4051 assembly code documentation continuation, covering CALL “EXEC”. CALL “EXEC”,A$ (or similar) is in every 4051 and it reads the string in a weird hex format (the cheapest possible form of hexadecimal), converting it to binary and writing the result into the scratch pad space. It then starts executing at the beginning of the scratch pad memory space.

As previously mentioned the scratch pad is 260 bytes long, and you should avoid using the last few bytes if you currently have EXTENDED BASIC or may get EXTENDED BASIC in the future since it steals a few bytes at the end of the scratch pad space.

The stupid hex format is simply a continuation of the ASCII chart immediately following 9, as shown:
0 1 2 3 4 5 6 7 8 9 : ; < = > ?

The way to create the above fake hex from the real hex created by the assembler is to first manually remove the first 7 and last 2 characters of each line (address and checksum) and remove the last line (final checksum). Then manually convert every A to a : and so on for the last 6 hex alpha digits.

The very last instruction in the hex string should always be a 39 (RTS, a return from subroutine). Since not much can be written in only 256 bytes of 6800 assembly code, a way around that limitation is to have the scratch pad code search the symbol table for a different string variable, say B$, and then to convert that from real hex to binary, which can be done in place within the string (but don’t ever try to print the converted string) and then JSR to the first 6800 instruction beginning of the converted string. When the B$ string program finishes execution it needs to perform an RTS to get back to the RTS in the scratch pad, unless the scratch pad code did a JMP instead of a JSR to execute the B$ string code, in which case the B$ RTS gets you back to BASIC.

This is the method that I use to do first time firmware programming of the flash memory (the first page of the flash holds the firmware, I/O and static RAM) since there is nothing in the flash for self programming it. After the RAMPACK firmware is installed it can reprogram itself using a special CALL, which also runs out of the built-in static RAM to do the flash programming. The self programming also checks the new firmware for correct formatting and using a very thorough checksum (unlike the idiot checksum in the 4051) to ensure that the new firmware is a valid copy. Those without an Option 1 (RS-232) would either need to type in the new firmware, use another RAMPACK with the new code, or else send their RAMPACK back to me for an update. I doubt that many want to type in the over 10K characters of new RAMPACK firmware. Currently, everyone has the latest copy of the firmware so no need to panic. I currently have just over 50 bytes of unused RAMPACK firmware space available in the flash memory for bug fixes, only enough for a couple of relatively simple bug fixes.
Given the 6800 instruction set and the arcane way that some things are done in the 4051 with the tokenized stack, fixing a bug with just 25 bytes generally requires that the fix be fairly simple.

If anyone wants to know how to search the 4051 symbol table just send me email and I will post an example program. Also note that every 4051 System variable in the symbol table starts with a NULL (Hex 00) character while all user variables always start with A…Z. One should not fool around with the 4051 System variables for quite obvious reasons. DELETE ALL or OLD wipes out ***all*** symbol table entries, even System ones, and then the firmware rebuilds the System variables in the new symbol table.

Here are some additional 4051 assembly code programming details. For assembling and linking I use ASXV5LXX, which is a Windows compatible cross assembler that creates code for dozens of different microprocessors (I use Windows 7 Ultimate 64 bit): https://shop-pdp.net/ashtml/asxbld.htm

I have some 14TB of disk drive space so ignore the disk drive letter since these files can be located anywhere.

My "ASM 6800.BAT" BATCH file looks like this:
E:\4051\RAMPack\as6800 -xgalosff E:\4051\RAMPack\NewRAMPack

And my "LINK 6800.BAT" file looks like this:
E:\4051\RAMPack\aslink -mxsu E:\4051\RAMPack\NewRAMPack E:\4051\RAMPack\NewRAMPack

The 4051 firmware has 24 sixteen bit pseudo registers located at the bottom on RAM. Many of the system calls use these pseudo registers for parameter passing and temporary space. There are a ***lot*** of other temporary locations that specific firmware routines use. Messing with the pseudo registers and other temporary locations needs to be carefully done to prevent system crashes. Unfortunately, I do not have any cross reference showing pseudo register and temporary location usage by the various routines (which should have existed!) so it is often necessary to examine the firmware listings to determine which registers are used by which routines. Since the firmware listings have errors some unintended surprises may still occur. There is also a 260 byte "Scratch Pad" area in RAM that is often (but ***not*** always) available. For example, the EXTENDED BASIC ROM PACK (also in a MAXIPACK) steals some of the scratch pad and other temporary locations in RAM, as I found out the hard way. The scratch pad is also used for print formatting.

Internally the 4051 firmware uses tokens on the stack to determine the stack contents. A BASIC program has tokens identifying each semi-compiled BASIC line, indicating what each line contains, as well as floating point values, return addresses, etc. What this means is that you cannot simply call a subroutine that may invoke the interpreter or the de-compiler since those rely on stack tags for parsing the stack, so tagging the stack is required in many situations. For example, if I need to call a routine that could examine the stack I need to manually push my return address on the stack, then push an ITEM2 stack tag constant and then JUMP to the firmware routine. That firmware routine will eventually perform a "dirty" exit by removing the stack tag and then moving the return address into one of the multiple special RAM locations, and then jump to that pre-setup on Power Up RAM routine that does a JUMP to the copied address. So instead of JSR and RTS one has to use this arcane method whenever any firmware stack examination routines are called.

Since error exits are possible, one must be ***sure*** to clean up the stack before returning to BASIC if any JSRs were performed or else the system will eventually go wonky and crash.

Since the RAMPACK required persistent data between BASIC program line evaluations I originally made entries into the symbol table to hold that system data, but it turned out that a BASIC OLD @51: statement actually blows away the symbol table, resulting in no open file (there is a special exception in the MAG TAPE routines to retain the state of the mag tape when an OLD statement is encountered). Trying to use the pseudo registers was also eventually abandoned since the poor documentation did not allow for immediately knowing which registers were "safe" to use, and my trial and error attempts to narrow the list to only safe registers ultimately ended up with way too little RAM space for even one RAMPACK, let alone multiple ones.

