intel®

APPLICATION NOTE

AP-55A

August 1979

A High-Speed Emulator for Intel MCS-48™ Microcomputers

Applications Staff
Microcontroller Operation

9801007-01

I. PURPOSE AND SCOPE

This Application Note presents a description of the design and operation of a high-speed emulator for the Intel® MCS-48™ family of single chip microcomputers. The HSE-49™ emulator provides a simple and inexpensive means for executing and debugging 8049 programs which require the full 11-MHz operating speed of the part.

Section II of this Application Note describes some of the features of this development tool and how it may be used. Section III briefly discusses the hardware used to implement these features, while Section IV describes the manner in which program execution status is made available to the operator.

A detailed description of all of the operator commands is presented in Section V of this note, along with the modifiers and options which may be specified for each command. Known restrictions and limitations of the HSE-49 system are listed and explained in Section VI. Section VII shows how the basic circuit may be modified to provide options on memory organization, I/O configurations, etc.

Full schematics of the system hardware, as well as monitor software listings, are presented in Appendices A and B, respectively. A short summary of the command syntax is presented in Appendix C,. Appendix D explains the error message codes which may appear during use.

It is assumed that the reader is already familiar with the operation of the 8048 or 8049 microcomputers. Some knowledge of the 8048 architecture is needed to understand sections of the command and modifier descriptions. Most users will already have this background. Other readers are referred to the *MCS-48 Microcomputer User's Manual,* Intel publication number 9800270.

II. THE HSE-49 DEVELOPMENT TOOL

In essence, the HSE-49 emulator provides the user a means for executing an MCS-48 program located in external RAM rather than internal ROM or EPROM. This allows programs being debugged to be modified easily and quickly during the debug cycle. A user's program may be entered into system RAM either manually or via a serial link from a host computer such as an Intellec® Microcomputer Development System. Once loaded, the program can be modified using an on-board keyboard and display, and executed in real-time in a number of breakpoint modes. The internal state of the processor, including RAM, accumulator, timer/counter, and status register contents, can also be read and modified through the keyboard.

Breakpoint and debug facilities are extremely flexible. The following execution modes are provided.

- Programs may be run in full (11 MHz) real time;
- Programs may be single-stepped;
- In break mode, programs run in full real time until break occurs;
- Breaks may be triggered by either program or external data RAM accesses;
- Any number of breakpoints may be used in any combination;
- "Auto-Step" operation causes the current program counter and Accumulator contents to be printed on the display for a short time on every instruction cycle;
- "Auto-Break" provides the above display only when a break flag is encountered, with real time operation otherwise;
- While running in non-break mode, a TTL-level pulse is generated whenever a break flag is encountered. This signal may be used to trigger an oscilloscope or Logic Analyzer to assist in hardware and software debug.
- While running in any mode, the keyboard and display are "alive". Execution may be suspended or terminated by commands from the keyboard.

Intent of this Note

While the HSE-49 emulator can assist a new microcomputer user in becoming familiar with the 8048 and 8049 microcomputers, its inherent debug capabilities will also prove helpful to design engineers. The design could be used for new system development and verification or adapted for prototype production.

The main concern in designing the HSE-49 emulator was to keep the basic design simple, while maximizing the system's flexibility. The design allows the use of jumpers, hardware and software switches, etc. to allow the user to reconfigure the system according to the way he dedicates chip-select pins, I/O, etc. The emulator can be changed to fit each user's unique needs, rather than forcing the user to alter his needs to what is provided.

The primary intent of note is to provide the reader with the information needed to reconstruct and make full use of the HSE-49 emulator. Less emphasis is placed on describing how the hardware operates or how the commands are implemented. This information may be found in the schematic diagrams and software listings included in the Appendices.

III. GENERAL HARDWARE OVERVIEW

User Program Emulation

The actual emulation of the user's program is done using an 8039 microcomputer (IC29 on the schematics in Appendix A) executing a program stored in external RAM. The basic minimum configuration includes the 8039 microcomputer, an 8282 address latch (IC19), and 2K bytes of 2114 RAM to use for program development and real-time execution (ICs B1, C1, B2, and C2). Additional RAM may be added to allow the user to expand his program and data memory to 4K each. (If an 11-MHz crystal is used with the microcomputer, type 2114-3 RAMs must be used.)

System Supervision

A second microcomputer — another 8039 (IC25) with an 8282 address latch (IC16) and off-chip program memory in a 2716 EPROM (IC15) — is used to scan the on-board keyboard and display, interpret and implement commands, drive serial interfaces, etc. In general, the master processor is used to interface the execution processor's memory spaces with the outside world and control the operation of the execution processor. In this note the two processors will be abbreviated "MP" and "EP", respectively. Figure 1 shows how the two processors interrelate with the rest of the system.
system.

Keyboard/Display

The 33-key keyboard shown in Figure 2 includes a 16-key hexidecimal keypad and 17 special function keys for specifying commands and modifiers. Readers already familiar with the PROMPT-48™ debug tool for the 8048 will find that 25 of the HSE-49 emulator keys are identical in function and layout to the PROMPT-48 keyboard, and use the PROMPT-48 command syntax. The eight additional keys are used to generalize and augment the PROMPT-48 capabilities, as described in Section V.

The eight-character seven-segment display (DS1-DS8) is used for displaying addresses, data, and pseudo-alphanumeric messages. The display responses printed in Section V and throughout this note use a mix of upper and lower case letters to indicate what seven-segment patterns appear. An 8243 (IC9) and eight DIP packages (resistor packs, current buffers, etc.) are used for multiplexing the display and scanning the keyboard.

Breakpoint Detection

Breakpoints are specified and detected using a 2102A 1K x 8 RAM corresponding to each pair of 2114s (ICs A1

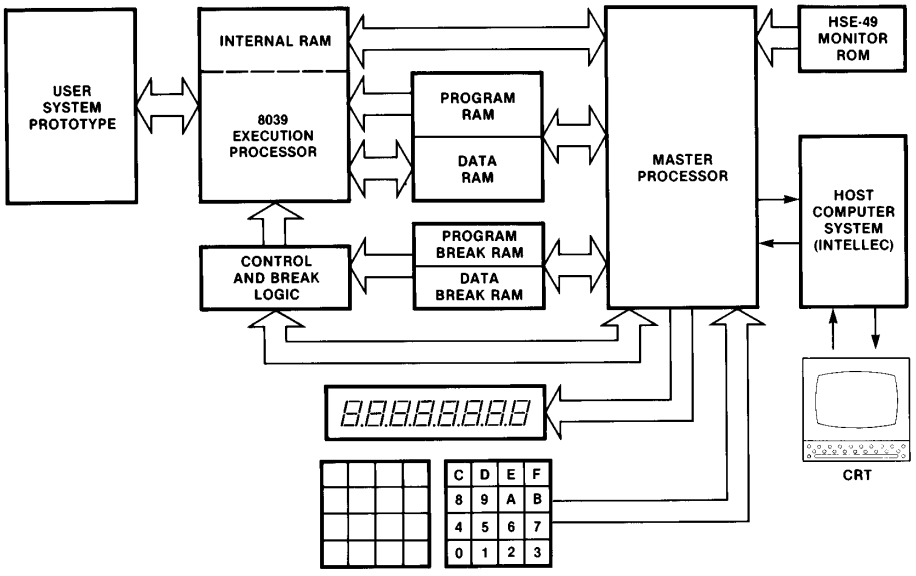


Figure 1. HSE-49™ Emulator Signal Flow Diagram

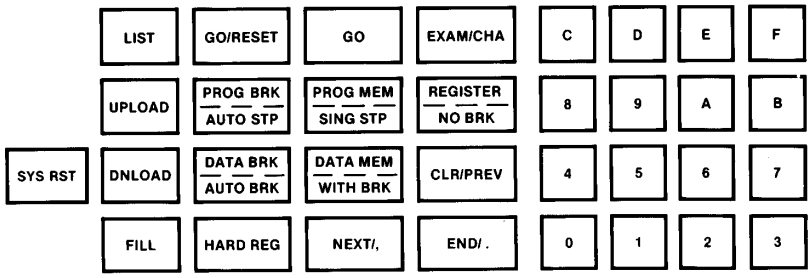


Figure 2. HSE-49™ Emulator Command Keyboard Organization

and A2). In effect, each program or data address accesses a 9-bit word. Eight bits are used normally for code or data storage. The ninth bit, accessed in parallel with the other eight, is used to indicate if a breakpoint has been set for that address. This output, when asserted, is latched (IC27 and IC36) and used to halt the execution processor via the single-step input. (In other modes, the break logic can be reconfigured to set the break requested flip-flop on any EP machine cycle or any EP "MOVX" instruction.)

Link Register

An 8212 8-bit latch (IC18) is used to communicate data and commands between the master and control processors. Under control of the MP, this register, called the "Link" register, may be logically mapped into either the program or data RAM address spaces. When this is done, the 2114s in the respective memory space are disabled and the link responds to all accesses, regardless of address. The link will be discussed in greater detail in Section IV.

Control Logic

In addition to the devices mentioned above, the minimum configuration requires about 10 additional ICs for bus arbitration, system control, and breakpoint and single-step logic. Additional parts may be optionally added for serial port interfacing, I/O reconstruction, etc.

MP Monitor

The monitor program executed by the MP includes commands for filling, reading, or writing the various memory spaces, including the execution processor's program RAM, external ("MOVX") data RAM, accumulator, PSW, PC, timer/counter, working registers, and internal RAM; to execute the user's program from arbitrary addresses in various debugging modes; and to upload or download object or data files from diskettes using an Intellec® development system. No special software is needed for the Intellec® other than ISIS Version 3.4 or later. The data format is compatible with the standard Intel hex file format produced by ASM-4; the baud rate may be altered from 110 baud (default state) up to 2400 baud from the on-board keybad. Blocks of data may be transmitted to a CRT or printer and displayed in a tabular format.

IV. INTERPROCESSOR COMMUNICATION

Program Break Sequence

When the MP detects that the EP has been halted by the breakpoint hardware, or when the operator presses a key while the program is executing, the program break sequence is initiated. The low-order 23 bytes of user program memory is read into a buffer within the internal RAM of the MP. A short program for reading and transmitting internal EP status is written over the low-order program memory. (This is one of several "mini-monitors" overlayed over the user program area.) The link register is mapped logically over the user program memory, and loaded with the 8049 machine code for a "CALL" instruction to the mini-monitor program area. The EP is then allowed to fetch a single instruction from the link, i.e., the "CALL" to the mini-monitor is forced onto the EP data bus.

From this point on, the EP executes code contained in the mini-monitor. The link is logically mapped over the data RAM address space (whether or not any 2114 data RAMs are present). A block diagram of the system at this point is shown in Figure 3. The break logic is reconfigured so that any "MOVX" (RD or WR) operation executed by the EP will cause it to halt.

For example, after entering the first mini-monitor, the EP executes a "MOVX @R0,A" instruction. This writes the contents of the accumulator prior to the execution termination into the link, and causes the EP to halt. The MP may then read and retain the link contents to determine the EP accumulator value. The EP timer/counter and PSW are preserved in the same manner.

Accessing EP Internal RAM

After reading and saving EP internal status, the MP loads a different mini-monitor into the same RAM area. This monitor allows the internal RAM of the EP to be read and written by the MP by passing address and data

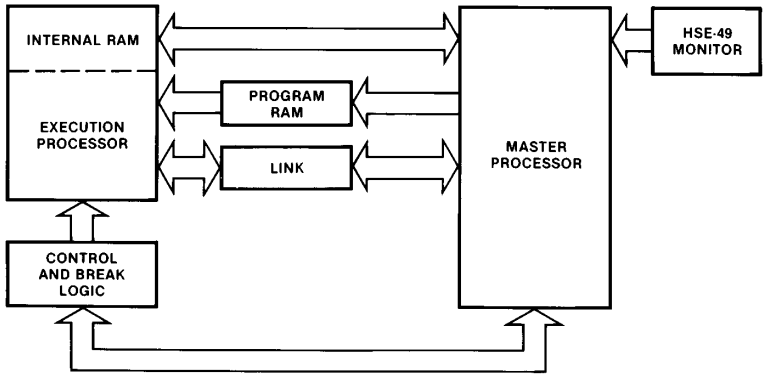


Figure 3. Communication between EP & MP

values between the two processors using the link register.

This is needed for two reasons. First, the EP program counter prior to the forced "CALL" instruction may be derived from the EP stack contents, and may be modified to cause the EP to resume execution at any desired address. Secondly, the internal RAM of the EP may then be accessed and modified in the process of executing a number of the monitor commands.

Resuming User Program Execution

In order to resume user program execution, a status-restoration mini-monitor is overlayed. This restores the EP internal status using a scheme analogous to the one in which the status was originally saved. The final step of the last mini-monitor is an "RETR" instruction, after which the EP is again halted. The low-order program memory saved earlier is rewritten into the appropriate area, the break logic is reconfigured for the desired execution mode, and the EP is released to run at full speed until the next break situation is encountered.

Note that all commands are implemented using "logical" rather than "physical" addressing. Thus the operator need not be concerned with the intricacies of the system design. For example, when any monitor command refers to low-order user program memory, the appropriate byte of storage within the MP internal RAM is accessed instead. If the location is altered, the internal RAM is modified appropriately. When program memory is reloaded prior to resuming user program execution, the modified version of the user program will be the one loaded.

Baud	HR06	HR07
110	93H	04H
150	96H	03H
300	45H	02H
600	9DH	01H
1200	44H	01H
2400	1AH	01H

Table 1. Serial Interface Data Rate Parameters

V. HSE-49 COMMAND DESCRIPTION

Whenever the characters "HSE-49" are present on the system display, a command string may be entered by the operator. In general, all command strings consist of a basic command initiator, an optional command modifier or type-designator, and a number of parameters or delimiters entered as hexidecimal digits. A command is executed, or a command in progress terminated, by pressing the [END/.] key. Logical default values are assumed for the modifier and parameters if either (or both) are omitted. A defualt parameter assumed for the command modifier will be presented on the display when the first parameter is entered.

Each parameter is a string of up to three hexidecimal digits. If more than three digits are entered, only the most recent three are considered. This allows an erroneous digit to be corrected without respecifying the entire command. A parameter is completed by pressing the [NEXT/,] key. Some commands may only need the low order part of a parameter; i.e., a command incorporating a data byte (such as [FILL]) will use only the low-order 8 bits of the corresponding parameter; Internal RAM and hardware register addressing uses only seven. In each case, higher order bits are ignored.

A command string is terminated and the command invoked by pressing the [END/.] key. The command will also be invoked by pressing the [NEXT/,] key when no additional parameters are allowed. A command string may be aborted at any point before the command is invoked by pressing the [CLEAR/PREV] key, and the sign-on message will appear.

Errors

An illegal command string, command terminator, or hardware failure will cause an error message and error code number to appear on the display (e.g., "Error-.3"). When this occurs, the monitor can be returned to command mode by pressing the [CLEAR] or [END/.] keys. An explanation of the various error codes is given in Appendix D.

Command Classes

Commands for the HSE-49 emulator are divided into general classes, where all commands in each class have the same choice of options or modifiers. A brief description of each command, followed by a description of the allowed options, is presented below by class.

Data Manipulation/Control Command Group

Commands:

[EXAM/CHA]

Display Response — "ECh."

Function — Examine/change memory location.

Causes the memory address specified to be read and presented on the display. New data may be entered (if desired) from the hexidecimal keypad. New data is verified before appearing on the display. Subsequent or previous locations may be read by pressing the [NEXT/,] or [PREV] keys, respectively. Command terminated with [END/.] key.

[FILL]

Display Response — "FIL."

Function — Fill range of memory addresses with a single data value.

Fill the appropriate memory space between the addresses specified by the first two parameters with the low-order byte of the third parameter. If second parameter less than first, only the location specified by the first is affected. If third parameter omitted, zero is assumed. If second and third parameters omitted, individual address specified is cleared. Command is useful for setting a large range of breakpoints; e.g., all of page 3 may be enabled for break with the command:

[FILL][PROG BRK]<300>[,]<3FF>[,]<1>[.]

[LIST]

Display Response — "LSt."

Function — List memory to output device through HSE-49 serial port.

Display the contents of a range of addresses given by two parameters to a teletype or CRT screen. Data is formatted, 16 separated bytes per line, with the starting address of each line printed. If used with an Intellec® system, the operator first uses ISIS-II to transfer the TTY input to the CRT output ("COPY :TI: TO :CO:") then invokes this command from the keypad. Alternatively, any ISIS device or disk file name(:TO:, :LP:, :F1:HRDREG.SAV, etc.) may be used as the destination.

[DNLOAD]

Display Response — "dnL."

Function — Download memory through HSE-49 serial port

Load data in hex file format through the serial input port. If used with Intellec® system, the operator first invokes this command from the keypad, then uses ISIS-II to transfer a disk file to the teletype port ("COPY : Fn:file.HEX TO :TO:").

The use of the checksum field for the download command is expanded slightly over the Intel hex file format standard. If the first character of the checksum field is a question mark ("?"), the checksum for that record will not be verified. This allows large object files produced by the assembler to be patched using the ISIS text editor without the necessity of manually recomputing the checksum value.

[UPLOAD]

Display Response — "UPL."

Function — Upload memory through HSE-49 serial port.

Output the contents of a range of addresses specified by the two parameters through the HSE-49 serial port in standard Intel hex file format. If used with Intellec® system, the operator first uses ISIS-II to transfer the TTY input to a disk file ("COPY :TI: TO :Fn:file.HEX"), then invokes this command from the keypad.

Data types allowed:

[PROG MEM]

Display Response — "Pr."

Function — User program memory.

Memory used to develop and execute user program. Addresses 000 through 7FF are the execution processor's memory bank 0; 800 through FFF are memory bank 1.

[REGISTER]

Display Response — "rG."

Function — Register memory and RAM.

Internal RAM of execution processor. Locations 0-7 are working register bank 0; 18-1F are working register bank 1. Only the low-order 7 bits of an address are considered.

[DATA MEM]

Display Response — "dA."

Function — External data memory (if installed).

Memory accessed by execution processor "MOVX A,@Rr" or "MOVX @Rr,A" instructions. High-order 4 bits may or may not be relevant, depending on jumpering option selected (explained in Section VII of this note).

[HARD REG]

Display Response — "Hr."

Function — Hardware registers.

The execution processor (EP) hardware registers (accumulator, timer/counter, etc.), as well as several parameters for controlling HSE-49 system status, are accessible through this catch-all memory space. Addresses are as follows:

00 — EP accumulator.

01 — EP PSW.
Bits correspond to 8049 PSW except that bit 3 (unused in the 8049) is used to monitor and alter the state of F1. Bits 2-0 correspond to the stack pointer value after the EP executes a CALL to the mini-monitor; i.e., one greater than when EP was running the user's program.

02 — EP timer/counter.

03 — EP internal RAM location 00.
(This value is also accessible through [REGISTER] space.)

04 — EP program counter (low byte).

05 — EP program counter (high nibble).

06-07 — HSE-49 serial interface baud rate parameters. Defaults to 110 baud; other rates may be selected by loading the values listed in Table 1.

08 — HSE-49 automatic sequencing rate parameter. Used in [GO][AUTO STP] and [GO][AUTO BRK] execution commands. 00 → fastest; FF → slowest. Defaults to 20H; approximately two steps per second.

09 — Monitor version/release number (packed BCD).

0A-0F — Currently unused by the monitor program.

10-7F — Variables used by master processor (MP) monitor. Should not be altered by operator.

[PROG BRK]

Display Response — "Pb."

Function — User program breakpoint memory.

Memory space used to indicate points where program execution should halt when running in a mode with breakpoints enabled ([GO][W/ BRK] and [GO][AUTOBRK]). Break will occur if enabled byte is read as the first or last byte of a 2-byte instruction, or read in executing a MOVP, MOVP3, or JMPP instruction. Memory is only 1 bit per location; 00 indicates continue, 01 causes a halt. Addresses 000 through 7FF are the execution processor's memory bank 0; 800 through FFF are memory bank 1.

[DATA BRK]

Display Response — "db."

Function — External data RAM breakpoint memory.

Memory space used to indicate points where data accesses should halt when running in a mode with breakpoints enabled ([GO][W/ BRK] and [GO][AUTOBRK]). Memory is only 1 bit per location; 00 indicates continue, 01 causes a halt. High-order 4 bits of breakpoint address may or may not be relevant, dependent on jumpering option selected for the corresponding data RAM (explained in Section VII of this note).

User Program Execution Control Group

Commands:

[GO]

Display Response — "Go."

Function — Begin execution.

If a parameter is given as part of the command string, execution will begin at that address. Otherwise, the EP program counter (hardware registers 04 and 05) will be used. These will contain the program counter from an earlier program execution break unless they have since been explicitly modified by the operator.

If command is terminated by [END/.], the EP's F1, PSW and stack pointer will be cleared. If command string is terminated by [NEXT/,], PSW will be taken from the EP PSW contents (hardware register 01).

While running the user's program, the characters "-run-." are written on the display. Execution may be halted and another command initiated by pressing the appropriate command key. Execution may be suspended at any time in any mode by pressing the [END/.] key. This will cause the current value of the execution processor program counter and accumulator to appear on the display in the form "PC.234-56". System status is saved in the appropriate hardware registers. At this point, or when an enabled breakpoint is encountered, pressing the [NEXT/,] key will cause the program to continue in the same mode as before. Any other command may be invoked by pressing the appropriate command string.

[GO/RESET]

Display Response — "Gr."

Function — Go from reset state.

EP is hardware-reset and released to execute the user's program from location 000H. No parameters are allowed. F0, F1, PSW, stack printer, memory bank flip-flop, etc., are cleared.

Note that this command does not require the use of mini-monitors to initiate program execution. As the last phase of the program development cycle, the 2114 program RAMs and address decoder may be removed and replaced by a ROM or EPROM part (not shown in schematics). This command may be used to start execution when the program RAM has been removed. No interrogation of EP status or internal RAM may be done, nor are break or single-step modes allowed in this case, though the 2102A breakpoint RAM outputs may still be used to trigger a logic analyzer.

Execution modes allowed:

[NO BRK]

Display Response — "nb."

Function — Without breakpoints.

Full-speed execution without breakpoints enabled. Does not affect the state of the breakpoint memories.

[SING STP]

Display Response — "SSt."

Function — Single Step.

Step through program one instruction at a time. After each instruction is executed, execution halts with the current value of the Execution Processor Program Counter and Accumulator appearing on the display in the form "PC.234-56". System status is saved in the appropriate Hardware Registers. At the point, [NEXT/,] will cause the program to execute one more instruction, or any other command may be invoked by pressing the appropriate command string. Does not affect the state of the Breakpoint Memories.

[W/ BRK]

Display Response — "br."

Function — With breakpoints.

Full-speed execution with breakpoints enabled. When a breakpoint is encountered, execution halts with the current value of the execution processor program counter and accumulator appearing on the display in the form "PC.234-56". System status is saved in the appropriate hardware registers. At this point, [NEXT./,] will cause the program to continue until the next breakpoint is reached, or any other command may be invoked by pressing the appropriate command string.

[AUTO STP]

Display Response — "ASt."

Function — Automatically sequence through a series of instructions.

Step through program one instruction at a time. After each instruction is executed, execution halts with the current value of the execution processor program counter and accumulator appearing on the display in the form "PC.234-56". System status is saved in the appropriate hardware registers. Execution resumes after a time determined by contents of hardware register 08. Does not affect the state of the breakpoint memories.

[AUTO BRK]

Display Response — "Abr."

Function — Automatically sequence between breakpoints.

Execute a series of instructions in real time between breakpoints. When breakpoint is encountered, halt EP temporarily while program counter and accumulator contents are displayed, then continue. Display is sustained after execution resumes. Does not affect the state of the breakpoint memories.

Breakpoint Control Command Group

Commands:

[B]

Display Response — "Stb."

Function — Breakpoint set.

Set breakpoint for the address given. Multiple breakpoints may be set by entering additional addresses, separated by the [NEXT/,] key. Command terminated by pressing [END/.]. Action taken is to fill the appropriate breakpoint memory locations with logical ones.

[C]

Display Response — "CLb."

Function — Clear breakpoint.

Clear breakpoint for the address given. Multiple breakpoints may be cleared by entering additional addresses, separated by the [NEXT/,] key. Command terminated by pressing [END/.]. Action taken is to fill the appropriate breakpoint memory locations with logical zeroes.

Data types allowed:

[PROG MEM]

Display Response — "Pr."

Function — Break on program memory fetch.

Applies command to the program breakpoint memory space.

[DATA MEM]

Display Response — "dA."

Function — Break on data memory access.

Applies command to the external data breakpoint memory space.

System Control Command Group

Command:

[SYS RST]

Display Response — "HSE-49."

Function — System reset.

Reset both the MP and EP and clear all breakpoints (requires approximately one second). CAUTION — If reset while EP is executing the user's program, the low order section of program memory (about 23 bytes) will be altered.

VI. SYSTEM LIMITATIONS

In designing the HSE-49 emulator, certain compromises were made in an attempt to maximize the usefulness of the emulator while keeping the circuitry simple and inexpensive. As a result, the following limitations exist and must be taken into account when using the system.

