2
                     591      ;68
F625 10BC            592              DW      MNE_XRL+A_OP1+R0_OP2
F627 14DC            593              DW      MNE_XRL+A_OP1+R1_OP2
F629 18FC            594              DW      MNE_XRL+A_OP1+R2_OP2
F62B 1D1C            595              DW      MNE_XRL+A_OP1+R3_OP2
                     596      ;6C
F62D 213C            597              DW      MNE_XRL+A_OP1+R4_OP2
F62F 255C            598              DW      MNE_XRL+A_OP1+R5_OP2
F631 297C            599              DW      MNE_XRL+A_OP1+R6_OP2
F633 2D9C            600              DW      MNE_XRL+A_OP1+R7_OP2
                     601      ;70
F635 03CD            602              DW      MNE_JNZ+REL8_OP1
F637 039E            603              DW      MNE_ACALL+EXP11_OP1
F639 489A            604              DW      MNE_ORL+C_OP1+BIT_EXP8_OP2
F63B 0416            605              DW      MNE_JMP+ATA_PLUS_DPTR_OP1
                     606      ;74
F63D 4A7B            607              DW      MNE_MOV+A_OP1+POUND_EXP_OP2
F63F 4D0F            608              DW      MNE_MOV+BYTE_EXP8_OP1+POUND_EXP_OP2
F641 4AA7            609              DW      MNE_MOV+ATR0_OP1+POUND_EXP_OP2
F643 4AD3            610              DW      MNE_MOV+ATR1_OP1+POUND_EXP_OP2
                     611      ;78
F645 4AFF            612              DW      MNE_MOV+R0_OP1+POUND_EXP_OP2
F647 4B2B            613              DW      MNE_MOV+R1_OP1+POUND_EXP_OP2
F649 4B57            614              DW      MNE_MOV+R2_OP1+POUND_EXP_OP2
F64B 4B83            615              DW      MNE_MOV+R3_OP1+POUND_EXP_OP2
                     616      ;7C
F64D 4BAF            617              DW      MNE_MOV+R4_OP1+POUND_EXP_OP2
F64F 4BDB            618              DW      MNE_MOV+R5_OP1+POUND_EXP_OP2
F651 4C07            619              DW      MNE_MOV+R6_OP1+POUND_EXP_OP2
F653 4C33            620              DW      MNE_MOV+R7_OP1+POUND_EXP_OP2
                     621      ;80
F655 03CC            622              DW      MNE_SJMP+REL8_OP1
F657 039F            623              DW      MNE_AJMP+EXP11_OP1
F659 4899            624              DW      MNE_ANL+C_OP1+BIT_EXP8_OP2
F65B 6336            625              DW      MNE_MOVC+A_OP1+ATA_PLUS_PC_OP2
                     626      ;84
F65D 0231            627              DW      MNE_DIV+AB_OP1
F65F 44CF            628              DW      MNE_MOV+BYTE_EXP8_OP1+BYTE_EXP8_OP2
F661 0B0F            629              DW      MNE_MOV+BYTE_EXP8_OP1+ATR0_OP2
F663 0F2F            630              DW      MNE_MOV+BYTE_EXP8_OP1+ATR1_OP2
                     631      ;88
F665 134F            632              DW      MNE_MOV+BYTE_EXP8_OP1+R0_OP2
```

```
LOC  OBJ            LINE     SOURCE

F667 176F            633              DW      MNE_MOV+BYTE_EXP8_OP1+R1_OP2
F669 1B8F            634              DW      MNE_MOV+BYTE_EXP8_OP1+R2_OP2
F66B 1FAF            635              DW      MNE_MOV+BYTE_EXP8_OP1+R3_OP2
                     636      ;8C
F66D 23CF            637              DW      MNE_MOV+BYTE_EXP8_OP1+R4_OP2
F66F 27EF            638              DW      MNE_MOV+BYTE_EXP8_OP1+R5_OP2
F671 2C0F            639              DW      MNE_MOV+BYTE_EXP8_OP1+R6_OP2
F673 302F            640              DW      MNE_MOV+BYTE_EXP8_OP1+R7_OP2
                     641      ;90
F675 54CB            642              DW      MNE_MOV+DPTR_OP1+EXP16_OP2
F677 039E            643              DW      MNE_ACALL+EXP11_OP1
F679 3CBB            644              DW      MNE_MOV+BIT_EXP8_OP1+C_OP2
F67B 5F16            645              DW      MNE_MOVC+A_OP1+ATA_PLUS_DPTR_OP2
                     646      ;94
F67D 4A7A            647              DW      MNE_SUBB+A_OP1+POUND_EXP_OP2
F67F 423A            648              DW      MNE_SUBB+A_OP1+BYTE_EXP8_OP2
F681 087A            649              DW      MNE_SUBB+A_OP1+ATR0_OP2
F683 0C9A            650              DW      MNE_SUBB+A_OP1+ATR1_OP2
                     651      ;98
F685 10BA            652              DW      MNE_SUBB+A_OP1+R0_OP2
F687 14DA            653              DW      MNE_SUBB+A_OP1+R1_OP2
F689 18FA            654              DW      MNE_SUBB+A_OP1+R2_OP2
F68B 1D1A            655              DW      MNE_SUBB+A_OP1+R3_OP2
                     656      ;9C
F68D 213A            657              DW      MNE_SUBB+A_OP1+R4_OP2
F68F 255A            658              DW      MNE_SUBB+A_OP1+R5_OP2
F691 297A            659              DW      MNE_SUBB+A_OP1+R6_OP2
F693 2D9A            660              DW      MNE_SUBB+A_OP1+R7_OP2
                     661      ;A0
F695 50DA            662              DW      MNE_ORL+C_OP1+SLASH_EXP_OP2
F697 039F            663              DW      MNE_AJMP+EXP11_OP1
F699 4897            664              DW      MNE_MOV+C_OP1+BIT_EXP8_OP2
F69B 0263            665              DW      MNE_INC+DPTR_OP1
                     666      ;A4
F69D 0230            667              DW      MNE_MUL+AB_OP1
F69F FFFF            668              DW      MNE_UNDEF
F6A1 4267            669              DW      MNE_MOV+ATR0_OP1+BYTE_EXP8_OP2
F6A3 4293            670              DW      MNE_MOV+ATR1_OP1+BYTE_EXP8_OP2
                     671      ;A8
F6A5 42BF            672              DW      MNE_MOV+R0_OP1+BYTE_EXP8_OP2
F6A7 42EB            673              DW      MNE_MOV+R1_OP1+BYTE_EXP8_OP2
F6A9 4317            674              DW      MNE_MOV+R2_OP1+BYTE_EXP8_OP2
F6AB 4343            675              DW      MNE_MOV+R3_OP1+BYTE_EXP8_OP2
                     676      ;AC
F6AD 436F            677              DW      MNE_MOV+R4_OP1+BYTE_EXP8_OP2
F6AF 439B            678              DW      MNE_MOV+R5_OP1+BYTE_EXP8_OP2
F6B1 43C7            679              DW      MNE_MOV+R6_OP1+BYTE_EXP8_OP2
F6B3 43F3            680              DW      MNE_MOV+R7_OP1+BYTE_EXP8_OP2
                     681      ;B0
F6B5 50D9            682              DW      MNE_ANL+C_OP1+SLASH_EXP_OP2
F6B7 039E            683              DW      MNE_ACALL+EXP11_OP1
F6B9 0307            684              DW      MNE_CPL+BIT_EXP8_OP1
F6BB 0283            685              DW      MNE_CPL+C_OP1
                     686      ;B4
F6BD 4A75            687              DW      MNE_CJNE+A_OP1+POUND_EXP_OP2
```

```
LOC  OBJ            LINE     SOURCE

F6BF 4235            688              DW      MNE_CJNE+A_OP1+BYTE_EXP8_OP2
F6C1 4AA1            689              DW      MNE_CJNE+ATR0_OP1+POUND_EXP_OP2
F6C3 4ACD            690              DW      MNE_CJNE+ATR1_OP1+POUND_EXP_OP2
                     691      ;B8
F6C5 4AF9            692              DW      MNE_CJNE+R0_OP1+POUND_EXP_OP2
F6C7 4B25            693              DW      MNE_CJNE+R1_OP1+POUND_EXP_OP2
F6C9 4B51            694              DW      MNE_CJNE+R2_OP1+POUND_EXP_OP2
F6CB 4B7D            695              DW      MNE_CJNE+R3_OP1+POUND_EXP_OP2
                     696      ;BC
F6CD 4BA9            697              DW      MNE_CJNE+R4_OP1+POUND_EXP_OP2
F6CF 4BD5            698              DW      MNE_CJNE+R5_OP1+POUND_EXP_OP2
F6D1 4C01            699              DW      MNE_CJNE+R6_OP1+POUND_EXP_OP2
F6D3 4C2D            700              DW      MNE_CJNE+R7_OP1+POUND_EXP_OP2
                     701      ;C0
F6D5 02DF            702              DW      MNE_PUSH+BYTE_EXP8_OP1
F6D7 039F            703              DW      MNE_AJMP+EXP11_OP1
F6D9 0306            704              DW      MNE_CLR+BIT_EXP8_OP1
F6DB 0282            705              DW      MNE_CLR+C_OP1
                     706      ;C4
F6DD 004A            707              DW      MNE_SWAP+A_OP1
F6DF 4239            708              DW      MNE_XCH+A_OP1+BYTE_EXP8_OP2
F6E1 0879            709              DW      MNE_XCH+A_OP1+ATR0_OP2
F6E3 0C99            710              DW      MNE_XCH+A_OP1+ATR1_OP2
                     711      ;C8
F6E5 10B9            712              DW      MNE_XCH+A_OP1+R0_OP2
F6E7 14D9            713              DW      MNE_XCH+A_OP1+R1_OP2
F6E9 18F9            714              DW      MNE_XCH+A_OP1+R2_OP2
F6EB 1D19            715              DW      MNE_XCH+A_OP1+R3_OP2
                     716      ;CC
F6ED 2139            717              DW      MNE_XCH+A_OP1+R4_OP2
F6EF 2559            718              DW      MNE_XCH+A_OP1+R5_OP2
F6F1 2979            719              DW      MNE_XCH+A_OP1+R6_OP2
F6F3 2D99            720              DW      MNE_XCH+A_OP1+R7_OP2
                     721      ;D0
F6F5 02DD            722              DW      MNE_POP+BYTE_EXP8_OP1
F6F7 039E            723              DW      MNE_ACALL+EXP11_OP1
F6F9 0305            724              DW      MNE_SETB+BIT_EXP8_OP1
F6FB 0281            725              DW      MNE_SETB+C_OP1
                     726      ;D4
F6FD 0048            727              DW      MNE_DA+A_OP1
F6FF 5D95            728              DW      MNE_DJNZ+BYTE_EXP8_OP1+REL8_OP2
F701 0878            729              DW      MNE_XCHD+A_OP1+ATR0_OP2
F703 0C98            730              DW      MNE_XCHD+A_OP1+ATR1_OP2
                     731      ;D8
F705 5B85            732              DW      MNE_DJNZ+R0_OP1+REL8_OP2
F707 5BB1            733              DW      MNE_DJNZ+R1_OP1+REL8_OP2
F709 5BDD            734              DW      MNE_DJNZ+R2_OP1+REL8_OP2
F70B 5C09            735              DW      MNE_DJNZ+R3_OP1+REL8_OP2
                     736      ;DC
F70D 5C35            737              DW      MNE_DJNZ+R4_OP1+REL8_OP2
F70F 5C61            738              DW      MNE_DJNZ+R5_OP1+REL8_OP2
F711 5C8D            739              DW      MNE_DJNZ+R6_OP1+REL8_OP2
F713 5CB9            740              DW      MNE_DJNZ+R7_OP1+REL8_OP2
                     741      ;E0
F715 3E17            742              DW      MNE_MOVX+A_OP1+ATDPTR_OP2
```

```
LOC  OBJ            LINE     SOURCE

F717 039F            743              DW      MNE_AJMP+EXP11_OP1
F719 0877            744              DW      MNE_MOVX+A_OP1+ATRO_OP2
F71B 0C97            745              DW      MNE_MOVX+A_OP1+ATR1_OP2
                     746      ;E4
F71D 0046            747              DW      MNE_CLR+A_OP1
F71F 423B            748              DW      MNE_MOV+A_OP1+BYTE_EXP8_OP2
F721 087B            749              DW      MNE_MOV+A_OP1+ATRO_OP2
F723 0C9B            750              DW      MNE_MOV+A_OP1+ATR1_OP2
                     751      ;E8
F725 10BB            752              DW      MNE_MOV+A_OP1+RO_OP2
F727 14DB            753              DW      MNE_MOV+A_OP1+R1_OP2
F729 18FB            754              DW      MNE_MOV+A_OP1+R2_OP2
F72B 1D1B            755              DW      MNE_MOV+A_OP1+R3_OP2
                     756      ;EC
F72D 213B            757              DW      MNE_MOV+A_OP1+R4_OP2
F72F 255B            758              DW      MNE_MOV+A_OP1+R5_OP2
F731 297B            759              DW      MNE_MOV+A_OP1+R6_OP2
F733 2D9B            760              DW      MNE_MOV+A_OP1+R7_OP2
                     761      ;F0
F735 06BF            762              DW      MNE_MOVX+ATDPTR_OP1+A_OP2
F737 039E            763              DW      MNE_ACALL+EXP11_OP1
F739 0483            764              DW      MNE_MOVX+ATRO_OP1+A_OP2
F73B 04AF            765              DW      MNE_MOVX+ATR1_OP1+A_OP2
                     766      ;F4
F73D 0047            767              DW      MNE_CPL+A_OP1
F73F 06EF            768              DW      MNE_MOV+BYTE_EXP8_OP1+A_OP2
F741 0487            769              DW      MNE_MOV+ATRO_OP1+A_OP2
F743 04B3            770              DW      MNE_MOV+ATR1_OP1+A_OP2
                     771      ;F8
F745 04DF            772              DW      MNE_MOV+RO_OP1+A_OP2
F747 050B            773              DW      MNE_MOV+R1_OP1+A_OP2
F749 0537            774              DW      MNE_MOV+R2_OP1+A_OP2
F74B 0563            775              DW      MNE_MOV+R3_OP1+A_OP2
                     776      ;FC
F74D 058F            777              DW      MNE_MOV+R4_OP1+A_OP2
F74F 05BB            778              DW      MNE_MOV+R5_OP1+A_OP2
F751 05E7            779              DW      MNE_MOV+R6_OP1+A_OP2
F753 0613            780              DW      MNE_MOV+R7_OP1+A_OP2;
                     781      ;*****************************************************************
                     782 +1   $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                     783     ;*****************************************************************
                     784     ;
                     785     ;    NAME: ONE_BYTE_TAIL/ MNEMONIC_SECOND_OPERAND_TAIL
                     786     ;
                     787     ;    ABSTRACT: This routine finds the opcode in the hash table which
                     788     ;       matches the token entered and sets the NUMBER_OF_BYTES according
                     789     ;       to the expression flags.  These are all one byte instructions
                     790     ;       regardless of actual NUMBER_OF_BYTES setting. Opcodes include
                     791     ;       NOP, RET etc..
                     792     ;
                     793     ;    INPUTS: None
                     794     ;
                     795     ;    OUTPUTS: OUR_CODE_LOW, OUR_CODE_HIGH
                     796     ;
                     797     ;    VARIABLES MODIFIED: None
                     798     ;
                     799     ;    ERROR EXITS: None
                     800     ;
                     801     ;    SUBROUTINES ACCESSED DIRECTLY: CALCULATE_INSTRUCTION_VALUE,
                     802     ;       CHECK_EXP_FLAG
                     803     ;
                     804     ;*****************************************************************
                     805     ONE_BYTE_TAIL:
                     806     MNEMONIC_SECOND_OPERAND_TAIL:
F755 12F99F          807             CALL    CALCULATE_INSTRUCTION_VALUE
F758 02FA68          808             JMP     CHECK_EXP_FLAG
                     809 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                     810     ;*****************************************************************
                     811     ;
                     812     ;    NAME: MNEMONIC_FIRST_OPERAND
                     813     ;
                     814     ;    ABSTRACT: This routine sets flags to indicate how to assemble
                     815     ;       one byte instructions with one operand.  It gets a hash
                     816     ;       value and passes the expression or expressions to run time
                     817     ;       routines.  Instructions include: CLR A, INC A, JMP @A+DPTR,
                     818     ;       etc.
                     819     ;
                     820     ;    INPUTS: None
                     821     ;
                     822     ;    OUTPUTS: NUMBER_OF_BYTES, ORDINAL, OUR_CODE_HIGH, OUR_CODE_LOW,
                     823     ;       VALLOW
                     824     ;
                     825     ;    VARIABLES MODIFIED: A, ORDINAL, NUMBER_OF_BYTES
                     826     ;
                     827     ;    ERROR EXITS: 10H (ASSEMBLY SYNTAX ERROR)
                     828     ;
                     829     ;    SUBROUTINES ACCESSED DIRECTLY: GETOKE, ONE_BYTE_TAIL,
                     830     ;       UPDATE_OUR_CODE, CALCULATE_INSTRUCTION_VALUE,
                     831     ;       GET_FIRST_OPERAND,  CHECK_AND_SET_EXP_FLAG
                     832     ;
                     833     ;*****************************************************************
                     834     MNEMONIC_FIRST_OPERAND:
F75B 12E056          835             CALL    GETOKE
F75E B40A14          836             CJNE    A,#ATA_TOKE,MF00                ;Check for @A+DPTR
F761 12E056          837             CALL    GETOKE
F764 B4056D          838             CJNE    A,#PLUS_TOKE,ASERR
F767 12E056          839             CALL    GETOKE
F76A B4A167          840             CJNE    A,#DPTR_TOKE,ASERR
F76D 755C17          841             MOV     ORDINAL,#17H
F770 12F9C7          842             CALL    UPDATE_OUR_CODE
F773 80E0            843             JMP     ONE_BYTE_TAIL
F775 300005          844     MF00:   JNB     B_0_T,MF01
F778 12FA02          845             CALL    GET_FIRST_OPERAND
F77B 80D8            846             JMP     ONE_BYTE_TAIL
F77D B4A10D          847     MF01:   CJNE    A,#DPTR_TOKE,MF02
F780 755C0D          848             MOV     ORDINAL,#0DH
F783 12F9C7          849             CALL    UPDATE_OUR_CODE
F786 12F99F          850             CALL    CALCULATE_INSTRUCTION_VALUE
F789 754D01          851             MOV     NUMBER_OF_BYTES,#01H
F78C 22              852             RET
F78D B40144          853     MF02:   CJNE    A,#NUMBER_TOKE,ASERR
F790 12FA47          854             CALL    CHECK_AND_SET_EXP_FLAG
F793 02F99F          855             JMP     CALCULATE_INSTRUCTION_VALUE
                     856 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                     857     ;*****************************************************************
                     858     ;
                     859     ;    NAME: MNEMONIC_TWO_OPERANDS
                     860     ;
                     861     ;    ABSTRACT: This routine sets flags to indicate how to assemble
                     862     ;       two operand instructions with 2 or 3 bytes.  It gets a hash
                     863     ;       value and passes the expression or expressions to run time
                     864     ;       routines.  Instructions include: MOV DPTR,#<addr>,
                     865     ;       MOV <data addr>,<data addr>.
                     866     ;
                     867     ;    INPUTS: None
                     868     ;
                     869     ;    OUTPUTS: NUMBER_OF_BYTES, ORDINAL, OUR_CODE_LOW, OUR_CODE_HIGH,
                     870     ;       TEMP_SEC, VALLOW
                     871     ;
                     872     ;    VARIABLES MODIFIED: A, ORDINAL, TEMP_SEC, ERRNUM
                     873     ;
                     874     ;    ERROR EXITS: 03H (NUMBER EXPECTED)
                     875     ;                 10H (ASSEMBLY SYNTAX)
                     876     ;
                     877     ;    SUBROUTINES ACCESSED DIRECTLY: GETOKE, UPDATE_OUR_CODE, GET_COMMA,
                     878     ;       GETNUM, MNEMONIC_SECOND_OPERAND_TAIL, CALCULATE_INSTRUCTION_VALUE,
                     879     ;       GET_SECOND_OPERAND, SET_POUND_EXP_FLAG, SET_SLASH_EXP_FLAG,
                     880     ;       CHECK_AND_SET_SECOND_EXP_FLAG
                     881     ;
                     882     ;*****************************************************************
                     883     MNEMONIC_TWO_OPERANDS:
F796 12E056          884             CALL    GETOKE
F799 B4A118          885             CJNE    A,#DPTR_TOKE,MTOO
F79C 755C0D          886             MOV     ORDINAL,#0DH
F79F 12F9C7          887             CALL    UPDATE_OUR_CODE
F7A2 12E06B          888             CALL    GET_COMMA
F7A5 12E056          889             CALL    GETOKE
F7A8 B40629          890             CJNE    A,#POUND_TOKE,ASERR
F7AB 12E050          891             CALL    GETNUM
F7AE 12FA57          892             CALL    SET_EXP_16_FLAG
F7B1 02F99F          893             JMP     CALCULATE_INSTRUCTION_VALUE
F7B4 300006          894     MTOO:   JNB     B_O_T,MFTOO                     ;MNEMONIC_FIRST_TWO_OPERANDS
F7B7 12FA02          895             CALL    GET_FIRST_OPERAND
F7BA 02F7C6          896             JMP     MTO1
F7BD B40114          897     MFTOO:  CJNE    A,#NUMBER_TOKE,ASERR
F7C0 12FA60          898             CALL    SET_EXP_FLAG
F7C3 854A62          899             MOV     TEMP_SEC,VALLOW
F7C6 12E06B          900     MTO1:   CALL    GET_COMMA
F7C9 12E056          901             CALL    GETOKE                          ;MNEMONIC_SECOND_OPERAND
F7CC 30000B          902             JNB     B_O_T,MSOO
F7CF 12FAAC          903             CALL    GET_SECOND_OPERAND
F7D2 8081            904             JMP     MNEMONIC_SECOND_OPERAND_TAIL
F7D4 754310          905     ASERR:  MOV     ERRNUM,#10H                     ;Assembly syntax
F7D7 02E05F          906             JMP     ERROR
F7DA E548            907     MSOO:   MOV     A,TOKSTR
F7DC B40608          908             CJNE    A,#POUND_TOKE,MSO1
F7DF 12FA87          909             CALL    SET_POUND_EXP_FLAG
F7E2 12E050          910             CALL    GETNUM
F7E5 E155            911             JMP     MNEMONIC_SECOND_OPERAND_TAIL
```