Similarly, since the RAMPACK requires that some of its firmware actually run in RAM rather than from the flash memory whenever the flash memory is being erased or written, trying to use the scratch pad space for that was not successful since a PRINT @51: would wipe the scratch pad memory that I would be trying to use for writing the flash memory.

Hence I ultimately ended up piggybacking a small static RAM on top of one of the flash memories to hold ***all*** of the RAMPACK state and relocating some of the RAMPACK firmware there so that it could run when the flash memory was unavailable. This also has the advantage that installing 1 or even multiple RAMPACKs does not steal any 4051 memory other than PIA table entries, which is always required for all ROM PACKs that perform I/O operations (i.e. PRINT @ and INPUT @, etc).

Below is a list of system constants and routine entry points. I have removed all of my RAMPACK constant and memory references. I've had very large portions my original assembly code stolen and then sold before now, so I avoid publically showing anything that I create anymore. Some people are so self centered that they have ***zero*** ethics.

```
            .TITLE     4051 RAMPACK VERS 1.0 BY MICHEAL D. CRANFORD

            .LIST

DEBUGGING   .EQU       0

            .SBTTL     4051 SYSTEM CONSTANT LABELS

ASCIIHORTAB .EQU       0H009          ; THE ASCII TAB CHARACTER
ASCIIRETURN .EQU       0H00D          ; THE ASCII CARRIAGE RETURN
CHARACTER
ASCIISPACE  .EQU       0H020          ; THE ASCII SPACE CHARACTER
ASCIIUPPERY .EQU       0H059          ; THE ASCII UPPER CASE Y
CHARACTER
ASCIICURSOR .EQU       0H0B2          ; THE ASCII CURSOR CHARACTER

LITERALTAG  .EQU       0H001          ; THE BASIC SYSTEM LITERAL
STRING TAG
STRINGTAG   .EQU       0H008          ; THE BASIC SYSTEM VARIABLE
STRING TAG
RETURNTAG   .EQU       0H015          ; THE BASIC SYSTEM RETURN
ADDRESS TAG
CALLINGTAG  .EQU       0H017          ; THE BASIC SYSTEM CALL TAG

APPENDTOKEN .EQU       0H059          ; THE BASIC PROGRAM APPEND TOKEN
INITTOKEN   .EQU       0H062          ; THE BASIC PROGRAM INITIALIZE
TOKEN
OLDTOKEN    .EQU       0H06E          ; THE BASIC PROGRAM OLD TOKEN


            .SBTTL     4051 SYSTEM VARIABLE LABELS
```

```
REGISTER00  .EQU       0H00000        ; PSEUDO REGISTER 00
K0          .EQU       0H00000        ;
K1          .EQU       0H00001        ;
REGISTER01  .EQU       0H00002        ; PSEUDO REGISTER 01
K2          .EQU       0H00002        ;
K3          .EQU       0H00003        ;
REGISTER02  .EQU       0H00004        ; PSEUDO REGISTER 02
K4          .EQU       0H00004        ;
K5          .EQU       0H00005        ;
REGISTER03  .EQU       0H00006        ; PSEUDO REGISTER 03
K6          .EQU       0H00006        ;
REGISTER04  .EQU       0H00008        ; PSEUDO REGISTER 04
REGISTER05  .EQU       0H0000A        ; PSEUDO REGISTER 05
REGISTER06  .EQU       0H0000C        ; PSEUDO REGISTER 06
REGISTER07  .EQU       0H0000E        ; PSEUDO REGISTER 07
REGISTER08  .EQU       0H00010        ; PSEUDO REGISTER 08
REGISTER09  .EQU       0H00012        ; PSEUDO REGISTER 09
REGISTER10  .EQU       0H00014        ; PSEUDO REGISTER 10
REGISTER11  .EQU       0H00016        ; PSEUDO REGISTER 11
INTEGERONE  .EQU       0H00016        ; INTEGER WORKING STORAGE
REGISTER12  .EQU       0H00018        ; PSEUDO REGISTER 12
INTEGERTWO  .EQU       0H00018        ; INTEGER WORKING STORAGE
REGISTER13  .EQU       0H0001A        ; PSEUDO REGISTER 13
DIMSUBSCRIP .EQU       0H0001A        ; DIMENSION SUBSCRIPT COUNTER
REGISTER14  .EQU       0H0001C        ; PSEUDO REGISTER 14
TABLEPOINT  .EQU       0H0001C        ; NAME TABLE POINTER
REGISTER15  .EQU       0H0001E        ; PSEUDO REGISTER 15
BYTEALLOC   .EQU       0H0001E        ; BYTE ALLOCATION COUNT
REGISTER16  .EQU       0H00020        ; PSEUDO REGISTER 16