1. As explained in Section IV, user program execution is terminated (by single-stepping, breakpoints, pressing the [END/.] key, etc.) by forcing the execution processor to execute a "CALL" instruction to the mini-monitor. This uses one level of the EP subroutine stack. The EP PSW reflects the value of the stack pointer *after* processing this CALL. As a result, the value indicated for stack depth by examining the EP PSW (hardware register 01) is one greater than the depth when the break was initiated. The user program must not be using all eight levels of stack when a break is initiated or the bottom level will be destroyed.
2. User program is initiated (by the [GO] command or when resuming execution after a breakpoint, single-stepping, etc.) by forcing the EP to execute an "RETR" instruction. This will clear the EP interrupt-in-progress flip-flop. If the user program allows both external and timer interrupts to be enabled at the same time, care must be taken to avoid causing a break while the EP is within an interrupt servicing routine. No limitation is placed on breakpoints or single-stepping in the background program because of this.
3. When the user program execution is terminated (by a break, single-stepping, etc.) and later resumed, the EP timer/counter is restored to its value when the break occurred (unless modified by the user). The prescaler, however, will have changed. Thus, up to 31 machine cycles may be "lost" or "gained" if a break occurs while the timer is running.
4. Timer interrupts occurring at the same time as an EP break may be ignored if the timer overflow occurs after breaking user program execution before the timer value is saved.
5. The 8049 "RET" and "RETR" instructions are each 1-byte, 2-cycle instructions. During the second cycle the byte following the return instruction is fetched and ignored. If a program breakpoint is set for a location following a "RET" or "RETR" instruction, a break will be initiated when the return is executed.

6. Breakpoints should not be placed in the last 3 bytes of an EP memory bank (locations 7FDH-7FFH and 0FFDH-0FFFH). User program should not be single-stepped or auto-stepped through these locations.
7. Since I/O configuration is determined by external hardware rather than software, I/O modes may not be altered while a program is executing. (See Section VII for further details.)
8. The "ANL BUS,#nn" and "ORL BUS,#nn" instructions may not be used in the user program, as external hardware cannot properly restore these functions.
9. The memory bank select flag is not affected by the user program break sequence. Upon resuming execution with the [GO] command this flag will remain in the same state as before the preceding break. The flag may be cleared only by executing the [GO/RESET] or [SYS RST] commands.

VII. HARDWARE CONFIGURATIONS

A number of control and status lines are available to the user. All are low-power Schottky TTL-compatible signals.

TP1 — Unused MP input.

TP2 — Unused MP output.

TP3 — User program suspended. Low when EP running user code. High when halted or running mini-monitors.

TP4 — Breakpoint encountered. Normally low. High-level pulse generated when breakpoint passed. Useful for triggering logic analyzers, oscilloscopes, etc.

TP5 & TP6 — Memory matrix mode control. Select program vs. data RAM, link mapping configuration, etc. (See Appendix B for details.)

TP7 — Bus control. Low when MP controls common memory buses. High when EP controls memory buses.

The HSE-49 emulator hardware is designed to allow the user to reconfigure the system for a wide variety of different applications by installing or removing jumper wires or additional components. The schematics in Appendix A show the components needed for a variety of different configurations. In general, not all of the devices are required (or allowed) for any one configuration. The devices which are required are included in the following description.

The types of options allowed are divided below into several general classes and subdivided into mutually-independent features. Within some of these features there are numbered, mutually exclusive configurations; i.e., the serial interface (if desired) may use either current-loop or RS-232C current buffers, but not both at one time.

Standard Operating Configuration

(Minimum system configurations — up to 4K program RAM; no data RAM; no serial interfaces; no execution processor I/O reconstruction.)

A. Basic 2K monitor from Appendix B:

Install resistors R4-R6
Install transistor Q1
Install crystals Y1-Y2
Install capacitors C5-C38
Install switches S1-S33
Install displays DS1-DS8
Install IC1-IC2
Install RP3-RP5
Install IC6-IC7
Install RP8
Install IC9
Install IC15-IC20
Install IC25-IC30
Install IC34
Install IC36-IC38
Install A1-A2
Install B1-B2
Install C1-C3
Install jumpers 13-15
Install jumpers 17-18
Install jumper 20

B. Expansion 2K monitor:

Install IC14
Remove jumper 17

Serial Interface Buffer Selection

A. Current loop serial interfaces (4N46s) installed for use with full Intellec® Model 800 development system TTY port.

Install IC21-IC22
Install resistor R1-R3
Install jumpers 4-9
(Remove RS-232 jumpers)

B. RS-232C serial interfaces (MC1488 and MC1489) installed for use with CRT as output device for data dumps:

Install IC23-IC24
Install jumpers 1-3
Install jumpers 10-11
(Remove current-loop jumpers)

External Data RAM Address Decoding Scheme for Execution Processor

A. Up to 16 pages of on-board external data RAM installed for execution processor (addresses 0 through

0FFFH = 4K bytes); port 2 used for addressing pages 0 through 15:

Install jumpers 21-25
Install jumper 27
Install A5-A8
Install B5-B8
Install C5-C8

B. One page of on-board external data RAM installed for execution processor (addresses 0 through 0FFH); port 2 not used for data addressing:

Install jumper 26
Install jumper 28
Install A5
Install B5
Install C5

Connect the outputs of IC20, pins 7, 9, 10, & 11 to the inputs of a 74LS21 AND gate (not shown). Connect the output to CE and CS inputs of A5-C5. (Note: these signals are all present at jumpers 21-24 on the schematics.)

Reconstructing I/O for Execution Processor

A. Application of port 2, pins P23-P20:

(1) Using P23-P20 for latched output data (used with "OUTL P2,A", "ANL P2,#data", and "ORL P2,#data" instructions):

Install IC31

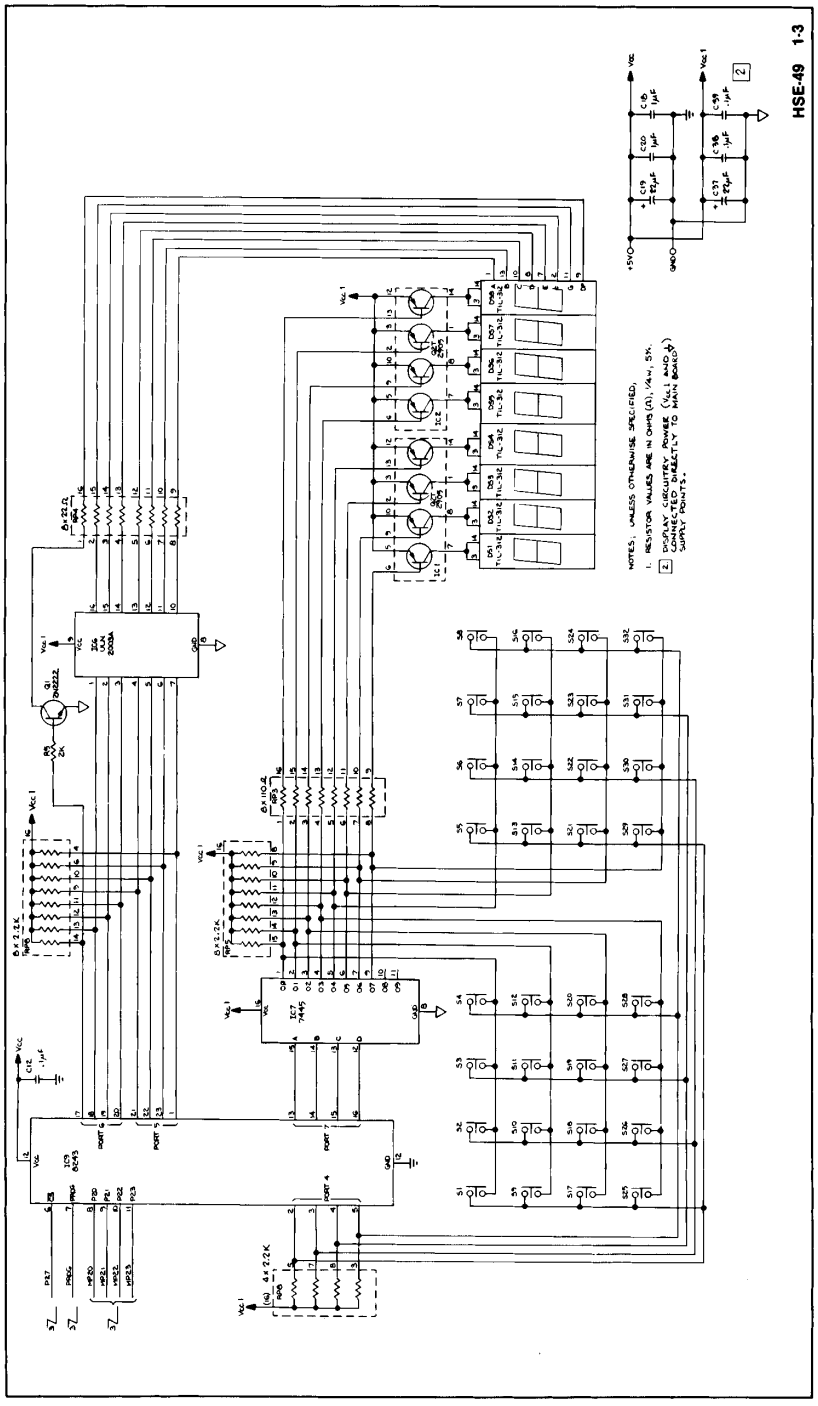
(2) Using P23-P20 for interfacing to an 8243 in user's prototype:

Connect D3-D0 pins on IC31 socket to corresponding Q3-Q0 pins.

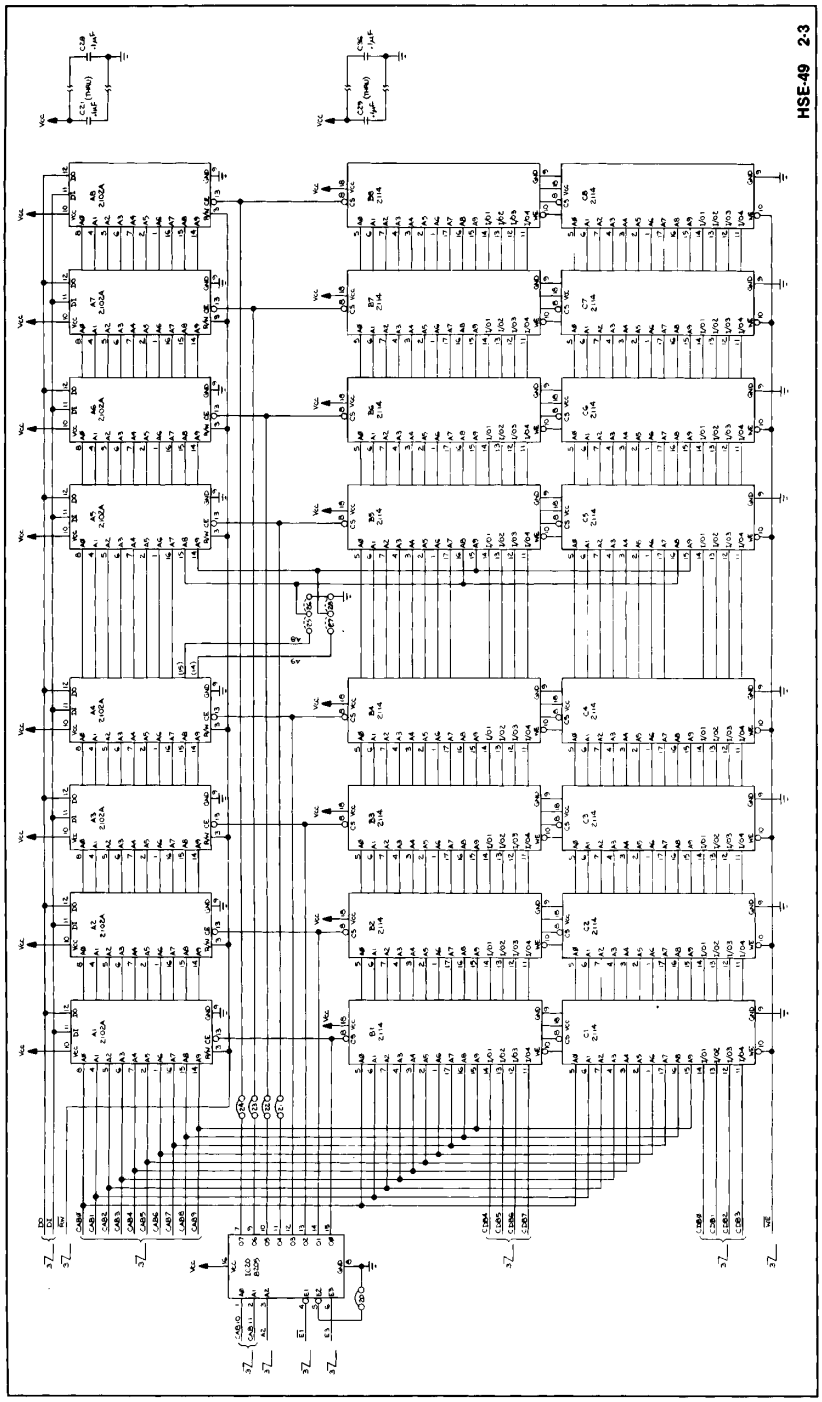
B. Application of execution processor BUS:

(1) Use of BUS as latched output port ("OUTL BUS,A"):