```
LOC  OBJ            LINE     SOURCE

F7E7 B40308          912     MS01:   CJNE    A,#BAR_TOKE,MS02
F7EA 12FA9B          913             CALL    SET_SLASH_EXP_FLAG
F7ED 12E050          914             CALL    GETNUM
F7F0 E155            915             JMP     MNEMONIC_SECOND_OPERAND_TAIL
F7F2 754303          916     MS02:   MOV     ERRNUM,#03H
F7F5 B40169          917             CJNE    A,#NUMBER_TOKE,TOERR
F7F8 12FA8F          918             CALL    CHECK_AND_SET_SECOND_EXP_FLAG
F7FB E155            919             JMP     MNEMONIC_SECOND_OPERAND_TAIL
                     920 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                     921     ;*****************************************************************
                     922     ;
                     923     ;    NAME: MOVC_OPERANDS
                     924     ;
                     925     ;    ABSTRACT: This routine divides operands into one of two possible
                     926     ;       cases and modifies the hash value. Instructions are
                     927     ;       MOVC A,@A+DPTR and MOVC A,@A+PC.
                     928     ;
                     929     ;    INPUTS: None
                     930     ;
                     931     ;    OUTPUTS: ORDINAL, OUR_CODE_LOW, OUR_CODE_HIGH
                     932     ;
                     933     ;    VARIABLES MODIFIED: A, ORDINAL
                     934     ;
                     935     ;    ERROR EXITS: 10H (ASSEMBLY SYNTAX)
                     936     ;
                     937     ;    SUBROUTINES ACCESSED DIRECTLY: GETOKE, GET_FIRST_OPERAND, GET_COMMA,
                     938     ;       UPDATE_OUR_CODE, ONE_BYTE_TAIL
                     939     ;
                     940     ;*****************************************************************
                     941     MOVC_OPERANDS:
F7FD 12E056          942             CALL    GETOKE
F800 3000D1          943             JNB     B_0_T,ASERR
F803 12FA02          944             CALL    GET_FIRST_OPERAND
F806 12E06B          945             CALL    GET_COMMA
F809 12E056          946             CALL    GETOKE                  ;MOVC_TAIL
F80C B40AC5          947             CJNE    A,#ATA_TOKE,ASERR
F80F 12E056          948             CALL    GETOKE
F812 12E056          949             CALL    GETOKE
F815 B4A109          950             CJNE    A,#DPTR_TOKE,MTO
F818 755C17          951             MOV     ORDINAL,#17H
F81B 12F9C7          952             CALL    UPDATE_OUR_CODE
F81E 02F755          953             JMP     ONE_BYTE_TAIL
F821 B4A0B0          954     MTO:    CJNE    A,#PC_TOKE,ASERR
F824 755C18          955             MOV     ORDINAL,#18H
F827 12F9C7          956             CALL    UPDATE_OUR_CODE
F82A 02F755          957             JMP     ONE_BYTE_TAIL
                     958 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                     959     ;*****************************************************************
                     960     ;
                     961     ;    NAME: THREE_OPERANDS
                     962     ;
                     963     ;    ABSTRACT: This routine parses the opcodes and modifies the
                     964     ;       hash value accordingly.  It saves the data address or
                     965     ;       immediate data field and the destination address.  Instructions
                     966     ;       are CJNE @R0,#<data>,<addr>; CJNE @R1,#<data>,<addr>;
                     967     ;       CJNE A,#<data>,<addr>; CJNE A,<data>,<addr>; CJNE Rn,#<data>,<data>
                     968     ;
                     969     ;    INPUTS: None
                     970     ;
                     971     ;    OUTPUTS: ORIDNAL, OUR_CODE_LOW, OUR_CODE_HIGH, VALLOW, TEMP_SEC,
                     972     ;       NUMBER_OF_BYTES
                     973     ;
                     974     ;    VARIABLES MODIFIED: NUMBER_OF_BYTES, TEMP_SEC, A
                     975     ;
                     976     ;    ERROR EXITS: 10H (ASSEMBLY SYNTAX)
                     977     ;                 03H (NUMBER EXPECTED)
                     978     ;
                     979     ;    SUBROUTINES ACCESSED DIRECTLY: GETOKE, GET_FIRST_OPERAND,
                     980     ;       GET_COMMA, SET_POUND_EXP_FLAG, CHECK_AND_SET_SECOND_EXP_FLAG,
                     981     ;       GETNUM, CALCULATE_INSTRUCTION_VALUE, ERROR
                     982     ;
                     983     ;*****************************************************************
                     984     THREE_OPERANDS:
F82D 12E056          985             CALL    GETOKE
F830 3000A1          986             JNB     B_0_T,ASERR
F833 12FA02          987             CALL    GET_FIRST_OPERAND
F836 12E06B          988             CALL    GET_COMMA
F839 12E056          989             CALL    GETOKE                          ;SECOND_THREE_OPERANDS
F83C B40609          990             CJNE    A,#POUND_TOKE,ST01
F83F 12FA87          991             CALL    SET_POUND_EXP_FLAG
F842 12E050          992             CALL    GETNUM
F845 02F851          993             JMP     STORET
F848 754303          994     ST01:   MOV     ERRNUM,#03H                     ;Number expected
F84B B40113          995             CJNE    A,#NUMBER_TOKE,TOERR
F84E 12FA8F          996             CALL    CHECK_AND_SET_SECOND_EXP_FLAG
F851 854A62          997     STORET: MOV     TEMP_SEC,VALLOW
F854 12E06B          998             CALL    GET_COMMA
F857 12E050          999             CALL    GETNUM
F85A 12F99F         1000             CALL    CALCULATE_INSTRUCTION_VALUE
F85D 754D05         1001             MOV     NUMBER_OF_BYTES,#05H
F860 22             1002             RET
F861 02E05F         1003     TOERR:  JMP     ERROR
F864 22             1004             RET
                    1005 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                    1006     ;*****************************************************************
                    1007     ;
                    1008     ;    NAME: JUMP_OPERAND
                    1009     ;
                    1010     ;    ABSTRACT: This routine gets the destination for a jump from
                    1011     ;       the command line and sets the relative operand flag to
                    1012     ;       indicate the method of assembly. Instructions are SJMP<addr>,
                    1013     ;       JNC<addr>, JC<addr>, JZ<addr>, JNZ<addr>.
                    1014     ;
                    1015     ;    INPUTS: None
                    1016     ;
                    1017     ;    OUTPUTS: OUR_CODE_LOW, OUR_CODE_HIGH, VALLOW
                    1018     ;
                    1019     ;    VARIABLES MODIFIED: None
                    1020     ;
                    1021     ;    ERROR EXITS: None
                    1022     ;
                    1023     ;    SUBROUTINES ACCESSED DIRECTLY: GETNUM, SET_REL_FLAG,
                    1024     ;       CALCULATE_INSTRUCTION_VALUE
                    1025     ;
                    1026     ;*****************************************************************
                    1027     JUMP_OPERAND:
F865 12E050         1028             CALL    GETNUM
F868 12FAA3         1029             CALL    SET_REL_FLAG
F86B 02F99F         1030             JMP     CALCULATE_INSTRUCTION_VALUE
                    1031 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                    1032     ;*****************************************************************
                    1033     ;
                    1034     ;    NAME: JUMP_TWO_OPERANDS
                    1035     ;
                    1036     ;    ABSTRACT: This routine gets an expression for an address bit
                    1037     ;       which will be tested by the jump.  It modifies OUR_CODE and
                    1038     ;       REL_FLAG to indicate proper means of assembly, then gets
                    1039     ;       the destination address. Instructions are JB<bit addr>,<addr>;
                    1040     ;       JBC<bit addr>,<addr>; JNB<bit addr>,<addr>; DJNZ<bit addr>,<addr>;
                    1041     ;       DJNZ Rn,<addr>.
                    1042     ;
                    1043     ;    INPUTS: B_0_T
                    1044     ;
                    1045     ;    OUTPUTS: NUMBER_OF_BYTES, TEMP_SEC, OUR_CODE_LOW, OUR_CODE_HIGH,
                    1046     ;       VALLOW
                    1047     ;
                    1048     ;    VARIABLES MODIFIED: NUMBER_OF_BYTES, TEMP_SEC
                    1049     ;
                    1050     ;    ERROR EXITS: None
                    1051     ;
                    1052     ;    SUBROUTINES ACCESSED DIRECTLY: GETOKE, GET_FIRST_OPERAND,
                    1053     ;       SET_REL_FLAG, CALCULATE_INSTRUCTION_VALUE, CHECK_AND_SET_EXP_FLAG,
                    1054     ;       GET_COMMA, GETNUM
                    1055     ;
                    1056     ;*****************************************************************
                    1057     JUMP_TWO_OPERANDS:
F86E 12E056         1058             CALL    GETOKE
F871 30000C         1059             JNB     B_0_T,JT00
F874 12FA02         1060             CALL    GET_FIRST_OPERAND
F877 12FAA3         1061             CALL    SET_REL_FLAG
F87A 12F99F         1062             CALL    CALCULATE_INSTRUCTION_VALUE
F87D 02F88F         1063             JMP     JTRET
F880 B401DE         1064     JT00:   CJNE    A,#NUMBER_TOKE,TOERR
F883 12FA47         1065             CALL    CHECK_AND_SET_EXP_FLAG
F886 12FAA3         1066             CALL    SET_REL_FLAG
F889 12F99F         1067             CALL    CALCULATE_INSTRUCTION_VALUE
F88C 754D05         1068             MOV     NUMBER_OF_BYTES,#05H
F88F 854A62         1069     JTRET:  MOV     TEMP_SEC,VALLOW
F892 12E06B         1070             CALL    GET_COMMA
F895 02E050         1071             JMP     GETNUM
                    1072 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                    1073     ;*****************************************************************
                    1074     ;
                    1075     ;    NAME: JUMP_ABSOLUTE_OPERAND
                    1076     ;
                    1077     ;    ABSTRACT: This routine gets the destination address and
                    1078     ;       modifies OUR_CODE to indicate that the upper 3 bits of
                    1079     ;       address must be included in the final opcode.  Instructions
                    1080     ;       of this type are AJMP <addr>, ACALL <addr>.
                    1081     ;
                    1082     ;    INPUTS: None
                    1083     ;
                    1084     ;    OUTPUTS: ORDINAL, NUMBER_OF_BYTES, OUR_CODE_LOW, OUR_CODE_HIGH,
                    1085     ;       VALLOW, VALHGH
                    1086     ;
                    1087     ;    VARIABLES MODIFIED: ORDINAL, NUMBER_OF_BYTES
                    1088     ;
                    1089     ;    ERROR EXITS: None
                    1090     ;
                    1091     ;    SUBROUTINES ACCESSED DIRECTLY: GETNUM, UPDATE_OUR_CODE,
                    1092     ;       CALCULATE_INSTRUCTION_VALUE
                    1093     ;
                    1094     ;*****************************************************************
                    1095     JUMP_ABSOLUTE_OPERAND:
F898 12E050         1096             CALL    GETNUM
F89B 755C15         1097             MOV     ORDINAL,#15H                    ;SET_EXP_11_FLAG
F89E 12F9C7         1098             CALL    UPDATE_OUR_CODE                 ;2K page jump
F8A1 754D06         1099             MOV     NUMBER_OF_BYTES,#06H            ;Absolute instruction
F8A4 02F99F         1100             JMP     CALCULATE_INSTRUCTION_VALUE
                    1101 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                    1102     ;*****************************************************************
                    1103     ;
                    1104     ;    NAME: JUMP_LONG_OPERAND
                    1105     ;
                    1106     ;    ABSTRACT: This routine gets the destination address and sets
                    1107     ;       the 16 bit expression flag.  It then searches the hash table
                    1108     ;       for a matching opcode.  Instructions are LCALL <addr> and
                    1109     ;       LJMP <addr>.
                    1110     ;
                    1111     ;    INPUTS: None
                    1112     ;
                    1113     ;    OUTPUTS: ORDINAL, NUMBER_OF_BYTES, OUR_CODE_LOW, OUR_CODE_HIGH,
                    1114     ;       VALHGH, VALLOW
                    1115     ;
                    1116     ;    VARIABLES MODIFIED: None
                    1117     ;
                    1118     ;    ERROR EXITS: None
                    1119     ;
                    1120     ;    SUBROUTINES ACCESSED DIRECTLY: GETNUM, SET_EXP_16_FLAG,
                    1121     ;       CALCULATE_INSTRUCTION_VALUE
                    1122     ;
                    1123     ;*****************************************************************
                    1124     JUMP_LONG_OPERAND:
F8A7 12E050         1125             CALL    GETNUM
F8AA 12FA57         1126             CALL    SET_EXP_16_FLAG
F8AD 02F99F         1127             JMP     CALCULATE_INSTRUCTION_VALUE
                    1128 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                    1129     ;*****************************************************************
                    1130     ;
                    1131     ;    NAME: MNEMONIC_INSTRUCTION_TAIL
                    1132     ;
                    1133     ;    ABSTRACT: This routine selects the type of instruction as determined
                    1134     ;       by the MNEMONIC INSTRUCTION TABLE and calls the handler for the
                    1135     ;       type specified.  The handler completes the parsing of the command
                    1136     ;       line and does the hash table look-up.
                    1137     ;
                    1138     ;    INPUTS: INSTRUCTION_VALUE
                    1139     ;
                    1140     ;    OUTPUTS: ORDINAL, VALLOW, VALHGH, TEMP_SEC, NUMBER_OF_BYTES,
                    1141     ;       OUR_CODE_LOW, OUR_CODE_HIGH
                    1142     ;
                    1143     ;    VARIABLES MODIFIED: DPTR, A, C, B, ERRNUM
                    1144     ;
                    1145     ;    ERROR EXITS: 10H (ASSEMBLY SYNTAX)
                    1146     ;
                    1147     ;    SUBROUTINES ACCESSED DIRECTLY: ONE_BYTE_TAIL, MNEMONIC_FIRST_OPERAND,
                    1148     ;       MNEMONIC_TWO_OPERANDS, MOVC_OPERANDS, THREE_OPERANDS, JUMP_OPERAND, '
                    1149     ;       JUMP_TWO_OPERANDS, JUMP_ABSOLUTE_OPERAND, JUMP_LONG_OPERAND
                    1150     ;
                    1151     ;*****************************************************************
                    1152     MNEMONIC_INSTRUCTION_TAIL:
F8B0 754310         1153             MOV     ERRNUM,#10H
F8B3 90F8C0         1154             MOV     DPTR,#MIT_JMP_TBL
F8B6 E55B           1155             MOV     A,INSTRUCTION_VALUE
F8B8 C3             1156             CLR     C
F8B9 9407           1157             SUBB    A,#07H
F8BB 75F003         1158             MOV     B,#03H
F8BE A4             1159             MUL     AB                      ;Mult by 3, each tbl entry is 3 bytes
F8BF 73             1160             JMP     @A+DPTR
                    1161     MIT_JMP_TBL:
F8C0 02F755         1162             LJMP    ONE_BYTE_TAIL
F8C3 02F75B         1163             LJMP    MNEMONIC_FIRST_OPERAND
F8C6 02F796         1164             LJMP    MNEMONIC_TWO_OPERANDS
F8C9 02F7FD         1165             LJMP    MOVC_OPERANDS
F8CC 02F82D         1166             LJMP    THREE_OPERANDS
F8CF 02F865         1167             LJMP    JUMP_OPERAND
F8D2 02F86E         1168             LJMP    JUMP_TWO_OPERANDS
F8D5 02F898         1169             LJMP    JUMP_ABSOLUTE_OPERAND
F8D8 02F8A7         1170             LJMP    JUMP_LONG_OPERAND
                    1171 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                    1172     ;*****************************************************************
                    1173     ;
                    1174     ;    NAME: MNEMONIC_INSTR_LIST_TAIL
                    1175     ;
                    1176     ;    ABSTRACT: This routine sets up information to be used in later
                    1177     ;       processing of the mnemonic by deciphering the information
                    1178     ;       in MNEMONIC_TAB then the call to MNEMONIC_INSTRUCTION_TAIL and
                    1179     ;       CHANGE_TO_INSTRUCTION_OP completes the assembly.
                    1180     ;
                    1181     ;    INPUTS: TOKSTR, ASM_PC_LOW, ASM_PC_HIGH
                    1182     ;
                    1183     ;    OUTPUTS: Code memory locations pointed to by ASM_PC.
                    1184     ;
                    1185     ;    VARIABLES MODIFIED: BIT_EXP, FIRST_EXP, SECOND_EXP, A, C, DPTR,
                    1186     ;       INSTRUCTION_VALUE, OUR_CODE_LOW
                    1187     ;
                    1188     ;    ERROR EXITS: None
                    1189     ;
                    1190     ;    SUBROUTINES ACCESSED DIRECTLY: MNEMONIC_INSTRUCTION_TAIL,
                    1191     ;       CHANGE_TO_INSTRUCTION_OP
                    1192     ;
                    1193     ;*****************************************************************
                    1194     MNEMONIC_INSTR_LIST_TAIL:
F8DB C202           1195             CLR     BIT_EXP                         ;MNEMONIC_INSTR
F8DD C203           1196             CLR     FIRST_EXP                       ;Initialize flags
F8DF C204           1197             CLR     SECOND_EXP
F8E1 754D00         1198             MOV     NUMBER_OF_BYTES,#00H
F8E4 754E00         1199             MOV     OUR_CODE_HIGH,#00H              ;SELECT_INSTRUCTION_TAIL
F8E7 C3             1200             CLR     C
F8E8 E548           1201             MOV     A,TOKSTR
F8EA 9410           1202             SUBB    A,#OFST
F8EC F54F           1203             MOV     OUR_CODE_LOW,A
F8EE 90F529         1204             MOV     DPTR,#MNEMONIC_TAB
F8F1 93             1205             MOVC    A,@A+DPTR
F8F2 F55B           1206             MOV     INSTRUCTION_VALUE,A             ;Search for corresponding match using
F8F4 7416           1207             MOV     A,#JUMP_END                     ;look-up and check to see if mnemonic
F8F6 B54F0C         1208             CJNE    A,OUR_CODE_LOW,OUR_GTRTHN       ;is JBC, JB, JNB, CPL or SETB.  If is,
                    1209     CONT_OUR_CODE:
F8F9 C3             1210             CLR     C                               ;then set BIT_EXP.  If OUR_CODE=#JUMP_EN
F8FA E54F           1211             MOV     A,OUR_CODE_LOW                  ;Then fall thru or check for OUR_CODE>#JUMP
                         _END
F8FC 941C           1212             SUBB    A,#(BIT_END+1)                  ;Check that OUR_CODE<=BIT_END
F8FE 5007           1213             JNC     END_SELECT_INSTRUCTION_TAIL
F900 D202           1214             SETB    BIT_EXP
F902 02F907         1215             JMP     END_SELECT_INSTRUCTION_TAIL
                    1216     OUR_GTRTHN:
F905 40F2           1217             JC      CONT_OUR_CODE
                    1218     END_SELECT_INSTRUCTION_TAIL:
F907 E54F           1219             MOV     A,OUR_CODE_LOW
F909 B42B03         1220             CJNE    A,#2BH,MI0
F90C 02F911         1221             JMP     MI1
F90F 5035           1222     MI0:    JNC     AMTERR
F911 11B0           1223     MI1:    CALL    MNEMONIC_INSTRUCTION_TAIL
F913 02FB21         1224             JMP     CHANGE_TO_INSTRUCTION_OP
                    1225 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                    1226     ;*****************************************************************
                    1227     ;
                    1228     ;    NAME: ASSEMBLY_CMD
                    1229     ;
                    1230     ;    ABSTRACT: This routine parses the rest of the command line
                    1231     ;       for ORG or carraige return and enters the ASM mode.  Once
                    1232     ;       in ASM mode, control remains here in a loop assembling
                    1233     ;       instructions until a carraige return is found on a line by
                    1234     ;       itself.
                    1235     ;
                    1236     ;    INPUTS: None
                    1237     ;
                    1238     ;    OUTPUTS: Code memeory locations pointed to in ORG clause or
                    1239     ;       pre-existing ASM_PC setting.
                    1240     ;
                    1241     ;    VARIABLES MODIFIED: ASM_PC_HIGH, ASM_PC_LOW, A, POINTO, PARAM1,
                    1242     ;       ERRNUM
                    1243     ;
                    1244     ;    ERROR EXITS: 10H (ASSEMBY SYNTAX)
                    1245     ;
                    1246     ;    SUBROUTINES ACCESSED DIRECTLY: GETOKE, NEWLINE, GETNUM,
                    1247     ;       SAVE_AND_DISPLAY, ERROR, MNEMONIC_INSTR_LIST_TAIL, GETEOL
                    1248     ;
                    1249     ;*****************************************************************
                    1250     ASSEMBLY_CMD:
F916 755205         1251             MOV     LINE_START,#05H
F919 12E056         1252             CALL    GETOKE
F91C B4D40F         1253             CJNE    A,#ORG_TOKE,AMT0
F91F 12E050         1254             CALL    GETNUM                  ;Get past address
F922 85494B         1255             MOV     ASM_PC_HIGH,VALHGH
F925 854A4C         1256             MOV     ASM_PC_LOW,VALLOW
F928 12E00F         1257             CALL    NEWLINE
F92B 12E056         1258             CALL    GETOKE
F92E B40715         1259     AMT0:   CJNE    A,#EOL_TOKE,AMTERR
F931 7824           1260     AMT1:   MOV     POINTO,#LINBUF
F933 AA4B           1261             MOV     PARAM1,ASM_PC_HIGH
F935 12E05C         1262             CALL    SAVE_AND_DISPLAY
F938 AA4C           1263             MOV     PARAM1,ASM_PC_LOW
F93A 12E05C         1264             CALL    SAVE_AND_DISPLAY
F93D 7620           1265             MOV     @POINTO,#' '
F93F 12E056         1266             CALL    GETOKE
F942 B40707         1267             CJNE    A,#EOL_TOKE,AMT2
F945 22             1268             RET
F946 754310         1269     AMTERR: MOV     ERRNUM,#10H             ;Assembly syntax
F949 02E05F         1270             JMP     ERROR
F94C 11DB           1271     AMT2:   CALL    MNEMONIC_INSTR_LIST_TAIL
F94E 12E053         1272             CALL    GETEOL
F951 80DE           1273             JMP     AMT1
                    1274
                    1275 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                    1276 +1  $INCLUDE(:F1:ASM.INC)
                  =1  1277     ;*****************************************************************
                  =1  1278     ;
                  =1  1279     ;       This is the include file called ASM.INC.  It contains the
                  =1  1280     ;       following subroutines in order:
                  =1  1281     ;
                  =1  1282     ;               START_DIVIDE
                  =1  1283     ;               CALCULATE_INSTRUCTION_VALUE
                  =1  1284     ;               UPDATE_OUR_CODE
                  =1  1285     ;               GET_FIRST_OPERAND
                  =1  1286     ;               CHECK_AND_SET_EXP_FLAG
                  =1  1287     ;                       SET_EXP_16_FLAG
                  =1  1288     ;                       SET_EXP_FLAG
                  =1  1289     ;                       CHECK_EXP_FLAG
                  =1  1290     ;                       SET_POUND_EXP_FLAG
                  =1  1291     ;                       CHECK_AND_SET_SECOND_EXP_FLAG
                  =1  1292     ;                       SET_SLASH_EXP_FLAG
                  =1  1293     ;                       SET_REL_FLAG
                  =1  1294     ;               GET_SECOND_EXP
                  =1  1295     ;
                  =1  1296     ;*****************************************************************
                  =1  1297 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1  1298     ;/*************************************************************************
                =1  1299     ; *                                                                       *
                =1  1300     ; * This module contains most procedures needed to implement the          *
                =1  1301     ; * assembler which processes the ASM command. The rest are contained     *
                =1  1302     ; * in the ASMA module.                                                   *
                =1  1303     ; *                                                                       *
                =1  1304     ; *   INSTRUCTION_VALUE - Public variable used at parse time.  The        *
                =1  1305     ; *     instruction is assembled into it.                                 *
                =1  1306     ; *                                                                       *
                =1  1307     ; *-----------------------------------------------------------------------*
                =1  1308     ; *                                                                       *
                =1  1309     ; * The assembler consists of three pieces:                              *
                =1  1310     ; * - Tables in the module ASM_TBL code which contain the details of the  *
                =1  1311     ; *   8051 assembly language,                                             *
                =1  1312     ; * - Parse time procedures in this module use these tables to:           *
                =1  1313     ; *   -Set up flags and variables to control actual memory     *
                =1  1314     ; *    writing operations, search the tables for matched to the hashed    *
                =1  1315     ; *    command line.
                =1  1316     ; *   - Assemble the instruction as if any expression, immediate data, or *
                =1  1317     ; *       jump addresses are zero (they are evaluated at run-time).       *
                =1  1318     ; * - Procedures selected by the above parse time procedures determine:   *
                =1  1319     ; *   - What the instruction format is,                                   *
                =1  1320     ; *   - How to combine the expressions, immediate data, or jump addresses *
                =1  1321     ; *       (if any) after being calculated with the instruction value      *
                =1  1322     ; *       assembled at parse time to create the final result of the       *
                =1  1323     ; *       assembly in memory.                                             *
                =1  1324     ; *                                                                       *
                =1  1325     ; * The opcode is found by generating a hash value as the parser scans the *
                =1  1326     ; * instruction.  How the hash value is calculated is discussed in ASM_TBL. *
                =1  1327     ; * All the hash values are stored in the table, #INSTRUCTION_CODE, and the *
                =1  1328     ; * ordinal corresponding to a hash value is the opcode for that instruction.*
                =1  1329     ; * Except for absolute instructions, in which case the opcode is further  *
                =1  1330     ; * calculated in CHANGE_TO_INSTRUCTION_OP, NUMBER_OF_BYTES contains either *
                =1  1331     ; * the actual number of bytes in the instruction or a code to enable      *
                =1  1332     ; * CHANGE_TO_INSTRUCTION_OP to write the correct number of bytes in the   *
                =1  1333     ; * correct order.  See CHANGE_TO_INSTRUCTION_OP for more details.         *
                =1  1334     ; *                                                                       *
                =1  1335     ; * Parsing the command line leaves the opcode in INSTRUCTION_VALUE at run *
                =1  1336     ; * time.  CHANGE_TO_INSTRUCTION_OP is called after each command line     *
                =1  1337     ; * to process the type of instruction appropriately to write it out to   *
                =1  1338     ; * memory.  Relative offsets and 2K jump or calls are generated here.    *
                =1  1339     ; *                                                                       *
                =1  1340     ; * Details on the use of the tables in the assembly can be found in the  *
                =1  1341     ; * documentation in the ASM_TBL module.                                  *
                =1  1342     ; *                                                                       *
                =1  1343     ; *-----------------------------------------------------------------------*
                =1  1344     ; *                                                                       *
                =1  1345     ; * In the operand_table the basic operands(ex. C,A,R0-R7,etc.) have the  *
                =1  1346     ; * ordinal+1 values of 1-15 but the values 16-24 were used to represent  *
                =1  1347     ; * certain expressions as follows:                                       *
                =1  1348     ; *                                                                       *
                =1  1349     ; *   16 - BYTE EXP8                  21 - EXP11                          *
                =1  1350     ; *   17 - BIT EXP8                   22 - RELATIVE OFFSET EXPRESSION     *
                =1  1351     ; *   18 - IMMEDIATE(#) EXP8          23 - @A+DPTR                        *
                =1  1352     ; *   19 - COMPLEMENT(/) EXP8         24 - @A+PC                          *
```