DIMENLOOPER .EQU       0H00020        ; DIMENSION LOOP COUNTER
REGISTER17  .EQU       0H00022        ; PSEUDO REGISTER 17
DIMSUBFLAG  .EQU       0H00022        ; DIMENSION SUBSCRIPT FLAG
REGISTER18  .EQU       0H00024        ; PSEUDO REGISTER 18
RETURNADDR  .EQU       0H00024        ; TEMPORARY RETURN ADDRESS
REGISTER19  .EQU       0H00026        ; PSEUDO REGISTER 19
REGISTER20  .EQU       0H00028        ; PSEUDO REGISTER 20
REGISTER21  .EQU       0H0002A        ; PSEUDO REGISTER 21
REGISTER22  .EQU       0H0002C        ; PSEUDO REGISTER 22
REGISTER23  .EQU       0H0002E        ; PSEUDO REGISTER 23
FLOATEQUAL  .EQU       0H00030        ; FLOATING POINT INPUT FLAG
FLOATCONVER .EQU       0H00031        ; FLOATING POINT CONVERSION FLAG
LITERALLENG .EQU       0H00032        ; UNCOMPRESS LITERAL LENGTH
COUNTER
SHUNTCOUNT  .EQU       0H00033        ; BYTES NEEDED FOR POSTFIX LINE
TRANSFLAGS  .EQU       0H00035        ; BASIC TRANSLATOR FLAGS
LEXSCRCOUNT .EQU       0H00036        ; LEXICAL SCRATCH BYTE COUNT
UNCOMPSCORE .EQU       0H00038        ; UNCOMPRESS SCORE
UNLEXSPACE  .EQU       0H00039        ; LAST CHARACTER WAS A SPACE
COMPLINEFLG .EQU       0H0003A        ; UNCOMPILE COMPRESS LINE FLAG
```

```
SYMBOLPOINT .EQU       0H0003B        ; VARIABLE SYMBOL TABLE POINTER
USERPROGRAM .EQU       0H0003D        ; PROGRAM FIRST LINE POINTER
HIGHERPOINT .EQU       0H0003F        ; RAM HIGH END POINTER
THEZEROWORD .EQU       0H00041        ; THE ZERO WORD, FOR ZEROING X
EXESTAKBASE .EQU       0H00043        ; EXECUTION STACK BASE POINTER
LOSTAKPOINT .EQU       0H00045        ; LOW RAM STACK POINTER
USERORIGIN  .EQU       0H00047        ; FIRST FREE RAM BYTE POINTER
EXECSTPOINT .EQU       0H00049        ; EXECUTION STACK POINTER
ERRORNUMBER .EQU       0H0004B        ; ERROR CODE HOLDING AREA
ERRORBACKUP .EQU       0H0004C        ; ERROR CODE BACKUP AREA
CURRENTLINE .EQU       0H0004D        ; CURRENTLY EXECUTING LINE
POINTER
NEXTLINEPTR .EQU       0H0004F        ; NEXT LINE TO EXECUTE POINTER
NEXTOKENPTR .EQU       0H00051        ; NEXT TOKEN TO EXECUTE POINTER
CURRENTOKEN .EQU       0H00053        ; CURRENT TOKEN HOLDING AREA
LOCALFLAGS  .EQU       0H00054        ; EVALUATOR LOCAL LINE FLAGS
GLOBALFLAGS .EQU       0H00055        ; EVALUATOR GLOBAL CONTROL FLAGS
OPERADDRESS .EQU       0H00056        ; EVALUATOR OPERATION HOLDING
AREA
DIRTYEXITA  .EQU       0H00058        ; JUMP TO RETURN ADDRESS DIRTY
EXIT A
DIRTYEXITB  .EQU       0H0005B        ; JUMP TO RETURN ADDRESS DIRTY
EXIT B
INTERRUPTED .EQU       0H0005E        ; INTERRUPT TEMPORARY REGISTER
EDITBUFFEND .EQU       0H00060        ; EDIT BUFFER END POINTER
EDITPOINTER .EQU       0H00062        ; LINE EDITOR CURSOR POINTER
EDITMAXIMUM .EQU       0H00064        ; LINE EDITOR WORKING REGISTER
AUTONUMBCUR .EQU       0H00066        ; AUTO NUMBER CURRENT
AUTONUMBINC .EQU       0H00068        ; AUTO NUMBER INCREMENT
INPUTSTATUS .EQU       0H0006A        ; BASIC INPUT BUFFER STATUS
KEYBOARDFLG .EQU       0H0006B        ; KEYBOARD STATUS BYTE FLAG
LASTKEYCODE .EQU       0H0006C        ; LAST VALID ASCII KEY CODE
PENDINGFLAG .EQU       0H0006D        ; PENDING FLAGS HOLDING AREA
PENDINGEOFS .EQU       0H0006E        ; PENDING EOF HOLDING AREA
MAGTAPEBUSY .EQU       0H0006F        ; MAG TAPE BUSY FLAG
MTSTAT2FLAG .EQU       0H00070        ; MAG TAPE STATUS 2 BYTE
DISPLAYSTAT .EQU       0H00071        ; CRT DISPLAY STATUS BYTE
KEYINPPOINT .EQU       0H00072        ; TYPE AHEAD INPUT POINTER
KEYOUTPOINT .EQU       0H00074        ; TYPE AHEAD OUTPUT POINTER
KEYLASTCHAR .EQU       0H00076        ; KEYBOARD LAST KEY TRANSFERRED
PERCENTMODE .EQU       0H00077        ; I/O SYSTEM PERCENT MODE FLAG
TEXTEOLCHAR .EQU       0H00078        ; I/O SYSTEM EOL CHARACTER
TEXTEOFCHAR .EQU       0H00079        ; I/O SYSTEM EOF CHARACTER