Install IC32

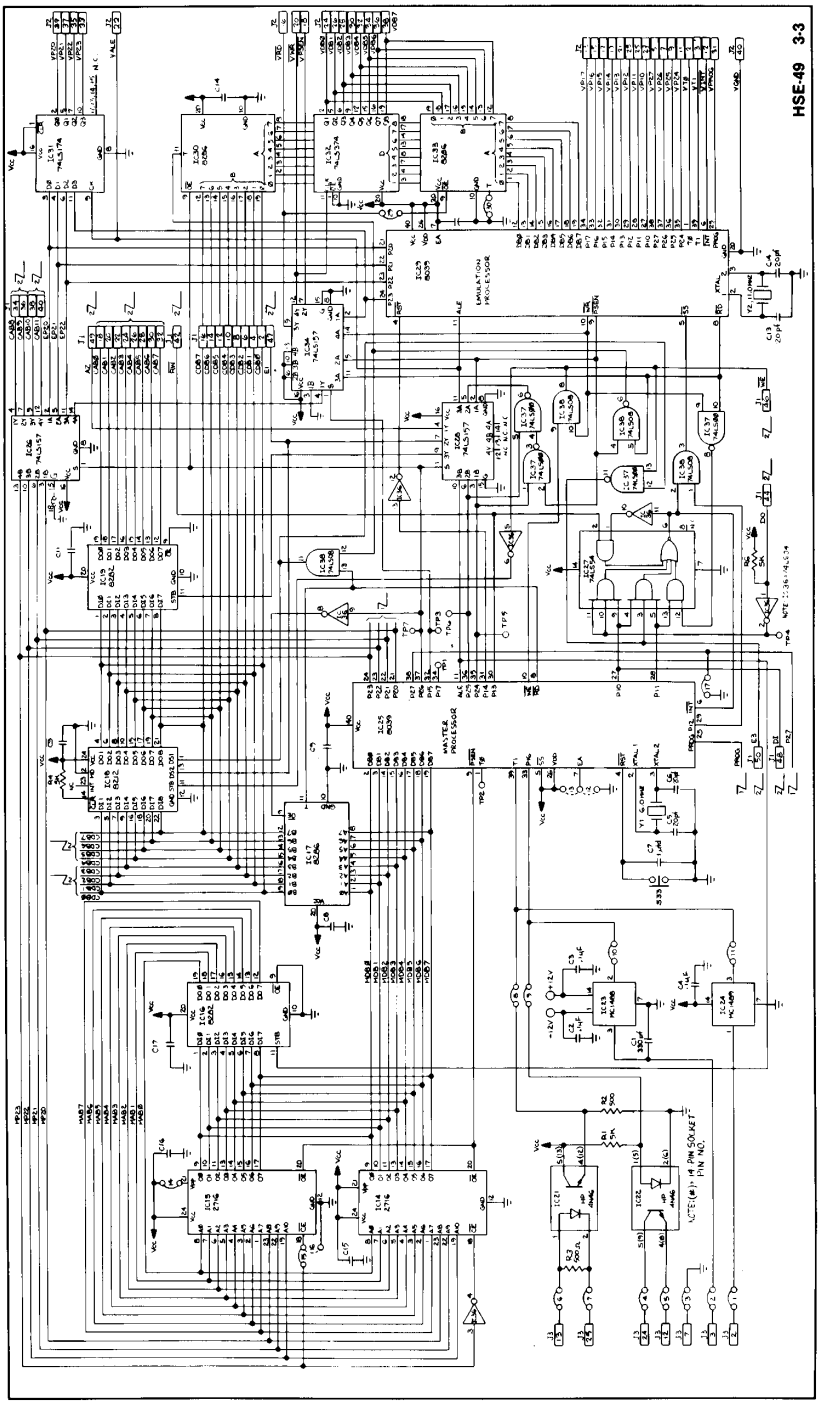


Key Board Display



HSE-49 2-3

Ram Memory



Central Processor

```
ASM49 HSE49 LNK PRINT( LF )


ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V3.0            PAGE    1
HSE-49(TM) EMULATOR MONITOR VERSION 2.5

 LOC  OBJ        LINE       SOURCE STATEMENT

                    1 $MACROFILE NOGEN NOCOND XREF
                    2 $TITLE('HSE-49(TM) EMULATOR MONITOR VERSION 2.5')
                    3 ;
                    4 ;*******************************************************************************
                    5 ;
                    6 ;                       PROGRAM: HSE-49(TM) EMULATOR MONITOR
                    7 ;                                VERS 2.5/709
                    8 ;
                    9 ;                       COPYRIGHT (C) 1979
                   10 ;                       INTEL CORPORATION
                   11 ;                       3065 BOWERS AVENUE
                   12 ;                       SANTA CLARA, CALIFORNIA 95051
                   13 ;
                   14 ;*******************************************************************************
                   15 ;
                   16 ;  ABSTRACT
                   17 ;  ========
                   18 ;
                   19 ; THIS PROGRAM CONTAINS THE SOFTWARE NECESSARY TO RUN THE HSE-49(TM)
                   20 ; HIGH-SPEED EMULATOR FOR INTEL'S MCS-48(TM) FAMILY FAMILY OF MICROCOMPUTERS.
                   21 ; THE EMULATOR PROVIDES AN ASSORTMENT OF UTILITY FUNCTIONS FOR
                   22 ; DEVELOPING AND DEBUGGING 8049-BASED APPLICATIONS, INCLUDING THE
                   23 ; ABILITY TO ENTER AND MODIFY PROGRAMS IN PROGRAM RAM,
                   24 ; ALTER DATA, SINGLE-STEP SECTIONS OF A PROGRAM, AND EXECUTE PROGRAMS
                   25 ; AT SPEEDS OF UP TO 11 MHZ. WITH OR WITHOUT BREAKPOINTS ENABLED.
                   26 ; THE EMULATOR IS DESCRIBED IN GREATER DEPTH IN INTEL'S APPLICATION NOTE
                   27 ; AP-55  "A HIGH-SPEED EMULATOR FOR INTEL MCS-48(TM) MICROCOMPUTERS."
                   28 ;
                   29 ;  PROGRAM ORGANIZATION
                   30 ;  ======= ============
                   31 ;
                   32 ; THIS LISTING IS ORGANIZED AS FOLLOWS:
                   33 ;
                   34 ;       INTRODUCTION AND HARDWARE OVERVIEW;
                   35 ;       VARIABLE DECLARATION AND DEFINITION;
                   36 ;       POWER-ON SYSTEM INITIALIZATION;
                   37 ;       KEYBOARD COMMAND PARSER AND ASSOCIATED TABLES;
                   38 ;       IMPLEMENTATIONS OF THE PRIMARY COMMANDS;
                   39 ;       DATA ACCESSING UTILITY SUBROUTINES USED THROUGHOUT;
                   40 ;       KEYBOARD SCANNING AND DISPLAY DRIVING SUBROUTINE;
                   41 ;       KEYBOARD AND DISPLAY INTERFACING UTILITIES;
                   42 ;       ROUTINES AND UTILITY SUBROUTINES WHICH INTERACT BETWEEN MP AND EP.
                   43 ;
                   44 ;
                   45 $EJECT
```

```
LOC  OBJ        LINE      SOURCE STATEMENT

                   46 ;
                   47 ;  INTRODUCTION AND HARDWARE OVERVIEW
                   48 ;  ============ === ======== ========
                   49 ;
                   50 ; THE EMULATOR DESIGN USES TWO MICROPROCESSORS.  ONE PROCESSOR CONTROLS
                   51 ; SYSTEM STATUS, INTERPRETS MONITOR COMMANDS, AND COMMUNICATES
                   52 ; WITH THE OUTSIDE WORLD THROUGH THE ON-BOARD KEYBOARD, DISPLAY, SERIAL
                   53 ; INTERFACES, CONTROL SIGNALS, ETC.
                   54 ; A SECOND PROCESSOR IS USED TO ACTUALLY
                   55 ; EXECUTE THE USER'S PROGRAM UNDER THE CONTROL OF THE FIRST.
                   56 ; THESE PROCESSORS ARE REFERRED TO
                   57 ; THROUGHOUT THIS PROGRAM AS THE MASTER PROCESSOR (MP) AND EXECUTION
                   58 ; PROCESSOR (EP) RESPECTIVELY.
                   59 ;
                   60 ; THE PROGRAM IN THIS LISTING IS EXECUTED BY THE MASTER PROCESSOR.
                   61 ; AT THE END OF THIS LISTING ARE SEVERAL SHORT "MINI-MONITOR OVERLAYS"
                   62 ; WHICH THE EXECUTION PROCESSOR EXECUTES WHEN INTERACTION BETWEEN THE
                   63 ; TWO PROCESSORS IS NECESSARY
                   64 ;
                   65 ; THIS PROGRAM WAS WRITTEN USING A NUMBER OF MACROS TO HANDLE THE ALLOCATION
                   66 ; OF MPU RESOURCES (WORKING REGISTERS, INTERNAL RAM, AND MP MONITOR ROM
                   67 ; FOR CODE AND DATA STORAGE).  THESE MACRO DEFINITIONS ARE INCLUDED IN A FILE
                   68 ; NAMED "ALLOC.MAC," AND ARE PRINTED IN THIS LISTING FOR REFERENCE.
                   69 ; ANOTHER SET OF MACROS IS USED TO SIMPLIFY THE ACCESSING OF VARIABLES
                   70 ; STORED IN INTERNAL RAM (AS OPPOSED TO WORKING REGISTERS) BY USING R1 TO
                   71 ; INDIRECTLY ADDRESS THE APPROPRIATE RAM LOCATION WHEN NECESSARY.
                   72 ; THESE MACROS ARE INCLUDED IN "MOMCOD.MAC", AND ARE ALSO PRINTED HERE.
                   73 ; COMPLETE UNDERSTANDING OF THESE MACROS IS NOT REQUIRED TO UNDERSTAND THE
                   74 ; MONITOR PROPER.  ALL LINES WHICH ACTUALLY PRODUCE OBJECT CODE APPEAR IN
                   75 ; THE LISTING ITSELF, INDENTED TWO SPACES FROM THE NORMAL TABULATION COLUMNS.
                   76 ; THE ACTUAL MONITOR PROGRAM FOR THE EMULATOR BEGINS AT APPROXIMATELY
                   77 ; SOURCE LINE NUMBER 500.
                   78 ;
                   79 ; LINES GENERATED BY MACRO EXPANSION ARE FLAGGED BY A PLUS SIGN ("+")
                   80 ; IMMEDIATELY FOLLOWING THE SOURCE LINE NUMBER.
                   81 ; A NUMBER OF LINES FROM THE VARIOUS MACRO DEFINITIONS WHICH DO NOT
                   82 ; PRODUCE ANY OBJECT CODE ARE PROCESSED BY THE ASSEMBLER
                   83 ; AS THESE MACROS ARE EXPANDED.  WHEN THIS IS THE CASE, THESE LINES ARE
                   84 ; SUPPRESSED FROM THE LIST FILE.  AS A RESULT, THE LINE NUMBERS ARE
                   85 ; NOT ALWAYS CONSECUTIVE WHERE A MACRO IS BEING INVOKED.
                   86 ;
                   87 ; NOTE:
                   88 ; ====
                   89 ; "SOURCE-LINE" REFERS TO THE DECIMAL NUMBERS LEFT OF EACH INSTRUCTION.
                   90 ; AT THE END OF THE LISTING IS AN ASSEMBLY CROSS-REFERENCE TABLE INDICATING
                   91 ; THE SEQUENTIAL SOURCE-LINE NUMBER OF ALL INSTANCES WHERE ANY VARIABLE
                   92 ; IS DEFINED OR REFERENCED.  THIS WILL BE OF GREAT ASSISTANCE IN
                   93 ; LOCATING SPECIFIC SUBROUTINES, ETC. IN THE LISTING.
                   94 ;
                   95 ; MNEMONICS COPYRIGHT (C) 1976 INTEL CORPORATION
                   96 ;
                   97 $EJECT
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

                  98 $       INCLUDE(:F0:ALLOC.MAC)
0004            = 99 ?R1     SET     0
                = 100 ;
0000            = 101 ?RB0   EQU     0
0001            = 102 ?RB1   EQU     1
0002            = 103 ?RAM   EQU     2
0003            = 104 ?CONST EQU     3
0004            = 105 ?R     EQU     4       ;ACCUMULATOR VARIABLE TYPE
                = 106 ;
                = 107 ;THE FOLLOWING INITIALIZES THE LINKED LIST POINTERS FOR
                = 108 ;THE REGISTER ALLOCATION AND DEALLOCATION ROUTINES.
                = 109 ;
0003            = 110 ?B0R2  SET     3
0004            = 111 ?B0R3  SET     4
0005            = 112 ?B0R4  SET     5
0006            = 113 ?B0R5  SET     6
0007            = 114 ?B0R6  SET     7
0008            = 115 ?B0R7  SET     8
                = 116 ,
0002            = 117 ?B0PNT SET     2
                = 118 ;
0003            = 119 ?B1R2  SET     3
0004            = 120 ?B1R3  SET     4
0005            = 121 ?B1R4  SET     5
0006            = 122 ?B1R5  SET     6
0007            = 123 ?B1R6  SET     7
0008            = 124 ?B1R7  SET     8
                = 125 ;
0002            = 126 ?B1PNT SET     2
                = 127 ;
0000            = 128 ORGPG0 SET     000H
0100            = 129 ORGPG1 SET     100H
0200            = 130 ORGPG2 SET     200H
0300            = 131 ORGPG3 SET     300H
0400            = 132 ORGPG4 SET     400H
0500            = 133 ORGPG5 SET     500H
0600            = 134 ORGPG6 SET     600H
0700            = 135 ORGPG7 SET     700H
                = 136 ;
                = 137 $EJECT
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

             = 138 ;*****************************************************
             = 139 ;
             = 140 ;       START OF ALLOCATION MACROS
             = 141 ;
             = 142 ;*****************************************************
             = 143 ;
             = 144 ?RSAVE  MACRO   SYMBOL,BANK,PNTVAL
-            = 145 IF      PNTVAL EQ 8
--           = 146         ERROR   2
-            = 147         EXITM
-            = 148 ENDIF
--           = 149 $       SAVE GEN
-            = 150         SYMBOL  SET  R&PNTVAL
--           = 151 $       RESTORE
-            = 152 ?B&BANK&PNT     SET     ?B&BANK&R&PNTVAL
             = 153         ENDM
             = 154 ;
             = 155 ;
 0020        = 156 ?MINDX  SET     20H
             = 157 ;
             = 158 ?MSAVE  MACRO   SYMBOL,LENGTH,ADDR
             = 159 $       SAVE GEN
--           = 160         SYMBOL  EQU  ADDR
-            = 161 $       RESTORE
.            = 162 ?MINDX  SET     ?MINDX+LENGTH
             = 163 ENDM
             = 164 ;
             = 165 MBLOCK  MACRO   SYMBOL,LENGTH
-            = 166 ?&SYMBOL        EQU   3
--           = 167         ?MSAVE  SYMBOL,LENGTH,%?MINDX
             = 168 ENDM
             = 169 ;
             = 170 DECLARE MACRO   SYMBOL,TYPE
-            = 171 ?&SYMBOL        SET     ?&TYPE
-            = 172 IF      ?&TYPE EQ 2
--           = 173         ?MSAVE  SYMBOL,1,%?MINDX
---          = 174         EXITM
-            = 175 ENDIF
--           = 176 IF      ?&TYPE EQ 0
-            = 177         ?RSAVE  SYMBOL,0,%?B0PNT
.            = 178         EXITM
-            = 179 ENDIF
.            = 180 IF      ?&TYPE EQ 1
--           = 181         ?RSAVE  SYMBOL,1,%?B1PNT
--           = 182         EXITM
.            = 183 ENDIF
             = 184         ENDM
             = 185 ,
             = 186 $       EJECT
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

                = 187 ;
                = 188 ;REORG  MACRO TO RESET THE INSTRUCTION LOCATION COUNTER
                = 189 ;       TO THE FIRST FREE LOCATION ON THE FIRST PAGE MODULE WILL
                = 190 ;       FIT WITHIN.
                = 191 REORG   MACRO LOCATION
                = 192 #SAVE GEN
                = 193           ORG     LOCATION
                = 194 $RESTORE
                = 195         ENDM
                = 196 ;
                = 197 ;CODEBLK        MACRO TO FIND A PAGE OF ROM
                = 198 ;       WHICH THIS BLOCK OF CODE WILL FIT WITHIN
                = 199 CODEBLK MACRO   LENGTH
                = 200 ?LENGTH SET     LENGTH
                = 201 IF      HIGH(ORGPG0+LENGTH-1) EQ 0
                = 202         REORG   %ORGPG0
                = 203 ?START  SET     $
                = 204 EXITM
                = 205 ENDIF
                = 206 IF      HIGH(ORGPG1+LENGTH-1) EQ 1
                = 207         REORG   %ORGPG1
                = 208 ?START  SET     $
                = 209 EXITM
                = 210 ENDIF
                = 211 IF      HIGH(ORGPG2+LENGTH-1) EQ 2
                = 212         REORG   %ORGPG2
                = 213 ?START  SET     $
                = 214 EXITM
                = 215 ENDIF
                = 216 IF      HIGH(ORGPG4+LENGTH-1) EQ 4
                = 217         REORG   %ORGPG4
                = 218 ?START  SET     $
                = 219 EXITM
                = 220 ENDIF
                = 221 IF      HIGH(ORGPG5+LENGTH-1) EQ 5
                = 222         REORG   %ORGPG5
                = 223 ?START  SET     $
                = 224 EXITM
                = 225 ENDIF
                = 226 IF      HIGH(ORGPG6+LENGTH-1) EQ 6
                = 227         REORG   %ORGPG6
                = 228 ?START  SET     $
                = 229 EXITM
                = 230 ENDIF
                = 231 IF      HIGH(ORGPG7+LENGTH-1) EQ 7
                = 232         REORG   %ORGPG7
                = 233 ?START  SET     $
                = 234 EXITM
                = 235 ENDIF
                = 236 IF      HIGH(ORGPG3+LENGTH-1) EQ 3
                = 237         REORG   %ORGPG3
                = 238 ?START  SET     $
                = 239 EXITM
                = 240 ENDIF
                = 241         ERROR   0       ;*** INSUFFICIENT SPACE FOR CODE ON ANY PAGE ***
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

                = 242          ENDM
                = 243 ;DATABLK         INSERTS ONTO PAGE 3
                = 244 DATABLK MACRO    LENGTH
-               = 245 ?LENGTH SET      LENGTH
--              = 246 IF       HIGH(ORGPG3+LENGTH-1) EQ 3
--              = 247          REORG   %ORGPG3
-               = 248 ?START   SET     $
--              = 249 EXITM
-               = 250 ENDIF
-               = 251          ERROR   0       ;*** INSUFFICIENT SPACE FOR DATA BLOCK ON PAGE 3 ***
                = 252          ENDM
                = 253 ;?SIZE   PRINTS A LINE TO THE SOURCE FILE GIVING BLOCK SIZE.
                = 254 ;        AND UPDATES APPROPRIATE ORGPG#
                = 255 ?SIZE    MACRO   BLK,PGE
                = 256 $SAVE GEN
-               = 257   SIZE   SET     BLK
-               = 258 ;
-               = 259 ;*******************************************************
.               = 260 IF ?LENGTH LT SIZE
-               = 261          ERROR   0       ;*** SIZE EXCEEDS SPACE CHECKED FOR BY CODEBLK MACRO
-               = 262 ENDIF
-               = 263 IF       HIGH($-1) NE HIGH(?START)
-               = 264          ERROR   0       ;*** CODE OR DATA BLOCK ROLLED OVER PAGE BOUNDARY ***
--              = 265 ENDIF
-               = 266 $RESTORE
--              = 267 ORGPG&PGE        SET     $
                = 268          ENDM
                = 269 ;SIZECHK         CHECKS SIZE OF PRECEDING BLOCK, PRINTS SIZE TO .LST FILE.
                = 270 SIZECHK MACRO
-               = 271          ?SIZE   %($-?START),%HIGH(?START)
                = 272          ENDM
                = 273 ;
                = 274 ;
                = 275 ;RSOURCE         CODE SPACE  ALLOCATION SUMMARY STATEMENT
                = 276 RSOURCE MACRO
                = 277 $SAVE LIST GEN
--              = 278            PGSIZE  SET   ORGPG0-000H      ;BYTES USED ON PAGE 0
-               = 279            PGSIZE  SET   ORGPG1-100H      ;BYTES USED ON PAGE 1
--              = 280            PGSIZE  SET   ORGPG2-200H      ;BYTES USED ON PAGE 2
-               = 281            PGSIZE  SET   ORGPG3-300H      ;BYTES USED ON PAGE 3
--              = 282            PGSIZE  SET   ORGPG4-400H      ;BYTES USED ON PAGE 4
--              = 283            PGSIZE  SET   ORGPG5-500H      ;BYTES USED ON PAGE 5
--              = 284            PGSIZE  SET   ORGPG6-600H      ;BYTES USED ON PAGE 6
-               = 285            PGSIZE  SET   ORGPG7-700H      ;BYTES USED ON PAGE 7
-               = 286 $EJECT
-               = 287 $RESTORE
                = 288          ENDM
                = 289 $EJECT
```

```
 LOC  OBJ        LINE       SOURCE STATEMENT

                   290 ;
                   291 $       INCLUDE(:F0:MOPCOD.MAC)
                 = 292 ;
                 = 293 ;?FORM1 MACRO   FOR GENERALIZING OPCODE INSTRUCTION
                 = 294 ;
                 = 295 ?FORM1  MACRO   OPCODE,SRC
-                = 296 IF      ?&SRC EQ 2
--               = 297 $       SAVE GEN
-                = 298           MOV     R1,#SRC
-                = 299           OPCODE          A,@R1
-                = 300 $       RESTORE
-                = 301         EXITM
--               = 302 ENDIF
--               = 303 IF      ?&SRC EQ 0 OR ?&SRC EQ 1
-                = 304 $       SAVE GEN
--               = 305           OPCODE          A,SRC
--               = 306 $       RESTORE
--               = 307         EXITM
-                = 308 ENDIF
--               = 309 IF      ?&SRC EQ 3
--               = 310 $       SAVE GEN
-                = 311           OPCODE          A,#SRC
--               = 312 $       RESTORE
--               = 313         EXITM
--               = 314 ENDIF
--               = 315         ERROR   1
                 = 316 ENDM
                 = 317 ;
                 = 318 ;?FORM2 MACRO   FOR GENERALIZING MOVES FROM THE ACC TO A VARIABLE
                 = 319 ?FORM2  MACRO   DEST
-                = 320 IF      ?&DEST EQ 2
-                = 321 $       SAVE GEN
-                = 322           MOV     R1,#DEST
--               = 323           MOV     @R1,A
--               = 324 $       RESTORE
--               = 325         EXITM
-                = 326 ENDIF
--               = 327 IF      ?&DEST EQ 0 OR ?&DEST EQ 1
--               = 328 $       SAVE GEN
-                = 329           MOV     DEST,A
-                = 330 $       RESTORE
--               = 331         EXITM
--               = 332 ENDIF
-                = 333         ERROR   1
                 = 334 ENDM
                 = 335 ;
                 = 336 ;?FORM3 MACRO   FOR GENERALIZING MOVES FROM THE ACC TO A VARIABLE
                 = 337 ;               WHEN IT IS KNOWN THAT R1 (IF NEEDED FOR INDIRECT ADDRESSING)
                 = 338 ;               IS ALREADY PRESET.
                 = 339 ?FORM3  MACRO   DEST
--               = 340 IF      ?&DEST EQ 2
-                = 341 $       SAVE GEN
--               = 342           MOV     @R1,A
--               = 343 $       RESTORE
-                = 344         EXITM
```

```
 LOC  OBJ        LINE       SOURCE STATEMENT

                 = 345 ENDIF
                 = 346 IF      ?&DEST EQ 0 OR ?&DEST EQ 1
                 = 347 $       SAVE GEN
                 = 348           MOV     DEST,A
                 = 349 $       RESTORE
                 = 350         EXITM
                 = 351 ENDIF
                 = 352         ERROR   1
                 = 353 ENDM
                 = 354 ;
                 = 355 ;?FORM4 MACRO   FOR GENERALIZING 'MOV   A,SRC' INSTRUCTION
                 = 356 ?FORM4  MACRO   SRC
                 = 357 IF      ?&SRC EQ 2
                 = 358 $       SAVE GEN
                 = 359           MOV     R1,#SRC
                 = 360           MOV     A,@R1
                 = 361 $       RESTORE
                 = 362         EXITM
                 = 363 ENDIF
                 = 364 IF      ?&SRC EQ 0 OR ?&SRC EQ 1
                 = 365 $       SAVE GEN
                 = 366           MOV     A,SRC
                 = 367 $       RESTORE
                 = 368         EXITM
                 = 369 ENDIF
                 = 370 IF      ?&SRC EQ 3
                 = 371 $       SAVE GEN
                 = 372           MOV     A,#SRC
                 = 373 $       RESTORE
                 = 374         EXITM
                 = 375 ENDIF
                 = 376         ERROR   1
                 = 377 ENDM
                 = 378 ;
                 = 379 ;?FORM5 MACRO   FOR GENERALIZING MOVING A CONSTANT INTO A VARIABLE
                 = 380 ?FORM5  MACRO   DEST,CONST
                 = 381 IF      ?&DEST EQ 0 OR ?&DEST EQ 1 OR ?&DEST EQ 4
                 = 382 $       SAVE GEN
                 = 383           MOV     DEST,#CONST
                 = 384 $       RESTORE
                 = 385         EXITM
                 = 386 ENDIF
                 = 387 IF      ?&DEST EQ 2
                 = 388 $       SAVE GEN
                 = 389           MOV     R1,#DEST
                 = 390           MOV     @R1,#CONST
                 = 391 $       RESTORE
                 = 392         EXITM
                 = 393 ENDIF
                 = 394         ERROR   1
                 = 395 ENDM
                 = 396 ;
                 = 397 ;MMOV   MACRO   GENERALIZED MOVE FROM SRC TO DEST
                 = 398 MMOV    MACRO   DEST,SRC
                 = 399 IF      ?&SRC EQ 3
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

                = 400          ?FORM5  DEST,SRC
                = 401          EXITM
                = 402 ENDIF
                = 403 IF       ?%DEST EQ 4
                = 404          ?FORM1  MOV,SRC
                = 405          EXITM
                = 406 ENDIF
                = 407 IF       ?%SRC EQ 4
                = 408          ?FORM2  DEST
                = 409          EXITM
                = 410 ENDIF
                = 411          ?FORM1  MOV,SRC
                = 412          ?FORM2  DEST
                = 413 ENDM
                = 414 ;?BINOP MACRO   GENERALIZES ARITHMETIC AND LOGICAL OPERATIONS
                = 415 ?BINOP  MACRO   OPCODE,DEST,SRC
                = 416 IF       ?%DEST EQ 4
                = 417          ?FORM1  OPCODE,SRC
                = 418          EXITM
                = 419 ENDIF
                = 420 IF       ?%SRC EQ 4
                = 421          ?FORM1  OPCODE,DEST
                = 422          ?FORM3  DEST
                = 423          EXITM
                = 424 ENDIF
                = 425          ?FORM1  MOV,SRC
                = 426          ?FORM1  OPCODE,DEST
                = 427          ?FORM3  DEST
                = 428 ENDM
                = 429 ;MADD   MACRO   FOR GENERALIZING ADD INSTRUCTION
                = 430 MADD    MACRO   DEST,SRC
                = 431          ?BINOP  ADD,DEST,SRC
                = 432          ENDM
                = 433 ;
                = 434 ;MADDC  MACRO   FOR GENERALIZING ADDC INSTRUCTION
                = 435 MADDC   MACRO   DEST,SRC
                = 436          ?BINOP  ADDC,DEST,SRC
                = 437          ENDM
                = 438 ;
                = 439 ;MANL   MACRO   FOR GENERALIZING ANL INSTRUCTION
                = 440 MANL    MACRO   DEST,SRC
                = 441          ?BINOP  ANL,DEST,SRC
                = 442          ENDM
                = 443 ;
                = 444 ;MORL   MACRO   FOR GENERALIZING ORL INSTRUCTION
                = 445 MORL    MACRO   DEST,SRC
                = 446          ?BINOP  ORL,DEST,SRC
                = 447          ENDM
                = 448 ;
                = 449 ;MXRL   MACRO   FOR GENERALIZING XRL INSTRUCTION
                = 450 MXRL    MACRO   DEST,SRC
                = 451          ?BINOP  XRL,DEST,SRC
                = 452          ENDM
                = 453 ;
                = 454 ;MXCH   MACRO   FOR GENERALIZING XCH INSTRUCTION
```

```
 LOC  OBJ        LINE        SOURCE STATEMENT

                = 455 MXCH    MACRO   DEST,SRC
-               = 456         ?BINOP  XCH,DEST,SRC
                = 457         ENDM
                = 458 ;
                = 459 ?UNARY  MACRO   OPCODE,DEST
-               = 460         ?FORM1  MOV,DEST
-               = 461 $SAVE GEN
-               = 462           OPCODE          A
-               = 463 $RESTORE
-               = 464         ?FORM3  DEST
                = 465         ENDM
                = 466 ;
                = 467 MINC    MACRO   DEST
-               = 468         ?UNARY  INC,DEST
                = 469         ENDM
                = 470 ;
                = 471 MDEC    MACRO   DEST
-               = 472         ?UNARY  DEC,DEST
                = 473         ENDM
                = 474 ;
                = 475 MDJNZ   MACRO   DEST,ADDR
-               = 476         ?UNARY  DEC,DEST
-               = 477 $SAVE GEN
--              = 478           JNZ     ADDR
--              = 479 $RESTORE
                = 480         ENDM
                = 481 ;
                = 482 MRL     MACRO   DEST