```
LOC  OBJ          LINE     SOURCE

                =1  1353     ; *   20 - EXP16                                                *
                =1  1354     ; *                                                             *
                =1  1355     ; *-------------------------------------------------------------*
                =1  1356     ; *                                                             *
                =1  1357     ; * A problem arose which made the software more involved: determining if    *
                =1  1358     ; * the eight bit expression was a bit or byte expression. Since disassembly *
                =1  1359     ; * uses the same tables as assembly the hash values had to be precise.      *
                =1  1360     ; * The following instructions had bit expressions:                          *
                =1  1361     ; *                                                                         *
                =1  1362     ; *   JBC  BIT EXP,CODE EXP       ORL C,BIT EXP        MOV BIT EXP,C        *
                =1  1363     ; *   JB   BIT EXP,CODE EXP       ANL C,BIT EXP                             *
                =1  1364     ; *   JNB  BIT EXP,CODE EXP       MOV C,BIT EXP                             *
                =1  1365     ; *   CLR  BIT EXP,CODE EXP                                                 *
                =1  1366     ; *   CPL  BIT EXP,CODE EXP                                                 *
                =1  1367     ; *   SETB BIT EXP,CODE EXP                                                 *
                =1  1368     ; *                                                                         *
                =1  1369     ; * In the first group, if the mnemonic was one of those six mnemonics the   *
                =1  1370     ; * BIT_EXP FLAG was set and if an expression was found we know it was a bit *
                =1  1371     ; * expression. The second group was a little more difficult.  If the first  *
                =1  1372     ; * operand of a two operand instruction was found to be a 'C' the BIT_EXP   *
                =1  1373     ; * flag was set and then if the second operand was an expression we knew it *
                =1  1374     ; * was a bit expression.  The third group was the real problem.  If the     *
                =1  1375     ; * second operand of a two operand instruction was a 'C' and the first      *
                =1  1376     ; * operand had been an expression then the hash value was re-calculated to  *
                =1  1377     ; * indicate a bit expression.                                              *
                =1  1378     ; *                                                                         *
                =1  1379     ; ***************************************************************************/
                =1  1380 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1  1381     ;*****************************************************************
                =1  1382     ;
                =1  1383     ;    NAME: START_DIVIDE
                =1  1384     ;
                =1  1385     ;    ABSTRACT: This is a software divide routine.  Inputs are an 8-bit
                =1  1386     ;       divisor and a 16-bit dividend.  The quotient is 16-bits and
                =1  1387     ;       the remainder is truncated to 8 bits.
                =1  1388     ;
                =1  1389     ;    INPUTS: DIVIDEND_HIGH, DIVIDEND_LOW, DIVISOR
                =1  1390     ;
                =1  1391     ;    OUTPUTS: QUOTIENT_HIGH, QUOTIENT_LOW
                =1  1392     ;
                =1  1393     ;    VARIABLES MODIFIED: A, PARAM6, DIVIDEND_LOW, QUOTIENT_HIGH,
                =1  1394     ;       PARAM5, PARAM4, C, DIVIDEND_HIGH, QUOTIENT_LOW
                =1  1395     ;
                =1  1396     ;    ERROR EXITS: None
                =1  1397     ;
                =1  1398     ;    SUBROUTINES ACCESSED DIRECTLY: None
                =1  1399     ;
                =1  1400     ;*****************************************************************
                =1  1401     START_DIVIDE:
F953 E570       =1  1402             MOV     A,DIVISOR
F955 7F09       =1  1403             MOV     PARAM6,#09H
F957 7E00       =1  1404             MOV     PARAM5,#00H
F959 7D00       =1  1405             MOV     PARAM4,#00H
                =1  1406     DIVIDE_1:
F95B C3         =1  1407             CLR C
                =1  1408     DIVIDE_2:
F95C E56E       =1  1409             MOV     A,DIVIDEND_HIGH
F95E 4011       =1  1410             JC      SUBTRACT_WITH_C         ;Carry occurs from rotate
F960 6021       =1  1411             JZ      ROTATE                  ;Rotate quotient and dividend if zero
F962 9570       =1  1412             SUBB    A,DIVISOR
F964 401D       =1  1413             JC      ROTATE                  ;A carry means divisor is larger than dividend
F966 F56E       =1  1414             MOV     DIVIDEND_HIGH,A         ;Replace DIVIDEND_HIGH with new number
F968 EE         =1  1415             MOV     A,PARAM5                ;PARAM5 holds lower byte of quotient
F969 2401       =1  1416             ADD     A,#01H                  ;Increment quotient
F96B 5001       =1  1417             JNC     DIVIDE_3
F96D 0D         =1  1418             INC     PARAM4                  ;High counter incremented if carry occurs
                =1  1419     DIVIDE_3:
F96E FE         =1  1420             MOV     PARAM5,A                ;Replace with new quotient
F96F 80EA       =1  1421             JMP     DIVIDE_1                ;Loop
                =1  1422     SUBTRACT_WITH_C:
F971 EE         =1  1423             MOV     A,PARAM5
F972 2401       =1  1424             ADD     A,#01H
F974 5001       =1  1425             JNC     DIVIDE_4
F976 0D         =1  1426             INC     PARAM4
                =1  1427     DIVIDE_4:
F977 FE         =1  1428             MOV     PARAM5,A                ;Quotient always incremented if carry set
F978 C3         =1  1429             CLR     C
F979 E56E       =1  1430             MOV     A,DIVIDEND_HIGH
F97B 9570       =1  1431             SUBB    A,DIVISOR
F97D F56E       =1  1432             MOV     DIVIDEND_HIGH,A         ;Subtract divisor from dividend
F97F 40DA       =1  1433             JC      DIVIDE_1                ;Jump to subtract with no carry if carry is set
F981 80EE       =1  1434             JMP     SUBTRACT_WITH_C         ;Loop in subtract with C if no carry
F983 DF05       =1  1435     ROTATE: DJNZ    PARAM6,ROTATE_CONTINUE  ;PARAM6 counts number of rotates
```

```
LOC  OBJ            LINE     SOURCE

F985 8D71       =1  1436             MOV     QUOTIENT_HIGH,PARAM4
F987 8E72       =1  1437             MOV     QUOTIENT_LOW,PARAM5
F989 22         =1  1438             RET                             ;Exit from divide routine
                =1  1439     ROTATE_CONTINUE:
F98A C3         =1  1440             CLR     C
F98B EE         =1  1441             MOV     A,PARAM5
F98C 33         =1  1442             RLC     A                       ;Rotate ;pwer byte of quotient first
F98D FE         =1  1443             MOV     PARAM5,A                ;Replace with new quotient low
F98E ED         =1  1444             MOV     A,PARAM4
F98F 33         =1  1445             RLC     A                       ;Rotate upper byte with MSB from lower
F990 FD         =1  1446             MOV     PARAM4,A                ;byte into LSB of upper byte
F991 C3         =1  1447             CLR     C
F992 E56F       =1  1448             MOV     A,DIVIDEND_LOW          ;Rotate dividend with every rotate of
                =1  1449                                             ;quotient
F994 33         =1  1450             RLC     A
F995 F56F       =1  1451             MOV     DIVIDEND_LOW,A
F997 E56E       =1  1452             MOV     A,DIVIDEND_HIGH
F999 33         =1  1453             RLC     A
F99A F56E       =1  1454             MOV     DIVIDEND_HIGH,A
F99C 80BE       =1  1455             SJMP    DIVIDE_2                ;Loop
F99E 22         =1  1456             RET                             ;End of divide routines
                =1  1457 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1  1458     ;*****************************************************************
                =1  1459     ;
                =1  1460     ;    NAME: CALCULATE_INSTRUCTION_VALUE
                =1  1461     ;
                =1  1462     ;    ABSTRACT: Parse-time action to assemble the instruction just parsed
                =1  1463     ;       into the public variable INSTRUCTION_VALUE.  The values may be
                =1  1464     ;       calculated and filled in at run-time.  Using the hash value,
                =1  1465     ;       the #INSTRUCTION_CODE table is searched for a corresponding match.
                =1  1466     ;       If one is found, the ordinal of the match (INSTRUCTION_VALUE) is
                =1  1467     ;       the opcode of the instruction.  If one is not found, an error is issued
                =1  1468     ;       and processing stops.
                =1  1469     ;
                =1  1470     ;    INPUTS: OUR_CODE_LOW, OUR_CODE_HIGH
                =1  1471     ;
                =1  1472     ;    OUTPUTS: INSTRUCTION, OUR_CODE_LOW, OUR_CODE_HIGH
                =1  1473     ;
                =1  1474     ;    VARIABLES MODIFIED: DPTR, A, ERRNUM, C, INSTRUCTION
                =1  1475     ;
                =1  1476     ;    ERROR EXITS: 10H (ASSEMBLY SYNTAX)
                =1  1477     ;
                =1  1478     ;    SUBROUTINES ACCESSED DIRECTLY: ERROR
                =1  1479     ;
                =1  1480     ;*****************************************************************
                =1  1481     CALCULATE_INSTRUCTION_VALUE:
F99F 90F555     =1  1482             MOV     DPTR,#INSTRUCTION_CODE
F9A2 755F00     =1  1483             MOV     INSTRUCTION,#00H
                =1  1484     INST_VALUE_LOOP:
F9A5 E4         =1  1485             CLR     A
F9A6 93         =1  1486             MOVC    A,@A+DPTR
F9A7 055F       =1  1487             INC     INSTRUCTION
F9A9 A3         =1  1488             INC     DPTR
F9AA B54E09     =1  1489             CJNE    A,OUR_CODE_HIGH,CHECK_AND_INC_HASH_TAB
F9AD E4         =1  1490             CLR     A
F9AE 93         =1  1491             MOVC    A,@A+DPTR
F9AF A3         =1  1492             INC     DPTR
F9B0 B54F04     =1  1493             CJNE    A,OUR_CODE_LOW,CHECK_HASH_TAB   ;Second byte is high byte (CS)
F9B3 155F       =1  1494             DEC     INSTRUCTION
F9B5 22         =1  1495             RET
                =1  1496     CHECK_AND_INC_HASH_TAB:
F9B6 A3         =1  1497             INC     DPTR
                =1  1498     CHECK_HASH_TAB:
F9B7 E583       =1  1499             MOV     A,DPH
F9B9 B4F7E9     =1  1500             CJNE    A,#HIGH(INSTRUCTION_CODE+200H),INST_VALUE_LOOP
F9BC E582       =1  1501             MOV     A,DPL
F9BE B455E4     =1  1502             CJNE    A,#LOW(INSTRUCTION_CODE+200H),INST_VALUE_LOOP
F9C1 754310     =1  1503             MOV     ERRNUM,#10H                     ;Assembly syntax
F9C4 02E05F     =1  1504             JMP     ERROR
                =1  1505 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1  1506     ;*****************************************************************
                =1  1507     ;
                =1  1508     ;    NAME: UPDATE_OUR_CODE
                =1  1509     ;
                =1  1510     ;    ABSTRACT: Local procedure used to determine whether to use
                =1  1511     ;       #MNEMONIC_FACTOR (first operand) or #OPERAND_FACTOR*#MNEMONIC_FACTOR
                =1  1512     ;       (second operand) and then update the hash value, OUR_CODE.
                =1  1513     ;
                =1  1514     ;    INPUTS: OUR_CODE_LOW, OUR_CODE_HIGH, ORDINAL
                =1  1515     ;
                =1  1516     ;    OUTPUTS: OUR_CODE_LOW, OUR_CODE_HIGH
                =1  1517     ;
                =1  1518     ;    VARIABLES MODIFIED: A, B, OUR_CODE_HIGH, OUR_CODE_LOW, PARAM6
                =1  1519     ;
                =1  1520     ;    ERROR EXITS: None
                =1  1521     ;
                =1  1522     ;    SUBROUTINES ACCESSED DIRECTLY: None
                =1  1523     ;
                =1  1524     ;*****************************************************************
                =1  1525     UPDATE_OUR_CODE:
F9C7 E54E       =1  1526             MOV     A,OUR_CODE_HIGH
F9C9 7017       =1  1527             JNZ     ULO
F9CB 742C       =1  1528             MOV     A,#MNEMONIC_FACTOR              ;Set-up for <=
F9CD 855CF0     =1  1529             MOV     B,ORDINAL                       ;Set-up for MUL AB
F9D0 B54F0D     =1  1530             CJNE    A,OUR_CODE_LOW,UPDATE_LSSTHN    ;Fall through if"=",or check
                =1  1531     CONT_UPDATE_LSSTHN:                             ;for "<"
F9D3 A4         =1  1532             MUL     AB                              ;ORDINAL * #MNEMONIC_FACTOR
F9D4 85F04E     =1  1533             MOV     OUR_CODE_HIGH,B                 ;Save high order byte
F9D7 254F       =1  1534             ADD     A,OUR_CODE_LOW                  ;Add old OUR_CODE
F9D9 5002       =1  1535             JNC     UL1
F9DB 054E       =1  1536             INC     OUR_CODE_HIGH
F9DD F54F       =1  1537     UL1:    MOV     OUR_CODE_LOW,A                  ;Replace with new code
F9DF 22         =1  1538             RET                                     ;Exit
                =1  1539     UPDATE_LSSTHN:
F9E0 50F1       =1  1540             JNC     CONT_UPDATE_LSSTHN
F9E2 742C       =1  1541     ULO:    MOV     A,#MNEMONIC_FACTOR
F9E4 855CF0     =1  1542             MOV     B,ORDINAL
F9E7 A4         =1  1543             MUL     AB                              ;ORDINAL * #MNEMONIC_FACTOR
F9E8 AFF0       =1  1544             MOV     PARAM6,B
F9EA 75F018     =1  1545             MOV     B,#OPERAND_FACTOR
F9ED A4         =1  1546             MUL     AB
F9EE 254F       =1  1547             ADD     A,OUR_CODE_LOW                  ;Add old OUR_CODE
F9F0 F54F       =1  1548             MOV     OUR_CODE_LOW,A                  ;Replace with new OUR_CODE
F9F2 E5F0       =1  1549             MOV     A,B
F9F4 354E       =1  1550             ADDC    A,OUR_CODE_HIGH                 ;Add high byte of OUR_CODE
F9F6 F54E       =1  1551             MOV     OUR_CODE_HIGH,A
F9F8 EF         =1  1552             MOV     A,PARAM6                        ;Recall upper byte of first mult.
F9F9 75F018     =1  1553             MOV     B,#OPERAND_FACTOR
F9FC A4         =1  1554             MUL     AB
F9FD 254E       =1  1555             ADD     A,OUR_CODE_HIGH                 ;Add upper byte if second multiply
                =1  1556                                                     ;to upper byte of first multiply
                =1  1557                                                     ;Multiplied by OPER_FACTOR
F9FF F54E       =1  1558             MOV     OUR_CODE_HIGH,A
FA01 22         =1  1559             RET                                     ;Exit
                =1  1560 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                    =1  1561     ;*********************************************************************
                    =1  1562     ;
                    =1  1563     ;    NAME: GET_FIRST_OPERAND
                    =1  1564     ;
                    =1  1565     ;    ABSTRACT: (ORDINAL + 1)*MNEMONIC_FACTOR is added to OUR_CODE
                    =1  1566     ;       (the hash value). If the operand was a 'C', then BIT_EXP is
                    =1  1567     ;       set to 1 (true).
                    =1  1568     ;
                    =1  1569     ;    INPUTS: TOKSTR, OUR_CODE_LOW, OUR_CODE_HIGH
                    =1  1570     ;
                    =1  1571     ;    OUTPUTS: BIT_EXP, OUR_CODE_LOW, OUR_CODE_HIGH
                    =1  1572     ;
                    =1  1573     ;    VARIABLES MODIFIED: B, A, C, OUR_CODE_LOW, OUR_CODE_HIGH, PARAM6,
                    =1  1574     ;       BIT_EXP
                    =1  1575     ;
                    =1  1576     ;    ERROR EXITS: None
                    =1  1577     ;
                    =1  1578     ;    SUBROUTINES ACCESSED DIRECTLY: None
                    =1  1579     ;
                    =1  1580     ;*********************************************************************
                    =1  1581     GET_FIRST_OPERAND:
FA02 75F02C         =1  1582             MOV     B,#MNEMONIC_FACTOR
FA05 E548           =1  1583             MOV     A,TOKSTR
FA07 C3             =1  1584             CLR     C
FA08 9490           =1  1585             SUBB    A,#90H
FA0A 401B           =1  1586             JC      FIRST_NOT_REGISTER
FA0C 9408           =1  1587             SUBB    A,#08H
FA0E 5017           =1  1588             JNC     FIRST_NOT_REGISTER      ;Check if TOKSTR=REGISTER token(0-7)
FA10 E548           =1  1589             MOV     A,TOKSTR
FA12 C3             =1  1590             CLR     C
FA13 948C           =1  1591             SUBB    A,#8CH
FA15 A4             =1  1592             MUL     AB
FA16 254F           =1  1593             ADD     A,OUR_CODE_LOW
FA18 F54F           =1  1594             MOV     OUR_CODE_LOW,A
FA1A 5002           =1  1595             JNC     GE_FI_OP_1
FA1C 054E           =1  1596             INC     OUR_CODE_HIGH
                    =1  1597     GE_FI_OP_1:
FA1E E5F0           =1  1598             MOV     A,B
FA20 254E           =1  1599             ADD     A,OUR_CODE_HIGH
FA22 F54E           =1  1600             MOV     OUR_CODE_HIGH,A
FA24 02FA3F         =1  1601             JMP     SET_BIT_EXP
                    =1  1602     FIRST_NOT_REGISTER:
FA27 7410           =1  1603             MOV     A,#OFST
FA29 2440           =1  1604             ADD     A,#REG
FA2B FF             =1  1605             MOV     PARAM6,A
FA2C E548           =1  1606             MOV     A,TOKSTR
FA2E C3             =1  1607             CLR     C
FA2F 9F             =1  1608             SUBB    A,PARAM6
FA30 A4             =1  1609             MUL     AB
FA31 254F           =1  1610             ADD     A,OUR_CODE_LOW
FA33 F54F           =1  1611             MOV     OUR_CODE_LOW,A
FA35 5002           =1  1612             JNC     GE_FI_OP_2
FA37 054E           =1  1613             INC     OUR_CODE_HIGH
                    =1  1614     GE_FI_OP_2:
FA39 E5F0           =1  1615             MOV     A,B
```