IGNOREDCHAR .EQU       0H0007A        ; I/O SYSTEM NULL CHARACTER
SECRETSTAT  .EQU       0H0007B        ; 4051 SECRET OUTPUT STATUS
ROMPACKBANK .EQU       0H0007C        ; ROM PACK BANK SWITCH NUMBER
LOGICALUNIT .EQU       0H0007D        ; FILE SYSTEM LOGICAL UNIT
NUMBER
RECORDNUMB  .EQU       0H0007E        ; PRESENT FILE RECORD NUMBER
REQUESTED
```

```
SECRETMODE  .EQU       0H00080        ; BASIC PROGRAM SECRET MODE
STATUS
MTSTATUSREG .EQU       0H00081        ; MAG TAPE FORMAT STATUS
REGISTER
MTSTATFLAGS .EQU       0H00082        ; MAG TAPE STATUS FLAG BYTE
MAGTAPEMAX  .EQU       0H00083        ; MAG TAPE BUFFER LAST CHARACTER
POINTER
MTBUFFPOINT .EQU       0H00085        ; MAG TAPE BUFFER POINTER
OUTBUFFSTAT .EQU       0H00087        ; OUTPUT BUFFER STATUS
IOSCANPOINT .EQU       0H00088        ; IO SCAN STACK POINTER
IOPROCFLAGS .EQU       0H0008A        ; IO PROCESSOR STATUS FLAGS
CHARCOUNTER .EQU       0H0008B        ; STRING INPUT CHARACTER COUNT
FIRSTSTRING .EQU       0H0008D        ; FIRST STRING CHARACTER
IOFUNCTION  .EQU       0H0008E        ; BASIC I/O FUNCTION KEYWORD
IOSTATUS    .EQU       0H0008F        ; SYSTEM I/O STATUS FLAGS
PRIMARYIO   .EQU       0H00090        ; CURRENT PRIMARY I/O ADDRESS
BUFFMINIMUM .EQU       0H00091        ; BASIC I/O BUFFER STARTING
POINTER
BUFFMAXIMUM .EQU       0H00093        ; BASIC I/O BUFFER STOPPING
POINTER
BUFFERTAIL  .EQU       0H00095        ; BASIC BUFFER STOPPING POINTER
SECONDARYIO .EQU       0H00097        ; CURRENT I/O SECONDARY ADDRESS
BUFFERHEAD  .EQU       0H00098        ; BASIC BUFFER STARTING POINTER
;STRDIMSIZE .EQU       0H0009A        ; STRING DIMENSIONED LENGTH
(*REDEFINED)
SYSTEMPOINT .EQU       0H0009C        ; IO SYSTEM POINTER TO VALUE OR
STRING
MATCOLCOUNT .EQU       0H0009E        ; MATRIX COLUMN COUNT
IONAMETABLE .EQU       0H000A0        ; IO SCAN NAME TABLE POINTER
MATCOLCOUNT .EQU       0H000A2        ; IO TEMPORARY MATRIX COLUMN
COUNTER
DISPYAXIS   .EQU       0H000A4        ; DISPLAY AXIS FLIP-FLOP FLAG
DISPTEMPONE .EQU       0H000A5        ; DISPLAY DRIVER TEMPORARY 1
DISPTEMPTWO .EQU       0H000A6        ; DISPLAY DRIVER TEMPORARY 2
DISPTEMPTWE .EQU       0H000A7        ; DISPLAY DRIVER TEMPORARY 3
CRTDISPVERT .EQU       0H000A8        ; CRT DISPLAY VERTICAL TEKPOINTS
CRTDISPHORZ .EQU       0H000AA        ; CRT DISPLAY HORIZONTAL
TEKPOINTS
CHARCOLUMN  .EQU       0H000AC        ; CHARACTER PAINTER COLUMN COUNT
CHARROW     .EQU       0H000AD        ; CHARACTER PAINTER ROW COUNT
CHARBLINKER .EQU       0H000AE        ; DRAW WRITE-THRU CHARACTERS
DOTCONTROL  .EQU       0H000AF        ; CRT DRIVER DOT CONTROL
CRTPAGEFULL .EQU       0H000B0        ; CRT DISPLAY PAGE FULL
CURSORCOUNT .EQU       0H000B1        ; CURSOR GENERATOR COUNT
CURSORCHAR  .EQU       0H000B2        ; CURSOR ASCII CHARACTER
CURRENTFONT .EQU       0H000B3        ; CURRENT DISPLAY FONT
MTEXTRABYTE .EQU       0H000B4        ; MAG TAPE EXTRA BYTE COUNT
MTLINECOUNT .EQU       0H000B5        ; MAG TAPE RECORD COUNT
MTDRIVEWARN .EQU       0H000B7        ; MAG TAPE DRIVE STATUS
MTCONDCODES .EQU       0H000B8        ; MAG TAPE CONDITION CODES
```

```
MTDRIVEFLAG .EQU     0H000B9        ; MAG TAPE DRIVER FLAGS
PIAMTBUFFER .EQU     0H000BA        ;
MTCURRFILE  .EQU     0H000BB        ; MAG TAPE CURRENT FILE
MTFINDFILE  .EQU     0H000BD        ; MAG TAPE FILE TO FIND
FPSCRATCHT1 .EQU     0H000BF        ; FP SCRATCH TEMPORARY BYTE 1
FPSCRATCHT2 .EQU     0H000C0        ; FP SCRATCH TEMPORARY BYTE 2
FPSCRATCHT3 .EQU     0H000C1        ; FP SCRATCH TEMPORARY BYTE 3
FPSCRATCHT4 .EQU     0H000C2        ; FP SCRATCH TEMPORARY BYTE 4
SIGNOFNUMB  .EQU     0H000C3        ; FP CONVERSION SIGN OF NUMBER
FPTEMPPOINT .EQU     0H000C4        ; FP TEMPORARY POINTER
FPMULTPOINT .EQU     0H000C6        ; FP OUTPUT POWER OF 10 POINTER
FPFRACTY5   .EQU     0H000C8        ; FP FRACTION Y BYTE 5