-               = 483         ?UNARY  RL,DEST
                = 484         ENDM
                = 485 ;
                = 486 MRR     MACRO   DEST
-               = 487         ?UNARY  RR,DEST
                = 488         ENDM
                = 489 ;
                = 490 MRRC    MACRO   DEST
-               = 491         ?UNARY  RRC,DEST
                = 492         ENDM
                = 493 ;
                = 494 MRLC    MACRO   DEST
-               = 495         ?UNARY  RLC,DEST
                = 496         ENDM
                = 497 ;
                = 498 $EJECT
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

                  499 ;
                  500 ;=======================================================
                  501 ;=======================================================
                  502 ;               BEGINNING OF PROGRAM PROPER
                  503 ;=======================================================
                  504 ;=======================================================
                  505 ;
                  506 ;
                  507 ;*********************************************************
                  508 ;
                  509 ;       ALLOCATION OF MP I/O PORTS:
                  510 ;
                  511 ;*********************************************************
                  512 ;
                  513 ;       BUS             ;USED FOR BIDIRECTIONAL ADDRESS AND DATA TRANSFERS
                  514 ;       P1              ;USED AS INDIVIDUAL CONTROL OUTPUTS AND BREAK LOGIC
                  515 ;       P2              ;HIGH-ORDER ADDRESS AND ADDRESS SPACE SELECTION
                  516 ;
000E              517 PDIGIT  EQU     P7      ;USED TO ENABLE CHARACTERS AND STROBE ROWS OF KEYBOARD
000D              518 PSEGHI  EQU     P6      ;USED TO TURN ON HI SEGMENTS OF CURRENTLY ENABLED DIGIT
000C              519 PSEGLO  EQU     P5      ;PORT FOR LOWER FOUR SEGMENTS
000B              520 PINPUT  EQU     P4      ;PORT USED TO SCAN FOR KEY CLOSURES
                  521 ;
                  522 ;*********************************************************
                  523 ;
                  524 ;       INDIVIDUAL PINS OF PORT 1 USED AS FOLLOWS:
                  525 ;
                  526 ;*********************************************************
                  527 ;
0001              528 ENBRAM  EQU     00000001B   ;P10 - HI ENABLES BREAK ON BREAK RAM OUTPUT SIGNAL
0002              529 ENBLNK  EQU     00000010B   ;P11 - HI ENABLES BREAK ON RD OR WR TO LINK BY EP
                  530                             ;      (NOTE: P11 & P10 BOTH HI ENABLES
                  531                             ;      BREAK ON ANY EP INSTRUCTION CYCLE)
0004              532 EPSSTP  EQU     00000100B   ;P12 - LO FORCES EP SS INPUT LOW,
                  533                             ;      HI GATES BREAKPOINT FLIP-FLOP TO EP SS INPUT.
0008              534 CLRBFF  EQU     00001000B   ;P13 - LO CLEARS BREAK FLIP-FLOP
                  535                             ;      AND ENABLES WR CONTROL TO BREAKPOINT RAM.
0010              536 EPRSET  EQU     00010000B   ;P14 - HI RESETS EP
0020              537 MODOUT  EQU     00100000B   ;P15 - LO WHEN EP IS EXECUTING USER PROGRAM,
                  538                             ;      HI WHEN EP FROZEN OR RUNNING OVERLAYS.
0040              539 TTYOUT  EQU     01000000B   ;P16 - SERIAL OUTPUT TO TTY OR CRT
                  540                             ;P17 - UNUSED
                  541 ;
                  542 $EJECT
```

```
LOC  OBJ        LINE        SOURCE STATEMENT

                543 ;*******************************************************
                544 ;
                545 ;       INDIVIDUAL PINS OF PORT 2 USED AS FOLLOWS
                546 ;
                547 ;*******************************************************
                548 ;
                549 ;       P23-P20             ;ADR11-ADR8 FOR ACCESSING PROGRAM OR DATA RAM ARRAY
                550 ;
0010            551 M0      EQU     00010000B   ;P24 - MEMORY MATRIX CONTROL PIN 0
0020            552 M1      EQU     00100000B   ;P25 - MEMORY MATRIX CONTROL PIN 1
0040            553 MPUSEL  EQU     01000000B   ;P26 - HIGH WHEN MP IN CONTROL OF COMMON MEM ARRAY,
                554                             ;      LOW WHEN EP IN CONTROL.
0080            555 EXPMON  EQU     10000000B   ;P27 - JUMPERED TO GROUND FOR STANDARD MONITOR,
                556                             ;      FLOATING WHEN EXPANSION MONITOR PRESENT.
                557 ;
                558 ;
                559 ;WHEN MP IN CONTROL OF MEMORY MATRIX M1-M0 USED AS FOLLOWS:
                560 ;
                561 ;       M1 M0   MODE
                562 ;        0  0   PROGRAM RAM ARRAY ENABLED FOR READ & WRITE
                563 ;        0  1   DATA RAM ARRAY ENABLED FOR READ & WRITE
                564 ;        1  X   LINK REGISTER ENABLED FOR READ, RAM ARRAYS DISABLED.
                565 ;               (NOTE: LINK REGISTER ALWAYS ENABLED FOR MP WRITES)
                566 ;
                567 ;WHEN EP IN CONTROL OF MATRIX M1-M0 USED AS FOLLOWS:
                568 ;
                569 ;       M1 M0   MODE
                570 ;        0  X   EP PSEN FETCHES FROM LINK REGISTER (USED TO FORCE OPCODES)
                571 ;        1  0   EP PSEN FETCHES FROM PROGRAM RAM ARRAY,
                572 ;               EP RD & WR CONTROL DATA RAM ARRAY
                573 ;        1  1   EP PSEN FETCHES FROM PROGRAM RAM ARRAY,
                574 ;               RD & WR CONTROL LINK REGISTER.
                575 ;
                576 $EJECT
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

                 577 ;
                 578 ;*******************************************************
                 579 ;
                 580 ;       SYSTEM CONSTANT DEFINITIONS
                 581 ;
                 582 ;*******************************************************
                 583 ;
                 584 DECLARE CHARNO,CONST    ;NUMBER OF DIGITS IN DISPLAY AND ROWS OF KEYS
0008             598         CHARNO  EQU     8
                 599 ;
                 600 DECLARE NCOLS,CONST     ;LESSER DIMENSION OF KEYBOARD MATRIX
0004             614         NCOLS   EQU     4
                 615 ;
                 616 DECLARE DEBNCE,CONST    ;NUMBER OF SUCESSIVE SCANS BEFORE KEY CLOSURE ACCEPTED
0008             630         DEBNCE  EQU     8
                 631 ;
                 632 DECLARE OVSIZE,CONST    ;SIZE OF LARGEST MINI-MONITOR OVERLAY FOR EP
0017             646         OVSIZE  EQU     23
                 647 ;
                 648 DECLARE BUFLEN,CONST    ;LENGTH OF HEX FORMAT XMIT BUFFER (MAX RECORD LENGTH)
0010             662         BUFLEN  EQU     16
                 663 ;
                 664 ;*******************************************************
                 665 ;
                 666 ;       UTILITY CONSTANT DECLARATIONS
                 667 ;
                 668 ;*******************************************************
                 669 ;
                 670 DECLARE ZERO,CONST
0000             684 ZERO    EQU     0
                 685 DECLARE PLUS1,CONST
0001             699 PLUS1   EQU     1
                 700 DECLARE PLUS3,CONST
0003             714 PLUS3   EQU     3
                 715 DECLARE NEG1,CONST
FFFF             729 NEG1    EQU     -1
                 730 ;
                 731 $EJECT
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

                  732 ;
                  733 ;*********************************************************
                  734 ;
                  735 ;       BANK 0 REGISTER ALLOCATION:
                  736 ;
                  737 ;*********************************************************
                  738 ;
                  739 DECLARE LDATA,RB0       ;DATA USED BY LOGICAL ADDRESSING READ/WRITE UTILITIES
0002              752+        LDATA   SET  R2
                  756 DECLARE KEY,RB0         ;HOLDS KEYCODE RETURNED FROM KBD INPUT ROUTINE.
0003              769+        KEY     SET  R3
                  773 DECLARE ITMP,RB0        ;COUNTER USED AS AN INDEX IN PARSER ROUTINE
0004              786+        ITMP    SET  R4
                  790 DECLARE CHKSUM,RB0      ;CHECKSUM OF DATA BYTES TRANSMITTED IN HEX FILE FORMAT
0005              803+        CHKSUM  SET  R5
                  807 DECLARE DSPTMP,RB0      ;TEMPORARY STORAGE FOR DISPLAY PATTERNS IN 'DSPACC'
0006              820+        DSPTMP  SET  R6
                  824 DECLARE XPCODE,RB0      ;EXPANSION MONITOR ROUTINE CODE NUMBER
0007              837+        XPCODE  SET  R7
                  841 ;
                  842 ;*********************************************************
                  843 ;
                  844 ;       BANK 1 REGISTER ALLOCATION
                  845 ;
                  846 ;*********************************************************
                  847 ;
                  848 DECLARE ROTPAT,RB1      ;USED TO HOLD INPUT PATTERN BEING ROTATED THROUGH CY
0002              865+        ROTPAT  SET  R2
                  869 DECLARE ROTCNT,RB1      ;COUNTS NUMBER OF BITS ROTATED THROUGH CY
0003              886+        ROTCNT  SET  R3
                  890 DECLARE LASTKY,RB1      ;HOLDS KEY POSITION OF LAST KEY DEPRESSION DETECTED
0004              907+        LASTKY  SET  R4
                  911 DECLARE CURDIG,RB1      ;HOLDS POSITION OF NEXT CHARACTER TO BE DISPLAYED
0005              928+        CURDIG  SET  R5
                  932 DECLARE KEYFLG,RB1      ;FLAG TO DETECT WHEN ALL KEYS ARE RELEASED
0006              949+        KEYFLG  SET  R6
                  953                         ;(REGISTER 7 NOT USED FOR PRIMARY MONITOR)
                  954 ;
                  955 ;*********************************************************
                  956 $EJECT
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

                 957 ;
                 958 ;*******************************************************
                 959 ;
                 960 ;       DATA RAM ALLOCATION
                 961 ;
                 962 ;*******************************************************
                 963 ;
                 964 DECLARE EPACC,RAM       ;STORAGE IN MP FOR EP ACCUMULATOR
0020             969+        EPACC   EQU  32
                 973 DECLARE EPPSW,RAM       ;STORAGE IN MP FOR EP PROGRAM STATUS WORD
0021             978+        EPPSW   EQU  33
                 982 DECLARE EPTIMR,RAM      ;STORAGE IN MP FOR EP TIMER/COUNTER REGISTER
0022             987+        EPTIMR  EQU  34
                 991 DECLARE EPR0,RAM        ;STORAGE IN MP FOR EP REGISTER 0 OF BANK 0
0023             996+        EPR0    EQU  35
                1000 DECLARE EPPCLO,RAM      ;STORAGE IN MP FOR LOW BYTE OF EP PROGRAM COUNTER
0024            1005+        EPPCLO  EQU  36
                1009 DECLARE EPPCHI,RAM      ;STORAGE IN MP FOR HIGH NIBBLE OF EP PROGRAM COUNTER
0025            1014+        EPPCHI  EQU  37
                1018 DECLARE HBITLO,RAM      ;PARAMETER 1 FOR SERIAL LINK DATA RATE GENERATOR
0026            1023+        HBITLO  EQU  38
                1027 DECLARE HBITHI,RAM      ;PARAMETER 2 FOR SERIAL LINK DATA RATE GENERATOR
0027            1032+        HBITHI  EQU  39
                1036 DECLARE DSPTIM,RAM      ;PARAMETER FOR AUTO-STEP AND AUTO-BREAK SEQUENCING RATE
0028            1041+        DSPTIM  EQU  40
                1045 DECLARE VERSNO,RAM      ;MONITOR VERSION NUMBER
0029            1050+        VERSNO  EQU  41
                1054 DECLARE HREGA,RAM       ;  (UNUSED)
002A            1059+        HREGA   EQU  42
                1063 DECLARE HREGB,RAM       ;  (UNUSED)
002B            1068+        HREGB   EQU  43
                1072 DECLARE HREGC,RAM       ;  (UNUSED)
002C            1077+        HREGC   EQU  44
                1081 DECLARE HREGD,RAM       ;  (UNUSED)
002D            1086+        HREGD   EQU  45
                1090 DECLARE HREGE,RAM       ;  (UNUSED)
002E            1095+        HREGE   EQU  46
                1099 DECLARE HREGF,RAM       ;  (UNUSED)
002F            1104+        HREGF   EQU  47
                1108 DECLARE SMALO,RAM       ;PRIMARY COMMAND STARTING MEMORY ADDRESS (LOW BYTE)
0030            1113+        SMALO   EQU  48
                1117 DECLARE SMAHI,RAM       ;PRIMARY COMMAND STARTING MEMORY ADDRESS (HIGH BYTE)
0031            1122+        SMAHI   EQU  49
                1126 DECLARE EMALO,RAM       ;PRIMARY COMMAND ENDING MEMORY ADDRESS (LOW BYTE)
0032            1131+        EMALO   EQU  50
                1135 DECLARE EMAHI,RAM       ;PRIMARY COMMAND ENDING MEMORY ADDRESS (HIGH BYTE)
0033            1140+        EMAHI   EQU  51
                1144 DECLARE MEMLO,RAM       ;THIRD PARSER PARAMETER & HEX RECORD ADDRESS (LOW)
0034            1149+        MEMLO   EQU  52
                1153 DECLARE MEMHI,RAM       ;THIRD PARSER PARAMETER & HEX RECORD ADDRESS (HIGH)
0035            1158+        MEMHI   EQU  53
                1162 DECLARE BCODE,RAM       ;PRIMARY COMMAND NUMBER FROM PARSER TABLES (0-9)
0036            1167+        BCODE   EQU  54
                1171 DECLARE TYPE,RAM        ;PRIMARY COMMAND MODIFIER/OPTION (0-5)
0037            1176+        TYPE    EQU  55
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

                1180 DECLARE NUMCON,RAM      ;MAX  NUMBER OF PARAMETERS ALLOWED FOR SELECTED COMMAND
0038            1185+        NUMCON  EQU  56
                1189 DECLARE OPTION,RAM      ;INDEX POINTER USED IN SEARCHING PARSER TABLES
0039            1194+        OPTION  EQU  57
                1198 DECLARE NEXTPL,RAM      ;CHARACTER POSITION FOR DISPLAY UTILITIES TO WRITE NEXT
003A            1203+        NEXTPL  EQU  58
                1207 DECLARE KBDBUF,RAM      ;POSITION OF KEY DEBOUNCED BY SCANNING SUBROUTINE
003B            1212+        KBDBUF  EQU  59
                1216 DECLARE KEYLOC,RAM      ;INCREMENTED AS SUCCESSIVE KEY LOCATIONS SCANNED
003C            1221+        KEYLOC  EQU  60
                1225 DECLARE NREPTS,RAM      ;KEEPS TRACK OF SUCCESSIVE READS OF SAME KEYSTROKE
003D            1230+        NREPTS  EQU  61
                1234 DECLARE ASAVL,RAM       ;HOLDS ACCUMULATOR VALUE DURING SERVICE ROUTINE
003E            1239+        ASAVE   EQU  62
                1243 DECLARE RDELAY,RAM      ;COUNTER DECREMENTED WHEN AUTO-STEP DELAY IN PROGRESS
003F            1248+        RDELAY  EQU  63
                1252 DECLARE STRTMP,RAM      ;INDEX POINTER FOR DISPLAY CHARACTER STRING ACCESSING
0040            1257+        STRTMP  EQU  64
                1261 DECLARE BUFCNT,RAM      ;COUNT OF DATA BYTES IN HEX FORMAT RECORD BUFFER
0041            1266+        BUFCNT  EQU  65
                1270 DECLARE RECTYP,RAM      ;TYPE OF HEX FORMAT RECORD (0 OR 1)
0042            1275+        RECTYP  EQU  66
                1279 DECLARE B,RAM           ;BIT COUNTER FOR ASCII SERIAL I/O UTILITY SUBROUTINES
0043            1284+        B       EQU  67
                1288 DECLARE REGC,RAM        ;CHARACTER BEING SHIFTED DURING SERIAL I/O PROCESS
0044            1293+        REGC    EQU  68
                1297 DECLARE H,RAM           ;COUNTER IN SOFTWARE DELAY DATA RATE GENERATOR
0045            1302+        H       EQU  69
                1306 ;
                1307 MBLOCK  SEGMAP,CHARNO   ;REGISTER ARRAY FOR DISPLAY PATTERNS
0046            1311+        SEGMAP  EQU  70
                1314 ;
                1315 MBLOCK  OVBUF,OVSIZE    ;LOW-ORDER USER PROGRAM DURING MINI-MONITOR OVERLAYS
004E            1319+        OVBUF   EQU  78
                1322 ;
                1323 MBLOCK  HEXBUF,BUFLEN   ;ALLOCATE BLOCK OF RAM FOR USE AS HEX RECORD BUFFER
0065            1327+        HEXBUF  EQU  101
                1330 ;
                1331 $EJECT
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

                1332        DATABLK 40
0300            1337+         ORG    768
                1341 ;INVALS TABLE OF CONSTANTS TO BE LOADED INTO MP INTERNAL RAM VARIABLES
                1342 ;       AS PART OF SYSTEM INITIALIZATION PROCEDURE:
                1343 ;
                1344 ;       INITIAL VALUE   VARIABLE       TYPE
                1345 ;       ======= =====   ========       ====
0300 00         1346 INVALS: DB      00H     ;ROTPAT        RB1
0301 00         1347         DB      00H     ;ROTCNT        RB1
0302 00         1348         DB      00H     ;LASTKY        RB1
0303 00         1349         DB      CHARNO  ;CURDIG        RB1
0304 00         1350         DB      00H     ;KEYFLG        RB1
0305 00         1351         DB      00H     ;<REG7>        RB1
0306 00         1352         DB      00H     ;EPACC         RAM
0307 01         1353         DB      01H     ;EPPSW         RAM
0308 00         1354         DB      00H     ;EPTIMR        RAM
0309 00         1355         DB      00H     ;EPR0          RAM
030A 00         1356         DB      00H     ;EPPCLO        RAM
030B 00         1357         DB      00H     ;EPPCHI        RAM
030C 93         1358         DB      93H     ;HBITLO        RAM
030D 04         1359         DB      04H     ;HBITHI        RAM
030E 20         1360         DB      20H     ;DSPTIM        RAM
030F 25         1361         DB      25H     ;VERSNO        RAM
0310 00         1362         DB      00H     ;HREGA         RAM
0311 00         1363         DB      00H     ;HREGB         RAM
0312 00         1364         DB      00H     ;HREGC         RAM
0313 00         1365         DB      00H     ;HREGD         RAM
0314 00         1366         DB      00H     ;HREGE         RAM
0315 00         1367         DB      00H     ;HREGF         RAM
0316 00         1368         DB      00H     ;SMALO         RAM
0317 00         1369         DB      00H     ;SMAHI         RAM
0318 FF         1370         DB      0FFH    ;EMALO         RAM
0319 0F         1371         DB      0FH     ;EMAHI         RAM
031A 00         1372         DB      00H     ;MEMLO         RAM
031B 00         1373         DB      00H     ;MEMHI         RAM
031C 00         1374         DB      00H     ;BCODE         RAM
031D 04         1375         DB      04H     ;TYPE          RAM
031E 01         1376         DB      01H     ;NUMCON        RAM
031F 00         1377         DB      00H     ;OPTION        RAM
0320 00         1378         DB      CHARNO  ;NEXTPL        RAM
0321 FF         1379         DB      0FFH    ;KBDBUF        RAM
0322 00         1380         DB      00H     ;KEYLOC        RAM
0023            1381 NOVALS  EQU     $-INVALS
                1382         SIZECHK
0023            1385+  SIZE  SET   35
                1386+;
                1387+;*********************************************************
                1396 $EJECT
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

                 1397 $      INCLUDE(:F0:PARSER.MOD)
                =1398        CODEBLK 45
0000            =1403+         ORG    0
                =1407 ;INIT  INITIALIZES PROCESSOR REGISTERS
                =1408 ;      AND RAM LOCATIONS TO DEFINED VALUES.
0000 C5         =1409 INIT:  SEL    RB0
0001 BF00       =1410        MOV    XPCODE,#0
0003 74D1       =1411        CALL   XPTEST
0005 27         =1412        CLR    A
0006 3D         =1413        MOVD   PSEGLO,A
0007 3E         =1414        MOVD   PSEGHI,A
0008 B81A       =1415        MOV    R0,#1AH ;START AT KB1 (REG2) = RAM LOC 1AH
000A B923       =1416        MOV    R1,#LOW NOVALS
000C BA00       =1417        MOV    R2,#LOW INVALS
000E FA         =1418 INITLP: MOV   A,R2
000F E3         =1419        MOVP3  A,@A
0010 A0         =1420        MOV    @R0,A
0011 18         =1421        INC    R0
0012 1A         =1422        INC    K2
0013 E90E       =1423        DJNZ   R1,INITLP
0015 55         =1424        STRT   T
0016 744F       =1425        CALL   EPSRK
0018 B8BB       =1426        MOV    R0,#LOW(OV1BAS+OVSIZE)
001A 746A       =1427        CALL   OVLORD
001C 54E5       =1428        CALL   COMFIL
001E B937       =1429        MOV    R1,#TYPE
0020 11         =1430        INC    @R1
0021 34F2       =1431        CALL   INCSMA
0023 54E5       =1432        CALL   COMFIL
0025 99EF       =1433        ANL    P1,#(NOT EPRSET)       ;REMOVE EP RESET SIGNAL
0027 0429       =1434        JMP    MAIN
                =1435 ;
                =1436        SIZECHK
0029            =1439+ SIZE  SET   41
                =1440+;
                =1441+;*****************************************************
                =1450 $EJECT
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

                =1451 ;
                =1452 ;       KEYBOARD LAYOUT:
                =1453 ;       ======== ======
                =1454 ;
                =1455 ;    ---------  ---------  ---------  ---------    -----  -----  -----  -----
                =1456 ;    !        !!        !!        !!        !  !     !!     !!     !!     !
                =1457 ;    !  LIST  !!GO/RESET!!   GO   !!EXAM/CHG!  !  C  !!  D  !!  E  !!  F  !
                =1458 ;    !        !!        !!        !!        !  !     !!     !!     !!     !
                =1459 ;    ---------  ---------  ---------  ---------    -----  -----  -----  -----
                =1460 ;    ---------  ---------  ---------  ---------    -----  -----  -----  -----
                =1461 ;    !        !!PROG BRK!!PROG MEM!!REGISTER!  !     !!     !!     !!     !
                =1462 ;    ! UPLOAD !!  ----  !!  ----  !!  ----  !  !  8  !!  9  !!  A  !!  B  !
                =1463 ;    !        !!AUTO STP!!SING STP!! NO BRK !  !     !!     !!     !!     !
                =1464 ;    ---------  ---------  ---------  ---------    -----  -----  -----  -----
                =1465 ;    ---------  ---------  ---------  ---------    -----  -----  -----  -----
                =1466 ;    !        !!DATA BRK!!DATA MEM!!        !  !     !!     !!     !!     !
                =1467 ;    ! DNLOAD !!  ----  !!  ----  !!CLR/PREV!  !  4  !!  5  !!  6  !!  7  !
                =1468 ;    !        !!AUTO BRK!!WITH BRK!!        !  !     !!     !!     !!     !
                =1469 ;    ---------  ---------  ---------  ---------    -----  -----  -----  -----
                =1470 ;    ---------  ---------  ---------  ---------    -----  -----  -----  -----
                =1471 ;    !        !!        !!        !!        !  !     !!     !!     !!     !
                =1472 ;    !  FILL  !!HARD REG!! NEXT/, !! END/   !  !  0  !!  1  !!  2  !!  3  !
                =1473 ;    !        !!        !!        !!        !  !     !!     !!     !!     !
                =1474 ;    ---------  ---------  ---------  ---------    -----  -----  -----  -----
                =1475 ;
                =1476 $EJECT
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

                =1477 ;
                =1478 ;       THE FOLLOWING EQUATES DETERMINES HOW THE PARSER INTERPRETS
                =1479 ;       VALUES RETURNED BY THE KEYBOARD SCANNING INPUT ROUTINE
                =1480 ;       WHEN THE VARIOUS KEYS OF THE KEYBOARD ARE PRESSED.
                =1481 ;
                =1482 ;
                =1483 ;KEY0   EQU     00H     VALUE RETURNED FOR EACH KEY OF KEYBOARD MATRIX
                =1484 ;KEY1   EQU     01H     BY KEYBOARD SCANNING SUBROUTINE "KBDIN".
                =1485 ;KEY2   EQU     02H
                =1486 ;KEY3   EQU     03H     +----+----+----+----+  +----+----+----+----+
                =1487 ;KEY4   EQU     04H     ! 1C ! 1D ! 1E ! 1F !  ! 0C ! 0D ! 0E ! 0F !
                =1488 ;KEY5   EQU     05H     +----+----+----+----+  +----+----+----+----+
                =1489 ;KEY6   EQU     06H     ! 18 ! 19 ! 1A ! 1B !  ! 08 ! 09 ! 0A ! 0B !