```
LOC  OBJ            LINE     SOURCE

FA3B 254E       =1  1616             ADD     A,OUR_CODE_HIGH
FA3D F54E       =1  1617             MOV     OUR_CODE_HIGH,A
                =1  1618     SET_BIT_EXP:
FA3F E548       =1  1619             MOV     A,TOKSTR
FA41 B45E02     =1  1620             CJNE    A,#C_TOKE,END_FIRST_OPERAND
FA44 D202       =1  1621             SETB    BIT_EXP
                =1  1622     END_FIRST_OPERAND:
FA46 22         =1  1623             RET                     ;Exit
                =1  1624 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1  1625     ;*****************************************************************
                =1  1626     ;
                =1  1627     ;    NAME: CHECK_AND_SET_EXP_FLAG, SET_EXP_16_FLAG, SET_EXP_FLAG,
                =1  1628     ;       CHECK_EXP_FLAG, SET_POUND_EXP_FLAG, CHECK_AND_SET_SECOND_EXP_FLAG,
                =1  1629     ;       SET_SLASH_EXP_FLAG, SET_REL_FLAG
                =1  1630     ;
                =1  1631     ;    ABSTRACT:
                =1  1632     ;       CHECK_AND_SET_EXP_FLAG: Parse-time action to check to see if
                =1  1633     ;       BIT_EXP is set(1).  If so, the EXP8 is a bit EXP8 (eight-bit
                =1  1634     ;       expression), otherwise is ti a byte EXP8.  The ordinal is set
                =1  1635     ;       appropriately and UPDATE_OUR_CODE is called to update the
                =1  1636     ;       hash value, OUR_CODE.  The FIRST_EXP flag is set(1) to signify
                =1  1637     ;       that the first operand was an expression of some sort.
                =1  1638     ;       NUMBER_OF_BYTES is set to 2 to signify that it is a two byte
                =1  1639     ;       instruction so far.
                =1  1640     ;
                =1  1641     ;       SET_EXP_16_FLAG:  Parse-time action to set the ordinal to 20 to
                =1  1642     ;       show that the operand has an EXP16 ad then cal UPDATE_OUR_CODE to
                =1  1643     ;       update the hash value, OUR_CODE.  SET_NUMBER_OF_BYTES equal to
                =1  1644     ;       7 to signify that the instruction was a long jump or call or
                =1  1645     ;       MOV DPTR,EXP16.
                =1  1646     ;
                =1  1647     ;       SET_EXP_FLAG:  Parse-time prodecure to set the  ordinal equal to
                =1  1648     ;       16 to show that the operand was a byte EXP8 expression ad call
                =1  1649     ;       UPDATE_OUR_CODE to update the hash value, OUR_CODE.  Set the
                =1  1650     ;       FIRST_EXP flag to show that the first operand was an expression
                =1  1651     ;       of some sort.
                =1  1652     ;
                =1  1653     ;       CHECK_EXP_FLAG:  Parse-time action that checks the FIRST_EXP
                =1  1654     ;       flag and the SECOND_EXP flag.  by determining which are set
                =1  1655     ;       and which are not, NUMBER_OF_BYTES is set according to the
                =1  1656     ;       number of bytes in the instruction.
                =1  1657     ;               FIRST_EXP       SECOND_EXP      NUMBER_OF_BYTES
                =1  1658     ;               0               0               1
                =1  1659     ;               0               1               2
                =1  1660     ;               1               0               2
                =1  1661     ;               1               1               3
                =1  1662     ;
                =1  1663     ;       SET_POUND_EXP_FLAG:  Parse-time action to set the ordinal equal
                =1  1664     ;       to 18 to show that the operand was an immediate(#) expression.
                =1  1665     ;       update the hash value, OUR_CODE, by calling UPDATE_OUR_CODE.
                =1  1666     ;       SECOND_EXP flag is set to signify that the second operand was an
                =1  1667     ;       expression of some sort.
                =1  1668     ;
                =1  1669     ;       CHECK_AND_SET_SECOND_EXP_FLAG:  Parse-time action to set the
                =1  1670     ;       SECOND_EXP flag to signify that the second operand was an expression
                =1  1671     ;       of some sort.  The BIT_EXP flag is checked.  If set, the ordinal
                =1  1672     ;       is set equal to 17 to show that the operand was a bit EXP8.  If
                =1  1673     ;       it was not set, the ordinal is set to 16 to show that the operand
                =1  1674     ;       was a byte EXP8.  The hash value is updated by calling UPDATE_OUR_CODE.
                =1  1675     ;
                =1  1676     ;       SET_SLASH_EXP_FLAG:  Parse-time action to set the ordinal equal to 19
                =1  1677     ;       to show that the operand was the complement(/) of a bit expression.
                =1  1678     ;       update the hash value, OUR_CODE, by calling UPDATE_OUR_CODE.
                =1  1679     ;       SECOND_EXP is set to signify that the second operand was an expression
```

```
LOC  OBJ            LINE     SOURCE

                =1  1680     ;       of some sort.
                =1  1681     ;
                =1  1682     ;       SET_REL_FLAG:  Parse-time action to set the ordinal equal to 22 to
                =1  1683     ;       show that the operand was a relative offset(EXP8).  The hash value,
                =1  1684     ;       OUR_CODE, is updated by calling UPDATE_OUR_CODE.  Set NUMBER_OF_BYTES
                =1  1685     ;       equal to 4 to signify that it was a jump instruction with a relative
                =1  1686     ;       operand.
                =1  1687     ;
                =1  1688     ;    INPUTS: BIT_EXP, OUR_CODE_LOW, OUR_CODE_HIGH, FIRST_EXP, SECOND_EXP
                =1  1689     ;
                =1  1690     ;    OUTPUTS: NUMBER_OF_BYTES, ORDINAL, FIRST_EXP, SECOND_EXP, OUR_CODE_LOW,
                =1  1691     ;       OUR_CODE_HIGH
                =1  1692     ;
                =1  1693     ;    VARIABLES MODIFIED: ORDINAL, FIRST_EXP, NUMBER_OF_BYTES, SECOND_EXP,
                =1  1694     ;       A, C, B, DPTR
                =1  1695     ;
                =1  1696     ;    ERROR EXITS: None
                =1  1697     ;
                =1  1698     ;    SUBROUTINES ACCESSED DIRECTLY: UPDATE_OUR_CODE
                =1  1699     ;
                =1  1700     ;*********************************************************************
                =1  1701     CHECK_AND_SET_EXP_FLAG:
FA47 755C10     =1  1702             MOV     ORDINAL,#10H            ;In case no bit 8
FA4A 300202     =1  1703             JNB     BIT_EXP,NO_BIT_8
FA4D 055C       =1  1704             INC     ORDINAL                 ;Bit 8 occurance
                =1  1705     NO_BIT_8:
FA4F 31C7       =1  1706             CALL    UPDATE_OUR_CODE
FA51 D203       =1  1707             SETB    FIRST_EXP
FA53 754D02     =1  1708             MOV     NUMBER_OF_BYTES,#02H    ;Two bytes so far
FA56 22         =1  1709             RET                             ;Exit
                =1  1710     ;*********************************************************************
                =1  1711     SET_EXP_16_FLAG:
FA57 755C14     =1  1712             MOV     ORDINAL,#14H
FA5A 31C7       =1  1713             CALL    UPDATE_OUR_CODE
FA5C 754D07     =1  1714             MOV     NUMBER_OF_BYTES,#07H    ;To signify an EXP16 instruction
FA5F 22         =1  1715             RET                             ;Exit
                =1  1716     ;*********************************************************************
                =1  1717     SET_EXP_FLAG:
FA60 755C10     =1  1718             MOV     ORDINAL,#10H
FA63 31C7       =1  1719             CALL    UPDATE_OUR_CODE
FA65 D203       =1  1720             SETB    FIRST_EXP               ;First operand of an exrpession
FA67 22         =1  1721             RET
                =1  1722     ;*********************************************************************
                =1  1723     CHECK_EXP_FLAG:
FA68 E4         =1  1724             CLR     A
FA69 A203       =1  1725             MOV     C,FIRST_EXP
FA6B 33         =1  1726             RLC     A
FA6C A204       =1  1727             MOV     C,SECOND_EXP
FA6E 33         =1  1728             RLC     A
FA6F 75F004     =1  1729             MOV     B,#04H
FA72 90FA77     =1  1730             MOV     DPTR,#EXP_FLAG_TABLE
FA75 A4         =1  1731             MUL     AB
FA76 73         =1  1732             JMP     @A+DPTR
                =1  1733     EXP_FLAG_TABLE:
FA77 754D01     =1  1734             MOV     NUMBER_OF_BYTES,#01H
```

```
LOC  OBJ            LINE     SOURCE

FA7A 22          =1  1735             RET
FA7B 754D02      =1  1736             MOV     NUMBER_OF_BYTES,#02H
FA7E 22          =1  1737             RET
FA7F 754D02      =1  1738             MOV     NUMBER_OF_BYTES,#02H
FA82 22          =1  1739             RET
FA83 754D03      =1  1740             MOV     NUMBER_OF_BYTES,#03H
FA86 22          =1  1741             RET                             ;Exit
                 =1  1742     ;*********************************************************************
                 =1  1743     SET_POUND_EXP_FLAG:
FA87 755C12      =1  1744             MOV     ORDINAL,#12H
FA8A 31C7        =1  1745             CALL    UPDATE_OUR_CODE
FA8C D204        =1  1746             SETB    SECOND_EXP
FA8E 22          =1  1747             RET                             ;Exit
                 =1  1748     ;*********************************************************************
                 =1  1749     CHECK_AND_SET_SECOND_EXP_FLAG:
FA8F D204        =1  1750             SETB    SECOND_EXP
FA91 7410        =1  1751             MOV     A,#10H
FA93 300201      =1  1752             JNB     BIT_EXP,SECOND_NO_BIT_8
FA96 04          =1  1753             INC     A
                 =1  1754     SECOND_NO_BIT_8:
FA97 F55C        =1  1755             MOV     ORDINAL,A
FA99 21C7        =1  1756             JMP     UPDATE_OUR_CODE
                 =1  1757     ;*********************************************************************
                 =1  1758     SET_SLASH_EXP_FLAG:
FA9B 755C13      =1  1759             MOV     ORDINAL,#13H            ;Complement of a bit expression
FA9E 31C7        =1  1760             CALL    UPDATE_OUR_CODE
FAA0 D204        =1  1761             SETB    SECOND_EXP
FAA2 22          =1  1762             RET                             ;Exit
                 =1  1763     ;*********************************************************************
                 =1  1764     SET_REL_FLAG:
FAA3 755C16      =1  1765             MOV     ORDINAL,#16H            ;Relative offset
FAA6 31C7        =1  1766             CALL    UPDATE_OUR_CODE
FAA8 754D04      =1  1767             MOV     NUMBER_OF_BYTES,#04H    ;Jump instruction with relative operand
FAAB 22          =1  1768             RET                             ;Exit
                 =1  1769 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                    =1  1770     ;*********************************************************************
                    =1  1771     ;
                    =1  1772     ;    NAME: GET_SECOND_EXP
                    =1  1773     ;
                    =1  1774     ;    ABSTRACT: (#MNEMONIC_FACTOR* #OPERAND_FACTOR) is added to the
                    =1  1775     ;       hash value, OUR_CODE.  If the operand was a 'C', then OUR_CODE
                    =1  1776     ;       must be re-calculated to allow for a bit EXP8 instead of a byte
                    =1  1777     ;       EXP8.
                    =1  1778     ;
                    =1  1779     ;    INPUTS: OUR_CODE_LOW, OUR_CODE_HIGH, TOKSTR
                    =1  1780     ;
                    =1  1781     ;    OUTPUTS: OUR_CODE_LOW, OUR_CODE_HIGH
                    =1  1782     ;
                    =1  1783     ;    VARIABLES MODIFIED: B, A, C, PARAM6, OUR_CODE_LOW, OUR_CODE_HIGH
                    =1  1784     ;
                    =1  1785     ;    ERROR EXITS: None
                    =1  1786     ;
                    =1  1787     ;    SUBROUTINES ACCESSED DIRECTLY: None
                    =1  1788     ;
                    =1  1789     ;*********************************************************************
                    =1  1790     GET_SECOND_OPERAND:
FAAC 75F02C         =1  1791             MOV     B,#MNEMONIC_FACTOR
FAAF E548           =1  1792             MOV     A,TOKSTR
FAB1 C3             =1  1793             CLR     C
FAB2 9490           =1  1794             SUBB    A,#90H
FAB4 4025           =1  1795             JC      SECOND_NOT_REGISTER
FAB6 9408           =1  1796             SUBB    A,#08H
FAB8 5021           =1  1797             JNC     SECOND_NOT_REGISTER     ;Check if TOKSTR=REGISTER token(0-7)
FABA E548           =1  1798             MOV     A,TOKSTR
FABC C3             =1  1799             CLR     C
FABD 948C           =1  1800             SUBB    A,#8CH
FABF A4             =1  1801             MUL     AB
FAC0 AFF0           =1  1802             MOV     PARAM6,B
FAC2 75F018         =1  1803             MOV     B,#OPERAND_FACTOR
FAC5 A4             =1  1804             MUL     AB
FAC6 254F           =1  1805             ADD     A,OUR_CODE_LOW
FAC8 F54F           =1  1806             MOV     OUR_CODE_LOW,A
FACA E5F0           =1  1807             MOV     A,B
FACC 354E           =1  1808             ADDC    A,OUR_CODE_HIGH
FACE F54E           =1  1809             MOV     OUR_CODE_HIGH,A
FAD0 EF             =1  1810             MOV     A,PARAM6
FAD1 75F018         =1  1811             MOV     B,#OPERAND_FACTOR
FAD4 A4             =1  1812             MUL     AB
FAD5 254E           =1  1813             ADD     A,OUR_CODE_HIGH
FAD7 F54E           =1  1814             MOV     OUR_CODE_HIGH,A
FAD9 8023           =1  1815             SJMP    OPERAND_C
                    =1  1816     SECOND_NOT_REGISTER:
FADB 7410           =1  1817             MOV     A,#OFST
FADD 2440           =1  1818             ADD     A,#REG
FADF FF             =1  1819             MOV     PARAM6,A
FAE0 E548           =1  1820             MOV     A,TOKSTR
FAE2 C3             =1  1821             CLR     C
FAE3 9F             =1  1822             SUBB    A,PARAM6
FAE4 A4             =1  1823             MUL     AB
FAE5 AFF0           =1  1824             MOV     PARAM6,B
```

```
LOC  OBJ            LINE     SOURCE

FAE7 75F018     =1  1825             MOV     B,#OPERAND_FACTOR
FAEA A4         =1  1826             MUL     AB
FAEB 254F       =1  1827             ADD     A,OUR_CODE_LOW
FAED F54F       =1  1828             MOV     OUR_CODE_LOW,A
FAEF E5F0       =1  1829             MOV     A,B
FAF1 354E       =1  1830             ADDC    A,OUR_CODE_HIGH
FAF3 F54E       =1  1831             MOV     OUR_CODE_HIGH,A
FAF5 EF         =1  1832             MOV     A,PARAM6
FAF6 75F018     =1  1833             MOV     B,#OPERAND_FACTOR
FAF9 A4         =1  1834             MUL     AB
FAFA 254E       =1  1835             ADD     A,OUR_CODE_HIGH
FAFC F54E       =1  1836             MOV     OUR_CODE_HIGH,A
                =1  1837     OPERAND_C:
FAFE E54E       =1  1838             MOV     A,OUR_CODE_HIGH
FB00 B43C08     =1  1839             CJNE    A,#03CH,END_SECOND_OPERAND
FB03 E54F       =1  1840             MOV     A,OUR_CODE_LOW
FB05 B48F03     =1  1841             CJNE    A,#08FH,END_SECOND_OPERAND
FB08 754FBB     =1  1842             MOV     OUR_CODE_LOW,#0BBH
                =1  1843     END_SECOND_OPERAND:
FB0B 22         =1  1844             RET                     ;EXIT
                    1845 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                    1846 +1  $INCLUDE(:F1:ASMA.INC)
                  =1  1847     ;*****************************************************************
                  =1  1848     ;
                  =1  1849     ;       This is the include file called ASMA.INC.  It contains the
                  =1  1850     ;       following subroutines in order:
                  =1  1851     ;
                  =1  1852     ;               CHECK_AND_CHANGE_ASM_PC
                  =1  1853     ;               CHANGE_TO_INSTRUCTION_OP
                  =1  1854     ;
                  =1  1855     ;*****************************************************************
                  =1  1856 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1  1857     ;*******************************************************************************
                =1  1858     ;
                =1  1859     ;    NAME: CHECK_AND_CHANGE_ASM_PC
                =1  1860     ;
                =1  1861     ;    ABSTRACT: Change the ASM_PC according to NUMBER_OF_BYTES
                =1  1862     ;       and check to make sure it does not wrap around.
                =1  1863     ;
                =1  1864     ;    INPUTS: NUMBER_OF_BYTES, ASM_PC_LOW, ASM_PC_HIGH
                =1  1865     ;
                =1  1866     ;    OUTPUTS: ASM_PC_LOW, ASM_PC_HIGH
                =1  1867     ;
                =1  1868     ;    VARIABLES MODIFIED: A, PARAM1, ASM_PC_HIGH, ASM_PC_LOW, ERRNUM
                =1  1869     ;
                =1  1870     ;    ERROR EXITS: 13H (ASM PC>OFFFH)
                =1  1871     ;
                =1  1872     ;    SUBROUTINES ACCESSED DIRECTLY: ERROR
                =1  1873     ;
                =1  1874     ;*******************************************************************************
                =1  1875
                =1  1876     CHECK_AND_CHANGE_ASM_PC:
FB0C E54D       =1  1877             MOV     A,NUMBER_OF_BYTES
FB0E 254C       =1  1878             ADD     A,ASM_PC_LOW
FB10 FA         =1  1879             MOV     PARAM1,A                ;Save to put in ASM_PC_LOW
FB11 E4         =1  1880             CLR     A
FB12 354B       =1  1881             ADDC    A,ASM_PC_HIGH           ;Add 1 to ASM_PC_HIGH if carry set
FB14 5006       =1  1882             JNC     CHANGE_ASM_PC_1         ;Error if carry set after add
FB16 754313     =1  1883             MOV     ERRNUM,#13H             ;ASM PC > OFFFH
FB19 02E05F     =1  1884             JMP     ERROR
                =1  1885     CHANGE_ASM_PC_1:
FB1C F54B       =1  1886             MOV     ASM_PC_HIGH,A
FB1E 8A4C       =1  1887             MOV     ASM_PC_LOW,PARAM1       ;Replace ASM_PC with new value
FB20 22         =1  1888             RET
                =1  1889 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1  1890     ;*******************************************************************************
                =1  1891     ;
                =1  1892     ;    NAME: CHANGE_TO_INSTRUCTION_OP
                =1  1893     ;
                =1  1894     ;    ABSTRACT: Run time action used to process the one, two or three bytes of
                =1  1895     ;       the assembled instruction and write it out to memory.  The assembly
                =1  1896     ;       program counter (ASM_PC) is updated according to the number of bytes
                =1  1897     ;       in the instruction.  A case statement will take care of all the
                =1  1898     ;       different types of instructions.  The byte(s) of the instruction are
                =1  1899     ;       stored in the appropriate order in a working area, WORKING_SPACE (3).
                =1  1900     ;       The opcode is always put in the fist byte.  If the instruction is
                =1  1901     ;       other than a one byte instruction, the other bytes are obtained from
                =1  1902     ;       VALLOW, VALHGH or TEMP_SEC as necessary.  NUMBER_OF_BYTES is updated
                =1  1903     ;       to reflect the number of bytes in the instruction to be written out
                =1  1904     ;       to memory and the ASM_PC is updated.  The individual cases are as
                =1  1905     ;       follows:
                =1  1906     ;
                =1  1907     ;       Case 1:  One byte instructions (ex. NOP)
                =1  1908     ;
                =1  1909     ;       Case 2:  Two byte instructions (ex. MOV R7,#DATA)
                =1  1910     ;                Put expression in second byte.
                =1  1911     ;
                =1  1912     ;       Case 3:  Three byte instructions (ex. MOV EXP8,#EXP)
                =1  1913     ;                Put the first expression in the second byte.
                =1  1914     ;                put the second expression in the third byte.
                =1  1915     ;
                =1  1916     ;       Case 4:  Jump instruction with one relative operand (ex. JC REL. OPER.)
                =1  1917     ;                Calculate the relative offset and put it in the second byte.
                =1  1918     ;
                =1  1919     ;       Case 5:  Jump instruction with an expression as the first operand
                =1  1920     ;                and a relative operand as the second operand
                =1  1921     ;                (ex. JNB EXP8,REL. OPER.)
                =1  1922     ;                Put the expression in the second byte, calculate the relative
                =1  1923     ;                offset and put it in the third byte.
                =1  1924     ;
                =1  1925     ;       Case 6:  Absolute call or jump instruction (ex. ACALL EXP11).
                =1  1926     ;                Calculate the 2K jump or call and incorporate it into the
                =1  1927     ;                opcode.  Put the  lower 8 bits of EXP11 in the second byte.
                =1  1928     ;
                =1  1929     ;       Case 7:  Long jump or call instruction or MOV DPTR,EXP16
                =1  1930     ;                (ex. LJMP EXP16).
                =1  1931     ;                The high byte of EXP16 is put in the second byte.  The low
                =1  1932     ;                byte of EXP16 is put in the third byte.
                =1  1933     ;
                =1  1934     ;    INPUTS: VALHGH, VALLOW, TEMP_SEC, INSTRUCTION_VALUE
                =1  1935     ;
                =1  1936     ;    OUTPUTS: Memory at address of ASM_PC
                =1  1937     ;
                =1  1938     ;    VARIABLES MODIFIED: NUMBER_OF_BYTES, REL_OFFSET_LOW, REL_OFFSET_HIGH,
                =1  1939     ;       A, ERRNUM, OLD_ASM_PC_HIGH, OLD_ASM_PC_LOW, POINTO, TEMP_SEC, C,
                =1  1940     ;       TEMP_LOW, SELECT, PNTLOW, PNTHGH, ASM_PC_HIGH, ASM_PC_LOW
                =1  1941     ;
                =1  1942     ;    ERROR EXITS: 10H (ASSEMBLY SYNTAX)
                =1  1943     ;                 11H (ADDRESS OUT OF RANGE-11 BIT ABSOLUTE OFFSET)
                =1  1944     ;                 12H (ADDRESS OUT OF RANGE-8 BIT RELATIVE OFFSET)
```