FPFRACTY4   .EQU     0H000C9        ; FP FRACTION Y BYTE 4
FPFRACTY3   .EQU     0H000CA        ; FP FRACTION Y BYTE 3
FPFRACTY2   .EQU     0H000CB        ; FP FRACTION Y BYTE 2
FPFRACTY1   .EQU     0H000CC        ; FP FRACTION Y BYTE 1
FPFRACTY0   .EQU     0H000CD        ; FP FRACTION Y BYTE 0
FPFRACTX5   .EQU     0H000CE        ; FP FRACTION X BYTE 5
FPFRACTX4   .EQU     0H000CF        ; FP FRACTION X BYTE 4
FPFRACTX3   .EQU     0H000D0        ; FP FRACTION X BYTE 3
FPFRACTX2   .EQU     0H000D1        ; FP FRACTION X BYTE 2
FPFRACTX1   .EQU     0H000D2        ; FP FRACTION X BYTE 1
FPFRACTX0   .EQU     0H000D3        ; FP FRACTION X BYTE 0
EXPONBUFFER .EQU     0H000D4        ; EXPONENT WORD BUFFER
IMGSTRPOINT .EQU     0H000D6        ; IMAGE STRING POINTER
DATAPOINTER .EQU     0H000D8        ; DATA POINTER
REPEATLOOP1 .EQU     0H000DA        ; REPEAT LOOP 1
RELSTPOINT1 .EQU     0H000DB        ; RELATIVE STRING POINTER 1
REPEATLOOP2 .EQU     0H000DD        ; REPEAT LOOP 2
RELSTPOINT2 .EQU     0H000DE        ; RELATIVE STRING POINTER 2
REPEATLOOP3 .EQU     0H000E0        ; REPEAT LOOP 3
RELSTPOINT3 .EQU     0H000E1        ; RELATIVE STRING POINTER 3
REPEATLOOP4 .EQU     0H000E3        ; REPEAT LOOP 4
RELSTPOINT4 .EQU     0H000E4        ; RELATIVE STRING POINTER 4
CURRELPOINT .EQU     0H000E6        ; RELATIVE CURSOR POINTER
PARENTHESIS .EQU     0H000E8        ; PARENTHESIS COUNT
IMAGSTRSIZE .EQU     0H000E9        ; IMAGE STRING LENGTH
DATALENGTH  .EQU     0H000EB        ; DATA LENGTH
NUMDATATYPE .EQU     0H000ED        ; NUMERIC DATA TYPE
LITERALFLAG .EQU     0H000EE        ; LITERAL FLAG
NOPRINTEDCR .EQU     0H000EF        ; NO PRINTF CR FLAG
NUMBERFLAG  .EQU     0H000F0        ; NUMBER FLAG
ASREQUIRED  .EQU     0H000F1        ; AS REQUIRED FLAG
POSITIVELY  .EQU     0H000F2        ; PLUS FLAG
NEGATIVELY  .EQU     0H000F3        ; MINUS FLAG
DOLLARFLAG  .EQU     0H000F4        ; DOLLAR FLAG
COMMAFORMAT .EQU     0H000F5        ; COMMA FORMAT FLAG
FORMATTYPE  .EQU     0H000F6        ; FORMAT TYPE TO PROCESS
PRINTFSTAT  .EQU     0H000F7        ; PRINT FORMATTED STATUS BYTE
IOSYSTEMNTP .EQU     0H000F8        ; I/O SYSTEM NAME TABLE POINTER
```

```
FRETMT      .EQU    0H000FA     ; MTCTL TEMPORARY RETURN ADDRESS
DUMWRITSTAT .EQU    0H000FC     ; IECDRV DUMMY WRITE STATUS
BANKADDRESS .EQU    0H000FD     ; CURRENT BANK BASE ADDRESS
;           .EQU    0H000FF     ;
KEYBOARDQUE .EQU    0H00100     ; KEYBOARD QUEUE START (31
CHARACTERS)
KEYBOARDEND .EQU    0H0011E     ; KEYBOARD QUEUE ENDING
SYSTEMTRASH .EQU    0H0011E     ; IO SYSTEM SCRATCH VARIABLE
MAGTAPEBUFF .EQU    0H0011F     ; START OF 258 BYTE MAG TAPE
BUFFER
EDITBUFFER  .EQU    0H00221     ; 74 CHARACTER EDIT LINE BUFFER
START
IOBUFFERONE .EQU    0H0026B     ; 74 CHARACTER IO LINE BUFFER
START
SCRATCHPAD  .EQU    0H002B5     ; 260 BYTE SCRATCH PAD RAM AREA
START
SCRATCHEND  .EQU    0H003B9     ; 260 BYTE SCRATCH PAD RAM AREA
END+1
ONUNITTABLE .EQU    0H003B9     ; ON UNIT PROCESSING COMMAND
TABLE
ACTEOFTABLE .EQU    0H003C9     ; TABLE OF ACTIVE EOF LINE
ADDRESSES
FUNCTTABLE  .EQU    0H003DD     ; BASIC FUNCTION TABLE LINE
ADDRESSES
USERDEFKEY  .EQU    0H00411     ; USER DEFINABLE KEY STACK
CRASHERROR  .EQU    0H0041A     ; SYSTEM ERROR HOLDING AREA
EXECUTSTACK .EQU    0H00424     ; EXECUTION STACK SPACE
ERRMESSHOLD .EQU    0H00434     ; ERROR MESSAGE WRITER WORK AREA
ERRORCOUNTA .EQU    0H00435     ; UNLEX ERROR TOKEN NUMBER
ERRORCOUNTB .EQU    0H00436     ; UNLEX EXIT ERROR TOKEN NUMBER
OPERATERET  .EQU    0H00437     ; OPERATOR RETURN
CURRDATAFMT .EQU    0H00439     ; CURRENT DATA STATEMENT POINTER
CURRDATAOBJ .EQU    0H0043B     ; CURRENT DATA STATEMENT OBJECT
POINTER
BRACKCOUNT  .EQU    0H0043D     ; DIMENSION SUBSCRIPT BRACKET
COUNT
DSTATPOINT  .EQU    0H0043D     ; CURRENT DATA STATEMENT POINTER