                =1490 ;KEY7   EQU     07H     +----+----+----+----+  +----+----+----+----+
                =1491 ;KEY8   EQU     08H     ! 14 ! 15 ! 16 ! 17 !  ! 04 ! 05 ! 06 ! 07 !
                =1492 ;KEY9   EQU     09H     +----+----+----+----+  +----+----+----+----+
                =1493 ;KEYA   EQU     0AH     ! 10 ! 11 ! 12 ! 13 !  ! 00 ! 01 ! 02 ! 03 !
                =1494 ;KEYB   EQU     0BH     +----+----+----+----+  +----+----+----+----+
                =1495 ;KEYC   EQU     0CH
                =1496 ;KEYD   EQU     0DH
                =1497 ;KEYE   EQU     0EH
                =1498 ;KEYF   EQU     0FH
0010            =1499 KEYFIL  EQU     10H     ;[FILL COMMAND]
0012            =1500 KEYNXT  EQU     12H     ;[NEXT/, ]
0013            =1501 KEYEND  EQU     13H     ;[END/. ]
0014            =1502 KEYREL  EQU     14H     ;[DOWNLOAD COMMAND]
0015            =1503 KEYPAT  EQU     15H     ;[AUTOBREAK MODIFIER]
0016            =1504 KEYDM   EQU     16H     ;[DATA MEMORY MODIFIER]
0017            =1505 KEYCLR  EQU     17H     ;[CLEAR/PREVIOUS]
0018            =1506 KEYREC  EQU     18H     ;[UPLOAD COMMAND]
0019            =1507 KEYTRA  EQU     19H     ;[AUTOSTEP MODIFIER]
001A            =1508 KEYPM   EQU     1AH     ;[PROGRAM MEMORY MODIFIER]
001B            =1509 KEYREG  EQU     1BH     ;[REGISTER MEMORY MODIFIER]
001C            =1510 KEYLST  EQU     1CH     ;[FORMATTED DATA OUTPUT COMMAND]
001D            =1511 KGORES  EQU     1DH     ;[GO FROM RESET STATE COMMAND]
001E            =1512 KEYGO   EQU     1EH     ;[GO COMMAND]
001F            =1513 KEYMOD  EQU     1FH     ;[EXAMINE/MODIFY COMMAND]
0008            =1514 KSETB   EQU     08H     ;[SET BREAKPOINT COMMAND]
000C            =1515 KCLRB   EQU     0CH     ;[CLEAR BREAKPOINT COMMAND]
                =1516 ;
                =1517 ;
0019            =1518 PBRK    EQU     19H     ;[PROGRAM BREAKPOINT MEMORY MODIFIER]
0015            =1519 DBRK    EQU     15H     ;[DATA BREAKPOINT MEMORY MODIFIER]
0011            =1520 RINT    EQU     11H     ;[HARDWARE REGISTER MEMORY MODIFIER]
001B            =1521 NOBRK   EQU     1BH     ;[WITHOUT BREAKPOINTS MODIFIER]
0016            =1522 WBRK    EQU     16H     ;[WITH BREAKPOINTS ENABLED MODIFIER]
001A            =1523 SING    EQU     1AH     ;[SINGLE STEP MODIFIER]
                =1524 ;
                =1525 $EJECT
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

                =1526          CODEBLK 160
0029            =1531+           ORG     41
                =1535 ;MAIN    OUTPUT_MESSAGE(COMMAND_PROMPT)
                =1536 ;        CALL INPUT_BYTE(KEY)
                =1537 ;MAIN2   IF THE KEY=END GO TO MAIN
                =1538 ;
0029 BF01       =1539 MAIN:    MOV     XPCODE,#1
002B 74D1       =1540          CALL    XPTEST
002D 2301       =1541          MOV     A,#1
002F 3400       =1542          CALL    OUTUTL
0031 14EC       =1543          CALL    INPKEY
0033 FB         =1544 MAIN2:   MOV     A,KEY
0034 D313       =1545          XRL     A,#KEYEND
0036 C629       =1546          JZ      MAIN
                =1547 ;
                =1548 ;FINDOP FIND OUT IF THE KEY PRESSED IS A LEGITIMATE COMMAND INITIATOR:
                =1549 ;        ITMP:=CTAB
                =1550 ;        BCODE:=TYPE:=0
                =1551 ;        WHILE CTAB(ITMP)<>0             /CTAB EXHAUSTED/
                =1552 ;          IF CTAB(ITMP)=KEY GOTO MAINA  /COMMAND ENTRY FOUND IN CTAB/
                =1553 ;          ELSE     ITMP:=ITMP+COMMAND_ENTRY_SIZE
                =1554 ;                   BCODE:=BCODE+1
                =1555 ;        ENDWHILE
                =1556 ;        GOTO ERROR
0038 BC23       =1557          MOV     ITMP,#CTAB
                =1558          MMOV    BCODE,ZERO
003A B936       =1569+           MOV     R1,#BCODE
003C B100       =1570+           MOV     @R1,#ZERO
                =1574          MMOV    TYPE,ZERO
003E B937       =1585+           MOV     R1,#TYPE
0040 B100       =1586+           MOV     @R1,#ZERO
0042 FC         =1590 FINDOP:  MOV     A,ITMP
0043 E3         =1591          MOVP3   A,@A
0044 B2BC       =1592          JB5     MERROR
0046 DB         =1593          XRL     A,KEY
0047 C652       =1594          JZ      MAINA
0049 FC         =1595          MOV     A,ITMP
004A 0303       =1596          ADD     A,#COMSIZ
004C AC         =1597          MOV     ITMP,A
004D B936       =1598          MOV     R1,#BCODE
004F 11         =1599          INC     @R1
0050 0442       =1600          JMP     FINDOP
                =1601 ;
                =1602 ;        OUTPUT_MESSAGE(STRCOM(BCODE))  /*PROMPT FOR THE CURRENT COMMAND*/
                =1603 ;        I:=1+1
                =1604 ;        OPTION:=MEM(I)
                =1605 ;        I:=I+1
                =1606 ;        NO_OF_PARAMETERS:=MEM(I)
                =1607 ;        I:=3
                =1608 ;
                =1609 MAINA:   MMOV    A,BCODE
0052 B936       =1618+           MOV     R1,#BCODE
0054 F1         =1619+           MOV     A,@R1
0055 031D       =1623          ADD     A,#STRCOM
0057 3402       =1624          CALL    OUTCLR
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

0059 1C         =1625         INC     ITMP
005A FC         =1626         MOV     A,ITMP
005B E3         =1627         MOVP3   A,@A            ; GET OPTION POINTER
                =1628         MMOV    OPTION,A
005C B939       =1641+          MOV     R1,#OPTION
005E A1         =1642+          MOV     @R1,A
005F 1C         =1646         INC     ITMP
0060 FC         =1647         MOV     A,ITMP
0061 E3         =1648         MOVP3   A,@A            ; GET NO OF PARAMETERS
                =1649         MMOV    NUMCON,A
0062 B938       =1662+          MOV     R1,#NUMCON
0064 A1         =1663+          MOV     @R1,A
                =1667 ;
                =1668 ;       PARAMETER_BUFFER(0=>5):=0
                =1669 ;
0065 B90C       =1670         MOV     R1,#6           ; EACH PARAM IS 2 BYTES
0067 B830       =1671         MOV     R0,#SMALO       ; START OF PARAM BUFFERS
0069 B000       =1672 MAINB:  MOV     @R0,#00H
006B 18         =1673         INC     R0
006C E969       =1674         DJNZ    R1,MAINB
006E 14EC       =1675         CALL    INPKEY
                =1676 ;
                =1677 ;       WHILE KEY<>MEM(OPTION+TYPE)[6-0] DO
                =1678 ;         IF MEM(OPTION+TYPE)[7]=1 GOTO MAIND1
                =1679 ;         TYPE.=TYPE+1
                =1680 ;       ENDWHILE
                =1681 ;
                =1682         MMOV    ITMP,OPTION
0070 B939       =1698+          MOV     R1,#OPTION
0072 F1         =1699+          MOV     A,@R1
0073 AC         =1712+          MOV     ITMP,A
0074 1C         =1715         INC     ITMP
                =1716 MAINC1: MMOV    A,ITMP
0075 FC         =1732+          MOV     A,ITMP
0076 E3         =1736         MOVP3   A,@A
0077 97         =1737         CLR     C
0078 F7         =1738         RLC     A
0079 77         =1739         RR      A       ; STRIP BIT SEVEN INTO CARRY
007A DB         =1740         XRL     A,KEY
007B C693       =1741         JZ      MAIND
007D F687       =1742         JC      MAIND1
                =1743         MINC    TYPE
007F B937       =1748+          MOV     R1,#TYPE
0081 F1         =1749+          MOV     A,@R1
0082 17         =1753+          INC     A
0083 A1         =1758+          MOV     @R1,A
0084 1C         =1761         INC     ITMP
0085 0475       =1762         JMP     MAINC1
                =1763 ;
                =1764 ;       MODIFIER NOT FOUND SO RESET TYPE INDEX TO DEFAULT CASE (ZERO).
                =1765 ;
                =1766 MAIND1: MMOV    TYPE,ZERO
0087 B937       =1777+          MOV     R1,#TYPE
0089 B100       =1778+          MOV     @R1,#ZERO
                =1782         MMOV    A,OPTION
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

008B B939       =1791+       MOV    R1,#OPTION
008D F1         =1792+       MOV    A,@R1
008E E3         =1796        MOVP3  A,@A
008F 3404       =1797        CALL   OUTMSG
0091 049E       =1798        JMP    MAINB0
                =1799 ;
                =1800 ;      CALL OUTPUT_MESSAGE(MODIFIER)
                =1801 MAIND: MMOV   A,OPTION
0093 B939       =1810+       MOV    R1,#OPTION
0095 F1         =1811+       MOV    A,@R1
0096 E3         =1815        MOVP3  A,@A
                =1816        MADD   A,TYPE
0097 B937       =1822+       MOV    R1,#TYPE
0099 61         =1823+       ADD    A,@R1
009A 3404       =1827        CALL   OUTMSG
009C 14EC       =1828        CALL   INPKEY
                =1829 ;
009E BC00       =1830 MAINB0: MOV     ITMP,#0
00A0 2330       =1831 MAINB1: MOV     A,#SMALO
00A2 6C         =1832        ADD    A,ITMP
00A3 6C         =1833        ADD    A,ITMP
00A4 A8         =1834        MOV    R0,A
00A5 14C0       =1835        CALL   INPADR
00A7 FGBA       =1836        JC     CMDINT
00A9 1C         =1837        INC    ITMP
00AA B938       =1838        MOV    R1,#NUMCON
00AC F1         =1839        MOV    A,@R1
00AD 07         =1840        DEC    A
00AE A1         =1841        MOV    @R1,A
00AF C6BA       =1842        JZ     CMDINT
00B1 FB         =1843        MOV    A,KEY
00B2 D313       =1844        XRL    A,#KEYEND
00B4 C6BA       =1845        JZ     CMDINT
00B6 14EC       =1846        CALL   INPKEY
00B8 04A0       =1847        JMP    MAINB1
                =1848 ;
                =1849 ;CMDINT ENTER THE COMMAND PROCESSOR WITH:
                =1850 ;       BASE_CODE=THE MAIN COMMAND TYPE
                =1851 ;       TYPE=SUBCOMMAND TYPE
                =1852 ;       PARAMETER(1)=FIRST ADDRESS
                =1853 ;       PARAMETER(2)=SECOND ADDRESS
                =1854 ;       PARAMETER(3)=DATA
00BA 4400       =1855 CMDINT: JMP     IMPLEM
                =1856 ;
                =1857 ;MERROR ERROR ENCOUNTERED IN MAIN PARSING ROUTINE.
00BC BA01       =1858 MERROR: MOV     LDATA,#1
00BE 249A       =1859        JMP     PERROR
                =1860        SIZECHK
0097            =1863+  SIZE  SET   151
                =1864+;
                =1865+;*************************************************************
                =1874 $EJECT
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

               =1875        DATABLK 50
0323           =1880+         ORG     803
               =1884 ;
               =1885 ;*************************************************************
               =1886 ;
               =1887 ;       TABLES FOR PARSER
               =1888 ;
               =1889 ;*************************************************************
               =1890 ;
               =1891 ;       THE CTAB TABLE CONTAINS (COMSIZ) ENTRIES FOR EACH COMMAND.  THE MEANING
               =1892 ;       OF THE ENTRIES IS AS FOLLOWS:
               =1893 ;
               =1894 ;       ENTRY 0  COMMAND KEY TO INITIATE
               =1895 ;       ENTRY 1  POINTER TO THE LIST OF OPTIONS APPLICABLE TO THIS COMMAND
               =1896 ;       ENTRY 2  NUMBER OF NUMERIC PARAMETERS REQUIRED BY THE COMMAND
               =1897 ;
0023           =1898 CTAB    EQU     $ AND 0FFH
0003           =1899 COMSIZ  EQU     3
               =1900 ;
0323 1F        =1901         DB      KEYMOD,LOW OPTAB1,1     ;EXAM
0324 3F        =
0325 01        =
0326 1E        =1902         DB      KEYGO,LOW OPTAB3,1      ;GO
0327 49        =
0328 01        =
0329 10        =1903         DB      KEYFIL,LOW OPTAB1,3     ;FILL
032A 3F        =
032B 03        =
032C 1C        =1904         DB      KEYLST,LOW OPTAB1,2     ;DUMP
032D 3F        =
032E 02        =
032F 18        =1905         DB      KEYREC,LOW OPTAB1,2     ;RECORD
0330 3F        =
0331 02        =
0332 14        =1906         DB      KEYREL,LOW OPTAB1,0     ;RELOAD
0333 3F        =
0334 00        =
0335 0C        =1907         DB      KSETB,LOW OPTAB2,1      ;SETBRK
0336 46        =
0337 01        =
0338 0C        =1908         DB      KCLRB,LOW OPTAB2,1      ;CLRBRK
0339 46        =
033A 01        =
033B 1D        =1909         DB      KGORES,LOW OPTAB3,0     ;GO FROM RESET STATE
033C 49        =
033D 00        =
033E FF        =1910         DB      0FFH                    ;ESCOP
               =1911 ;
               =1912 $EJECT
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

                =1913 ;
                =1914 ;       THE OPTION TABLE GIVES THE VARIOUS OPTIONS ALLOWED FOR EACH
                =1915 ;       BASIC COMMAND, AS FOLLOWS:
                =1916 ;
                =1917 ;       ENTRY 0:  START OF TABLE OF MODIFIER RESPONSES.
                =1918 ;       ENTRY 1+:  ALLOWED MODIFIER KEYSTROKES CORRESPONDING TO OPTIONS 0-5.
                =1919 ;                NOTE THAT THE LAST BYTE IN EACH OPTION GROUP HAS BIT
                =1920 ;                SEVEN SET TO INDICATE THE END.
                =1921 ;
033F 26         =1922 OPTAB1: DB       STRMEM
0340 1A         =1923         DB       KEYPM,KEYDM,KEYREG,RINT
0341 16         =
0342 1B         =
0343 11         =
0344 13         =1924         DB       PBRK,DBRK OR 80H
0345 95         =
0346 26         =1925 OPTAB2: DB       STRMEM
0347 1A         =1926         DB       KEYPM,KEYDM OR 80H
0348 96         =
0349 2C         =1927 OPTAB3: DB       STRGOC
034A 1B         =1928         DB       NOBRK,WBRK,SING
034B 16         =
034C 1A         =
034D 15         =1929         DB       KEYPAT,KEYTRA OR 80H
034E 99         =
                =1930         SIZECHK
002C            =1933+  SIZE  SET   44
                =1934+;
                =1935+;*****************************************************************
                =1944 $EJECT
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

                =1945          CODEBLK 130
0100            =1955+           ORG     256
                =1959 ;OUTUTL OUTPUT ONE OF FOUR UTILITY DISPLAY PROMPTS (LEFT JUSTIFIED)
                =1960 ;       ACCORDING TO ACC CONTENTS (0-3).
                =1961 ;OUTCLR CLEAR DISPLAY AND OUTPUT CHARACTER STRING STARTING
                =1962 ;       AT THE ADDRESS POINTED TO BY BYTE AT ADDRESS IN ACCUMULATOR.
                =1963 ;OUTMSG SUBROUTINE TO COPY A STRING OF BIT PATTERNS FROM ROM TO THE
                =1964 ;       DISPLAY REGISTERS.
                =1965 ;       STRING SELECTED IS DETERMINED BY ACC WHEN CALLED.
                =1966 ;       ON ENTERING OUTMSG, ACC CONTENTS ARE USED TO ADDRESS A BYTE IN A
                =1967 ;       LOOKUP TABLE ON THE CURRENT PAGE WHICH CONTAINS THE ADDRESS OF
                =1968 ;       A STRING OF SEGMENT PATTERN DATA BYTES TO BE PRINTED ONTO THE
                =1969 ;       DISPLAY.
                =1970 ;       THE END OF THE STRING IS INDICATED WHEN BIT7 =1.
                =1971 ;       CALLS SUBROUTINE 'WDISP'
                =1972 ;       TO ACTUALLY EFFECT WRITING INTO THE DISPLAY REGISTERS.
0100 0319       =1973 OUTUTL: ADD     A,#STRUTL
0102 B4F1       =1974 OUTCLR: CALL    CLEAR
0104 A3         =1975 OUTMSG: MOVP    A,@A
                =1976         MMOV    STRTMP,A
0105 B940       =1989+          MOV     R1,#STRTMP
0107 A1         =1990+          MOV     @R1,A
                =1994 PRNT2:  MMOV    A,STRTMP        ;LOAD NEXT CHARACTER LOCATION
0108 B940       =2003+          MOV     R1,#STRTMP
010A F1         =2004+          MOV     A,@R1
010B A3         =2008         MOVP    A,@A            ;LOAD BIT PATTERN INDIRECT
010C F217       =2009         JB7     PRNT1
010E D4D8       =2010         CALL    WDISP           ;OUTPUT TO NEXT CHARACTER POSITION
                =2011         MINC    STRTMP          ;INDEX POINTER
0110 B940       =2016+          MOV     R1,#STRTMP
0112 F1         =2017+          MOV     A,@R1
0113 17         =2021+          INC     A
0114 A1         =2026+          MOV     @R1,A
0115 2408       =2029         JMP     PRNT2
0117 C4D8       =2030 PRNT1:  JMP     WDISP           ;DONE
                =2031 ;
0019            =2032 STRUTL  EQU     LOW $
0119 31         =2033         DB      LOW(DERROR)     ;UTILITY MESSAGE 0 ADDRESS
011A 37         =2034         DB      LOW(DSGNON)     ;UTILITY MESSAGE 1 ADDRESS
011B 3E         =2035         DB      LOW(DRUN)       ;UTILITY MESSAGE 2 ADDRESS
011C 44         =2036         DB      LOW(DBPNT)      ;UTILITY MESSAGE 3 ADDRESS
001D            =2037 STRCOM  EQU     LOW $
011D 46         =2038         DB      LOW(DMOD)       ;BASIC COMMAND 0 RESPONSE ADDRESS
011E 49         =2039         DB      LOW(DGO)        ;BASIC COMMAND 1 RESPONSE ADDRESS
011F 4B         =2040         DB      LOW(DFILL)      ;BASIC COMMAND 2 RESPONSE ADDRESS
0120 4E         =2041         DB      LOW(DLST)       ;BASIC COMMAND 3 RESPONSE ADDRESS
0121 51         =2042         DB      LOW(DREC)       ;BASIC COMMAND 4 RESPONSE ADDRESS
0122 54         =2043         DB      LOW(DREL)       ;BASIC COMMAND 5 RESPONSE ADDRESS
0123 57         =2044         DB      LOW(DSB)        ;BASIC COMMAND 6 RESPONSE ADDRESS
0124 5A         =2045         DB      LOW(DCB)        ;BASIC COMMAND 7 RESPONSE ADDRESS
0125 5D         =2046         DB      LOW(DGR)        ;BASIC COMMAND 8 RESPONSE ADDRESS
0026            =2047 STRMEM  EQU     LOW $
0126 5F         =2048         DB      LOW(DPRMEM)     ;DATA TYPE MODIFIER 0 RESPONSE ADDRESS
0127 61         =2049         DB      LOW(DDAMEM)     ;DATA TYPE MODIFIER 1 RESPONSE ADDRESS
0128 63         =2050         DB      LOW(DRM)        ;DATA TYPE MODIFIER 2 RESPONSE ADDRESS
```

```
LOC  OBJ       LINE       SOURCE STATEMENT

0129 69        =2051        DB      LOW(DINTRG)     ;DATA TYPE MODIFIER 3 RESPONSE ADDRESS
012A 65        =2052        DB      LOW(DPRERK)     ;DATA TYPE MODIFIER 4 RESPONSE ADDRESS
012B 67        =2053        DB      LOW(DDABRK)     ;DATA TYPE MODIFIER 5 RESPONSE ADDRESS
002C           =2054 STRGOC EQU     LOW $
012C 6B        =2055        DB      LOW(DNOBRK)     ;EXECUTION MODE MODIFIER 0
012D 6D        =2056        DB      LOW(DWBRK)      ;EXECUTION MODE MODIFIER 1
012E 6F        =2057        DB      LOW(DSS)        ;EXECUTION MODE MODIFIER 2
012F 72        =2058        DB      LOW(DPA)        ;EXECUTION MODE MODIFIER 3
0130 75        =2059        DB      LOW(DTR)        ;EXECUTION MODE MODIFIER 4
               =2060 ;
               =2061 ;      UTILITY OUTPUT MESSAGES
               =2062 ;
               =2063 DERROR:
0131 79        =2064        DB      01111001B       ;"E"
0132 50        =2065        DB      01010000B       ;"R"
0133 50        =2066        DB      01010000B       ;"R"
0134 5C        =2067        DB      01011100B       ;"O"
0135 50        =2068        DB      01010000B       ;"R"
0136 C0        =2069        DB      11000000B       ;"-."
               =2070 DSGNON:
0137 00        =2071        DB      00000000B       ;" "
0138 76        =2072        DB      01110110B       ;"H"
0139 6D        =2073        DB      01101101B       ;"S"
013A 79        =2074        DB      01111001B       ;"E"
013B 40        =2075        DB      01000000B       ;"-"
013C 66        =2076        DB      01100110B       ;"4"
013D E7        =2077        DB      11100111B       ;"9."(TM)
               =2078 DRUN:
013E 00        =2079        DB      00000000B       ;" "
013F 40        =2080        DB      01000000B       ;"-"
0140 50        =2081        DB      01010000B       ;"R"
0141 1C        =2082        DB      00011100B       ;"U"
0142 54        =2083        DB      01010100B       ;"N"
0143 C0        =2084        DB      11000000B       ;"-."
               =2085 DBMNT:
0144 73        =2086        DB      01110011B       ;"P"
0145 B9        =2087        DB      10111001B       ;"C."
               =2088 $EJECT
```

```
LOC  OBJ        LINE        SOURCE STATEMENT

                =2089 ;
                =2090 ;       PRIMARY COMMAND RESPONSE STRING PATTERNS
                =2091 ;
                =2092 DMOD:
0146 79         =2093         DB      01111001B,00111001B,11110100B   ;"ECH."
0147 39         =
0148 F4         =
                =2094 DGO:
0149 3D         =2095         DB      00111101B,11011100B             ;"GO."
014A DC         =
                =2096 DFILL:
014B 71         =2097         DB      01110001B,00110000B,10111000B   ;"FIL."
014C 30         =
014D B8         =
                =2098 DLST:
014E 38         =2099         DB      00111000B,01101101B,11111000B   ;"LST."
014F 6D         =
0150 F8         =
                =2100 DREC:
0151 3E         =2101         DB      00111110B,01110011B,10111000B   ;"UPL."
0152 73         =
0153 B8         =
                =2102 DREL:
0154 5E         =2103         DB      01011110B,01010100B,10111000B   ;"DNL."
0155 54         =
0156 B8         =
                =2104 DSB:
0157 6D         =2105         DB      01101101B,01111000B,11111100B   ;"STB."
0158 78         =
0159 FC         =
                =2106 DCB:
015A 39         =2107         DB      00111001B,00111000B,11111100B   ;"CLB."
015B 38         =
015C FC         =
                =2108 DGR:
015D 3D         =2109         DB      00111101B,11010000B             ;"GR."
015E D0         =
                =2110 $EJECT
```

```
LOC  OBJ        LINE        SOURCE STATEMENT

                =2111 ;
                =2112 ;       MEMORY SPACE MODIFIER OPTION RESPONSE STRINGS:
                =2113 ;
                =2114 DFRMEM:
015F 73         =2115         DB      01110011B,11010000B             ,"PR."
0160 D0         =
                =2116 DDAMEM:
0161 5E         =2117         DB      01011110B,11110111B             ;"DA."
0162 F7         =
                =2118 DRM:
0163 50         =2119         DB      01010000B,10111101B             ,"RG."
0164 BD         =
                =2120 DFRBRK:
0165 73         =2121         DB      01110011B,11111100B             ;"PB."
0166 FC         =
                =2122 DDABRK:
0167 5E         =2123         DB      01011110B,11111100B             ;"DB."
0168 FC         =
                =2124 DINTRG:
0169 76         =2125         DB      01110110B,11010000B             ;"HR."
016A D0         =
                =2126 ;
                =2127 ;       RESPONSE MESSAGES FOR GO CONDITION MODIFIERS.
                =2128 ;
                =2129 DNOBRK:
016B 54         =2130         DB              01010100B,11111100B     ;"NB."
016C FC         =
                =2131 DWBRK:
016D 7C         =2132         DB      01111100B,11010000B             ;"BR."
016E D0         =
                =2133 DSS:
016F 6D         =2134         DB      01101101B,01101101B,11111000B   ;"SST."
0170 6D         =
0171 F8         =
                =2135 DPA:
0172 77         =2136         DB      01110111B,01111100B,11010000B   ;"ABR."
0173 7C         =
0174 D0         =
                =2137 DTR:
0175 77         =2138         DB      01110111B,01101101B,11111000B   ;"AST."
0176 6D         =
0177 F8         =
                =2139 ;
                =2140         SIZECHK
0078            =2143+  SIZE  SET   120
                =2144+;
                =2145+;*****************************************************************
                =2154 $EJECT
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

                =2155        CODEBLK 45
00C0            =2160+         ORG     192
                =2164 ;INPADR INPUT DATA INTO TWO-BYTE PARAMETER BUFFER INDICATED BY R0.
                =2165 ;       RECEIVE NUMERIC KEYS FROM KEYBOARD UNTIL ',' OR '.'.
                =2166 ;       SHIFT INTO ADDRESS BUFFER;
                =2167 ;       RE-WRITE DISPLAY.
                =2168 ;       IF NUMBER OF CONSTANTS NEEDED IS ZERO, NO NEW PARAMETERS ARE ALLOWED.
                =2169 ;
00C0 97         =2170 INPADR: CLR     C
00C1 A7         =2171         CPL     C
                =2172         MMOV    A,NUMCON
00C2 B938       =2181+          MOV     R1,#NUMCON
00C4 F1         =2182+          MOV     A,@R1
00C5 C6D7       =2186         JZ      ELSIF1
00C7 FB         =2187 INPAD1: MOV     A,KEY
00C8 92D7       =2188         JB4     ELSIF1