```
LOC  OBJ            LINE     SOURCE

                    =1  1945     ;
                    =1  1946     ;    SUBROUTINES ACCESSED DIRECTLY: CHECK_AND_CHANGE_ASM_PC, ERROR
                    =1  1947     ;
                    =1  1948     ;*******************************************************************************
                    =1  1949     CHANGE_TO_INSTRUCTION_OP:
FB21 854B5D         =1  1950             MOV     OLD_ASM_PC_HIGH,ASM_PC_HIGH
FB24 854C5E         =1  1951             MOV     OLD_ASM_PC_LOW,ASM_PC_LOW
FB27 E54D           =1  1952             MOV     A,NUMBER_OF_BYTES
FB29 B40109         =1  1953             CJNE    A,#01H,CHANGE_CASE_2            ;Change case 1
FB2C 710C           =1  1954             CALL    CHECK_AND_CHANGE_ASM_PC         ;Update ASM PC
FB2E 7840           =1  1955             MOV     POINTO,#WORKING_SPACE           ;Get opcode
FB30 A65F           =1  1956             MOV     @POINTO,INSTRUCTION
FB32 02FC73         =1  1957             JMP     CHANGE_END
                    =1  1958     CHANGE_CASE_2:
FB35 B4020C         =1  1959             CJNE    A,#02H,CHANGE_CASE_3
FB38 710C           =1  1960             CALL    CHECK_AND_CHANGE_ASM_PC
FB3A 7840           =1  1961             MOV     POINTO,#WORKING_SPACE
FB3C A65F           =1  1962             MOV     @POINTO,INSTRUCTION             ;Put opcode in 1st byte
FB3E 08             =1  1963             INC     POINTO
FB3F A64A           =1  1964             MOV     @POINTO,VALLOW
FB41 02FC73         =1  1965             JMP     CHANGE_END
                    =1  1966     CHANGE_CASE_3:
FB44 B4031B         =1  1967             CJNE    A,#03H,CHANGE_CASE_4
FB47 710C           =1  1968             CALL    CHECK_AND_CHANGE_ASM_PC
FB49 7840           =1  1969             MOV     POINTO,#WORKING_SPACE
FB4B A65F           =1  1970             MOV     @POINTO,INSTRUCTION             ;Put opcode in 1st byte
FB4D E55F           =1  1971             MOV     A,INSTRUCTION
FB4F B48506         =1  1972             CJNE    A,#85H,CASE_3_MORE
FB52 E562           =1  1973             MOV     A,TEMP_SEC
FB54 C54A           =1  1974             XCH     A,VALLOW
FB56 F562           =1  1975             MOV     TEMP_SEC,A
                    =1  1976     CASE_3_MORE:
FB58 7841           =1  1977             MOV     POINTO,#(WORKING_SPACE+1)
FB5A A662           =1  1978             MOV     @POINTO,TEMP_SEC
FB5C 08             =1  1979             INC     POINTO
FB5D A64A           =1  1980             MOV     @POINTO,VALLOW
FB5F 02FC73         =1  1981             JMP     CHANGE_END
                    =1  1982     CHANGE_CASE_4:
FB62 B40460         =1  1983             CJNE    A,#04H,CHANGE_CASE_5
FB65 754312         =1  1984             MOV     ERRNUM,#12H                     ;Adr out of range-8 bit
FB68 754D02         =1  1985             MOV     NUMBER_OF_BYTES,#02H            ;2 byte instruction
FB6B 710C           =1  1986             CALL    CHECK_AND_CHANGE_ASM_PC
FB6D 854A61         =1  1987             MOV     REL_OFFSET_LOW,VALLOW
FB70 854960         =1  1988             MOV     REL_OFFSET_HIGH,VALHGH          ;Move value into relative offset
FB73 E560           =1  1989             MOV     A,REL_OFFSET_HIGH
FB75 B54B03         =1  1990             CJNE    A,ASM_PC_HIGH,CHANGE_CASE_4A
FB78 02FB80         =1  1991             JMP     CHANGE_CASE_4AA
                    =1  1992     CHANGE_CASE_4A:
FB7B 4024           =1  1993             JC      BACKWARD_JUMP_CASE_4
FB7D 02FB87         =1  1994             JMP     FORWARD_JUMP_CASE_4
                    =1  1995     CHANGE_CASE_4AA:
FB80 E561           =1  1996             MOV     A,REL_OFFSET_LOW
FB82 B54C00         =1  1997             CJNE    A,ASM_PC_LOW,CHANGE_CASE_4C
                    =1  1998     CHANGE_CASE_4C:
FB85 401A           =1  1999             JC      BACKWARD_JUMP_CASE_4            ;Jump if rel. offset if < ASM_PC
```

```
LOC  OBJ            LINE     SOURCE

                    =1  2000     FORWARD_JUMP_CASE_4:
FB87 C3             =1  2001             CLR     C
FB88 E561           =1  2002             MOV     A,REL_OFFSET_LOW
FB8A 954C           =1  2003             SUBB    A,ASM_PC_LOW
FB8C F561           =1  2004             MOV     REL_OFFSET_LOW,A
FB8E E560           =1  2005             MOV     A,REL_OFFSET_HIGH
FB90 954B           =1  2006             SUBB    A,ASM_PC_HIGH              ;Subtract ASM_PC from relative offset
FB92 7067           =1  2007             JNZ     CHANGE_ERROR               ;Error if relative offset > OFFH
FB94 747F           =1  2008             MOV     A,#7FH
FB96 B56100         =1  2009             CJNE    A,REL_OFFSET_LOW,CHANGE_CASE_4D
                    =1  2010     CHANGE_CASE_4D:
FB99 4060           =1  2011             JC      CHANGE_ERROR               ;Error if relative offset > 7FH
FB9B 7841           =1  2012             MOV     POINTO,#(WORKING_SPACE+1)
FB9D A661           =1  2013             MOV     @POINTO,REL_OFFSET_LOW     ;Move offset into WORKING_SPACE (1)
FB9F 801D           =1  2014             SJMP    CHANGE_CASE_4_END
                    =1  2015
                    =1  2016     BACKWARD_JUMP_CASE_4:
FBA1 C3             =1  2017             CLR     C
FBA2 E54C           =1  2018             MOV     A,ASM_PC_LOW
FBA4 9561           =1  2019             SUBB    A,REL_OFFSET_LOW
FBA6 F561           =1  2020             MOV     REL_OFFSET_LOW,A
FBA8 E54B           =1  2021             MOV     A,ASM_PC_HIGH
FBAA 9560           =1  2022             SUBB    A,REL_OFFSET_HIGH          ;Subtract rel. offset from ASM_PC
FBAC F560           =1  2023             MOV     REL_OFFSET_HIGH,A
FBAE 704B           =1  2024             JNZ     CHANGE_ERROR               ;Error if relative offset > OFFH
FBB0 7480           =1  2025             MOV     A,#80H
FBB2 B56100         =1  2026             CJNE    A,REL_OFFSET_LOW,CHANGE_CASE_4F
                    =1  2027     CHANGE_CASE_4F:
FBB5 4044           =1  2028             JC      CHANGE_ERROR               ;Error if relative offset is > 80H
FBB7 7841           =1  2029             MOV     POINTO,#(WORKING_SPACE+1)
FBB9 E561           =1  2030             MOV     A,REL_OFFSET_LOW
FBBB F4             =1  2031             CPL     A
FBBC 04             =1  2032             INC     A
FBBD F6             =1  2033             MOV     @POINTO,A                  ;Move REL_OFFSET_LOW into WORKING_SPACE
                    =1  2034
                    =1  2035     CHANGE_CASE_4_END:
FBBE 7840           =1  2036             MOV     POINTO,#WORKING_SPACE
FBC0 A65F           =1  2037             MOV     @POINTO,INSTRUCTION        ;Move 8-bit inst into WORKING_SPACE (1)
FBC2 02FC73         =1  2038             JMP     CHANGE_END
                    =1  2039     CHANGE_CASE_5:
FBC5 B4056D         =1  2040             CJNE    A,#05H,CHANGE_CASE_6
FBC8 754312         =1  2041             MOV     ERRNUM,#12H                ;Adr out of range - 8 bit
FBCB 754D03         =1  2042             MOV     NUMBER_OF_BYTES,#03H       ;3 byte instruction
FBCE 710C           =1  2043             CALL    CHECK_AND_CHANGE_ASM_PC    ;Update ASM PC
FBD0 7840           =1  2044             MOV     POINTO,#WORKING_SPACE
FBD2 A65F           =1  2045             MOV     @POINTO,INSTRUCTION        ;Move instruction into WORKING_SPACE (0)
FBD4 854A61         =1  2046             MOV     REL_OFFSET_LOW,VALLOW
FBD7 854960         =1  2047             MOV     REL_OFFSET_HIGH,VALHGH     ;Move value into relative offset
FBDA E560           =1  2048             MOV     A,REL_OFFSET_HIGH
FBDC B54B03         =1  2049             CJNE    A,ASM_PC_HIGH,CHANGE_CASE_5A
FBDF 02FBE7         =1  2050             JMP     CHANGE_CASE_5AA
                    =1  2051     CHANGE_CASE_5A:
FBE2 402D           =1  2052             JC      BACKWARD_JUMP_CASE_5
FBE4 02FBEE         =1  2053             JMP     FORWARD_JUMP_CASE_5
                    =1  2054     CHANGE_CASE_5AA:
```

```
LOC  OBJ            LINE     SOURCE

FBE7 E561       =1  2055             MOV     A,REL_OFFSET_LOW
FBE9 B54C00     =1  2056             CJNE    A,ASM_PC_LOW,CHANGE_CASE_5C
                =1  2057     CHANGE_CASE_5C:
FBEC 4023       =1  2058             JC      BACKWARD_JUMP_CASE_5
                =1  2059     FORWARD_JUMP_CASE_5:
FBEE C3         =1  2060             CLR     C
FBEF E561       =1  2061             MOV     A,REL_OFFSET_LOW
FBF1 954C       =1  2062             SUBB    A,ASM_PC_LOW
FBF3 F561       =1  2063             MOV     REL_OFFSET_LOW,A
FBF5 E560       =1  2064             MOV     A,REL_OFFSET_HIGH
FBF7 954B       =1  2065             SUBB    A,ASM_PC_HIGH                ;Subtract ASM_PC from dest. addr
                =1  2066                                                  ;and place in relative offset
FBF9 6009       =1  2067             JZ      FJC_5_CONTINUE               ;Error if relative offset < OFFH
                =1  2068     CHANGE_ERROR:
FBFB 855D4B     =1  2069             MOV     ASM_PC_HIGH,OLD_ASM_PC_HIGH
FBFE 855E4C     =1  2070             MOV     ASM_PC_LOW,OLD_ASM_PC_LOW
FC01 02E05F     =1  2071             JMP     ERROR
                =1  2072     FJC_5_CONTINUE:
FC04 747F       =1  2073             MOV     A,#7FH
FC06 B56100     =1  2074             CJNE    A,REL_OFFSET_LOW,CHANGE_CASE_5D
                =1  2075     CHANGE_CASE_5D:
FC09 40F0       =1  2076             JC      CHANGE_ERROR                 ;Error if relative offset < 07FH
FC0B 7842       =1  2077             MOV     POINT0,#(WORKING_SPACE+2)
FC0D A661       =1  2078             MOV     @POINT0,REL_OFFSET_LOW       ;Move REL_OFFSET_LOW into WORKING_SPACE
FC0F 801D       =1  2079             SJMP    CHANGE_CASE_5_END
                =1  2080
                =1  2081     BACKWARD_JUMP_CASE_5:
FC11 C3         =1  2082             CLR     C
FC12 E54C       =1  2083             MOV     A,ASM_PC_LOW
FC14 9561       =1  2084             SUBB    A,REL_OFFSET_LOW
FC16 F561       =1  2085             MOV     REL_OFFSET_LOW,A
FC18 E54B       =1  2086             MOV     A,ASM_PC_HIGH
FC1A 9560       =1  2087             SUBB    A,REL_OFFSET_HIGH            ;Subtract relative offset from ASM_PC
FC1C F560       =1  2088             MOV     REL_OFFSET_HIGH,A            ;and store in relative offset
FC1E 70DB       =1  2089             JNZ     CHANGE_ERROR                 ;Error if relative offset > OFFH
FC20 7480       =1  2090             MOV     A,#80H
FC22 B56100     =1  2091             CJNE    A,REL_OFFSET_LOW,CHANGE_CASE_5F
                =1  2092     CHANGE_CASE_5F:
FC25 40D4       =1  2093             JC      CHANGE_ERROR                 ;Error if relative offset > 080H
FC27 7842       =1  2094             MOV     POINT0,#(WORKING_SPACE+2)
FC29 E561       =1  2095             MOV     A,REL_OFFSET_LOW
FC2B F4         =1  2096             CPL     A
FC2C 04         =1  2097             INC     A
FC2D F6         =1  2098             MOV     @POINT0,A                    ;Move REL_OFFSET_LOW into WORKING_SPACE
                =1  2099     CHANGE_CASE_5_END:
FC2E 7841       =1  2100             MOV     POINT0,#(WORKING_SPACE+1)
FC30 A662       =1  2101             MOV     @POINT0,TEMP_SEC             ;Move TEMP_LOW into WORKING_SPACE (1)
FC32 02FC73     =1  2102             JMP     CHANGE_END
                =1  2103     CHANGE_CASE_6:
FC35 B40626     =1  2104             CJNE    A,#06H,CHANGE_CASE_7
FC38 754D02     =1  2105             MOV     NUMBER_OF_BYTES,#02H         ;2 byte instruction
FC3B 710C       =1  2106             CALL    CHECK_AND_CHANGE_ASM_PC
FC3D E549       =1  2107             MOV     A,VALHGH                     ;Move value into TEMP
FC3F 54F8       =1  2108             ANL     A,#0F8H                      ;Use 3 top bits of 11 to determine
FC41 F547       =1  2109             MOV     TEMP_LOW,A                   ;which 2k page JMP or CALL it is
```

```
LOC  OBJ            LINE     SOURCE

FC43 74F8       =1  2110             MOV     A,#0F8H
FC45 554B       =1  2111             ANL     A,ASM_PC_HIGH
FC47 754311     =1  2112             MOV     ERRNUM,#11H                ;Adr out of range (11 bit)
FC4A B547AE     =1  2113             CJNE    A,TEMP_LOW,CHANGE_ERROR
FC4D 7840       =1  2114             MOV     POINTO,#WORKING_SPACE
FC4F E549       =1  2115             MOV     A,VALHGH                   ;TEMP HIGH <= 07
FC51 5407       =1  2116             ANL     A,#07H
FC53 C4         =1  2117             SWAP    A
FC54 23         =1  2118             RL      A                          ;TEMP_HIGH now rotated right 3X
FC55 255F       =1  2119             ADD     A,INSTRUCTION
FC57 F6         =1  2120             MOV     @POINTO,A                  ;Put result in WORKING_SPACE (0)
FC58 08         =1  2121             INC     POINTO
FC59 A64A       =1  2122             MOV     @POINTO,VALLOW             ;TEMP_LOW stored in WORKING_SPACE (1)
FC5B 02FC73     =1  2123             JMP     CHANGE_END                 ;truncates to 8 bits
                =1  2124     CHANGE_CASE_7:
FC5E 754310     =1  2125             MOV     ERRNUM,#10H                ;Assembly syntax
FC61 B40797     =1  2126             CJNE    A,#07H,CHANGE_ERROR        ;Error if orig NUMBER_OF_BYTES > 7
FC64 754D03     =1  2127             MOV     NUMBER_OF_BYTES,#03H       ;3 byte instruction
FC67 710C       =1  2128             CALL    CHECK_AND_CHANGE_ASM_PC
FC69 7840       =1  2129             MOV     POINTO,#WORKING_SPACE
FC6B A65F       =1  2130             MOV     @POINTO,INSTRUCTION        ;Store instruction in WORKING_SPACE (0)
FC6D 08         =1  2131             INC     POINTO
FC6E A649       =1  2132             MOV     @POINTO,VALHGH             ;Store VALHGH in WORKING_SPACE (1)
FC70 08         =1  2133             INC     POINTO
FC71 A64A       =1  2134             MOV     @POINTO,VALLOW             ;Store VALLOW in WORKING_SPACE (2)
                =1  2135     CHANGE_END:
FC73 754600     =1  2136             MOV     SELECT,#00H                ;Select external ROM
FC76 855E45     =1  2137             MOV     PNTLOW,OLD_ASM_PC_LOW
FC79 855D44     =1  2138             MOV     PNTHGH,OLD_ASM_PC_HIGH     ;Load pointer for store
FC7C 855E4C     =1  2139             MOV     ASM_PC_LOW,OLD_ASM_PC_LOW
FC7F 855D4B     =1  2140             MOV     ASM_PC_HIGH,OLD_ASM_PC_HIGH
FC82 7840       =1  2141             MOV     POINTO,#WORKING_SPACE
                =1  2142     CHANGE_END_LOOP:
FC84 E6         =1  2143             MOV     A,@POINTO                  ;Parameter to be stored
FC85 FA         =1  2144             MOV     PARAM1,A
FC86 12E04D     =1  2145             CALL    STORE
FC89 08         =1  2146             INC     POINTO
FC8A 0545       =1  2147             INC     PNTLOW
FC8C E545       =1  2148             MOV     A,PNTLOW
FC8E 7002       =1  2149             JNZ     CHANGE_END_A
FC90 0544       =1  2150             INC     PNTHGH
                =1  2151     CHANGE_END_A:
FC92 D54DEF     =1  2152             DJNZ    NUMBER_OF_BYTES,CHANGE_END_LOOP
                =1  2153                                                ;Store until NUMBER_OF_BYTES=0
FC95 85454C     =1  2154             MOV     ASM_PC_LOW,PNTLOW
FC98 85444B     =1  2155             MOV     ASM_PC_HIGH,PNTHGH
FC9B 22         =1  2156             RET                                ;End of change routine
                    2157 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                    2158 +1  $INCLUDE(:F1:SDKDSM.INC)
                  =1  2159 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                    =1  2160     ;*****************************************************************
                    =1  2161     ;
                    =1  2162     ;    NAME: DISASSEMBLY_CMD
                    =1  2163     ;
                    =1  2164     ;    ABSTRACT: This routine gets a token and partition and displays
                    =1  2165     ;       <address>=.  It then gets a byte of memory from code memory,
                    =1  2166     ;       searches the hash table for a match to that byte and dissassembles
                    =1  2167     ;       it if one is found.
                    =1  2168     ;
                    =1  2169     ;    INPUTS: None
                    =1  2170     ;
                    =1  2171     ;    OUTPUTS: None
                    =1  2172     ;
                    =1  2173     ;    VARIABLES MODIFIED: PARAM1, PARAM2, MEMORY_TRACE_ADDR_LOW,
                    =1  2174     ;       MEMORY_TRACE_ADDR_HIGH, A, POINT1, PNTLOW, PNTHGH, SELECT,
                    =1  2175     ;       TEMP_LOW, POINTO, PARTIT_LO_HIGH
                    =1  2176     ;
                    =1  2177     ;    ERROR EXITS: None
                    =1  2178     ;
                    =1  2179     ;    SUBROUTINES ACCESSED DIRECTLY: GETOKE, GET_PART, EOL_CHECK,
                    =1  2180     ;       NEWLINE, LSTWRD, CO, FETCH, GET_HASH_VALUE, DISASSEMBLE,
                    =1  2181     ;       CONTINUATION_LINE, WAIT_FOR_USER
                    =1  2182     ;
                    =1  2183     ;*****************************************************************
                    =1  2184     DISASSEMBLY_CMD:
FC9C 12E056         =1  2185             CALL    GETOKE
FC9F 12E065         =1  2186             CALL    GET_PART
FCA2 12E06E         =1  2187             CALL    EOL_CHECK
                    =1  2188     DS0:
FCA5 12E00F         =1  2189             CALL    NEWLINE
FCA8 AA57           =1  2190             MOV     PARAM1,PARTIT_LO_HIGH
FCAA AB58           =1  2191             MOV     PARAM2,PARTIT_LO_LOW
FCAC 12E018         =1  2192             CALL    LSTWRD
FCAF 7A3D           =1  2193             MOV     PARAM1,#'='                     ;Display Adr = to console
FCB1 12E006         =1  2194             CALL    CO
FCB4 85586A         =1  2195             MOV     MEMORY_TRACE_ADDR_LOW,PARTIT_LO_LOW
FCB7 855769         =1  2196             MOV     MEMORY_TRACE_ADDR_HIGH,PARTIT_LO_HIGH
FCBA 7900           =1  2197             MOV     POINT1,#00H
                    =1  2198     DS4:
FCBC E9             =1  2199             MOV     A,POINT1
FCBD B40300         =1  2200             CJNE    A,#03H,DS1
                    =1  2201     DS1:
FCC0 501D           =1  2202             JNC     DS2
FCC2 E558           =1  2203             MOV     A,PARTIT_LO_LOW
FCC4 29             =1  2204             ADD     A,POINT1
FCC5 F545           =1  2205             MOV     PNTLOW,A
FCC7 855744         =1  2206             MOV     PNTHGH,PARTIT_LO_HIGH
FCCA 5002           =1  2207             JNC     DS3
FCCC 0544           =1  2208             INC     PNTHGH
FCCE 754600         =1  2209     DS3:    MOV     SELECT,#(CBYTE_TOKE AND 07H)
FCD1 12E04A         =1  2210             CALL    FETCH                           ;Get a byte from code memory
FCD4 F547           =1  2211             MOV     TEMP_LOW,A
FCD6 7440           =1  2212             MOV     A,#WORKING_SPACE
FCD8 29             =1  2213             ADD     A,POINT1
FCD9 F8             =1  2214             MOV     POINTO,A
```