DOBJEPOINT  .EQU    0H0043F     ; CURRENT DATA OBJECT ENTRY
POINTER
AUTONUMHOLD .EQU    0H00462     ; AUTO NUMBER HOLD FLAG
CRLFIOMODE  .EQU    0H00463     ; CARRIAGE RETURN LINE FEED MODE
PERCENTEOL  .EQU    0H00464     ; PERCENT MODE END OF LINE
CHARACTER
PERCENTEOF  .EQU    0H00465     ; PERCENT MODE END OF FILE
CHARACTER
PERCENTNULL .EQU    0H00466     ; PERCENT MODE NULL CHARACTER
ROMIOSYSADD .EQU    0H00467     ; ROM PACK IO SYSTEM ADDRESSED
FLAG
MAGTAPEMASK .EQU    0H00468     ; UNUSED BY THE MAG TAPE
ROUTINES ?
```

```
EXTENDFUNCT .EQU       0H00469        ; FOUR EXTENDED FUNCTION BANKS
CHARHANDLER .EQU       0H0046D        ; OPTIONAL CHARACTER HANDLER
DISKBANK    .EQU       0H0046E        ; DISK BANK, IF PRESENT
ONFULLBANK  .EQU       0H0046F        ; ON FULL BANK, IF PRESENT
IDLEVECTOR  .EQU       0H00470        ; MASTER IDLE LOOP ALTERNATE
VECTOR
;           .EQU       0H00472        ;
CHARVECTOR  .EQU       0H00473        ; ALTERNATE CHARACTER PAINTER
POINTER
FULLPOINTER .EQU       0H00476        ; IO PROCESSOR PAGE FULL POINTER
OPT1EOTFLAG .EQU       0H00478        ; OPTION 1 EOT DETECTED FLAG
SCRAMBSEED  .EQU       0H00479        ; PRERSENT SCRAMBLE CHARACTER
XMINWINDOW  .EQU       0H0047A        ; PRESENT XMIN WINDOW FPN
XMAXWINDOW  .EQU       0H00482        ; PRESENT XMAX WINDOW FPN
YMINWINDOW  .EQU       0H0048A        ; PRESENT YMIN WINDOW FPN
YMAXWINDOW  .EQU       0H00492        ; PRESENT YMAX WINDOW FPN
DELTAXAXIS  .EQU       0H0049A        ; XMAX VIEWPORT - XMIN VIEWPORT
XMINVIEWPRT .EQU       0H004A2        ; PRESENT XMIN VIEWPORT FPN
DELTAYAXIS  .EQU       0H004AA        ; YMAX VIEWPORT - YMIN VIEWPORT
YMINVIEWPRT .EQU       0H004B2        ; PRESENT YMIN VIEWPORT FPN
XSCALEFACT  .EQU       0H004BA        ; X AXIS SCALE FACTOR FPN
USERLASTX   .EQU       0H004C2        ; PRESENT LAST X POSITION FPN
YSCALEFACT  .EQU       0H004CA        ; Y AXIS SCALE FACTOR FPN
USERLASTY   .EQU       0H004D2        ; PRESENT LAST Y POSITION FPN
SINETHETA   .EQU       0H004DA        ; RMOVE AND RDRAW ROTATION ANGLE
SINE
COSINETHETA .EQU       0H004E2        ; RMOVE AND RDRAW ROTATION ANGLE
COSINE
XNEW        .EQU       0H004EA        ; USER NEW X POSITION FPN
GRAPHTEMPX  .EQU       0H004F2        ; TEMPORARY GRAPHICS FLOATING
POINT X
YNEW        .EQU       0H004FA        ; USER NEW Y POSITION FPN
GRAPHTEMPY  .EQU       0H00502        ; TEMPORARY GRAPHICS FLOATING
POINT Y
SYSMASKCNTR .EQU       0H0050A        ; SYSTEM INTERRUPT MASK COUNTER
ANYINTERRUP .EQU       0H0050B        ; ANY PENDING ROM PACK
INTERRUPTS
PIATABLEBEG .EQU       0H0050C        ; 4051 PIA TABLE STARTING
ADDRESS
PIATABLEEND .EQU       0H0058A        ; 4051 PIA TABLE ENDING ADDRESS
AERASEDELAY .EQU       0H0058C        ; AUTO ERASE DELAY COUNTER
FULLACTION  .EQU       0H0058E        ; PAGE FULL ACTION INDICATOR
CRTLINESIZE .EQU       0H0058F        ; DISPLAY LINE LENGTH (OPT 1)
REREADCOUNT .EQU       0H00590        ; MAG TAPE REREAD ERROR COUNT
KEYCONTFLAG .EQU       0H00591        ; KEYBOARD CONTROL KEY FLAG
KEYLOCKFLAG .EQU       0H00592        ; KEYBOARD CAPS LOCK KEY FLAG
KEYSHIFTONE .EQU       0H00593        ; KEYBOARD DRIVER SHIFT KEY 1
KEYSHIFTTWO .EQU       0H00594        ; KEYBOARD DRIVER SHIFT KEY 2
KEYBOARDROW .EQU       0H00595        ; KEYBOARD ROW FLAG
KEYBOARDCOL .EQU       0H00596        ; KEYBOARD COLUMN FLAG
```

```
KEYSTACKCNT .EQU       0H00597        ; KEYBOARD DRIVER STACK COUNTER
KEYDELAYTIM .EQU       0H00598        ; CURRENT KEY DELAY TIME
KEYOLDDELAY .EQU       0H00599        ; OLDEST KEY DELAY COUNT
KEYSTKPOINT .EQU       0H0059A        ; KEYBOARD STACK POINTER
KEYDRVPOINT .EQU       0H0059C        ; KEYBOARD DRIVER PSEUDO STACK
KEYACCELRAT .EQU       0H005A4        ; KEYBOARD KEY ACCELERATION RATE
KEYMAXRATE  .EQU       0H005A5        ; KEYBOARD MAXIMUM RATE
KEYHANDTEMP .EQU       0H005A6        ; KEYBOARD HANDLER TEMPORARY KEY