00CA 20         =2189         XCH     A,@R0
00CB 47         =2190         SWAP    A
00CC 20         =2191         XCH     A,@R0
00CD 30         =2192         XCHD    A,@R0
00CE 18         =2193         INC     R0
00CF 30         =2194         XCHD    A,@R0
00D0 3478       =2195         CALL    UPDADR
00D2 14EC       =2196         CALL    INPKEY
00D4 97         =2197         CLR     C
00D5 04C7       =2198         JMP     INPAD1
                =2199 ;
                =2200 ;ELSIF1 IF KEY=',' OR '.' THEN RETURN.
                =2201 ;
00D7 FB         =2202 ELSIF1: MOV     A,KEY
00D8 D312       =2203         XRL     A,#KEYNXT
00DA C6E5       =2204         JZ      ELSIF2
00DC FB         =2205         MOV     A,KEY
00DD D313       =2206         XRL     A,#KEYEND
00DF C6E5       =2207         JZ      ELSIF2
                =2208 ;
                =2209 ;       ELSE GOTO PERROR.
                =2210 ;
00E1 BA02       =2211         MOV     LDATA,#2
00E3 249A       =2212         JMP     PERROR
00E5 B846       =2213 ELSIF2: MOV     R0,#SEGMAP
00E7 B903       =2214         MOV     R1,#3
00E9 B4F5       =2215         CALL    DBLANK
00EB 83         =2216         RET
                =2217         SIZECHK
002C            =2220+  SIZE  SET   44
                =2221+;
                =2222+;*****************************************************************
                =2231 $EJECT
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

                =2232          CODEBLK 35
0178            =2242+           ORG     376
                =2246 ;UPDADR UPDATE ADDRESS FIELD
                =2247 ;        (LAST THREE CHARACTERS OF DISPLAY) WITH ADDRESS BUFFER
                =2248 UPDADR:  MMOV    NEXTPL,PLUS3
0178 B93A       =2259+           MOV     R1,#NEXTPL
017A B103       =2260+           MOV     @R1,#PLUS3
                =2264 ;        WRITE ADDR INTO NEXT THREE BUFFER LOCATIONS.
017C F0         =2265 UPDAD1:  MOV     A,@R0
017D C8         =2266          DEC     R0
017E 530F       =2267          ANL     A,#0FH
0180 968E       =2268          JNZ     DSPHI
0182 D4D8       =2269          CALL    WDISP
0184 F0         =2270          MOV     A,@R0
0185 47         =2271          SWAP    A
0186 530F       =2272          ANL     A,#0FH
0188 9692       =2273          JNZ     DSPM1
018A D4D8       =2274          CALL    WDISP
018C 2494       =2275          JMP     DSPLO
018E D4D3       =2276 DSPHI.   CALL    DSPACC
0190 F0         =2277 DSPMID.  MOV     A,@R0
0191 47         =2278          SWAP    A
0192 D4D3       =2279 DSPM1    CALL    DSPACC
0194 F0         =2280 DSPLO:   MOV     A,@R0
0195 D4D3       =2281          CALL    DSPACC
0197 83         =2282          RET
                =2283          SIZECHK
0020            =2286+  SIZE   SET   32
                =2287+;
                =2288+;*****************************************************************
                =2297 $EJECT
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

                =2298         CODEBLK 35
0198            =2308+          ORG     408
                =2312 ;PERROR:        REPEAT
                =2313 ;         OUTPUT_MESSAGE(PERROR_PROMPT)
                =2314 ;         OUTPUT(LDATA)
                =2315 ;         CALL INPUT_BYTE(KEY)
                =2316 ;       UNTIL KEY='CLEAR/PREVIOUS'
019C BA04       =2317 RERROR: MOV     LDATA,#4
019A BF02       =2318 PERROR: MOV     XPCODE,#2
019C 74D1       =2319         CALL    XPTEST
019E 27         =2320         CLR     A
019F D7         =2321         MOV     PSW,A
01A0 FB         =2322         MOV     A,KEY
01A1 D317       =2323         XRL     A,#KEYCLR
01A3 C6D6       =2324         JZ      ERROR2
01A5 27         =2325         CLR     A
01A6 3400       =2326         CALL    OUTUTL
01A8 FA         =2327         MOV     A,LDATA
01A9 D4D3       =2328         CALL    DSPACC
                =2329         MMOV    KBDBUF,NEG1
01AC B93B       =2340+          MOV     R1,#KBDBUF
01AD B1FF       =2341+          MOV     @R1,#NEG1
01AF 14EC       =2345         CALL    INPKEY
01B1 FB         =2346         MOV     A,KEY
01B2 D313       =2347         XRL     A,#KEYEND
01B4 9698       =2348         JNZ     RERROR
01B6 0429       =2349 ERROR2: JMP     MAIN
                =2350         SIZECHK
0020            =2353+  SIZE  SET   $2
                =2354+;
                =2355+;*************************************************************
                =2364 ;
                =2365         CODEBLK 80
0200            =2380+          ORG     512
                =2384 ;IMPLEM IMPLEMENT COMMAND
0200 2306       =2385 IMPLEM: MOV     A,#LOW(JMPTBL)
                =2386         MADD    A,BCODE
0202 B936       =2392+          MOV     R1,#BCODE
0204 61         =2393+          ADD     A,@R1
0205 B3         =2397         JMPP    @A
                =2398 ;
                =2399 JMPTBL:
0206 0F         =2400         DB      LOW(JTOMOD)
0207 20         =2401         DB      LOW(JTOGO)
0208 22         =2402         DB      LOW(JTOFIL)
0209 1A         =2403         DB      LOW(JTOLST)
020A 11         =2404         DB      LOW(JTOREC)
020B 16         =2405         DB      LOW(JTOREL)
020C 2C         =2406         DB      LOW(COMSBR)
020D 28         =2407         DB      LOW(COMCBR)
020E 26         =2408         DB      LOW(JGORES)
                =2409 ;
020F 444F       =2410 JTOMOD: JMP     EXAMIN
                =2411 ;
0211 85         =2412 JTOREC: CLR     F0      ;F0=0 ==> HEX FORMAT DATA DUMP
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

0212 B472       =2413          CALL    HFILEO
0214 0429       =2414          JMP     MAIN
                =2415 ;
0216 5497       =2416 JTOREL   CALL    HRECIN
0218 0429       =2417          JMP     MAIN
                =2418 ;
021A 85         =2419 JTOLST:  CLR     F0
021B 95         =2420          CPL     F0
021C B472       =2421          CALL    HFILEO
021E 0429       =2422          JMP     MAIN
                =2423 ;
0220 8400       =2424 JTOGO:   JMP     EPRUN
                =2425 ;
0222 54E5       =2426 JTOFIL:  CALL    COMFIL
0224 0429       =2427          JMP     MAIN
                =2428 ;
0226 8461       =2429 JGORES:  JMP     COMGOR
                =2430 ;
                =2431 ;COMCBR COMMAND TO CLEAR BREAKPOINTS
0228 BA00       =2432 COMCBR:  MOV     LDATA,#0
022A 442E       =2433          JMP     BRKFIL
                =2434 ;
                =2435 ;COMSBR COMMAND TO SET BREAKPOINTS
022C BA01       =2436 COMSBR:  MOV     LDATA,#1
022E 2304       =2437 BRKFIL:  MOV     A,#4
                =2438          MADD    TYPE,A
0230 B937       =2448+           MOV     R1,#TYPE
0232 61         =2449+           ADD     A,@R1
0233 A1         =2455+           MOV     @R1,A
0234 F400       =2459 BRKNXT:  CALL    LSTORE
0236 FB         =2460          MOV     A,KEY
0237 D313       =2461          XRL     A,#KEYEND
0239 C64D       =2462          JZ      BRKEND
023B 14EC       =2463          CALL    INPKEY
                =2464          MMOV    NUMCON,PLUS1
023D B938       =2475+           MOV     R1,#NUMCON
023F D101       =2476+           MOV     @R1,#PLUS1
0241 B830       =2480          MOV     R0,#SMALO
0243 B000       =2481          MOV     @R0,#0
                =2482          MMOV    SMAHI,ZERO
0245 B931       =2493+           MOV     R1,#SMAHI
0247 B100       =2494+           MOV     @R1,#ZERO
0249 14C0       =2498          CALL    INPADR
024B E634       =2499          JNC     BRKNXT
024D 0429       =2500 BRKEND:  JMP     MAIN
                =2501          SIZECHK
004F            =2504+  SIZE   SET   79
                =2505+;
                =2506+;*****************************************************************
                =2515 $EJECT
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

                =2516         CODEBLK 75
024F            =2531+          ORG     591
                =2535 ;EXAMIN EXAMINE/MODIFY MEMORY COMMAND.
                =2536 ;       DISPLAYS MEMORY ADDRESS SPACE OPTION, ADDRESS VALUE, AND CURRENT DATA
                =2537 ;       READS KEYBOARD AND INTERPRETS RESPONSE.
                =2538 ;
                =2539 ;       OUTPUT_MESSAGE(<MEMORY_SPACE_OPTION><SMA>'='<DATA_BYTE>)
024F 85         =2540 EXAMIN: CLR     F0
                =2541 EXAM0:  MMOV    A,TYPE
0250 B937       =2550+          MOV     R1,#TYPE
0252 F1         =2551+          MOV     A,@R1
0253 0326       =2555         ADD     A,#STRMEM       ;OFFSET FOR FIRST MEMORY TYPE STRING
0255 3402       =2556         CALL    OUTCLR
0257 B831       =2557         MOV     R0,#SMALO+1
0259 347C       =2558         CALL    UPDAD1
025B 2348       =2559         MOV     A,#01001000B    ;'='
025D D4D8       =2560         CALL    WDISP
025F 14FC       =2561         CALL    LFETCH
0261 FA         =2562         MOV     A,LDATA
0262 47         =2563         SWAP    A
0263 D4D3       =2564         CALL    DSPACC
0265 FA         =2565         MOV     A,LDATA
0266 D4D3       =2566         CALL    DSPACC
                =2567 ;
                =2568 ;
                =2569 ;       INPUT_KEY(KEY)
                =2570 ;       IF (KEY IS NOT NUMERIC)
                =2571 ;               IF (KEY=KEYEND) GO TO PARSER
                =2572 ;               ELSEIF (KEY=KEYNEXT)
                =2573 ;                       INCREMENT <SMA>
                =2574 ;                       GOTO EXAMIN
                =2575 ;               ELSEIF (KEY=KEYPREVIOUS)
                =2576 ;                       DECREMENT <SMA>
                =2577 ;                       GOTO EXAMIN
                =2578 ;               ELSE GOTO PERROR
                =2579 ;
0268 14EC       =2580         CALL    INPKEY
                =2581         MMOV    A,KEY
026A FB         =2597+          MOV     A,KEY
026B 927B       =2601         JB4     EXAM1
                =2602 ;
                =2603 ;       APPEND DATA WITH <LOWNIB_<KEY>>
                =2604 ;       CALL LSTORE
                =2605 ;       GOTO EXAMIN
                =2606 ;
026D FA         =2607         MOV     A,LDATA
026E 47         =2608         SWAP    A
026F 53F0       =2609         ANL     A,#0F0H
0271 B675       =2610         JF0     EXAM5
0273 27         =2611         CLR     A
0274 95         =2612         CPL     F0
0275 6B         =2613 EXAM5:  ADD     A,KEY
0276 AA         =2614         MOV     LDATA,A
0277 F400       =2615         CALL    LSTORE
0279 4450       =2616         JMP     EXAM0
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

                =2617 ;
027B D313       =2618 EXAM1:  XRL     A,#(KEYEND)
027D 9681       =2619         JNZ     EXAM2
027F 0429       =2620         JMP     MAIN
                =2621 ;
0281 FB         =2622 EXAM2:  MOV     A,KEY
0282 D312       =2623         XRL     A,#KEYNXT
0284 968A       =2624         JNZ     EXAM3
0286 34F2       =2625         CALL    INCSMA
0288 444F       =2626         JMP     EXAMIN
028A FB         =2627 EXAM3:  MOV     A,KEY
028B D317       =2628         XRL     A,#KEYCLR
028D 9693       =2629         JNZ     EXAM4
028F 54F4       =2630         CALL    DECSMA
0291 444F       =2631         JMP     EXAMIN
0293 B003       =2632 EXAM4:  MOV     LDATA,#03H
0295 249A       =2633         JMP     PERROR
                =2634         SIZECHK
0048            =2637+  SIZE  SET   72
                =2638+;
                =2639+;*******************************************************
                =2648 ;
                =2649         CODEBLK 4
00EC            =2654+          ORG     236
00EC D4C2       =2658 INPKEY: CALL    KBDIN   ; RETURNS KEY DEPRESSION IN A
00EE AB         =2659         MOV     KEY,A
00EF 83         =2660         RET
                =2661         SIZECHK
0004            =2664+  SIZE  SET   4
                =2665+;
                =2666+;*******************************************************
                =2675 $EJECT
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

                 2676 $      INCLUDE(:F0:GOCOMS.MOD)
                =2677        CODEBLK 210
0400            =2697+         ORG     1024
                =2701 ;EPRUN RUN EMULATION MODE.
                =2702 ;      RELOAD EP WITH SYSTEM STATUS AND RELEASE.
                =2703 ;      SEQUENCE IS AS FOLLOWS:
                =2704 ;      IF COMMAND WAS TERMINATED BY THE 'NEXT' KEY:
                =2705 ;          STORE SMA INTO EP PC;
                =2706 ;      STORE EP PC INTO TOP-OF-STACK (RELATIVE TO EP PSW);
                =2707 ;      PASS EP R0;
                =2708 ;      PASS EP PSW;
                =2709 ;      PASS EP TIMER;
                =2710 ;      PASS EP ACCUMULATOR;
                =2711 ;
0400 2302       =2712 EPRUN: MOV     A,#2
0402 3400       =2713        CALL    OUTUTL
                =2714        MMOV    A,NUMCON
0404 B938       =2723+         MOV     R1,#NUMCON
0406 F1         =2724+         MOV     A,@R1
0407 9615       =2728        JNZ     EPCONT
                =2729        MMOV    EPPCLO,SMALO
0409 B930       =2745+         MOV     R1,#SMALO
040B F1         =2746+         MOV     A,@R1
040C B924       =2752+         MOV     R1,#EPPCLO
040E A1         =2753+         MOV     @R1,A
                =2756        MMOV    EPPCHI,SMAHI
040F B931       =2772+         MOV     R1,#SMAHI
0411 F1         =2773+         MOV     A,@R1
0412 B925       =2779+         MOV     R1,#EPPCHI
0414 A1         =2780+         MOV     @R1,A
0415 FB         =2783 EPCONT: MOV     A,KEY
0416 D312       =2784        XRL     A,#KEYNXT
0418 C61F       =2785        JZ      EPCON1
041A 2301       =2786        MOV     A,#01H  ;STACK ONE LEVEL DEEP TO HOLD USER STARTING ADDRESS
                =2787        MMOV    EPPSW,A
041C B921       =2800+         MOV     R1,#EPPSW
041E A1         =2801+         MOV     @R1,A
                =2805 EPCON1: MMOV    LDATA,EPPCLO
041F B924       =2821+         MOV     R1,#EPPCLO
0421 F1         =2822+         MOV     A,@R1
0422 AA         =2835+         MOV     LDATA,A
                =2838        MMOV    A,EPPSW
0423 B921       =2847+         MOV     R1,#EPPSW
0425 F1         =2848+         MOV     A,@R1
0426 07         =2852        DEC     A
0427 5307       =2853        ANL     A,#07H
0429 E7         =2854        RL      A
042A 0308       =2855        ADD     A,#08H
                =2856        MMOV    SMALO,A
042C B930       =2869+         MOV     R1,#SMALO
042E A1         =2870+         MOV     @R1,A
042F F4C3       =2874        CALL    EPSTOR
                =2875        MINC    SMALO
0431 B930       =2880+         MOV     R1,#SMALO
0433 F1         =2881+         MOV     A,@R1
```

```
LOC  OBJ       LINE       SOURCE STATEMENT

0434 17       =2885+          INC     A
0435 A1       =2890+          MOV     @R1,A
              =2893          MMOV    A,EPPSW
0436 B921     =2902+          MOV     R1,#EPPSW
0438 F1       =2903+          MOV     A,@R1
0439 53F0     =2907          ANL     A,#0F0H
              =2908          MORL    A,EPPCHI
043B B925     =2914+          MOV     R1,#EPPCHI
043D 41       =2915+          ORL     A,@R1
043E AA       =2919          MOV     LDATA,A
043F F4C3     =2920          CALL    EPSTOR
0441 B8D1     =2921 EPCNT:   MOV     R0,#LOW(OV2BAS+OVSIZE)
0443 746A     =2922          CALL    OVLOAD
              =2923          MMOV    A,EPR0
0445 B923     =2932+          MOV     R1,#EPR0
0447 F1       =2933+          MOV     A,@R1
0448 F4D0     =2937          CALL    EPPASS
              =2938          MMOV    A,EPPSW
044A B921     =2947+          MOV     R1,#EPPSW
044C F1       =2948+          MOV     A,@R1
044D F4D0     =2952          CALL    EPPASS
              =2953          MMOV    A,EPTIMR
044F B922     =2962+          MOV     R1,#EPTIMR
0451 F1       =2963+          MOV     A,@R1
0452 F4D0     =2967          CALL    EPPASS
              =2968          MMOV    A,EPACC
0454 B920     =2977+          MOV     R1,#EPACC
0456 F1       =2978+          MOV     A,@R1
0457 F4D0     =2982          CALL    EPPASS
0459 8903     =2983          ORL     P1,#00000011B
045B F4DB     =2984          CALL    EPSTEP
045D 745A     =2985          CALL    OVSWAP
045F 846B     =2986          JMP     CGO
              =2987 ;
              =2988 ;COMGOR GO FROM RESET COMMAND
              =2989 ;       RESET PROCESSOR
              =2990 ;       RELOAD LOW ORDER PROGRAM BYTES INTO PROGRAM MEMORY
              =2991 ;
0461 2302     =2992 COMGOR:  MOV     A,#2
0463 3400     =2993          CALL    OUTUTL
0465 8910     =2994          ORL     P1,#EPRSET
0467 745A     =2995          CALL    OVSWAP
0469 99EF     =2996          ANL     P1,#(NOT EPRSET)
              =2997 ;
              =2998 ;
              =2999 ;CGO    SET UP BREAK LOGIC FOR APPROPRIATE BREAK CONDITIONS,
              =3000 ;       DEPENDING ON CONTENTS OF 'TYPE'.
              =3001 ;
              =3002 CGO:     MMOV    A,TYPE
046B B937     =3011+          MOV     R1,#TYPE
046D F1       =3012+          MOV     A,@R1
046E 0371     =3016          ADD     A,#LOW GOTBL
0470 B3       =3017          JMPP    @A
              =3018 ;
0471 7C       =3019 GOTBL:   DB      LOW(CGONB)
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

0472 76         =3020         DB      LOW(CGOWB)
0473 80         =3021         DB      LOW(CGOSS)
0474 76         =3022         DB      LOW(CGOPAT)
0475 80         =3023         DB      LOW(CGOTRA)
                =3024 ;
                =3025 CGOPAT:
0476 99FD       =3026 CGOWB:  ANL     P1,#NOT 00000010B
0478 8901       =3027         ORL     P1,#00000001B
047A 8482       =3028         JMP     EPRUN4
                =3029 ;
047C 99FC       =3030 CGONB:  ANL     P1,#NOT 00000011B
047E 8482       =3031         JMP     EPRUN4
                =3032 ;
                =3033 CGOTRA:
0480 8903       =3034 CGOSS:  ORL     P1,#00000011B
                =3035 ;
                =3036 ;EPRUN4 SET UP CONTROL LOGIC TO RUN USER'S PROGRAM.
                =3037 ;       RELEASE PROCESSOR TO RUN.
                =3038 ;
0482 8A20       =3039 EPRUN4: ORL     P2,#00100000B          ;DISABLE EP LINK REFERENCES.
0484 9AEF       =3040         ANL     P2,#NOT 00010000B      ;SET ALL REFERENCES TO RAM ARRAY.
0486 99DF       =3041         ANL     P1,#NOT MODOUT
0488 F4F4       =3042         CALL    EPREL
                =3043 ;
                =3044 ;       WAIT FOR KEYSTROKE INPUT OR HARDWARE BREAK TO OCCUR.
                =3045 ;
048A F4AC       =3046 EPRUN1: CALL    IOFPOL
048C F4AF       =3047         CALL    KBDPOL
048E 37         =3048         CPL     A
048F F295       =3049         JB7     EPRUN3
0491 8699       =3050         JNI     EPRUN2
0493 848A       =3051         JMP     EPRUN1
                =3052 ;
                =3053 ;EPRUN3 A KEYSTROKE WAS DETECTED WHILE EP WAS RUNNING.
                =3054 ;       BREAK EXECUTION,
                =3055 ;       PROCESS KEYSTROKE.
0495 B400       =3056 EPRUN3: CALL    STSAVE
0497 84B3       =3057         JMP     EPRUN5
                =3058 ;
                =3059 ;EPRUN2 AN ENABLED BREAK CONDITION OCCURRED.
                =3060 ;       BREAK EMULATION MODE,
                =3061 ;       CONTINUE ACCORDING TO GO COMMAND TYPE.
0499 B400       =3062 EPRUN2: CALL    STSAVE
                =3063         MMOV    A,TYPE
                =3072+          MOV     R1,#TYPE
049D F1         =3073+          MOV     A,@R1
049E 03A1       =3077         ADD     A,#LOW CNTTBL
04A0 B3         =3078         JMPP    @A
                =3079 ;
04A1 A6         =3080 CNTTBL: DB      LOW(BRKERR)
04A2 BA         =3081         DB      LOW(EPRUN6)
04A3 BA         =3082         DB      LOW(EPRUN6)
04A4 AA         =3083         DB      LOW(CNTTRA)
04A5 AA         =3084         DB      LOW(CNTTRA)
                =3085 ;
```

```
LOC  OBJ        LINE        SOURCE STATEMENT

                =3086 ;BRKERR BREAKPOINT LATCH WAS SET THOUGH BREAKPOINTS NOT ENABLED.
                =3087 ;       DISPLAY HARDWARE ERROR MESSAGE.
04A6 BA0B       =3088 BRKERR: MOV     LDATA,#0BH
04A8 249A       =3089         JMP     PERROR
                =3090 ;
                =3091 CNTTRA: MMOV    A,DSPTIM
04AA D928       =3100+          MOV     R1,#DSPTIM
04AC F1         =3101+          MOV     A,@R1
04AD 94F2       =3105         CALL    DELAY
04AF F4AF       =3106         CALL    KBDPOL
04B1 F241       =3107         JB7     EPCNT           ;B7 SET INDICATES NO KEYSTROKE.
                =3108 ;
                =3109 ;EPRUN5 INPUT(KEY),
                =3110 ;       IF KEY=END GO TO PARSER,
                =3111 ;       INPUT KEY,
                =3112 ;       IF KEY<>NEXT GO TO PARSER,
                =3113 ;       CONTINUE IN SAME MODE.
                =3114 ;
04B3 14EC       =3115 EPRUN5: CALL    INPKEY
04B5 FB         =3116         MOV     A,KEY
04B6 D313       =3117         XRL     A,#KEYEND
04B8 96C7       =3118         JNZ     EPRET
04BA 14EC       =3119 EPRUN6: CALL    INPKEY
04BC FB         =3120         MOV     A,KEY
04BD D312       =3121         XRL     A,#KEYNXT
04BF 96C7       =3122         JNZ     EPRET
04C1 2302       =3123         MOV     A,#2
04C3 3400       =3124         CALL    OUTUTL
04C5 8441       =3125         JMP     EPCNT
                =3126 ;
                =3127 ;EPRET  EXECUTION MODE IS TO BE TERMINATED.
                =3128 ;       JUMP INTO PARSER TO INTERPRET KEY ALREADY DETECTED.
04C7 8433       =3129 EPRET:  JMP     MAIN2
                =3130 ;
                =3131         SIZECHK
00C9            =3134+  SIZE  SET   201
                =3135+;
                =313G+;*******************************************************
                =3145 $EJECT
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

                =3146         CODEBLK 115
0500            =3171+          ORG     1200
                =3175 ;STSAVE EP STATUS SAVE SUBROUTINE.
                =3176 ;       FORCE CALL TO LOC 014H;
                =3177 ;       SAVE EP ACC;
                =3178 ;       SAVE EP TIMER;
                =3179 ;       SAVE EP PSW;
                =3180 ;       SAVE EP R0;
                =3181 ;       SAVE EP TOP-OF-STACK IN EP PC;
                =3182 ;       RETURN.
0500 744F       =3183 STSAVE: CALL    EPBRK
0502 2303       =3184         MOV     A,#3
0504 3400       =3185         CALL    OUTUTL
0506 7450       =3186         CALL    OVSWAP
0508 B88F       =3187         MOV     R0,#LOW(OV0BAS+OVSIZE)
050A 746A       =3188         CALL    OVLOAD
050C 8A20       =3189         ORL     P2,#00100000B
050E 2314       =3190         MOV     A,#14H
0510 91         =3191         MOVX    @R1,A
0511 9ADF       =3192         ANL     P2,#NOT 00100000B
0513 8903       =3193         ORL     P1,#00000011B
0515 F4D8       =3194         CALL    EPSTEP
0517 8A20       =3195         ORL     P2,#00100000B
0519 9AEF       =3196         ANL     P2,#NOT 00010000B
051B 8903       =3197         ORL     P1,#(ENBRAM OR ENBLNK)
051D F4D8       =3198         CALL    EPSTEP
                =3199 ;
                =3200 ;       EXECUTION PROCESSOR IS NOW AT LOCATION 009H INTERNAL WITH
                =3201 ;       (RETURN ADDRESS+2) PUSHED ON STACK.
                =3202 ;
051F B8A5       =3203         MOV     R0,#LOW(OV3BAS+OVSIZE)
0521 746A       =3204         CALL    OVLOAD
0523 F4D0       =3205         CALL    EPPASS
                =3206         MMOV    EPACC,A
0525 B920       =3219+          MOV     R1,#EPACC
0527 A1         =3220+          MOV     @R1,A
0528 F4D0       =3224         CALL    EPPASS
                =3225         MMOV    EPTIMR,A
052A B922       =3238+          MOV     R1,#EPTIMR
052C A1         =3239+          MOV     @R1,A
052D F4D0       =3243         CALL    EPPASS
                =3244         MMOV    EPPSW,A
052F B921       =3257+          MOV     R1,#EPPSW
0531 A1         =3258+          MOV     @R1,A
0532 F4D0       =3262         CALL    EPPASS
                =3263         MMOV    EPR0,A