```
LOC  OBJ            LINE     SOURCE

FCDA A647       =1  2215.            MOV     @POINTO,TEMP_LOW
FCDC 09         =1  2216             INC     POINT1
FCDD 80DD       =1  2217             JMP     DS4
FCDF 12FD02     =1  2218     DS2:    CALL    GET_HASH_VALUE              ;Search hash table for match
FCE2 12FF23     =1  2219             CALL    DISASSEMBLE
FCE5 C558       =1  2220             XCH     A,PARTIT_LO_LOW
FCE7 254D       =1  2221             ADD     A,NUMBER_OF_BYTES
FCE9 C558       =1  2222             XCH     A,PARTIT_LO_LOW
FCEB 5002       =1  2223             JNC     DS5
FCED 0557       =1  2224             INC     PARTIT_LO_HIGH
FCEF C3         =1  2225     DS5:    CLR     C
FCF0 E55A       =1  2226             MOV     A,PARTIT_HI_LOW
FCF2 9558       =1  2227             SUBB    A,PARTIT_LO_LOW             ;Subtract actual partition address low
                =1  2228                                                 ;From ending address and carry borrow
FCF4 E559       =1  2229             MOV     A,PARTIT_HI_HIGH
FCF6 9557       =1  2230             SUBB    A,PARTIT_LO_HIGH            ;Subtract actual partition address high
                =1  2231                                                 ;From ending address high
FCF8 4005       =1  2232             JC      DSRET                       ;Exit if carry generated
FCFA 12E068     =1  2233             CALL    CONTINUATION_LINE
FCFD 80A6       =1  2234             JMP     DSO
FCFF 02E062     =1  2235     DSRET:  JMP     WAIT_FOR_USER
                =1  2236 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                    =1  2237     ;*****************************************************************
                    =1  2238     ;
                    =1  2239     ;    NAME: GET_HASH_VALUE
                    =1  2240     ;
                    =1  2241     ;    ABSTRACT: This routine takes the hash value in OUR_CODE and
                    =1  2242     ;       divides it into one the 4 ordinals.  They are MNEMONIC_ORDINAL,
                    =1  2243     ;       FIRST_OPER_ORDINAL, SECOND_OPER_ORDINAL and THIRD_OPER_ORDINAL.
                    =1  2244     ;
                    =1  2245     ;    INPUTS: WORKING_SPACE
                    =1  2246     ;
                    =1  2247     ;    OUTPUTS: MNEMONIC_ORDINAL, FIRST_OPER_ORDINAL, SECOND_OPER_ORDINAL,
                    =1  2248     ;       THIRD_OPER_ORDINAL
                    =1  2249     ;
                    =1  2250     ;    VARIABLES MODIFIED: A, ERRNUM, DPTR, C, TEMP_LOW, OUR_CODE_LOW,
                    =1  2251     ;       OUR_CODE_HIGH, DIVISOR, DIVIDEND_HIGH, DIVIDEND_LOW, PARAM5,
                    =1  2252     ;       PARAM6, B, QUOTIENT_LOW, QUOTIENT_HIGH, MNEMONIC_ORDINAL,
                    =1  2253     ;       NUMBER_OF_OPERANDS, FIRST_OPER_ORDINAL, SECOND_OPER_ORDINAL,
                    =1  2254     ;       OPERAND_CHECK, NUMBER_OF_BYTES, THIRD_OPER_ORDINAL
                    =1  2255     ;
                    =1  2256     ;    ERROR EXITS: OFH (UNDEFINED OPCODE)
                    =1  2257     ;
                    =1  2258     ;    SUBROUTINES ACCESSED DIRECTLY: ERROR, START_DIVIDE, OPERAND_BYTE_CHECK
                    =1  2259     ;
                    =1  2260     ;*****************************************************************
                    =1  2261     GET_HASH_VALUE:
FD02 E540           =1  2262             MOV     A,WORKING_SPACE                     ;Memory containing opcode to be
FD04 B4A506         =1  2263             CJNE    A,#UNDEFINED_OPCODE,HASH_CONTINUE   ;disassembled
FD07 75430F         =1  2264             MOV     ERRNUM,#OFH                         ;Undefined opcode
FD0A 02E05F         =1  2265             JMP     ERROR
                    =1  2266     HASH_CONTINUE:
FD0D 90F555         =1  2267             MOV     DPTR,#INSTRUCTION_CODE              ;Starting adr of hash tbl
FD10 C3             =1  2268             CLR     C
FD11 33             =1  2269             RLC     A                                   ;Multiply pointer by two
FD12 5002           =1  2270             JNC     GHV_A1
FD14 0583           =1  2271             INC     DPH                                 ;Increment DPH if rotate overflows
FD16 F547           =1  2272     GHV_A1: MOV     TEMP_LOW,A
FD18 93             =1  2273             MOVC    A,@A+DPTR
FD19 F54E           =1  2274             MOV     OUR_CODE_HIGH,A
FD1B 0547           =1  2275             INC     TEMP_LOW
FD1D E547           =1  2276             MOV     A,TEMP_LOW
FD1F 93             =1  2277             MOVC    A,@A+DPTR
FD20 F54F           =1  2278             MOV     OUR_CODE_LOW,A                      ;Ordinal of hashed value
FD22 75702C         =1  2279             MOV     DIVISOR,#MNEMONIC_FACTOR
FD25 854E6E         =1  2280             MOV     DIVIDEND_HIGH,OUR_CODE_HIGH
FD28 854F6F         =1  2281             MOV     DIVIDEND_LOW,OUR_CODE_LOW
FD2B 3153           =1  2282             CALL    START_DIVIDE
FD2D AE72           =1  2283             MOV     PARAM5,QUOTIENT_LOW
FD2F AF71           =1  2284             MOV     PARAM6,QUOTIENT_HIGH
FD31 E572           =1  2285             MOV     A,QUOTIENT_LOW
FD33 75F02C         =1  2286             MOV     B,#MNEMONIC_FACTOR
FD36 A4             =1  2287             MUL     AB
FD37 F572           =1  2288             MOV     QUOTIENT_LOW,A
FD39 85F071         =1  2289             MOV     QUOTIENT_HIGH,B
FD3C EF             =1  2290             MOV     A,PARAM6
FD3D 75F02C         =1  2291             MOV     B,#MNEMONIC_FACTOR
```

```
LOC  OBJ            LINE     SOURCE

FD40 A4         =1  2292             MUL     AB
FD41 2571       =1  2293             ADD     A,QUOTIENT_HIGH
FD43 F571       =1  2294             MOV     QUOTIENT_HIGH,A
FD45 E54F       =1  2295             MOV     A,OUR_CODE_LOW
FD47 C3         =1  2296             CLR     C
FD48 9572       =1  2297             SUBB    A,QUOTIENT_LOW
FD4A F56D       =1  2298             MOV     MNEMONIC_ORDINAL,A                  ;Mnemonic ord
FD4C 8F4E       =1  2299             MOV     OUR_CODE_HIGH,PARAM6
FD4E 8E4F       =1  2300             MOV     OUR_CODE_LOW,PARAM5
FD50 E54F       =1  2301             MOV     A,OUR_CODE_LOW
FD52 700A       =1  2302             JNZ     GHV1
FD54 E54E       =1  2303             MOV     A,OUR_CODE_HIGH
FD56 7006       =1  2304             JNZ     GHV1
FD58 756B00     =1  2305             MOV     NUMBER_OF_OPERANDS,#00H
FD5B 02FDB6     =1  2306             JMP     GHV9
                =1  2307     GHV1:
FD5E 757018     =1  2308             MOV     DIVISOR,#OPERAND_FACTOR
FD61 854E6E     =1  2309             MOV     DIVIDEND_HIGH,OUR_CODE_HIGH
FD64 854F6F     =1  2310             MOV     DIVIDEND_LOW,OUR_CODE_LOW
FD67 3153       =1  2311             CALL    START_DIVIDE
FD69 AE72       =1  2312             MOV     PARAM5,QUOTIENT_LOW
FD6B AF71       =1  2313             MOV     PARAM6,QUOTIENT_HIGH
FD6D E572       =1  2314             MOV     A,QUOTIENT_LOW
FD6F 75F018     =1  2315             MOV     B,#OPERAND_FACTOR
FD72 A4         =1  2316             MUL     AB
FD73 F572       =1  2317             MOV     QUOTIENT_LOW,A
FD75 85F071     =1  2318             MOV     QUOTIENT_HIGH,B
FD78 EF         =1  2319             MOV     A,PARAM6
FD79 75F018     =1  2320             MOV     B,#OPERAND_FACTOR
FD7C A4         =1  2321             MUL     AB
FD7D 2571       =1  2322             ADD     A,QUOTIENT_HIGH
FD7F F571       =1  2323             MOV     QUOTIENT_HIGH,A
FD81 E54F       =1  2324             MOV     A,OUR_CODE_LOW
FD83 C3         =1  2325             CLR     C
FD84 9572       =1  2326             SUBB    A,QUOTIENT_LOW
FD86 F563       =1  2327             MOV     FIRST_OPER_ORDINAL,A                ;First operand ord
FD88 B40F03     =1  2328             CJNE    A,#0FH,GHV2
FD8B 02FD90     =1  2329             JMP     GHV2_2
                =1  2330     GHV2:
FD8E 5002       =1  2331             JNC     GHV3
                =1  2332     GHV2_2:
FD90 1563       =1  2333             DEC     FIRST_OPER_ORDINAL
                =1  2334     GHV3:
FD92 8F4E       =1  2335             MOV     OUR_CODE_HIGH,PARAM6
FD94 8E4F       =1  2336             MOV     OUR_CODE_LOW,PARAM5
FD96 E54F       =1  2337             MOV     A,OUR_CODE_LOW
FD98 700A       =1  2338             JNZ     GHV5
FD9A E54E       =1  2339             MOV     A,OUR_CODE_HIGH
FD9C 7006       =1  2340             JNZ     GHV5
FD9E 756B01     =1  2341             MOV     NUMBER_OF_OPERANDS,#01H
FDA1 02FDB6     =1  2342             JMP     GHV9
                =1  2343     GHV5:
FDA4 854F64     =1  2344             MOV     SECOND_OPER_ORDINAL,OUR_CODE_LOW    ;Second operand ord
FDA7 E564       =1  2345             MOV     A,SECOND_OPER_ORDINAL
FDA9 B40F03     =1  2346             CJNE    A,#0FH,GHV6
```

```
LOC  OBJ            LINE     SOURCE

FDAC 02FDB1     =1  2347             JMP     GHV6_6
                =1  2348     GHV6:
FDAF 5002       =1  2349             JNC     GHV7
                =1  2350     GHV6_6:
FDB1 1564       =1  2351             DEC     SECOND_OPER_ORDINAL
                =1  2352     GHV7:
FDB3 756B02     =1  2353             MOV     NUMBER_OF_OPERANDS,#02H
                =1  2354     GHV9:
FDB6 E56D       =1  2355             MOV     A,MNEMONIC_ORDINAL
FDB8 B40909     =1  2356             CJNE    A,#09H,GHV10
FDBB 754D02     =1  2357             MOV     NUMBER_OF_BYTES,#02H
FDBE 756516     =1  2358             MOV     THIRD_OPER_ORDINAL,#16H
FDC1 02FDC7     =1  2359             JMP     GHV11
                =1  2360     GHV10:
FDC4 754D01     =1  2361             MOV     NUMBER_OF_BYTES,#01H
                =1  2362     GHV11:
FDC7 90FDD1     =1  2363             MOV     DPTR,#GHVTBL
FDCA E56B       =1  2364             MOV     A,NUMBER_OF_OPERANDS
FDCC 85636C     =1  2365             MOV     OPERAND_CHECK,FIRST_OPER_ORDINAL
FDCF 23         =1  2366             RL      A
FDD0 73         =1  2367             JMP     @A+DPTR
                =1  2368     GHVTBL:
FDD1 22         =1  2369             RET                                     ;Entry 1 for GHVTBL
FDD2 00         =1  2370             NOP
FDD3 8006       =1  2371             SJMP    OPERAND_BYTE_CHECK              ;Entry 2 for GHVTBL
FDD5 12FDDB     =1  2372             CALL    OPERAND_BYTE_CHECK              ;Entry 3 for GHVTBL
FDD8 85646C     =1  2373             MOV     OPERAND_CHECK,SECOND_OPER_ORDINAL
                =1  2374
                =1  2375 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1  2376     ;*****************************************************************
                =1  2377     ;
                =1  2378     ;    NAME: OPERAND_BYTE_CHECK
                =1  2379     ;
                =1  2380     ;    ABSTRACT: This routine is updating the number of bytes in the
                =1  2381     ;       opcode based on OPERAND_CHECK.
                =1  2382     ;
                =1  2383     ;       CAUTION: This routine is position sensitive.  It is entered from
                =1  2384     ;       the previous routine, GET_HASH_VALUE as 'in line' code.
                =1  2385     ;
                =1  2386     ;    INPUTS: OPERAND_CHECK
                =1  2387     ;
                =1  2388     ;    OUTPUTS: NUMBER_BYTES
                =1  2389     ;
                =1  2390     ;    VARIABLES MODIFIED: A, NUMBER_OF_BYTES
                =1  2391     ;
                =1  2392     ;    ERROR EXITS: None
                =1  2393     ;
                =1  2394     ;    SUBROUTINES ACCESSED DIRECTLY: None
                =1  2395     ;
                =1  2396     ;
                =1  2397     ;*****************************************************************
                =1  2398     OPERAND_BYTE_CHECK:
FDDB E56C       =1  2399             MOV     A,OPERAND_CHECK
FDDD B41000     =1  2400             CJNE    A,#10H,OBC0
                =1  2401     OBC0:
FDE0 400A       =1  2402             JC      OBC1
FDE2 B41603     =1  2403             CJNE    A,#16H,OBC2
FDE5 02FDEA     =1  2404             JMP     OBC2_2
                =1  2405     OBC2:
FDE8 5002       =1  2406             JNC     OBC1
                =1  2407     OBC2_2:
FDEA 054D       =1  2408             INC     NUMBER_OF_BYTES
                =1  2409     OBC1:
FDEC B41402     =1  2410             CJNE    A,#14H,OBCRET
FDEF 054D       =1  2411             INC     NUMBER_OF_BYTES
FDF1 22         =1  2412     OBCRET: RET
                =1  2413 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                   =1  2414     ;*****************************************************************
                   =1  2415     ;
                   =1  2416     ;    NAME: DISPLAY_OPERAND
                   =1  2417     ;
                   =1  2418     ;    ABSTRACT: This routine displays an operand of the disassembled
                   =1  2419     ;       opcode to the console.
                   =1  2420     ;
                   =1  2421     ;    INPUTS: NUNBER_OF_OPERANDS_PRINTED, FIRST_OPER_ORDINAL,
                   =1  2422     ;       SECOND_OPER_ORDINAL, THIRD_OPER_ORDINAL
                   =1  2423     ;
                   =1  2424     ;    OUTPUTS: NUMBER_OF_OPERANDS_PRINTED
                   =1  2425     ;
                   =1  2426     ;    VARIABLES MODIFIED: A, DPTR, CURRENT_OPERAND, C, PARAM1, POINTO,
                   =1  2427     ;       VALHGH, VALLOW, PARAM2, EXPRESSIONS_PRINTED, MEMORY_TRACE_ADDR_HIGH,
                   =1  2428     ;       TEMP_LOW, NO_OF_OPERANDS_PRINTED
                   =1  2429     ;
                   =1  2430     ;    ERROR EXITS: None
                   =1  2431     ;
                   =1  2432     ;    SUBROUTINES ACCESSED DIRECTLY: DISPLAY_TOKEN, LSTBYT, CO, LSTWRD,
                   =1  2433     ;       PRINT_STRING
                   =1  2434     ;
                   =1  2435     ;
                   =1  2436     ;*****************************************************************
                   =1  2437     DISPLAY_OPERAND:
FDF2 E567          =1  2438             MOV     A,NO_OF_OPERANDS_PRINTED
FDF4 14            =1  2439             DEC     A
FDF5 23            =1  2440             RL      A
FDF6 23            =1  2441             RL      A
FDF7 90FDFB        =1  2442             MOV     DPTR,#DDTBL
FDFA 73            =1  2443             JMP     @A+DPTR
FDFB E563          =1  2444     DDTBL:  MOV     A,FIRST_OPER_ORDINAL
FDFD 8006          =1  2445             SJMP    DDO
FDFF E564          =1  2446             MOV     A,SECOND_OPER_ORDINAL
FE01 8002          =1  2447             SJMP    DDO
FE03 E565          =1  2448             MOV     A,THIRD_OPER_ORDINAL
FE05 F566          =1  2449     DDO:    MOV     CURRENT_OPERAND,A
FE07 B40C05        =1  2450             CJNE    A,#0CH,DDO_1
FE0A 74A1          =1  2451             MOV     A,#0A1H
FE0C 02FE2D        =1  2452             JMP     DD4_1
FE0F B40F03        =1  2453     DDO_1:  CJNE    A,#0FH,DD1
FE12 02FE17        =1  2454             JMP     DD1_1
FE15 501A          =1  2455     DD1:    JNC     DD2
FE17 B40300        =1  2456     DD1_1:  CJNE    A,#03H,DD3
FE1A 400E          =1  2457     DD3:    JC      DD4
FE1C B40A03        =1  2458             CJNE    A,#0AH,DD5
FE1F 02FE24        =1  2459             JMP     DD5_5
FE22 5006          =1  2460     DD5:    JNC     DD4
FE24 C3            =1  2461     DD5_5:  CLR     C
FE25 248D          =1  2462             ADD     A,#8DH
FE27 02FE2D        =1  2463             JMP     DD4_1
FE2A C3            =1  2464     DD4:    CLR     C
FE2B 2451          =1  2465             ADD     A,#(OFST+REG+1)
FE2D FA            =1  2466     DD4_1:  MOV     PARAM1,A
FE2E 12E059        =1  2467             CALL    DISPLAY_TOKEN
FE31 E566          =1  2468     DD2:    MOV     A,CURRENT_OPERAND
```

```
LOC  OBJ            LINE     SOURCE

FE33 C3         =1  2469             CLR     C
FE34 9410       =1  2470             SUBB    A,#10H
FE36 B4000F     =1  2471             CJNE    A,#00H,DD_CASE_1            ;Byte expression 8-bits
                =1  2472     DD_CASE_EXP8:                          ;Generalized byte expression display
FE39 7440       =1  2473             MOV     A,#WORKING_SPACE
FE3B 2568       =1  2474             ADD     A,EXPRESSIONS_PRINTED
FE3D F8         =1  2475             MOV     POINTO,A
FE3E E6         =1  2476             MOV     A,@POINTO
FE3F FA         =1  2477             MOV     PARAM1,A
FE40 12E015     =1  2478             CALL    LSTBYT
FE43 0568       =1  2479             INC     EXPRESSIONS_PRINTED
FE45 02FF1B     =1  2480             JMP     DD_CASE_END
                =1  2481     DD_CASE_1:
FE48 B40102     =1  2482             CJNE    A,#01H,DD_CASE_2            ;Bit expression, 8-bits
FE4B 80EC       =1  2483             JMP     DD_CASE_EXP8
                =1  2484     DD_CASE_2:
FE4D B40207     =1  2485             CJNE    A,#02H,DD_CASE_3            ;Immediate expression, 8-bits
FE50 7A23       =1  2486             MOV     PARAM1,#'#'
FE52 12E006     =1  2487             CALL    CO
FE55 80E2       =1  2488             JMP     DD_CASE_EXP8
                =1  2489     DD_CASE_3:
FE57 B40307     =1  2490             CJNE    A,#03H,DD_CASE_4            ;Complimented byte expression, 8-bits
FE5A 7A2F       =1  2491             MOV     PARAM1,#'/'
FE5C 12E006     =1  2492             CALL    CO
FE5F 80D8       =1  2493             JMP     DD_CASE_EXP8
                =1  2494     DD_CASE_4:
FE61 B4043F     =1  2495             CJNE    A,#04H,DD_CASE_5            ;Expression, 16-bits
FE64 7840       =1  2496             MOV     POINTO,#WORKING_SPACE
FE66 08         =1  2497             INC     POINTO
FE67 8649       =1  2498             MOV     VALHGH,@POINTO
FE69 08         =1  2499             INC     POINTO
FE6A 864A       =1  2500             MOV     VALLOW,@POINTO
FE6C E56D       =1  2501             MOV     A,MNEMONIC_ORDINAL
FE6E B40F0F     =1  2502             CJNE    A,#0FH,DD_CASE_4_0
FE71 7A23       =1  2503             MOV     PARAM1,#'#'
FE73 12E006     =1  2504             CALL    CO
                =1  2505     DD_CASE_EXP16:                         ;Generalized word expression display
FE76 AA49       =1  2506             MOV     PARAM1,VALHGH
FE78 AB4A       =1  2507             MOV     PARAM2,VALLOW
FE7A 12E018     =1  2508             CALL    LSTWRD
FE7D 02FF1B     =1  2509             JMP     DD_CASE_END
                =1  2510     DD_CASE_4_0:
FE80 E566       =1  2511             MOV     A,CURRENT_OPERAND
FE82 B41403     =1  2512             CJNE    A,#14H,SS0
FE85 02FE91     =1  2513             JMP     SS3
FE88 B41503     =1  2514     SS0:    CJNE    A,#21,SS1
FE8B 02FE91     =1  2515             JMP     SS3
FE8E B4160A     =1  2516     SS1:    CJNE    A,#16H,SS2
FE91 AA49       =1  2517     SS3:    MOV     PARAM1,VALHGH
FE93 AB4A       =1  2518             MOV     PARAM2,VALLOW
FE95 12E018     =1  2519             CALL    LSTWRD
FE98 02FF1B     =1  2520             JMP     DD_CASE_END
FE9B AA4A       =1  2521     SS2:    MOV     PARAM1,VALLOW
FE9D 12E015     =1  2522             CALL    LSTBYT
FEA0 02FF1B     =1  2523             JMP     DD_CASE_END
```

```
LOC  OBJ            LINE     SOURCE

                    =1  2524     DD_CASE_5:
FEA3 B4050E         =1  2525             CJNE    A,#05H,DD_CASE_6                ;Expression, 11-bits
FEA6 7840           =1  2526             MOV     POINTO,#WORKING_SPACE
FEA8 E6             =1  2527             MOV     A,@POINTO
FEA9 54E0           =1  2528             ANL     A,#0E0H
FEAB C4             =1  2529             SWAP    A
FEAC 03             =1  2530             RR      A
FEAD F549           =1  2531             MOV     VALHGH,A
FEAF 08             =1  2532             INC     POINTO
FEB0 864A           =1  2533             MOV     VALLOW,@POINTO
FEB2 80C2           =1  2534             JMP     DD_CASE_EXP16
                    =1  2535     DD_CASE_6:
FEB4 B4063C         =1  2536             CJNE    A,#06H,DD_CASE_7                ;Relative offset
FEB7 E56A           =1  2537             MOV     A,MEMORY_TRACE_ADDR_LOW
FEB9 254D           =1  2538             ADD     A,NUMBER_OF_BYTES
FEBB F56A           =1  2539             MOV     MEMORY_TRACE_ADDR_LOW,A
FEBD 5002           =1  2540             JNC     DD_CASE_6_0
FEBF 0569           =1  2541             INC     MEMORY_TRACE_ADDR_HIGH
                    =1  2542     DD_CASE_6_0:
FEC1 7440           =1  2543             MOV     A,#WORKING_SPACE
FEC3 2568           =1  2544             ADD     A,EXPRESSIONS_PRINTED
FEC5 F8             =1  2545             MOV     POINTO,A
FEC6 E6             =1  2546             MOV     A,@POINTO
FEC7 B47F03         =1  2547             CJNE    A,#07FH,DD_CASE_6_1
FECA 02FEE4         =1  2548             JMP     DD_CASE_6_2
                    =1  2549     DD_CASE_6_1:
FECD 4015           =1  2550             JC      DD_CASE_6_2
FECF F4             =1  2551             CPL     A
FED0 04             =1  2552             INC     A
FED1 F547           =1  2553             MOV     TEMP_LOW,A
FED3 E56A           =1  2554             MOV     A,MEMORY_TRACE_ADDR_LOW
FED5 C3             =1  2555             CLR     C
FED6 9547           =1  2556             SUBB    A,TEMP_LOW
FED8 F54A           =1  2557             MOV     VALLOW,A
FEDA E569           =1  2558             MOV     A,MEMORY_TRACE_ADDR_HIGH
FEDC 5001           =1  2559             JNC     DD_CASE_6_3
FEDE 14             =1  2560             DEC     A
                    =1  2561     DD_CASE_6_3:
FEDF F549           =1  2562             MOV     VALHGH,A
FEE1 02FEEF         =1  2563             JMP     DD_CASE_6_5
                    =1  2564     DD_CASE_6_2:
FEE4 256A           =1  2565             ADD     A,MEMORY_TRACE_ADDR_LOW
FEE6 F54A           =1  2566             MOV     VALLOW,A
FEE8 E569           =1  2567             MOV     A,MEMORY_TRACE_ADDR_HIGH
FEEA 5001           =1  2568             JNC     DD_CASE_6_4
FEEC 04             =1  2569             INC     A
                    =1  2570     DD_CASE_6_4:
FEED F549           =1  2571             MOV     VALHGH,A
                    =1  2572     DD_CASE_6_5:
FEEF 0568           =1  2573             INC     EXPRESSIONS_PRINTED
FEF1 8083           =1  2574             JMP     DD_CASE_EXP16
                    =1  2575     DD_CASE_7:
FEF3 B40712         =1  2576             CJNE    A,#07H,DD_CASE_8                ;Special case for @A+DPTR
FEF6 7AFF           =1  2577             MOV     PARAM1,#HIGH DD_CASE_7_MSG
FEF8 7B00           =1  2578             MOV     PARAM2,#LOW DD_CASE_7_MSG
```