NODIGITSEEN .EQU       0H005A7        ; NO INPUT DIGITS SEEN YET FLAG
FPISINTEGER .EQU       0H005A8        ; FP INPUT IS AN INTEGER FLAG
FLOATCONVE  .EQU       0H005A9        ; FLOATING POINT CONVERSION SAW
AN "E"
AFLOATING   .EQU       0H005AA        ; 8 BYTE FLOATING POINT BUFFER A
BFLOATING   .EQU       0H005B2        ; 8 BYTE FLOATING POINT BUFFER B
FLOATINGTAG .EQU       0H005B9        ; 8 BYTE FLOATING POINT BUFFER C
TAG
CFLOATING   .EQU       0H005BA        ; 8 BYTE FLOATING POINT BUFFER C
NOTZEROFUZZ .EQU       0H005C2        ; NOT ZERO COMPARE FUZZ FACTOR
ZEROCMPFUZZ .EQU       0H005C3        ; ZERO COMPARE FUZZ FACTOR
NORMALSHIFT .EQU       0H005CB        ; NORMALIZATION SHIFT COUNT
TRIGCONVERT .EQU       0H005CC        ; TRIG MODE CONVERSION FACTOR
POINTER
DETERMHOLD  .EQU       0H005CE        ; DETERMINATE HOLDING AREA
RANDKERNEL  .EQU       0H005D6        ; RANDOM NUMBER GENERATOR KERNEL
IMAGSTRBASE .EQU       0H005DE        ; IMAGE STRING BASE FOR PRINT
USING
DATABASEPNT .EQU       0H005E0        ; PRINT USING DATA BASE POINTER
ASCIIEXPSGN .EQU       0H005E2        ; ASCII EXPONENT SIGN
EXPTOPDIGIT .EQU       0H005E3        ; EXPONENT MOST SIGNIFICANT
DIGIT
EXPMIDDIGIT .EQU       0H005E4        ; EXPONENT MIDDLE SIGNIFICANT
DIGIT
EXPLOWDIGIT .EQU       0H005E5        ; EXPONENT LEAST SIGNIFICANT
DIGIT
ASCIINUMLSD .EQU       0H005E6        ; ASCII NUMBER LSD
ASCIINUMDIG .EQU       0H005F0        ; ASCII NUMBER DIGIT
ASCIINUMMSD .EQU       0H005F1        ; ASCII NUMBER MSD
ASCIINUMSGN .EQU       0H005F2        ; ASCII NUMBER SIGN
DIGDECPOINT .EQU       0H005F3        ; DIGITS DECIMAL POINT
LEFTDIGITS  .EQU       0H005F4        ; NUMBER OF LEFT DIGITS
RIGHTDIGITS .EQU       0H005F5        ; NUMBER OF RIGHT DIGITS
OUTDIGCOUNT .EQU       0H005F6        ; NUMBER OF OUTPUT DIGITS
ASLEFTDEC   .EQU       0H005F7        ; AS REQUIRED LEFT OF DECIMAL
POINT
ASRIGHTDEC  .EQU       0H005F8        ; AS REQUIRED RIGHT OF DECIMAL
POINT
CONEXPONENT .EQU       0H005F9        ; CONDENSED EXPONENT
BINEXPSIGN  .EQU       0H005FA        ; EXPONENT BINARY SIGN
PRINTFSPACE .EQU       0H005FB        ; PRINT FORMATTED LEADING SPACES
LEFTDIGCNTR .EQU       0H005FC        ; LEFT DIGIT COUNTER
```

```
LEFTDIGADDR .EQU       0H005FD        ; LEFT DIGIT ADDRESS
LCOMMACOUNT .EQU       0H005FF        ; COMMA COUNTER
LEFTTRAILER .EQU       0H00600        ; LEFT TRAILING ZERO COUNT
RIGHTLEADER .EQU       0H00601        ; RIGHT LEADING ZERO COUNT
RIGHTPRINT  .EQU       0H00602        ; RIGHT DIGITS PRINTING COUNT
RIGHTPOINT  .EQU       0H00603        ; RIGHT PRINT ADDRESS POINTER
RIGHTRAILER .EQU       0H00605        ; RIGHT TRAILING ZERO COUNT
COMMONZERO  .EQU       0H00606        ; COMMON ZERO COUNT
ZEROSAVEADD .EQU       0H00607        ; ZERO SAVE ADDRESS
RCOMMACOUNT .EQU       0H00609        ; COMMA REMAINDER COUNTER
DECIMALCONT .EQU       0H0060A        ; DECIMAL POINT CONTROL
TABCOUNTER  .EQU       0H0060B        ; CURRENT LOGICAL OUTPUT
POSITION


            .SBTTL     4051 SYSTEM ROUTINE LABELS

JUMP2APLUSX .EQU       0H0043E        ; SELF MODIFYING CODE, JUMP TO
THE ADDRESS AT [A+X]
;           .EQU       0H00445        ; SELF MODIFYING CODE, LDX FROM
ADDRESS [A,X]
;           .EQU       0H0044B        ; SELF MODIFYING CODE, LDAA FROM
ADDRESS [A,X]
;           .EQU       0H00451        ; SELF MODIFYING CODE, LDAB FROM
ADDRESS [A,X]
;           .EQU       0H00457        ; SELF MODIFYING CODE, STAB AT
ADDRESS [A,X]
;           .EQU       0H0045B        ; SELF MODIFYING CODE, JUMP TO
ADDRESS [A,X]
;           .EQU       0H006BE        ;
;           .EQU       0H006CA        ;
;           .EQU       0H006CD        ;
;           .EQU       0H006D0        ;
;           .EQU       0H006D3        ;
;           .EQU       0H006D6        ;
;           .EQU       0H006D9        ;