0534 B923       =3276+          MOV     R1,#EPR0
0536 A1         =3277+          MOV     @R1,A
0537 B8BB       =3281         MOV     R0,#LOW(OV1BAS+OVSIZE)
0539 746A       =3282         CALL    OVLOAD
                =3283         MMOV    A,EPPSW
053B B921       =3292+          MOV     R1,#EPPSW
053D F1         =3293+          MOV     A,@R1
053E 07         =3297         DEC     A
053F 5307       =3298         ANL     A,#07H
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

0541 E7         =3299        RL      A
0542 0308       =3300        ADD     A,#08H
                =3301        MMOV    SMALO,A
0544 B930       =3314+         MOV     R1,#SMALO
0546 A1         =3315+         MOV     @R1,A
0547 F4B7       =3319        CALL    EPFET
0549 03FE       =3320        ADD     A,#-2
054B AA         =3321        MOV     LDATA,A
                =3322        MMOV    EPPCLO,A
054C B924       =3335+         MOV     R1,#EPPCLO
054E A1         =3336+         MOV     @R1,A
054F F4C3       =3340        CALL    EPSTOR
0551 B930       =3341        MOV     R1,#SMALO
0553 11         =3342        INC     @R1
0554 F4B7       =3343        CALL    EPFET
0556 AA         =3344        MOV     LDATA,A
0557 53F0       =3345        ANL     A,#11110000B
0559 2A         =3346        XCH     A,LDATA
055A 13FF       =3347        ADDC    A,#-1
055C 530F       =3348        ANL     A,#00001111B
                =3349        MMOV    EPPCHI,A
055E B925       =3362+         MOV     R1,#EPPCHI
0560 A1         =3363+         MOV     @R1,A
0561 4A         =3367        ORL     A,LDATA
0562 AA         =3368        MOV     LDATA,A
0563 F4C3       =3369        CALL    EPSTOR
0565 B825       =3370        MOV     R0,#EPPCHI
0567 347C       =3371        CALL    UFDAD1
0569 2340       =3372        MOV     A,#01000000B    ;"-" FOR DISPLAY
056B D4D8       =3373        CALL    WDISP
056D B820       =3374        MOV     R0,#EPACC
056F 3490       =3375        CALL    DSPMID
0571 83         =3376        RET
                =3377        SIZECHK
0072            =3380+  SIZE  SET   114
                =3381+;
                =3382+;*******************************************************
                =3391 $EJECT
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

                 3392 $      INCLUDE(:F0:HFILE.MOD)
000D            =3393 CHARCR  EQU    0DH            ;<CR>
000A            =3394 CHARLF  EQU    0AH            ;<LF>
001A            =3395 CNTRLZ  EQU    1AH            ;CONTROL-Z
                =3396 ;
                =3397         CODEBLK 80
0297            =3412+          ORG    663
                =3416 ;HRECIN HEXFILE RECORD INPUT ROUTINE
0297 34CD       =3417 HRECIN: CALL   CHARIN
0299 D31A       =3418         XRL    A,#CNTRLZ
029B C6E0       =3419         JZ     DONE
029D D31A       =3420         XRL    A,#CNTRLZ
029F D33A       =3421         XRL    A,#(':')
02A1 9697       =3422         JNZ    HRECIN
                =3423         MMOV   CHKSUM,ZERO
02A3 BD00       =3428+          MOV    CHKSUM,#ZERO
02A5 14F0       =3432         CALL   BYTEIN
                =3433         MMOV   BUFCNT,A
02A7 B941       =3446+          MOV    R1,#BUFCNT
02A9 A1         =3447+          MOV    @R1,A
02AA 14F0       =3451         CALL   BYTEIN
                =3452         MMOV   SMAHI,A
02AC B931       =3465+          MOV    R1,#SMAHI
02AE A1         =3466+          MOV    @R1,A
02AF 14F0       =3470         CALL   BYTEIN
                =3471         MMOV   SMALO,A
02B1 B930       =3484+          MOV    R1,#SMALO
02B3 A1         =3485+          MOV    @R1,A
02B4 14F0       =3489         CALL   BYTEIN
                =3490         MMOV   RECTYP,A
02B6 B942       =3503+          MOV    R1,#RECTYP
02B8 A1         =3504+          MOV    @R1,A
                =3508 ;
                =3509 ;HDATIN HEX DATA BYTE IN
                =3510 HDATIN: MMOV   A,BUFCNT
02B9 B941       =3519+          MOV    R1,#BUFCNT
02BB F1         =3520+          MOV    A,@R1
02BC C6CC       =3524         JZ     RECDON
02BE 14F0       =3525         CALL   BYTEIN
02C0 AA         =3526         MOV    LDATA,A
02C1 F400       =3527         CALL   LSTORE
02C3 34F2       =3528         CALL   INCSMA
                =3529         MDEC   BUFCNT
02C5 B941       =3534+          MOV    R1,#BUFCNT
02C7 F1         =3535+          MOV    A,@R1
02C8 07         =3539+          DEC    A
02C9 A1         =3544+          MOV    @R1,A
02CA 44B9       =3547         JMP    HDATIN
                =3548 ;
02CC 34CD       =3549 RECDON: CALL   CHARIN
02CE D33F       =3550         XRL    A,#('?')
02D0 C6DB       =3551         JZ     CKSMOK
02D2 D33F       =3552         XRL    A,#('?')       ;SWITCH BACK TO DATA CHARACTER
02D4 34BA       =3553         CALL   NIBIN2         ;JOIN SUBROUTINE ALREADY IN PROGRESS
02D6 14F2       =3554         CALL   BYTEI1         ;DITTO
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

                =3555                               ;(RESULT FOR NON-'?' CHARACTERS IS AS IF
                =3556                               ;  BYTEIN WAS CALLED.)
                =3557          MMOV    A,CHKSUM
02D8 FD         =3573+           MOV     A,CHKSUM
02D9 96E1       =3577          JNZ     CHKERR
                =3578 CKSMOK:  MMOV    A,RECTYP
02DB B942       =3587+           MOV     R1,#RECTYP
02DD F1         =3580+           MOV     A,@R1
02DE C697       =3592          JZ      HRECIN
                =3593 ;
                =3594 ;DONE    HEX FILE CORRECTLY RECEIVED
02E0 83         =3595 DONE:    RET
                =3596 ;
                =3597 ;CHKERR CHECKSUM ERROR IN INPUT RECORD DETECTED
02E1 BA0C       =3598 CHKERR:  MOV     LDATA,#0CH
02E3 249A       =3599          JMP     PLRROR
                =3600          SIZECHK
004E            =3603+  SIZE   SET   78
                =3604+,
                =3605+;*************************************************************
                =3614 ;
                =3615          CODEBLK 12
00F0            =3620+           ORG     240
                =3624 ;BYTEIN BYTE INPUT SUBROUTINE.
                =3625 ;        RECEIVES TWO HEXIDECIMAL CHARACTERS FROM THE TAPE INPUT DEVICE
                =3626 ;        AND ASSEMBLES THEM INTO A SINGLE BYTE OF DATA.
00F0 34B8       =3627 BYTEIN:  CALL    NIBIN
00F2 47         =3628 BYTEI1:  SWAP    A
00F3 AA         =3629          MOV     LDATA,A
00F4 34B8       =3630          CALL    NIBIN
                =3631          MORL    LDATA,A
00F6 4A         =3648+           ORL     A,LDATA
00F7 AA         =3660+           MOV     LDATA,A
00F8 6D         =3664          ADD     A,CHKSUM
00F9 AD         =3665          MOV     CHKSUM,A
00FA FA         =3666          MOV     A,LDATA
00FB 83         =3667          RET
                =3668          SIZECHK
000C            =3671+  SIZE   SET   12
                =3672+;
                =3673+;*************************************************************
                =3682 ;
                =3683          CODEBLK 25
01B8            =3693+           ORG     440
                =3697 ;NIBIN  RECEIVES A HEXIDECIMAL CHARACTER AND PRODUCES A MASKED FOUR BIT VALUE.
                =3698 ;        NOTE- ERROR CHECKING DONE TO VERIFY HEXIDECIMAL VALIDITY
01B8 34CD       =3699 NIBIN:   CALL    CHARIN
01BA 03C6       =3700 NIBIN2:  ADD     A,#-3AH         ;ACC=0F6-0FF FOR CHARACTERS '0'-'9'
                =3701                                  ;CHARACTERS > '9' PRODUCE OVERFLOW
01BC E6C2       =3702          JNC     NIBI3
01BE 03F9       =3703          ADD     A,#-7           ;ACC=0-5 FOR CHARACTERS 'A'-'F'
01C0 E6C9       =3704          JNC     ASCLRR          ;ERROR IF CHARACTER BETWEEN '9' AND 'A'
                =3705 ;
                =3706 ;        ACC=0F6H-05H FOR CHARACTERS '0'-'F'
                =3707 ;
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

01C2 03FA      =3708 NIBI3:  ADD     A,#-6          ;ACC=0F0H-0FFH FOR CHARACTERS '0'-'F'
01C4 0310      =3709         ADD     A,#10H         ;ACC=00H-0FH FOR CHARACTERS '0'-'F';
               =3710                                ;OVERFLOW IF ABOVE IS TRUE.
01C6 E6C9      =3711         JNC     ASCERR
01C8 83        =3712         RET
               =3713 ;
               =3714 ;ASCERR ILLEGAL HEXIDECIMAL CHARACTER RECEIVED
01C9 BA0A      =3715 ASCERR: MOV     LDATA,#0AH
01CB 249A      =3716         JMP     PERROR
               =3717         SIZECHK
0015           =3720+  SIZE  SET   21
               =3721+;
               =3722+;*************************************************************
               =3731 ;
               =3732 ;
               =3733         CODEBLK 5
01CD           =3743+          ORG     461
               =3747 ;CHARIN CHARACTER INPUT ROUTINE.
               =3748 ;       RECEIVES ONE ASCII CHARACTER FROM THE LOGICAL READER DEVICE.
01CD D449      =3749 CHARIN: CALL    CIN
01CF 537F      =3750         ANL     A,#7FH
01D1 83        =3751         RET
               =3752         SIZECHK
0005           =3755+  SIZE  SET   5
               =3756+;
               =3757+;*************************************************************
               =3766 ;
               =3767 ;
               =3768 $EJECT
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

                =3769          CODEBLK 100
0572            =3794+           ORG     1394
                =3798 ;HFILEO HEX FILE OUTPUT SUBROUTINE
                =3799 ;       WHEN CALLED WITH F0=0 OUTPUT IS STANDARD HEX FILE FORMAT.
                =3800 ;       WHEN CALLED WITH F0=1 OUTPUT IS FORMATTED DATA DUMP TO CRT.
                =3801 HFILEO: MMOV    MEMHI,SMAHI
0572 B931       =3817+          MOV     R1,#SMAHI
0574 F1         =3818+          MOV     A,@R1
0575 B935       =3824+          MOV     R1,#MEMHI
0577 A1         =3825+          MOV     @R1,A
                =3828          MMOV    MEMLO,SMALO
0578 B930       =3844+          MOV     R1,#SMALO
057A F1         =3845+          MOV     A,@R1
057B B934       =3851+          MOV     R1,#MEMLO
057D A1         =3852+          MOV     @R1,A
                =3855          MMOV    CHKSUM,ZERO
057E BD00       =3860+          MOV     CHKSUM,#ZERO
0580 B865       =3864          MOV     R0,#HEXBUF
                =3865 ;
                =3866 ;LDBYTE LOAD NEXT BYTE FROM MEMORY INTO HEX BUFFER
0582 14FC       =3867 LDBYTE: CALL    LFETCH
0584 FA         =3868          MOV     A,LDATA
0585 A0         =3869          MOV     @R0,A
0586 18         =3870          INC     R0
0587 B4E2       =3871          CALL    CMPMAS
0589 E696       =3872          JNC     ENDFIL
058B 34F2       =3873          CALL    INCSMA
058D F8         =3874          MOV     A,R0
058E 038B       =3875          ADD     A,#-(BUFLEN+HEXBUF)
0590 E682       =3876          JNC     LDBYTE
0592 D400       =3877          CALL    HRECO
0594 A472       =3878          JMP     HFILEO
                =3879 ;
                =3880 ;ENDFIL END HEX FILE TRANSMISSION:
                =3881 ;       PRINT OUT BUFFER FOR LAST DATA RECORD
                =3882 ;       PRINT OUT CANNED 'END-OF-FILE' RECORD
                =3883 ;       RETURN.
0596 D400       =3884 ENDFIL: CALL    HRECO
0598 B6A7       =3885          JF0     HFDONE
059A 34D2       =3886          CALL    TCRLFO
059C B8AE       =3887          MOV     R0,#(LOW EOFREC)
059E F8         =3888 ENDF1:  MOV     A,R0
059F A3         =3889          MOVP    A,@A
05A0 C6A7       =3890          JZ      HFDONE
05A2 B4BD       =3891          CALL    CHARO
05A4 18         =3892          INC     R0
05A5 A49E       =3893          JMP     ENDF1
05A7 34D2       =3894 HFDONE: CALL    TCRLFO
05A9 231A       =3895          MOV     A,#CNTRLZ
05AB B4BD       =3896          CALL    CHARO
05AD 83         =3897          RET
                =3898 ;
                =3899 ;EOFREC CHARACTER STRING FOR CANNED END-OF-FILE RECORD FOR
                =3900 ;       INTEL HEX FILE FORMAT STANDARD.
05AE 203A3030   =3901 EOFREC: DB      ' :00000001FF'
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

05B2 30303030
05B6 30314646
05BA 00         =3902        DB     0              ;END OF STRING CODE BYTE
                =3903        SIZECHK
0049            =3906+ SIZE  SET  73
                =3907+;
                =3908+;*****************************************************
                =3917 ;
                =3918 ;
                =3919        CODEBLK 90
0600            =3949+         ORG    1536
                =3953 ;HRECO  HEXIDECIMAL RECORD OUTPUT SEQUENCE.
                =3954 ;       HEX BUFFER ALREADY LOADED.
0600 F8         =3955 HRECO:  MOV    A,R0
0601 039B       =3956         ADD    A,#-HEXBUF
                =3957         MMOV   BUFCNT,A
0603 B941       =3970+          MOV    R1,#BUFCNT
0605 A1         =3971+          MOV    @R1,A
0606 34D2       =3975         CALL   TCRLFO
0608 2320       =3976         MOV    A,#' '
060A B4BD       =3977         CALL   CHARO
060C D617       =3978         JF0    FDUMP1
060E 233A       =3979         MOV    A,#':'
0610 B4BD       =3980         CALL   CHARO
                =3981         MMOV   A,BUFCNT
0612 B941       =3990+          MOV    R1,#BUFCNT
0614 F1         =3991+          MOV    A,@R1
0615 34DB       =3995         CALL   BYTEO
                =3996 FDUMP1: MMOV   A,MEMHI
0617 B935       =4005+          MOV    R1,#MEMHI
0619 F1         =4006+          MOV    A,@R1
061A 34DB       =4010         CALL   BYTEO
                =4011         MMOV   A,MEMLO
061C B934       =4020+          MOV    R1,#MEMLO
061E F1         =4021+          MOV    A,@R1
061F 34DB       =4025         CALL   BYTEO
0621 B628       =4026         JF0    FDUMP2
0623 27         =4027         CLR    A
0624 34DB       =4028         CALL   BYTEO
0626 C42C       =4029         JMP    DATO
0628 233D       =4030 FDUMP2: MOV    A,#'='
062A B4BD       =4031         CALL   CHARO
                =4032 ;DATO   DATA OUTPUT
062C E865       =4033 DATO:   MOV    R0,#HEXBUF
062E B632       =4034 DATO1:  JF0    FDUMP5
0630 C436       =4035         JMP    FDUMP3
0632 2320       =4036 FDUMP5: MOV    A,#' '
0634 B4BD       =4037         CALL   CHARO
0636 F0         =4038 FDUMP3: MOV    A,@R0
0637 34DB       =4039         CALL   BYTEO
0639 18         =4040         INC    R0
                =4041         MDJNZ  BUFCNT,DATO1
063A B941       =4046+          MOV    R1,#BUFCNT
063C F1         =4047+          MOV    A,@R1
063D 07         =4051+          DEC    A
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

063E A1         =4056+       MOV     @R1,A
063F 962E       =4060+       JNZ     DAT01
                =4062 ;
                =4063 ;ENDREC END RECORD BEING TRANSMITTED
0641 B648       =4064 ENDREC: JF0     FDUMP4
                =4065        MMOV    A,CHKSUM
0643 FD         =4081+       MOV     A,CHKSUM
0644 37         =4085        CPL     A
0645 17         =4086        INC     A
0646 34DB       =4087        CALL    BYTE0
0648 83         =4088 FDUMP4: RET
                =4089        SIZECHK
0049            =4092+ SIZE  SET   73
                =4093+;
                =4094+;*************************************************************
                =4103 ;
                =4104        CODEBLK 9
01D2            =4114+       ORG     466
                =4118 ;TCRLF0 TAPE <CR><LF> OUTPUT
01D2 23OD       =4119 TCRLF0: MOV     A,#CHARCR
01D4 B4BD       =4120        CALL    CHAR0
01D6 23OA       =4121        MOV     A,#CHARLF
01D8 B4BD       =4122        CALL    CHAR0
01DA 83         =4123        RET
                =4124        SIZECHK
0009            =4127+ SIZE  SET   9
                =4128+;
                =4129+;*************************************************************
                =4138 ;
                =4139        CODEBLK 11
01DB            =4149+       ORG     475
                =4153 ;BYTE0  BYTE OUTPUT
01DB AA         =4154 BYTE0:  MOV     LDATA,A
01DC 6D         =4155        ADD     A,CHKSUM
01DD AD         =4156        MOV     CHKSUM,A
01DE FA         =4157        MOV     A,LDATA
01DF 47         =4158        SWAP    A
01E0 B4DB       =4159        CALL    NIB0
01E2 FA         =4160        MOV     A,LDATA
01E3 B4BB       =4161        CALL    NIB0
01E5 83         =4162        RET
                =4163        SIZECHK
000B            =4166+ SIZE  SET   11
                =4167+;
                =4168+;*************************************************************
                =4177 ;
                =4178        CODEBLK 12
01E6            =4188+       ORG     48C
                =4192 ;HEXASC HEXIDECIMAL NIBBLE TO ASCII CHARACTER CONVERSION.
01E6 530F       =4193 HEXASC: ANL     A,#0FH
01E8 03F6       =4194        ADD     A,#(-10)
01EA F6EF       =4195        JC      HEXNIB
01EC 033A       =4196        ADD     A,#(10+'0')
01EE 83         =4197        RET
01EF 0341       =4198 HEXNIB: ADD     A,#('A')
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

01F1 83        =4199         RET
               =4200         SIZECHK
000C           =4203+  SIZE  SET   12
               =4204+;
               =4205+;*******************************************************
               =4214 ;
               =4215 ;
               =4216 DECLARE BITSO,CONST
000B           =4230 BITSO   EQU     11      ;DATA BITS PUT OUT (INCLUDING TWO STOP BITS)
               =4231 ;
               =4232         CODEBLK 30
04C9           =4252+          ORG     1225
               =4256 ;HBDLAY HALF-BIT TIME DELAY
               =4257 HBDLAY: MMOV    H,HBITHI
04C9 B927      =4273+          MOV     R1,#HBITHI
04CB F1        =4274+          MOV     A,@R1
04CC B945      =4280+          MOV     R1,#H
04CE A1        =4281+          MOV     @R1,A
               =4284         MMOV    R1,HBITLO
04CF B926      =4300+          MOV     R1,#HBITLO
04D1 F1        =4301+          MOV     A,@R1
04D2 A9        =4314+          MOV     R1,A
04D3 84D7      =4317         JMP     HBD1
04D5 B900      =4318 HBD2:   MOV     R1,#0
04D7 E9D7      =4319 HBD1:   DJNZ    R1,HBD1
               =4320         MDJNZ   H,HBD2
04D9 B945      =4325+          MOV     R1,#H
04DB F1        =4326+          MOV     A,@R1
04DC 07        =4330+          DEC     A
04DD A1        =4335+          MOV     @R1,A
04DE 96D5      =4339+          JNZ     HBD2
04E0 83        =4341         RET
               =4342         SIZECHK
0018           =4345+  SIZE  SET   24
               =4346+;
               =4347+;*******************************************************
               =4356 ;
               =4357 $EJECT
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

                =4358         CODEBLK 40
05B8            =4383+          ORG     1467
                =4387 ;NIBO   MASK ACC TO MAKE HEX NIBBLE, TRANSLATE TO ASCII AND OUTPUT
05B8 34E6       =4388 NIBO:   CALL    HEXASC
                =4389 ;
                =4390 ;CHARO  CONSOLE OUTPUT SUBROUTINE
                =4391 ;       WRITES THE CONTENTS OF THE ACC TO THE CRT DISPLAY SCREEN
                =4392 CHARO:  MMOV    REGC,A
05BD B944       =4405+          MOV     R1,#REGC
05BF A1         =4406+          MOV     @R1,A
                =4410         MMOV    B,BITSO ;SET NUMBER OF BITS TO BE TRANSMITTED
05C0 B943       =4421+          MOV     R1,#B
05C2 B108       =4422+          MOV     @R1,#BITSO
05C4 97         =4426         CLR     C       ;CLEAR CARRY
05C5 F6C8       =4427 CO1:    JC      CO2
05C7 99BF       =4428         ANL     P1,#NOT TTYOUT
05C9 A4CF       =4429         JMP     CO3
05CB 8940       =4430 CO2:    ORL     P1,#TTYOUT
05CD 00         =4431         NOP             ;EVEN OUT TWO BRANCH EXECUTION TIMES
05CE 00         =4432         NOP
05CF 94C9       =4433 CO3:    CALL    HBDLAY
05D1 94C9       =4434         CALL    HBDLAY
05D3 97         =4435         CLR     C       ;SET WHAT WILL EVENTUALLY BECOME A STOP BIT
05D4 A7         =4436         CPL     C
                =4437         MRRC    REGC    ;ROTATE CHARACTER RIGHT ONE BIT,
05D5 B944       =4442+          MOV     R1,#REGC
05D7 F1         =4443+          MOV     A,@R1
05D8 67         =4447+          RRC     A
05D9 A1         =4452+          MOV     @R1,A
                =4455                                   ;\ MOVING NEXT DATA BIT INTO CARRY
                =4456         MDJNZ   B,CO1   ;CHECK IF CHARACTER (AND STOP BIT(S)) DONE
05DA B943       =4461+          MOV     R1,#B
05DC F1         =4462+          MOV     A,@R1
05DD 07         =4466+          DEC     A
05DE A1         =4471+          MOV     @R1,A
05DF 96C5       =4475+          JNZ     CO1
05E1 83         =4477         RET
                =4478         SIZECHK
0027            =4481+  SIZE  SET   39
                =4482+;
                =4483+;*******************************************************
                =4492 ;
                =4493         CODEBLK 47
0649            =4523+          ORG     1609
                =4527 ;CIN    CONSOL INPUT SUBROUTINE WAITS FOR A KEYSTROKE AND
                =4528 ;       RETURNS WITH 8 BITS IN REG ACC.
0649 B943       =4529 CIN:    MOV     R1,#B
064B B108       =4530         MOV     @R1,#8  ;DATA BITS TO BE READ
064D 464D       =4531 CI0:    JNT1    CI0
064F 464D       =4532         JNT1    CI0
0651 5651       =4533 CI1:    JT1     CI1
0653 5651       =4534         JT1     CI1
0655 94C9       =4535         CALL    HBDLAY
0657 5651       =4536         JT1     CI1
0659 94C9       =4537 CI2:    CALL    HBDLAY
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

065B 94C9      =4538        CALL    MBDLAY
065D 5662      =4539        JT1     CI3     ;CHECK SID LINE LEVEL
065F 97        =4540        CLR     C       ;DATA BIT IN CY
0660 C465      =4541        JMP     CI4
0662 97        =4542 CI3:   CLR     C
0663 A7        =4543        CPL     C
0664 00        =4544        NOP             ;EVEN OUT BRANCH EXECUTION TIMES
0665 00        =4545 CI4:   NOP
0666 00        =4546        NOP
0667 00        =4547        NOP
               =4548        MRRC    REGC
0668 B944      =4553+         MOV     R1,#REGC
066A F1        =4554+         MOV     A,@R1
066B 67        =4558+         RRC     A
066C A1        =4563+         MOV     @R1,A
               =4566        MDJNZ   B,CI2
066D B943      =4571+         MOV     R1,#B
066F F1        =4572+         MOV     A,@R1
0670 07        =4576+         DEC     A
0671 A1        =4581+         MOV     @R1,A
0672 9659      =4585+         JNZ     CI2
               =4587        MMOV    A,REGC
0674 B944      =4596+         MOV     R1,#REGC
0676 F1        =4597+         MOV     A,@R1
0677 83        =4601        RET             ;CHARACTER COMPLETE
               =4602        SIZECHK
002F           =4605+  SIZE  SET   47
               =4606+;
               =4607+;*****************************************************
               =4616 $EJECT
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

                 4617 $       INCLUDE(:F0:MEMREF.MOD)
                =4618         CODEBLK 15
02E5            =4633+          ORG     741
                =4637 ;COMFIL COMMAND TO FILL ADDRESS SPACE BETWEEN SMA AND EMA WITH DATA
                =4638 ;       IN LOW BYTE OF MEM.
                =4639 COMFIL: MMOV    LDATA,MEMLO
02E5 B934       =4655+          MOV     R1,#MEMLO
02E7 F1         =4656+          MOV     A,@R1
02E8 AA         =4669+          MOV     LDATA,A
02E9 F400       =4672 LFILL:  CALL    LSTORE
02EB B4E2       =4673         CALL    CMPMAS
02ED E6F3       =4674         JNC     LFILL1
02EF 34F2       =4675         CALL    INCSMA
02F1 44E9       =4676         JMP     LFILL
02F3 83         =4677 LFILL1: RET
                =4678         SIZECHK
000F            =4681+  SIZE  SET   15
                =4682+;
                =4683+;*******************************************************
                =4692 ;
                =4693         CODEBLK 4
00FC            =4698+          ORG     252