```
LOC  OBJ            LINE     SOURCE

FEFA 12E01E      =1  2579             CALL    PRINT_STRING
FEFD 02FF1B      =1  2580             JMP     DD_CASE_END
                 =1  2581     DD_CASE_7_MSG:
FF00 07          =1  2582             DB      07,'@A+DPTR'
FF01 40412B44
FF05 505452
                 =1  2583     DD_CASE_8:
FF08 B40810      =1  2584             CJNE    A,#8,DD_CASE_END                ;Special case for @A+PC
FF0B 7AFF        =1  2585             MOV     PARAM1,#HIGH DD_CASE_8_MSG
FF0D 7B15        =1  2586             MOV     PARAM2,#LOW DD_CASE_8_MSG
FF0F 12E01E      =1  2587             CALL    PRINT_STRING
FF12 02FF1B      =1  2588             JMP     DD_CASE_END
                 =1  2589     DD_CASE_8_MSG:
FF15 05          =1  2590             DB      05,'@A+PC'
FF16 40412B50
FF1A 43
                 =1  2591     DD_CASE_END:
FF1B 0567        =1  2592             INC     NO_OF_OPERANDS_PRINTED
FF1D 22          =1  2593             RET
                 =1  2594 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1  2595     ;*********************************************************************
                =1  2596     ;
                =1  2597     ;    NAME: DISPLAY_COMMA
                =1  2598     ;
                =1  2599     ;    ABSTRACT: This routine displays a comma symbol to the console.
                =1  2600     ;
                =1  2601     ;    INPUTS: None
                =1  2602     ;
                =1  2603     ;    OUTPUTS: None
                =1  2604     ;
                =1  2605     ;    VARIABLES MODIFIED: PARAM1
                =1  2606     ;
                =1  2607     ;    ERROR EXITS: None
                =1  2608     ;
                =1  2609     ;    SUBROUTINES ACCESSED DIRECTLY: CO
                =1  2610     ;
                =1  2611     ;
                =1  2612     ;*********************************************************************
                =1  2613     DISPLAY_COMMA:
FF1E 7A2C       =1  2614             MOV     PARAM1,#','
FF20 02E006     =1  2615             JMP     CO
                =1  2616 +1  $EJECT
```

```
LOC  OBJ            LINE     SOURCE

                =1  2617     ;*****************************************************************
                =1  2618     ;
                =1  2619     ;    NAME: DISASSEMBLE
                =1  2620     ;
                =1  2621     ;    ABSTRACT: This routine displays one disassembled opcode on the
                =1  2622     ;       console.
                =1  2623     ;
                =1  2624     ;    INPUTS: MNEMONIC_ORDINAL
                =1  2625     ;
                =1  2626     ;    OUTPUTS: None
                =1  2627     ;
                =1  2628     ;    VARIABLES MODIFIED: A, PARAM1, DPTR, INSTRUCTION_VALUE,
                =1  2629     ;       NO_OF_OPERANDS_PRINTED, EXPRESSIONS_PRINTED, C
                =1  2630     ;
                =1  2631     ;    ERROR EXITS: None
                =1  2632     ;
                =1  2633     ;    SUBROUTINES ACCESSED DIRECTLY: DISPLAY_TOKEN, CO, DISPLAY_OPERAND,
                =1  2634     ;       DISPLAY_COMMA,
                =1  2635     ;
                =1  2636     ;*****************************************************************
                =1  2637     DISASSEMBLE:
FF23 E56D       =1  2638             MOV     A,MNEMONIC_ORDINAL
FF25 2410       =1  2639             ADD     A,#OFST
FF27 FA         =1  2640             MOV     PARAM1,A
FF28 12E059     =1  2641             CALL    DISPLAY_TOKEN
FF2B 90F529     =1  2642             MOV     DPTR,#MNEMONIC_TAB
FF2E E56D       =1  2643             MOV     A,MNEMONIC_ORDINAL
FF30 93         =1  2644             MOVC    A,@A+DPTR
FF31 F55B       =1  2645             MOV     INSTRUCTION_VALUE,A
FF33 7A20       =1  2646             MOV     PARAM1,#' '
FF35 12E006     =1  2647             CALL    CO
FF38 756701     =1  2648             MOV     NO_OF_OPERANDS_PRINTED,#1
FF3B 756801     =1  2649             MOV     EXPRESSIONS_PRINTED,#1
FF3E E55B       =1  2650             MOV     A,INSTRUCTION_VALUE
FF40 C3         =1  2651             CLR     C
FF41 9407       =1  2652             SUBB    A,#07H
FF43 B40001     =1  2653             CJNE    A,#00H,DISCASE_1
FF46 22         =1  2654             RET
                =1  2655     DISCASE_1:
FF47 B40102     =1  2656             CJNE    A,#01H,DISCASE_2
FF4A A1F2       =1  2657             JMP     DISPLAY_OPERAND
                =1  2658     DISCASE_2:
FF4C B40212     =1  2659             CJNE    A,#02H,DISCASE_3
FF4F E540       =1  2660             MOV     A,WORKING_SPACE
FF51 B48507     =1  2661             CJNE    A,#85H,DISCASE_2_1                  ;Check for special case
FF54 E541       =1  2662             MOV     A,(WORKING_SPACE+1)                 ;of MOV /,/ where operands
FF56 854241     =1  2663             MOV     (WORKING_SPACE+1),(WORKING_SPACE+2) ;are in reverse order.
FF59 F542       =1  2664             MOV     (WORKING_SPACE+2),A
                =1  2665     DISCASE_2_1:
FF5B B1F2       =1  2666             CALL    DISPLAY_OPERAND
FF5D F11E       =1  2667             CALL    DISPLAY_COMMA
FF5F A1F2       =1  2668             JMP     DISPLAY_OPERAND
                =1  2669     DISCASE_3:
FF61 B40306     =1  2670             CJNE    A,#03H,DISCASE_4
FF64 B1F2       =1  2671             CALL    DISPLAY_OPERAND
```

```
LOC  OBJ             LINE     SOURCE

FF66 F11E        =1  2672             CALL    DISPLAY_COMMA
FF68 A1F2        =1  2673             JMP     DISPLAY_OPERAND
                 =1  2674     DISCASE_4:
FF6A B4040A      =1  2675             CJNE    A,#04H,DISCASE_5
FF6D B1F2        =1  2676             CALL    DISPLAY_OPERAND
FF6F F11E        =1  2677             CALL    DISPLAY_COMMA
FF71 B1F2        =1  2678             CALL    DISPLAY_OPERAND
FF73 F11E        =1  2679             CALL    DISPLAY_COMMA
FF75 A1F2        =1  2680             JMP     DISPLAY_OPERAND
                 =1  2681     DISCASE_5:
FF77 B40502      =1  2682             CJNE    A,#05H,DISCASE_6
FF7A A1F2        =1  2683             JMP     DISPLAY_OPERAND
                 =1  2684     DISCASE_6:
FF7C B40606      =1  2685             CJNE    A,#06H,DISCASE_7
FF7F B1F2        =1  2686             CALL    DISPLAY_OPERAND
FF81 F11E        =1  2687             CALL    DISPLAY_COMMA
FF83 A1F2        =1  2688             JMP     DISPLAY_OPERAND
                 =1  2689     DISCASE_7:
FF85 B40702      =1  2690             CJNE    A,#07H,DISCASE_8
FF88 A1F2        =1  2691             JMP     DISPLAY_OPERAND
                 =1  2692     DISCASE_8:
FF8A B40802      =1  2693             CJNE    A,#08H,DISCASE_END
FF8D B1F2        =1  2694             CALL    DISPLAY_OPERAND
                 =1  2695     DISCASE_END:
FF8F 22          =1  2696             RET
                     2697     END
```

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

NAME	TYPE	VALUE AND REFERENCES
A_OP1	N	002CH 367# 465 467 485 487 505 507 508 509 510 512 513 514 515 517 518 519 520 525 527 528 529 530 532 533 534 535 537 538 539 540 547 548 549 550 552 553 554 555 557 558 559 560 567 568 569 570 572 573 574 575 577 578 579 580 587 588 589 590 592 593 594 595 597 598 599 600 607 625 645 647 648 649 650 652 653 654 655 657 658 659 660 687 688 707 708 709 710 712 713 714 715 717 718 719 720 727 729 730 742 744 745 747 748 749 750 752 753 754 755 757 758 759 760 767
A_OP2	N	0420H 391# 544 564 584 762 764 765 768 769 770 772 773 774 775 777 778 779 780
AB_OP1	N	0210H 378# 627 667
AMT0	L CSEG	F92EH 1253 1259#
AMT1	L CSEG	F931H 1260# 1273
AMT2	L CSEG	F94CH 1267 1271#
AMTERR	L CSEG	F946H 1222 1259 1269#
ASERR	L CSEG	F7D4H 838 840 853 890 897 905# 943 947 954 986
ASM_PC_HIGH	L DSEG	004BH 86# 1255 1261 1881 1886 1950 1990 2006 2021 2049 2065 2069 2086 2111 2140 2155
ASM_PC_LOW	L DSEG	004CH 87# 1256 1263 1878 1887 1951 1997 2003 2018 2056 2062 2070 2083 2139 2154
ASMBASE	N	F523H 39# 40
ASSEMBLY_CMD	L CSEG	F916H 41 1250#
ATA_PLUS_DPTR_OP1	N	03F4H 387# 605
ATA_PLUS_DPTR_OP2	N	5EE0H 410# 645
ATA_PLUS_PC_OP2	N	6300H 411# 625
ATA_TOKE	N	000AH 54# 836 947
ATDPTR_OP1	N	0294H 381# 762
ATDPTR_OP2	N	3DE0H 403# 742
ATR0_OP1	N	0058H 368# 469 489 609 669 689 764 769
ATR0_OP2	N	0840H 392# 509 529 549 569 589 629 649 709 729 744 749
ATR1_OP1	N	0084H 369# 470 490 610 670 690 765 770
ATR1_OP2	N	0C60H 393# 510 530 550 570 590 630 650 710 730 745 750
B	N DSEG	00F0H PREDEFINED 1158 1529 1533 1542 1544 1545 1549 1553 1582 1598 1615 1729 1791 1802 1803 1807 1811 1824 1825 1829 1833 2286 2289 2291 2315 2318 2320
B_0_T	L BSEG	0000H 107# 844 894 902 943 986 1059
BACKWARD_JUMP_CASE_4	L CSEG	FBA1H 1993 1999 2016#
BACKWARD_JUMP_CASE_5	L CSEG	FC11H 2052 2058 2081#
BAR_TOKE	N	0003H 50# 912
BASE	N	E000H 46# 126 127 128 129 130 131 132 133 135 136 137 138 139 140 141 142 143 144 145 146 147
BIT_END	N	001BH 455# 1212
BIT_EXP	L BSEG	0002H 448# 1195 1214 1621 1703 1752
BIT_EXP8_OP1	N	02ECH 383# 482 502 522 644 684 704 724
BIT_EXP8_OP2	N	4620H 405# 604 624 664
BLINK	N	0080H 64#
BYTE_EXP8_OP1	N	02C0H 382# 468 488 544 545 564 565 584 585 608 628 629 630 632 633 634 635 637 638 639 640 702 722 728 768
BYTE_EXP8_OP2	N	4200H 404# 508 528 548 568 588 628 648 669 670 672 673 674 675 677 678 679 680 688 708 748
C_OP1	N	0268H 380# 604 624 662 664 682 685 705 725
C_OP2	N	39C0H 402# 644
C_TOKE	N	005EH 55# 1620
CALCULATE_INSTRUCTION_VALUE	L CSEG	F99FH 807 850 855 893 1000 1030 1062 1067 1100 1127 1481#

NAME	TYPE	VALUE	AND REFERENCES
CASE_3_MORE	L CSEG	FB58H	1972 1976#
CBYTE_TOKE.	N	0080H	56# 2209
CHANGE_ASM_PC_1	L CSEG	FB1CH	1882 1885#
CHANGE_CASE_2	L CSEG	FB35H	1953 1958#
CHANGE_CASE_3	L CSEG	FB44H	1959 1966#
CHANGE_CASE_4	L CSEG	FB62H	1967 1982#
CHANGE_CASE_4_END	L CSEG	FBBEH	2014 2035#
CHANGE_CASE_4A.	L CSEG	FB7BH	1990 1992#
CHANGE_CASE_4AA	L CSEG	FB80H	1991 1995#
CHANGE_CASE_4C.	L CSEG	FB85H	1997 1998#
CHANGE_CASE_4D.	L CSEG	FB99H	2009 2010#
CHANGE_CASE_4F.	L CSEG	FBB5H	2026 2027#
CHANGE_CASE_5	L CSEG	FBC5H	1983 2039#
CHANGE_CASE_5_END	L CSEG	FC2EH	2079 2099#
CHANGE_CASE_5A.	L CSEG	FBE2H	2049 2051#
CHANGE_CASE_5AA	L CSEG	FBE7H	2050 2054#
CHANGE_CASE_5C.	L CSEG	FBECH	2056 2057#
CHANGE_CASE_5D.	L CSEG	FC09H	2074 2075#
CHANGE_CASE_5F.	L CSEG	FC25H	2091 2092#
CHANGE_CASE_6	L CSEG	FC35H	2040 2103#
CHANGE_CASE_7	L CSEG	FC5EH	2104 2124#
CHANGE_END.	L CSEG	FC73H	1957 1965 1981 2038 2102 2123 2135#
CHANGE_END_A.	L CSEG	FC92H	2149 2151#
CHANGE_END_LOOP	L CSEG	FC84H	2142# 2152
CHANGE_ERROR.	L CSEG	FBFBH	2007 2011 2024 2028 2068# 2076 2089 2093 2113 2126
CHANGE_TO_INSTRUCTION_OP.	L CSEG	FB21H	1224 1949#
CHARIN.	L DSEG	0050H	91#
CHECK_AND_CHANGE_ASM_PC	L CSEG	FB0CH	1876# 1954 1960 1968 1986 2043 2106 2128
CHECK_AND_INC_HASH_TAB.	L CSEG	F9B6H	1489 1496#
CHECK_AND_SET_EXP_FLAG.	L CSEG	FA47H	854 1065 1701#
CHECK_AND_SET_SECOND_EXP_FLAG	L CSEG	FA8FH	918 996 1749#
CHECK_EXP_FLAG.	L CSEG	FA68H	808 1723#
CHECK_HASH_TAB.	L CSEG	F9B7H	1493 1498#
CHECKSUM.	N REG	R6	120#
CHRCNT.	L DSEG	0051H	92#
CI.	N	E009H	127#
CO.	N	E006H	126# 2194 2487 2492 2504 2615 2647
CONT_OUR_CODE	L CSEG	F8F9H	1209# 1217
CONT_UPDATE_LSSTHN.	L CSEG	F9D3H	1531# 1540
CONTINUATION_LINE	N	E068H	145# 2233
COUNT	N REG	R7	119#
CSTS.	N	E00CH	128#
CURRENT_OPERAND	L DSEG	0066H	429# 2449 2468 2511
DD_CASE_1	L CSEG	FE48H	2471 2481#
DD_CASE_2	L CSEG	FE4DH	2482 2484#
DD_CASE_3	L CSEG	FE57H	2485 2489#
DD_CASE_4	L CSEG	FE61H	2490 2494#
DD_CASE_4_0	L CSEG	FE80H	2502 2510#
DD_CASE_5	L CSEG	FEA3H	2495 2524#
DD_CASE_6	L CSEG	FEB4H	2525 2535#
DD_CASE_6_0	L CSEG	FEC1H	2540 2542#
DD_CASE_6_1	L CSEG	FECDH	2547 2549#
DD_CASE_6_2	L CSEG	FEE4H	2548 2550 2564#
DD_CASE_6_3	L CSEG	FEDFH	2559 2561#
DD_CASE_6_4	L CSEG	FEEDH	2568 2570#

```
NAME                          TYPE     VALUE AND REFERENCES

DD_CASE_6_5 . . . . . .  L CSEG   FEEFH  2563 2572#
DD_CASE_7 . . . . . .  L CSEG   FEF3H  2536 2575#
DD_CASE_7_MSG . . . . . .  L CSEG   FF00H  2577 2578 2581#
DD_CASE_8 . . . . . .  L CSEG   FF08H  2576 2583#
DD_CASE_8_MSG . . . . . .  L CSEG   FF15H  2585 2586 2589#
DD_CASE_END . . . . . .  L CSEG   FF1BH  2480 2509 2520 2523 2580 2584 2588 2591#
DD_CASE_EXP16 . . . . . .  L CSEG   FE76H  2505# 2534 2574
DD_CASE_EXP8. . . . . .  L CSEG   FE39H  2472# 2483 2488 2493
DD0 . . . . . .  L CSEG   FE05H  2445 2447 2449#
DD0_1 . . . . . .  L CSEG   FE0FH  2450 2453#
DD1 . . . . . .  L CSEG   FE15H  2453 2455#
DD1_1 . . . . . .  L CSEG   FE17H  2454 2456#
DD2 . . . . . .  L CSEG   FE31H  2455 2468#
DD3 . . . . . .  L CSEG   FE1AH  2456 2457#
DD4 . . . . . .  L CSEG   FE2AH  2457 2460 2464#
DD4_1 . . . . . .  L CSEG   FE2DH  2452 2463 2466#
DD5 . . . . . .  L CSEG   FE22H  2458 2460#
DD5_5 . . . . . .  L CSEG   FE24H  2459 2461#
DDTBL . . . . . .  L CSEG   FDFBH  2442 2444#
DISASSEMBLE . . . . . .  L CSEG   FF23H  2219 2637#
DISASSEMBLY_CMD . . . . . .  L CSEG   FC9CH  42 2184#
DISCASE_1 . . . . . .  L CSEG   FF47H  2653 2655#
DISCASE_2 . . . . . .  L CSEG   FF4CH  2656 2658#
DISCASE_2_1 . . . . . .  L CSEG   FF5BH  2661 2665#
DISCASE_3 . . . . . .  L CSEG   FF61H  2659 2669#
DISCASE_4 . . . . . .  L CSEG   FF6AH  2670 2674#
DISCASE_5 . . . . . .  L CSEG   FF77H  2675 2681#
DISCASE_6 . . . . . .  L CSEG   FF7CH  2682 2684#
DISCASE_7 . . . . . .  L CSEG   FF85H  2685 2689#
DISCASE_8 . . . . . .  L CSEG   FF8AH  2690 2692#
DISCASE_END . . . . . .  L CSEG   FF8FH  2693 2695#
DISPLAY_COMMA . . . . . .  L CSEG   FF1EH  2613# 2667 2672 2677 2679 2687
DISPLAY_OPERAND . . . . . .  L CSEG   FDF2H  2437# 2657 2666 2668 2671 2673 2676 2678 2680 2683 2686 2688 2691 2694
DISPLAY_TOKEN . . . . . .  N        E059H  140# 2467 2641
DIVIDE_1. . . . . .  L CSEG   F95BH  1406# 1421 1433
DIVIDE_2. . . . . .  L CSEG   F95CH  1408# 1455
DIVIDE_3. . . . . .  L CSEG   F96EH  1417 1419#
DIVIDE_4. . . . . .  L CSEG   F977H  1425 1427#
DIVIDEND_HIGH . . . . . .  L DSEG   0006EH  437# 1409 1414 1430 1432 1452 1454 2280 2309
DIVIDEND_LOW. . . . . .  L DSEG   0006FH  438# 1448 1451 2281 2310
DIVISOR . . . . . .  L DSEG   00070H  439# 1402 1412 1431 2279 2308
DPH . . . . . .  N DSEG   0083H PREDEFINED  1499 2271
DPL . . . . . .  N DSEG   0082H PREDEFINED  1501
DPTR_OP1. . . . . .  N        023CH  379# 642 665
DPTR_TOKE . . . . . .  N        00A1H  57# 840 847 885 950
DS0 . . . . . .  L CSEG   FCA5H  2188# 2234
DS1 . . . . . .  L CSEG   FCC0H  2200 2201#
DS2 . . . . . .  L CSEG   FCDFH  2202 2218#
DS3 . . . . . .  L CSEG   FCCEH  2207 2209#
DS4 . . . . . .  L CSEG   FCBCH  2198# 2217
DS5 . . . . . .  L CSEG   FCEFH  2223 2225#
DSRET . . . . . .  L CSEG   FCFFH  2232 2235#
END_FIRST_OPERAND . . . . . .  L CSEG   FA46H  1620 1622#
END_SECOND_OPERAND. . . . . .  L CSEG   FB0BH  1839 1841 1843#
END_SELECT_INSTRUCTION_TAIL .  L CSEG   F907H  1213 1215 1218#
```