SWITCHBANK  .EQU       0H0A9D5        ; UPDATE THE EXTENDED ROM BANK
SWITCH NUMBER
PUSHRETURN  .EQU       0H0A9E7        ; PUSH RETURN ADDRESS ON
SECONDARY STACK
RETURNBASIC .EQU       0H0A9FD        ; RETURN BACK TO BASIC
ADD16INDEX  .EQU       0H0AA33        ; INCREMENT THE INDEX REGISTER
BY 16
ADD9INDEX   .EQU       0H0AA3A        ; INCREMENT THE INDEX REGISTER
BY 9
ADD8INDEX   .EQU       0H0AA3B        ; INCREMENT THE INDEX REGISTER
BY 8
ADD7INDEX   .EQU       0H0AA3C        ; INCREMENT THE INDEX REGISTER
BY 7
```

```
ADD6INDEX   .EQU      0H0AA3D       ; INCREMENT THE INDEX REGISTER
BY 6
ADD5INDEX   .EQU      0H0AA3E       ; INCREMENT THE INDEX REGISTER
BY 5
ADD4INDEX   .EQU      0H0AA3F       ; INCREMENT THE INDEX REGISTER
BY 4
SUB8INDEX   .EQU      0H0AA49       ; DECREMENT THE INDEX REGISTER
BY 8
ADDRDEVICE  .EQU      0H0AE22       ; ADDRESS I/O SYSTEM DEVICE
UNADDRESSIO .EQU      0H0AF1D       ; UNADDRESS I/O SYSTEM DEVICE
INTTOASCII  .EQU      0H0B004       ; CONVERT 16 BIT INTEGER TO
ASCII
FLOATASCII  .EQU      0H0B014       ; CONVERT FLOATING POINT TO
ASCII
ASCIIFLOAT  .EQU      0H0B05D       ; CONVERT ASCII TO FLOATING
POINT
;           .EQU      0H0B0C2       ;
SYMBOLTABLE .EQU      0H0B0D6       ; FIND OR CREATE A SYMBOL TABLE
ENTRY
FIXEDPOINT  .EQU      0H0B1A2       ; CONVERT FLOATING POINT TO
INTEGER
FLOATPOINT  .EQU      0H0B1FB       ; CONVERT INTEGER TO FLOATING
POINT
STRINGFLOAT .EQU      0H0B56F       ; CONVERT ASCII STRING TO
FLOATING POINT
PUSHFLOAT   .EQU      0H0B6EB       ; PUSH FLOATING POINT NUMBER
PULLFLOAT   .EQU      0H0B70F       ; PULL FLOATING POINT NUMBER
;           .EQU      0H0BC30       ;
POWERINGON  .EQU      0H0BC4B       ; 4051 POWERING ON ENTRY POINT
DECODERRING .EQU      0H0C4A9       ; DECODE THE SECRET PROGRAM TEXT
SCRAMBUFFER .EQU      0H0C4CD       ; SCRAMBLE THE OUTPUT BUFFER
TEXT
GETKEYBOARD .EQU      0H0C64A       ; GET A KEY FROM THE KEYBOARD
QUEUE
PUTKEYBOARD .EQU      0H0C69A       ; PUT A KEY INTO THE KEYBOARD
QUEUE
CRTRESET    .EQU      0H0CBBF       ; RESET THE 4051 CRT DISPLAY
;           .EQU      0H0CBEE       ;
;           .EQU      0H0CE77       ;
BACKUPSTACK .EQU      0H0D11D       ; BACK UP THE STACK ONE ENTRY
;           .EQU      0H0DBC8       ;
;           .EQU      0H0DCA8       ;
;           .EQU      0H0DD3F       ;
;           .EQU      0H0DD53       ;
MAGTAPESEND .EQU      0H0DD98       ; SEND A BUFFER TO THE MAG TAPE
MAGTAPEREAD .EQU      0H0DF35       ; GET A BUFFER FROM THE MAG TAPE
;           .EQU      0H0E00C       ;
KEYBOARDINP .EQU      0H0F163       ; THE KEYBOARD INPUT ROUTINE
PRINTCHAR   .EQU      0H0F22E       ; PRINT AN ASCII CHARACTER
;           .EQU      0H0F2A6       ;
```

```
;           .EQU        0H0F36F         ;
;           .EQU        0H0F7DC         ;


            .SBTTL      4051 ROM PACK HEADER
            .AREA       ROMPACKCODE (ABS)
            .ORG        0H08800

            .WORD       0H04051         ; BANK ID
            .WORD       POWERINGUP      ; POWER UP ENTRY
            .WORD       CLOSINGFILE     ; INITIALIZE ENTRY
            .WORD       DELETINGALL     ; DELETING ALL ENTRY
            .WORD       CLOSINGFILE     ; CLOSE THE FILE ENTRY
            .WORD       0H00000         ; SPECIAL FUNCTION ENTRY
            .BYTE       0H000           ; ROM PACK SPECIAL ID BYTE
            .ASCII      "CALL1 "        ; ROM PACK CALL NAME #1 (6
letters max, space filled)
            .WORD       CALL1ADDRESS
            .ASCII      "CALL2 "        ; ROM PACK CALL NAME #2
            .WORD       CALL2ADDRESS
            .ASCII      "CALL3 "        ; ROM PACK CALL NAME #3
            .WORD       CALL3ADDRESS
            .ASCII      "CALL4 "        ; ROM PACK CALL NAME #4
            .WORD       CALL4ADDRESS
            …
            .WORD       0H00000         ; CALL FUNCTION TABLE END
```