```
NAME                          TYPE     VALUE AND REFERENCES

EOL_CHECK . . . . . .  N        E06EH  147# 2187
EOL_TOKE. . . . . .  N        0007H  53# 1259 1267
ERRNUM. . . . . .  L DSEG   0043H  78# 905 916 994 1153 1269 1503 1883 1984 2041 2112 2125 2264
ERROR . . . . . .  N        E05FH  142# 906 1003 1270 1504 1884 2071 2265
EXP_FLAG_TABLE. . . . . .  L CSEG   FA77H  1730 1733#
EXP11_OP1 . . . . . .  N        039CH  385# 463 483 503 523 543 563 583 603 623 643 663 683 703 723 743 763
EXP16_OP1 . . . . . .  N        0370H  384# 464 484
EXP16_OP2 . . . . . .  N        5280H  408# 642
EXPRESSIONS_PRINTED . . . . .  L DSEG   0068H  431# 2474 2479 2544 2573 2649
FETCH . . . . . .  N        E04AH  135# 2210
FIRST_EXP . . . . . .  L BSEG   0003H  449# 1196 1707 1720 1725
FIRST_NOT_REGISTER. . . . . .  L CSEG   FA27H  1586 1588 1602#
FIRST_OPER_ORDINAL. . . . . .  L DSEG   0063H  426# 2327 2333 2365 2444
FJC_5_CONTINUE. . . . . .  L CSEG   FC04H  2067 2072#
FORWARD_JUMP_CASE_4 . . . . .  L CSEG   FB87H  1994 2000#
FORWARD_JUMP_CASE_5 . . . . .  L CSEG   FBEEH  2053 2059#
GE_FI_OP_1. . . . . .  L CSEG   FA1EH  1595 1597#
GE_FI_OP_2. . . . . .  L CSEG   FA39H  1612 1614#
GET_COMMA . . . . . .  N        E06BH  146# 888 900 945 988 998 1070
GET_FIRST_OPERAND . . . . . .  L CSEG   FA02H  845 895 944 987 1060 1581#
GET_HASH_VALUE. . . . . .  L CSEG   FD02H  2218 2261#
GET_PART. . . . . .  N        E065H  144# 2186
GET_SECOND_OPERAND. . . . . .  L CSEG   FAACH  903 1790#
GETEOL. . . . . .  N        E053H  138# 1272
GETNUM. . . . . .  N        E050H  137# 891 910 914 992 999 1028 1071 1096 1125 1254
GETOKE. . . . . .  N        E056H  139# 835 837 839 884 889 901 942 946 948 949 985 989 1058 1252 1258 1266
                                     2185
GHV_A1. . . . . .  L CSEG   FD16H  2270 2272#
GHV1. . . . . .  L CSEG   FD5EH  2302 2304 2307#
GHV10 . . . . . .  L CSEG   FDC4H  2356 2360#
GHV11 . . . . . .  L CSEG   FDC7H  2359 2362#
GHV2. . . . . .  L CSEG   FD8EH  2328 2330#
GHV2_2. . . . . .  L CSEG   FD90H  2329 2332#
GHV3. . . . . .  L CSEG   FD92H  2331 2334#
GHV5. . . . . .  L CSEG   FDA4H  2338 2340 2343#
GHV6. . . . . .  L CSEG   FDAFH  2346 2348#
GHV6_6. . . . . .  L CSEG   FDB1H  2347 2350#
GHV7. . . . . .  L CSEG   FDB3H  2349 2352#
GHV9. . . . . .  L CSEG   FDB6H  2306 2342 2354#
GHVTBL. . . . . .  L CSEG   FDD1H  2363 2368#
HASH_CONTINUE . . . . . .  L CSEG   FD0DH  2263 2266#
INST_VALUE_LOOP . . . . . .  L CSEG   F9A5H  1484# 1500 1502
INSTRUCTION . . . . . .  L DSEG   005FH  422# 1483 1487 1494 1956 1962 1970 1971 2037 2045 2119 2130
INSTRUCTION_CODE. . . . . .  L CSEG   F555H  460# 1482 1500 1502 2267
INSTRUCTION_VALUE . . . . . .  L DSEG   005BH  418# 1155 1206 2645 2650
JT00. . . . . .  L CSEG   F880H  1059 1064#
JTRET . . . . . .  L CSEG   F88FH  1063 1069#
JUMP_ABSOLUTE_OPERAND . . . .  L CSEG   F898H  1095# 1169
JUMP_END. . . . . .  N        0016H  454# 1207
JUMP_LONG_OPERAND . . . . . .  L CSEG   F8A7H  1124# 1170
JUMP_OPERAND. . . . . .  L CSEG   F865H  1027# 1167
JUMP_TWO_OPERANDS . . . . . .  L CSEG   F86EH  1057# 1168
LINBUF. . . . . .  L DSEG   0024H  72# 1260
LINCNT. . . . . .  L DSEG   0053H  94#
LINE_START. . . . . .  L DSEG   0052H  93# 1251
```

```
NAME                          TYPE     VALUE AND REFERENCES

LINMAX. . . . . .  N        0018H  62# 72
LNLGTH. . . . . .  L DSEG   0054H  95#
LSTBYT. . . . . .  N        E015H  131# 2478 2522
LSTFLG. . . . . .  L BSEG   0001H  108# 447
LSTWRD. . . . . .  N        E018H  132# 2192 2508 2519
MEMORY_TRACE_ADDR_HIGH. . . .  L DSEG   0069H  432# 2196 2541 2558 2567
MEMORY_TRACE_ADDR_LOW . . . .  L DSEG   006AH  433# 2195 2537 2539 2554 2565
MF00. . . . . .  L CSEG   F775H  836 844#
MF01. . . . . .  L CSEG   F77DH  844 847#
MF02. . . . . .  L CSEG   F78DH  847 853#
MFT00 . . . . . .  L CSEG   F7BDH  894 897#
MI0 . . . . . .  L CSEG   F90FH  1220 1222#
MI1 . . . . . .  L CSEG   F911H  1221 1223#
MIT_JMP_TBL . . . . . .  L CSEG   F8C0H  1154 1161#
MNE_ACALL . . . . . .  N        0002H  321# 483 523 563 603 643 683 723 763
MNE_ADD . . . . . .  N        0014H  339# 507 508 509 510 512 513 514 515 517 518 519 520
MNE_ADDC. . . . . .  N        0013H  338# 527 528 529 530 532 533 534 535 537 538 539 540
MNE_AJMP. . . . . .  N        0003H  322# 463 503 543 583 623 663 703 743
MNE_ANL . . . . . .  N        0011H  336# 564 565 567 568 569 570 572 573 574 575 577 578 579 580 624 682
MNE_CJNE. . . . . .  N        0009H  328# 687 688 689 690 692 693 694 695 697 698 699 700
MNE_CLR . . . . . .  N        001AH  345# 704 705 747
MNE_CPL . . . . . .  N        001BH  346# 684 685 767
MNE_DA. . . . . .  N        001CH  347# 727
MNE_DEC . . . . . .  N        0025H  356# 487 488 489 490 492 493 494 495 497 498 499 500
MNE_DIV . . . . . .  N        0021H  352# 627
MNE_DJNZ. . . . . .  N        0015H  340# 728 732 733 734 735 737 738 739 740
MNE_INC . . . . . .  N        0027H  358# 467 468 469 470 472 473 474 475 477 478 479 480 665
MNE_JB. . . . . .  N        0017H  342# 502
MNE_JBC . . . . . .  N        0018H  343# 482
MNE_JC. . . . . .  N        0008H  327# 542
MNE_JMP . . . . . .  N        0022H  353# 605
MNE_JNB . . . . . .  N        0016H  341# 522
MNE_JNC . . . . . .  N        0007H  326# 562
MNE_JNZ . . . . . .  N        0005H  324# 602
MNE_JZ. . . . . .  N        0006H  325# 582
MNE_LCALL . . . . . .  N        0000H  319# 484
MNE_LJMP. . . . . .  N        0001H  320# 464
MNE_MOV . . . . . .  N        000FH  334# 607 608 609 610 612 613 614 615 617 618 619 620 628 629 630 632 633
                                       634 635 637 638 639 640 642 644 664 669 670 672 673 674 675 677 678 679 680
                                       748 749 750 752 753 754 755 757 758 759 760 768 769 770 772 773 774 775 777
                                       778 779 780
MNE_MOVC. . . . . .  N        000AH  329# 625 645
MNE_MOVX. . . . . .  N        000BH  330# 742 744 745 762 764 765
MNE_MUL . . . . . .  N        0020H  351# 667
MNE_NOP . . . . . .  N        002BH  362# 462
MNE_ORL . . . . . .  N        0012H  337# 544 545 547 548 549 550 552 553 554 555 557 558 559 560 604 662
MNE_POP . . . . . .  N        001DH  348# 722
MNE_PUSH. . . . . .  N        001FH  350# 702
MNE_RET . . . . . .  N        002AH  361# 504
MNE_RETI. . . . . .  N        0029H  360# 524
MNE_RL. . . . . .  N        0024H  355# 505
MNE_RLC . . . . . .  N        0023H  354# 525
MNE_RR. . . . . .  N        0028H  359# 465
MNE_RRC . . . . . .  N        0026H  357# 485
MNE_SETB. . . . . .  N        0019H  344# 724 725
```

```
NAME                          TYPE     VALUE AND REFERENCES

MNE_SJMP. . . . . .           N        0004H  323# 622
MNE_SUBB. . . . . .           N        000EH  333# 647 648 649 650 652 653 654 655 657 658 659 660
MNE_SWAP. . . . . .           N        001EH  349# 707
MNE_UNDEF . . . . . .         N        FFFFH  267# 668
MNE_XCH . . . . . .           N        000DH  332# 708 709 710 712 713 714 715 717 718 719 720
MNE_XCHD. . . . . .           N        000CH  331# 729 730
MNE_XRL . . . . . .           N        0010H  335# 584 585 587 588 589 590 592 593 594 595 597 598 599 600
MNEMONIC_FACTOR . . . . . .   N        002CH  456# 1528 1541 1582 1791 2279 2286 2291
MNEMONIC_FIRST_OPERAND. . . . L CSEG   F75BH  834# 1163
MNEMONIC_INSTR_LIST_TAIL. . . L CSEG   F8DBH  1194# 1271
MNEMONIC_INSTRUCTION_TAIL . . L CSEG   F8B0H  1152# 1223
MNEMONIC_ORDINAL. . . . . .   L DSEG   006DH  436# 2298 2355 2501 2638 2643
MNEMONIC_SECOND_OPERAND_TAIL. L CSEG   F755H  806# 904 911 915 919
MNEMONIC_TAB. . . . . .       L CSEG   F529H  269# 1204 2642
MNEMONIC_TWO_OPERANDS . . . . L CSEG   F796H  883# 1164
MOVC_OPERANDS . . . . . .     L CSEG   F7FDH  941# 1165
MS00. . . . . .               L CSEG   F7DAH  902 907#
MS01. . . . . .               L CSEG   F7E7H  908 912#
MS02. . . . . .               L CSEG   F7F2H  912 916#
MT0 . . . . . .               L CSEG   F821H  950 954#
MT00. . . . . .               L CSEG   F7B4H  885 894#
MT01. . . . . .               L CSEG   F7C6H  896 900#
NEWLINE . . . . . .           N        E00FH  129# 1257 2189
NO_BIT_8. . . . . .           L CSEG   FA4FH  1703 1705#
NO_OF_OPERANDS_PRINTED. . . . L DSEG   0067H  430# 2438 2592 2648
NUMBER_OF_BYTES . . . . . .   L DSEG   004DH  88# 851 1001 1068 1099 1198 1708 1714 1734 1736 1738 1740 1767 1877 1952
                                         1985 2042 2105 2127 2152 2221 2357 2361 2408 2411 2538
NUMBER_OF_OPERANDS. . . . . .  L DSEG  006BH  434# 2305 2341 2353 2364
NUMBER_TOKE . . . . . .       N        0001H  49# 853 897 917 995 1064
OBC0. . . . . .               L CSEG   FDE0H  2400 2401#
OBC1. . . . . .               L CSEG   FDECH  2402 2406 2409#
OBC2. . . . . .               L CSEG   FDE8H  2403 2405#
OBC2_2. . . . . .             L CSEG   FDEAH  2404 2407#
OBCRET. . . . . .             L CSEG   FDF1H  2410 2412#
OFST. . . . . .               N        0010H  61# 1202 1603 1817 2465 2639
OLD_ASM_PC_HIGH . . . . . .   L DSEG   005DH  420# 1950 2069 2138 2140
OLD_ASM_PC_LOW. . . . . .     L DSEG   005EH  421# 1951 2070 2137 2139
ONE_BYTE_TAIL . . . . . .     L CSEG   F755H  805# 843 846 953 957 1162
OPERAND_BYTE_CHECK. . . . . .  L CSEG  FDDBH  2371 2372 2398#
OPERAND_C . . . . . .         L CSEG   FAFEH  1815 1837#
OPERAND_CHECK . . . . . .     L DSEG   006CH  435# 2365 2373 2399
OPERAND_FACTOR. . . . . .     N        0018H  457# 1545 1553 1803 1811 1825 1833 2308 2315 2320
ORDINAL. . . . . .            L DSEG   005CH  419# 841 848 886 951 955 1097 1529 1542 1702 1704 1712 1718 1744 1755
                                         1759 1765
ORG_TOKE. . . . . .           N        00D4H  58# 1253
OUR_CODE_HIGH . . . . . .     L DSEG   004EH  89# 1199 1489 1526 1533 1536 1550 1551 1555 1558 1596 1599 1600 1613
                                         1616 1617 1808 1809 1813 1814 1830 1831 1835 1836 1838 2274 2280 2299 2303
                                         2309 2335 2339
OUR_CODE_LOW. . . . . .       L DSEG   004FH  90# 1203 1208 1211 1219 1493 1530 1534 1537 1547 1548 1593 1594 1610
                                         1611 1805 1806 1827 1828 1840 1842 2278 2281 2295 2300 2301 2310 2324 2336
                                         2337 2344
OUR_GTRTHN. . . . . .         L CSEG   F905H  1208 1216#
PARAM1. . . . . .             N REG    R2     113# 1261 1263 1879 1887 2144 2190 2193 2466 2477 2486 2491 2503 2506
                                         2517 2521 2577 2585 2614 2640 2646
PARAM2. . . . . .             N REG    R3     114# 2191 2507 2518 2578 2586
```

```
NAME                          TYPE     VALUE AND REFERENCES

PARAM3. . . . . .  N REG    R4     115#
PARAM4. . . . . .  N REG    R5     116# 1405 1418 1426 1436 1444 1446
PARAM5. . . . . .  N REG    R6     117# 1404 1415 1420 1423 1428 1437 1441 1443 2283 2300 2312 2336
PARAM6. . . . . .  N REG    R7     118# 1403 1435 1544 1552 1605 1608 1802 1810 1819 1822 1824 1832 2284
                                     2290 2299 2313 2319 2335
PARTIT_HI_HIGH. . . . . .  L DSEG   0059H  100# 2229
PARTIT_HI_LOW . . . . . .  L DSEG   005AH  101# 417 2226
PARTIT_LO_HIGH. . . . . .  L DSEG   0057H  98# 2190 2196 2206 2224 2230
PARTIT_LO_LOW . . . . . .  L DSEG   0058H  99# 2191 2195 2203 2220 2222 2227
PC_TOKE . . . . . .  N      00A0H  59# 954
PLUS_TOKE . . . . . .  N    0005H  52# 838
PNTHGH. . . . . .  L DSEG   0044H  79# 2138 2150 2155 2206 2208
PNTLOW. . . . . .  L DSEG   0045H  80# 2137 2147 2148 2154 2205
POINT0. . . . . .  N REG    R0     111# 1260 1265 1955 1956 1961 1962 1963 1964 1969 1970 1977 1978 1979
                                     1980 2012 2013 2029 2033 2036 2037 2044 2045 2077 2078 2094 2098 2100 2101
                                     2114 2120 2121 2122 2129 2130 2131 2132 2133 2134 2141 2143 2146 2214 2215
                                     2475 2476 2496 2497 2498 2499 2500 2526 2527 2532 2533 2545 2546
POINT1. . . . . .  N REG    R1     112# 2197 2199 2204 2213 2216
POUND_EXP_OP2 . . . . . .  N        4A40H  406# 507 527 545 547 565 567 585 587 607 608 609 610 612 613 614 615 617
                                     618 619 620 647 687 689 690 692 693 694 695 697 698 699 700
POUND_TOKE. . . . . .  N    0006H  51# 890 908 990
PRINT_STRING. . . . . .  N  E01EH  133# 2579 2587
QUOTIENT_HIGH . . . . . .  L DSEG   0071H  440# 1436 2284 2289 2293 2294 2313 2318 2322 2323
QUOTIENT_LOW. . . . . .  L DSEG   0072H  441# 1437 2283 2285 2288 2297 2312 2314 2317 2326
R0_OP1. . . . . .  N        00B0H  370# 472 492 612 672 692 732 772
R0_OP2. . . . . .  N        1080H  394# 512 532 552 572 592 632 652 712 752
R1_OP1. . . . . .  N        00DCH  371# 473 493 613 673 693 733 773
R1_OP2. . . . . .  N        14A0H  395# 513 533 553 573 593 633 653 713 753
R2_OP1. . . . . .  N        0108H  372# 474 494 614 674 694 734 774
R2_OP2. . . . . .  N        18C0H  396# 514 534 554 574 594 634 654 714 754
R3_OP1. . . . . .  N        0134H  373# 475 495 615 675 695 735 775
R3_OP2. . . . . .  N        1CE0H  397# 515 535 555 575 595 635 655 715 755
R4_OP1. . . . . .  N        0160H  374# 477 497 617 677 697 737 777
R4_OP2. . . . . .  N        2100H  398# 517 537 557 577 597 637 657 717 757
R5_OP1. . . . . .  N        018CH  375# 478 498 618 678 698 738 778
R5_OP2. . . . . .  N        2520H  399# 518 538 558 578 598 638 658 718 758
R6_OP1. . . . . .  N        01B8H  376# 479 499 619 679 699 739 779
R6_OP2. . . . . .  N        2940H  400# 519 539 559 579 599 639 659 719 759
R7_OP1. . . . . .  N        01E4H  377# 480 500 620 680 700 740 780
R7_OP2. . . . . .  N        2D60H  401# 520 540 560 580 600 640 660 720 760
REG . . . . . .  N          0040H  60# 1604 1818 2465
REL_OFFSET_HIGH . . . . . .  L DSEG   0060H  423# 1988 1989 2005 2022 2023 2047 2048 2064 2087 2088
REL_OFFSET_LOW. . . . . .  L DSEG   0061H  424# 1987 1996 2002 2004 2009 2013 2019 2020 2026 2030 2046 2055 2061
                                     2063 2074 2078 2084 2085 2091 2095
REL8_OP1. . . . . .  N      03C8H  386# 542 562 582 602 622
REL8_OP2. . . . . .  N      5AC0H  409# 482 502 522 728 732 733 734 735 737 738 739 740
ROTATE. . . . . .  L CSEG   F983H  1411 1413 1435#
ROTATE_CONTINUE . . . . . .  L CSEG  F98AH  1435 1439#
SAVE_AND_DISPLAY. . . . . .  N      E05CH  141# 1262 1264
SECOND_EXP. . . . . .  L BSEG   0004H  450# 1197 1727 1746 1750 1761
SECOND_NO_BIT_8 . . . . . .  L CSEG  FA97H  1752 1754#
SECOND_NOT_REGISTER . . . . . .  L CSEG  FADBH  1795 1797 1816#
SECOND_OPER_ORDINAL . . . . . .  L DSEG  0064H  427# 2344 2345 2351 2373 2446
SELECT. . . . . .  L DSEG   0046H  81# 2136 2209
SELECT_CON. . . . . .  N    0000H  65#
```

```
NAME                          TYPE     VALUE AND REFERENCES

SET_BIT_EXP . . . . . . . . . L CSEG   FA3FH  1601 1618#
SET_EXP_16_FLAG . . . . . . . L CSEG   FA57H  892 1126 1711#
SET_EXP_FLAG. . . . . . . . . L CSEG   FA60H  898 1717#
SET_POUND_EXP_FLAG. . . . . . L CSEG   FA87H  909 991 1743#
SET_REL_FLAG. . . . . . . . . L CSEG   FAA3H  1029 1061 1066 1764#
SET_SLASH_EXP_FLAG. . . . . . L CSEG   FA9BH  913 1758#
SLASH_EXP_OP2 . . . . . . . . N        4E60H  407# 662 682
SS0 . . . . . . . . . . . . . L CSEG   FE88H  2512 2514#
SS1 . . . . . . . . . . . . . L CSEG   FE8EH  2514 2516#
SS2 . . . . . . . . . . . . . L CSEG   FE9BH  2516 2521#
SS3 . . . . . . . . . . . . . L CSEG   FE91H  2513 2515 2517#
START_DIVIDE. . . . . . . . . L CSEG   F953H  1401# 2282 2311
STO1. . . . . . . . . . . . . L CSEG   F848H  990 994#
STORE . . . . . . . . . . . . N        E04DH  136# 2145
STORET. . . . . . . . . . . . L CSEG   F851H  993 997#
STRGBF. . . . . . . . . . . . L DSEG   003CH  73#
STRGCT. . . . . . . . . . . . L DSEG   0055H  96#
SUBTRACT_WITH_C . . . . . . . L CSEG   F971H  1410 1422# 1434
TEMP. . . . . . . . . . . . . N REG    R5     121#
TEMP_LOW. . . . . . . . . . . L DSEG   0047H  82# 2109 2113 2211 2215 2272 2275 2276 2553 2556
TEMP_SEC. . . . . . . . . . . L DSEG   0062H  425# 899 997 1069 1973 1975 1978 2101
TEMP1 . . . . . . . . . . . . L DSEG   0056H  97#
THIRD_OPER_ORDINAL. . . . . . L DSEG   0065H  428# 2358 2448
THREE_OPERANDS. . . . . . . . L CSEG   F82DH  984# 1166
TIME. . . . . . . . . . . . . N        E012H  130#
TOERR . . . . . . . . . . . . L CSEG   F861H  917 995 1003# 1064
TOKSIZ. . . . . . . . . . . . N        0004H  63# 73
TOKSTR. . . . . . . . . . . . L DSEG   0048H  83# 907 1201 1583 1589 1606 1619 1792 1798 1820
UL0 . . . . . . . . . . . . . L CSEG   F9E2H  1527 1541#
UL1 . . . . . . . . . . . . . L CSEG   F9DDH  1535 1537#
UNDEFINED_OPCODE. . . . . . . N        00A5H  458# 2263
UPDATE_LSSTHN . . . . . . . . L CSEG   F9E0H  1530 1539#
UPDATE_OUR_CODE . . . . . . . L CSEG   F9C7H  842 849 887 952 956 1098 1525# 1706 1713 1719 1745 1756 1760 1766
VALHGH. . . . . . . . . . . . L DSEG   0049H  84# 1255 1988 2047 2107 2115 2132 2498 2506 2517 2531 2562 2571
VALLOW. . . . . . . . . . . . L DSEG   004AH  85# 899 997 1069 1256 1964 1974 1980 1987 2046 2122 2134 2500 2507 2518
                                         2521 2533 2557 2566
WAIT_FOR_USER . . . . . . . . N        E062H  143# 2235
WORKING_SPACE . . . . . . . . L DSEG   0040H  74# 1955 1961 1969 1977 2012 2029 2036 2044 2077 2094 2100 2114 2129
                                         2141 2212 2262 2473 2496 2526 2543 2660 2662 2663 2663 2664

ASSEMBLY COMPLETE, NO ERRORS FOUND
```

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

Printed in U.S.A.