                =4702 ;LFETCH FETCHES CONTENTS OF LOGICAL MEMORY ADDRESS DETERMINED BY
                =4703 ;       (TYPE),(SMAHI), & (SMALO) INTO (LDATA).
00FC D478       =4704 LFETCH: CALL    AFETCH
00FE AA         =4705         MOV     LDATA,A
00FF 83         =4706         RET
                =4707         SIZECHK
0004            =4710+  SIZE  SET   4
                =4711+;
                =4712+;*******************************************************
                =4721 ;
                =4722         CODEBLK 75
0678            =4752+          ORG     1656
                =4756 ;
                =4757 ;AFETCH LOGICAL FETCH SUBROUTINE
                =4758 ;       FETCHS CONTENTS OF VARIOUS MEMORY SPACES TO ACC.
                =4759 AFETCH: MMOV    A,TYPE
0678 B937       =4768+          MOV     R1,#TYPE
067A F1         =4769+          MOV     A,@R1
067B 037E       =4773         ADD     A,#LOW LFETBL
067D B3         =4774         JMPP    @A
                =4775 ;
067E 04         =4776 LFETBL: DB      LOW LFEPM
067F 98         =4777         DB      LOW LFEDM
0680 9C         =4778         DB      LOW LFEREG
0681 A9         =4779         DB      LOW LFEINT
0682 B1         =4780         DB      LOW LFEBRK
0683 B1         =4781         DB      LOW LFEBRK
                =4782 ;
                =4783 LFEPM:  MMOV    A,SMAHI
0684 B931       =4792+          MOV     R1,#SMAHI
0686 F1         =4793+          MOV     A,@R1
0687 9698       =4797         JNZ     LFEDM
                =4798         MMOV    A,SMALO
```

```
LOC  OBJ       LINE       SOURCE STATEMENT

0689 B930      =4807+       MOV     R1,#SMALO
068B F1        =4808+       MOV     A,@R1
068C 03E9      =4812        ADD     A,#-OVSIZE
068E F698      =4813        JC      LFEDM
               =4814        MMOV    A,SMALO
0690 B930      =4823+       MOV     R1,#SMALO
0692 F1        =4824+       MOV     A,@R1
0693 034E      =4828        ADD     A,#OVBUF
0695 A9        =4829        MOV     R1,A
0696 F1        =4830        MOV     A,@R1
0697 83        =4831        RET
0698 94E1      =4832 LFEDM: CALL    LPGSEL
069A 81        =4833        MOVX    A,@R1
069B 83        =4834        RET
               =4835 ;
               =4836 LFEREG: MMOV   A,SMALO
069C B930      =4845+       MOV     R1,#SMALO
069E F1        =4846+       MOV     A,@R1
069F 537F      =4850        ANL     A,#01111111B    ;CHECK IF LOW 7 BITS =0
06A1 C6A5      =4851        JZ      LFER0
06A3 E4B7      =4852        JMP     EPFET
               =4853 ;
               =4854 LFER0:  MMOV   A,EPR0
06A5 B923      =4863+       MOV     R1,#EPR0
06A7 F1        =4864+       MOV     A,@R1
06A8 83        =4868        RET
               =4869 ;
               =4870 LFEINT: MMOV   A,SMALO
06A9 B930      =4879+       MOV     R1,#SMALO
06AB F1        =4880+       MOV     A,@R1
06AC 0320      =4884        ADD     A,#EPACC
06AE A9        =4885        MOV     R1,A
06AF F1        =4886        MOV     A,@R1
06B0 83        =4887        RET
               =4888 ;
               =4889 ;LFEBRK LOGICAL FETCH OF BREAK-POINT DATA
06B1 94E1      =4890 LFEBRK: CALL   LPGSEL
06B3 99F7      =4891        ANL     P1,#NOT 00001000B
06B5 8908      =4892        ORL     P1,#00001000B
06B7 99FD      =4893        ANL     P1,#NOT 00000010B
06B9 8901      =4894        ORL     P1,#00000001B
06BB 81        =4895        MOVX    A,@R1
06BC 2301      =4896        MOV     A,#01H
06BE 86C1      =4897        JNI     LFEBR1
06C0 27        =4898        CLR     A
06C1 83        =4899 LFEBR1: RET
               =4900        SIZECHK
004A           =4903+  SIZE  SET   74
               =4904+;
               =4905+;*************************************************************
               =4914 $EJECT
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

                =4915        CODEBLK 85
0700            =4950+         ORG    1792
                =4954 ;
                =4955 ;LSTORE LOGICAL STORE SUBROUTINE
                =4956 ;       STORES CONTENTS OF LDATA INTO VARIOUS MEMORY SPACES.
                =4957 LSTORE: MMOV   A,TYPE
0700 B937       =4966+         MOV    R1,#TYPE
0702 F1         =4967+         MOV    A,@R1
0703 0306       =4971        ADD    A,#LOW LSTTBL
0705 B3         =4972        JMPP   @A
                =4973 ;
0706 0C         =4974 LSTTBL: DB     LOW LSTPM
0707 21         =4975        DB     LOW LSTDM
0708 26         =4976        DB     LOW LSTREG
0709 34         =4977        DB     LOW LSTINT
070A 3D         =4978        DB     LOW LSTBRK
070B 3D         =4979        DB     LOW LSTBRK
                =4980 ;
                =4981 LSTPM:  MMOV   A,SMAHI
070C B931       =4990+         MOV    R1,#SMAHI
070E F1         =4991+         MOV    A,@R1
070F 9621       =4995        JNZ    LSTDM
                =4996        MMOV   A,SMALO
0711 B930       =5005+         MOV    R1,#SMALO
0713 F1         =5006+         MOV    A,@R1
0714 03E9       =5010        ADD    A,#-OVSIZE
0716 F621       =5011        JC     LSTDM
                =5012        MMOV   A,SMALO
0718 B930       =5021+         MOV    R1,#SMALO
071A F1         =5022+         MOV    A,@R1
071B 034E       =5026        ADD    A,#OVBUF
071D A9         =5027        MOV    R1,A
071E FA         =5028        MOV    A,LDATA
071F A1         =5029        MOV    @R1,A
0720 83         =5030        RET
                =5031 ;
0721 94E1       =5032 LSTDM:  CALL   LPGSEL
0723 FA         =5033        MOV    A,LDATA
0724 91         =5034        MOVX   @R1,A
0725 83         =5035        RET
                =5036 ;
                =5037 LSTREG: MMOV   A,SMALO
0726 B930       =5046+         MOV    R1,#SMALO
0728 F1         =5047+         MOV    A,@R1
0729 537F       =5051        ANL    A,#01111111B    ;CHECK IF LOW ORDER BITS = 0
072B C62F       =5052        JZ     LSTR0
072D E4C3       =5053        JMP    EPSTOR
                =5054 ;
                =5055 LSTR0:  MMOV   EPR0,LDATA
072F FA         =5078+         MOV    A,LDATA
0730 B923       =5084+         MOV    R1,#EPR0
0732 A1         =5085+         MOV    @R1,A
0733 83         =5088        RET
                =5089 ;
                =5090 LSTINT: MMOV   A,SMALO
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

0734 B930       =5099+       MOV    R1,#SMALO
0736 F1         =5100+       MOV    A,@R1
0737 0320       =5104        ADD    A,#EPACC
0739 A9         =5105        MOV    R1,A
073A FA         =5106        MOV    A,LDATA
073B A1         =5107        MOV    @R1,A
073C 83         =5108        RET
                =5109 ;
                =5110 ;LSTBRK LOGICAL STORE OF BREAK-POINT DATA
073D 94E1       =5111 LSTBRK: CALL   LPGSEL
073F FA         =5112        MOV    A,LDATA
0740 1246       =5113        JB0    LSTBR1
0742 8901       =5114        ORL    P1,#00000001B
0744 E448       =5115        JMP    LSTBR2
0746 99FE       =5116 LSTBR1: ANL    P1,#NOT 00000001B
0748 99F7       =5117 LSTBR2: ANL    P1,#NOT 00001000B
074A 81         =5118        MOVX   A,@R1
074B 8908       =5119        ORL    P1,#00001000B
074D 83         =5120        RET
                =5121        SIZECHK
004E            =5124+  SIZE SET  78
                =5125+;
                =5126+;*************************************************************
                =5135 ;
                =5136        CODEBLK 17
04E1            =5156+       ORG    1249
                =5160 ;LPGSEL LOGICAL PAGE SELECT.
                =5161 ;      SETS UP PORT 2 TO ADDRESS APPROPRIATE BYTE OF RAM BLOCK.
                =5162 LPGSEL: MMOV   A,TYPE
04E1 B937       =5171+       MOV    R1,#TYPE
04E3 F1         =5172+       MOV    A,@R1
04E4 5301       =5176        ANL    A,#00000001B          ;MASK OFF DATA TYPE SELECTOR BIT
04E6 47         =5177        SWAP   A
                =5178        MORL   A,SMAHI
04E7 B931       =5184+       MOV    R1,#SMAHI
04E9 41         =5185+       ORL    A,@R1
04EA 4340       =5189        ORL    A,#01000000B
04EC 3A         =5190        OUTL   P2,A
                =5191        MMOV   A,SMALO
04ED B930       =5200+       MOV    R1,#SMALO
04EF F1         =5201+       MOV    A,@R1
04F0 A9         =5205        MOV    R1,A
04F1 83         =5206        RET
                =5207        SIZECHK
0011            =5210+  SIZE SET  17
                =5211+;
                =5212+;*************************************************************
                =5221 ;
                =5222 $EJECT
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

                =5223         CODEBLK 11
01F2            =5233+          ORG     498
                =5237 ;INCSMA INCREMENT STARTING MEMORY ADDRESS WORD.
01F2 B930       =5238 INCSMA: MOV     R1,#SMALO
01F4 11         =5239 INCW:   INC     @R1
01F5 F1         =5240         MOV     A,@R1
01F6 96FC       =5241         JNZ     INCW1
01F8 19         =5242         INC     R1
01F9 F1         =5243         MOV     A,@R1
01FA 17         =5244         INC     A
01FB 31         =5245         XCHD    A,@R1
01FC 83         =5246 INCW1:  RET
                =5247         SIZECHK
000B            =5250+  SIZE  SET   11
                =5251+;
                =5252+;*************************************************************
                =5261 ;
                =5262         CODEBLK 12
02F4            =5277+          ORG     756
                =5281 ;DECSMA DECREMENT SMA WORD.
02F4 B930       =5282 DECSMA: MOV     R1,#SMALO
02F6 F1         =5283         MOV     A,@R1
02F7 07         =5284         DEC     A
02F8 21         =5285         XCH     A,@R1
02F9 96FF       =5286         JNZ     DECSM1
02FB 19         =5287         INC     R1
02FC F1         =5288         MOV     A,@R1
02FD 07         =5289         DEC     A
02FE 31         =5290         XCHD    A,@R1
02FF 83         =5291 DECSM1: RET
                =5292         SIZECHK
000C            =5295+  SIZE  SET   12
                =5296+;
                =5297+;*************************************************************
                =5306 ;
                =5307         CODEBLK 15
05E2            =5332+          ORG     1506
                =5336 ;CMPMAS COMPARE MEMORY ADDRESSES
                =5337 ;       COMPARE SMA BYTES WITH EMA BYTES TO DETERMINE RELATIVE MAGNITUDE.
                =5338 ;       RETURNS WITH CARRY=1 IFF <SMA> >= <EMA>.
                =5339 ;       IS CALLED AFTER ACTION HAS BEEN PERFORMED ON <SMA> TO DETERMINE IF
                =5340 ;       TASK IS COMPLETED:
                =5341 ;           IF CY=0 THEN <SMA> >= <EMA>   ==>  TERMINATE TASK.
                =5342 ;           IF CY=1 THEN <SMA> <  <EMA>   ==>  INC SMA AND REPEAT.
                =5343 CMPMAS: MMOV    A,SMALO
05E2 B930       =5352+          MOV     R1,#SMALO
05E4 F1         =5353+          MOV     A,@R1
05E5 37         =5357         CPL     A
                =5358         MADD    A,EMALO
05E6 B932       =5364+          MOV     R1,#EMALO
05E8 61         =5365+          ADD     A,@R1
                =5369         MMOV    A,SMAHI
05E9 B931       =5378+          MOV     R1,#SMAHI
05EB F1         =5379+          MOV     A,@R1
05EC 37         =5383         CPL     A
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

              =5384         MADDC   A,EMAHI
05ED B933     =5390+          MOV     R1,#EMAHI
05EF 71       =5391+          ADDC    A,@R1
05F0 83       =5395 CMPRET: RET
              =5396         SIZECHK
000F          =5399+  SIZE  SET   15
              =5400+;
              =5401+;*****************************************************
              =5410 $EJECT
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

                 5411 $      INCLUDE(:F0:KBD.MOD)
                =5412        CODEBLK 100
074E            =5447+         ORG    1870
                =5451 ;
                =5452 ;      KEYBOARD AND DISPLAY PROCESSING ROUTINE
                =5453 ;      CALLED PERIODICALLY WHEN KBD AND DISPLAY ARE TO BE ALIVE.
074E D5         =5454 TIINT: SEL    RB1
                =5455        MMOV   ASAVE,A
074F B93E       =5468+         MOV    R1,#ASAVE
0751 A1         =5469+         MOV    @R1,A
0752 23F0       =5473        MOV    A,#(-10H)
0754 62         =5474        MOV    T,A             ;RELOAD TIMER INTERVAL
0755 27         =5475        CLR    A
0756 3E         =5476        MOVD   PSEGHI,A        ;WRITE BLANK PATTERN TO SEG DRIVERS
0757 3D         =5477        MOVD   PSEGLO,A
0758 FD         =5478        MOV    A,CURDIG
0759 07         =5479        DEC    A
075A 3F         =5480        MOVD   PDIGIT,A        ;ENERGIZE CHARACTER
075B 0C         =5481        MOVD   A,PINPUT        ;LOAD ANY SWITCH CLOSURES
075C AA         =5482        MOV    ROTPAT,A
                =5483                               ;WRITE NEXT SEGMENT PATTERN
075D FD         =5484        MOV    A,CURDIG
075E 07         =5485        DEC    A
075F 0346       =5486        ADD    A,#SEGMAP       ;ADD CURDIG DISPLACMENT TO BASE
0761 A8         =5487        MOV    R0,A
0762 F0         =5488        MOV    A,@R0           ;LOAD ACC W/ NEXT SEGMENT PATTERN
0763 3D         =5489        MOVD   PSEGLO,A        ;ENABLE APPROPRIATE SEGMENTS
0764 47         =5490        SWAP   A
0765 3E         =5491        MOVD   PSEGHI,A
                =5492 ;
                =5493 ;*********************************************************
                =5494 ;      THE NEXT CHARACTER IS NOW BEING DISPLAYED.
                =5495 ;      THE KEYBOARD SCAN ROUTINE IS INTEGRATED INTO THE DISPLAY SCAN.
                =5496 ;      WITH THE CURRENT ROW ENERGIZED, CHECK IF THERE ARE ANY INPUTS.
                =5497 ;*********************************************************
                =5498 ;
                =5499 ;      ROTATE BITS THROUGH THE CY WHILE INCREMENTING KEYLOC.
                =5500 ;
0766 B804       =5501        MOV    ROTCNT,#NCOLS   ;SET UP FOR <NCOLS> LOOPS THROUGH 'NXTLOC'
                =5502 NXTLOC: MRRC  ROTPAT
0768 FA         =5514+         MOV    A,ROTPAT
0769 67         =5518+         RRC    A
076A AA         =5529+         MOV    ROTPAT,A
076B F688       =5532        JC     SCAN5           ;ONE BIT IN CY INDICATES KEY NOT DOWN
076D BE01       =5533        MOV    KEYFLG,#1       ;MARK THAT AT LEAST ONE KEY WAS DETECTED
                =5534                               ;\ IN THE CURRENT SCAN
                =5535 ;
                =5536 ;*********************************************************
                =5537 ;      A KEYSTROKE WAS DETECTED FOR THE CURRENT COLUMN.  ITS
                =5538 ;      POSITION IS IN REGISTER KEYLOC.  SEE IF SAME KEY SENSED LAST CYCLE.
                =5539 ;*********************************************************
                =5540 ;
                =5541        MMOV   A,KEYLOC
076F B93C       =5550+         MOV    R1,#KEYLOC
0771 F1         =5551+         MOV    A,@R1
```

```
LOC  OBJ       LINE       SOURCE STATEMENT

0772 2C        =5555        XCH     A,LASTKY
0773 DC        =5556        XRL     A,LASTKY
0774 C67C      =5557        JZ      SCAN3
               =5558 ;
               =5559 ;*************************************************************
               =5560 ;      A DIFFERENT KEY WAS READ ON THIS CYCLE THAN ON THE PREVIOUS CYCLE.
               =5561 ;      SET NREPTS TO THE DEBOUNCE PARAMETER FOR A NEW COUNTDOWN.
               =5562 ;*************************************************************
               =5563 ;
0776 B93D      =5564        MOV     R1,#NREPTS
0778 B106      =5565        MOV     @R1,#6
077A C488      =5566        JMP     SCAN5
               =5567 ;
               =5568 ;*************************************************************
               =5569 ;      SAME KEY WAS DETECTED 3S ON PREVIOUS CYCLE
               =5570 ;      LOOK AT NREPTS: IF ALREADY ZERO, DO NOTHING.
               =5571 ;      ELSE DECREMENT NREPTS.
               =5572 ;      IF THIS RESULTS IN ZERO, MOVE LASTKY INTO KBDBUF.
               =5573 ;*************************************************************
               =5574 ;
               =5575 SCAN3:  MMOV    A,NREPTS
077C B93D      =5584+          MOV     R1,#NREPTS
077E F1        =5585+          MOV     A,@R1
077F C688      =5589        JZ      SCAN5           ;IF ALREADY ZERO
0781 07        =5590        DEC     A               ;INDICATE ONE MORE SUCCESIVE KEY DETECTION
               =5591        MMOV    NREPTS,A
0782 B93D      =5604+          MOV     R1,#NREPTS
0784 A1        =5605+          MOV     @R1,A
0785 9688      =5609        JNZ     SCAN5           ;IF DECREMENT DOES NOT RESULT IN ZERO
               =5610        MMOV    KBDBUF,LASTKY   ;TO MARK NEW KEY CLOSURE
0787 FC        =5633+          MOV     A,LASTKY
0788 B93B      =5639+          MOV     R1,#KBDBUF
078A A1        =5640+          MOV     @R1,A
               =5643 ;
078B B93C      =5644 SCAN5:  MOV     R1,#KEYLOC
078D 11        =5645        INC     @R1
078E EDG8      =5646        DJNZ    ROTCNT,NXTLOC
0790 EDA8      =5647        DJNZ    CURDIG,TIRET1
0792 BD08      =5648        MOV     CURDIG,#CHARNO
               =5649 ;
               =5650 ;*************************************************************
               =5651 ;      THE FOLLOWING CODE SEGMENT IS USED BY THE KEYBOARD SCANNING ROUTINE.
               =5652 ;      IT IS EXECUTED ONLY AFTER A REFRESH SEQUENCE IS COMPLETED
               =5653 ;*************************************************************
               =5654 ;
               =5655        MMOV    KEYLOC,ZERO
0794 B93C      =5666+          MOV     R1,#KEYLOC
0796 B100      =5667+          MOV     @R1,#ZERO
0798 FE        =5671        MOV     A,KEYFLG
0799 969D      =5672        JNZ     SCAN8           ;JUMP IF ANY KEYS WERE DETECTED
               =5673        MMOV    LASTKY,NEG1     ;CHANGE <LASTKY> WHEN NO KEYS ARE DOWN
079B BCFF      =5678+          MOV     LASTKY,#NEG1
079D BE00      =5682 SCAN8:  MOV     KEYFLG,#0
               =5683 ;
               =5684 ;*************************************************************
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

                =5685 ;
                =5686 ;       KBD/DISP RETURN CODE- RESTORES SYSTEM STATUS.
                =5687         MMOV    A,RDELAY
079F B93F       =5696+          MOV     R1,#RDELAY
07A1 F1         =5697+          MOV     A,@R1
07A2 C6A8       =5701         JZ      TIRET1
07A4 07         =5702         DEC     A
                =5703         MMOV    RDELAY,A
07A5 B93F       =5716+          MOV     R1,#RDELAY
07A7 A1         =5717+          MOV     @R1,A
                =5721 TIRET1: MMOV    A,ASAVE
07A8 B93E       =5730+          MOV     R1,#ASAVE
07AA F1         =5731+          MOV     A,@R1
07AB 93         =5735         RETR
                =5736 ;
                =5737 ;
                =5738 ;TOFPOL TIMER OVERFLOW POLLING SUBROUTINE.
                =5739 ;       CALLED REPEATEDLY FROM WHEREVER KBD/DISP MUST BE ALIVE.
                =5740 ;       MONITORS THE TIMER OVERFLOW FLAG (TOF) AND CALLS SERVICE
                =5741 ;       ROUTINE WHEN APPROPRIATE.
07AC 164E       =5742 TOFPOL: JTF     TIINT
07AE 83         =5743         RET
                =5744         SIZECHK
0061            =5747+  SIZE  SET   97
                =5748+;
                =5749+;*****************************************************
                =5750 $EJECT
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

                =5759         CODEBLK 17
06C2            =5789+          ORG     1730
                =5793 ;
                =5794 ;KBDIN  KEYBOARD INPUT SUBROUTINE.
                =5795 ;       RETURNS ONLY AFTER A NEW KEYSTROKE HAS BEEN DETECTED AND DEBOUNCED.
                =5796 ;       VALUE OF KEY POSITION IN SWITCH MATRIX IS
                =5797 ;       RETURNED IN THE ACCUMULATOR.
                =5798 ;       DISPLAY CHARACTER NOW ON BLANKED BEFORE RETURNING.
06C2 DF03       =5799 KBDIN:  MOV     XPCODE,#3
06C4 74D1       =5800         CALL    XPTEST
06C6 F4AC       =5801 KBDI1:  CALL    TOFPOL
                =5802         MMOV    A,KBDBUF
06C8 B93B       =5811+          MOV     R1,#KBDBUF
06CA F1         =5812+          MOV     A,@R1
06CB F2C6       =5816         JB7     KBDI1
06CD 27         =5817         CLR     A
06CE 3E         =5818         MOVD    PSEGHI,A
06CF 3D         =5819         MOVD    PSEGLO,A
06D0 37         =5820         CPL     A
06D1 21         =5821         XCH     A,@R1
06D2 83         =5822         RET
                =5823         SIZECHK
0011            =5826+  SIZE  SET   17
                =5827+,
                =5828+;*****************************************************************
                =5837 ;
                =5838         CODEBLK 15
05F1            =5863+          ORG     1521
                =5867 ;CLEAR  WRITES 'BLANK' CHARACTERS INTO ALL DISPLAY REGISTERS.
                =5868 ;       RETURNS WITH NEXTPL SET TO LEFTMOST CHARACTER POSITION
                =5869 ;       DOES NOT AFFECT ACC OR CY.
05F1 B846       =5870 CLEAR:  MOV     R0,#SEGMAP
05F3 B908       =5871         MOV     R1,#CHARNO
05F5 B000       =5872 DBLANK: MOV     @R0,#0          ;STORE THE BLANK CODE
05F7 18         =5873         INC     R0              ;POINT TO NEXT CHARACTER TO THE LEFT
05F8 E9F5       =5874         DJNZ    R1,DBLANK
                =5875         MMOV    NEXTPL,CHARNO
05FA B93A       =5886+          MOV     R1,#NEXTPL
05FC B108       =5887+          MOV     @R1,#CHARNO
05FE 83         =5891         RET
                =5892         SIZECHK
000E            =5895+  SIZE  SET   14
                =5896+,
                =5897+;*****************************************************************
                =5906 ;
                =5907         CODEBLK 44
06D3            =5937+          ORG     1747
                =5941 ;DSPACC DISPLAY VALUE OF LOW NIBBLE OF ACC
06D3 530F       =5942 DSPACC: ANL     A,#0FH
06D5 03EF       =5943         ADD     A,#DGPATS
06D7 A3         =5944         MOVP    A,@A
                =5945 ;WDISP  WRITES BIT PATTERN NOW IN ACC INTO NEXT CHARACTER POSITION
                =5946 ;       OF THE DISPLAY (NEXTPL).  INCREMENTS NEXTPL.
                =5947 ;       RESULTS IN DISPLAY BEING FILLED LEFT TO RIGHT, THEN RESTARTING
06D8 AE         =5948 WDISP:  MOV     DSPTMP,A
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

06D9 BF04       =5949         MOV     XPCODE,#4
06DB 74D1       =5950         CALL    XPTEST
                =5951         MMOV    A,NEXTPL
06DD B93A       =5960+          MOV     R1,#NEXTPL
06DF F1         =5961+          MOV     A,@R1
06E0 0345       =5965         ADD     A,#SEGMAP-1
06E2 A9         =5966         MOV     R1,A
06E3 FE         =5967         MOV     A,DSPTMP
06E4 A1         =5968         MOV     @R1,A
                =5969         MDJNZ   NEXTPL,WDISP1
06E5 B93A       =5974+          MOV     R1,#NEXTPL
06E7 F1         =5975+          MOV     A,@R1
06E8 07         =5979+          DEC     A
06E9 A1         =5984+          MOV     @R1,A
06EA 96EE       =5988+          JNZ     WDISP1
06EC B108       =5990         MOV     @R1,#CHARNO
06EE 83         =5991 WDISP1: RET
                =5992 ;
                =5993 ;DGPATS IS THE BASE FOR THE TABLE OF SEGMENT PATTERNS FOR HEX DIGITS.
                =5994 ;HERE THE FULL HEX SET (0-F) IS INCLUDED.
                =5995 ;
00EF            =5996 DGPATS  EQU     $ AND 0FFH
                =5997 ;
                =5998 ;FORMAT IS      PGFEDCBA        IN STANDARD SEVEN-SEGMENT ENCODING CONVENTION
                =5999 ;                               WHERE P REPRESENTS THE DECIMAL POINT
06EF 3F         =6000         DB      00111111B       ;SEGMENT PATTERN FOR DIGIT '0'
06F0 06         =6001         DB      00000110B       ;SEGMENT PATTERN FOR DIGIT '1'
06F1 5B         =6002         DB      01011011B       ;SEGMENT PATTERN FOR DIGIT '2'
06F2 4F         =6003         DB      01001111B       ;SEGMENT PATTERN FOR DIGIT '3'
06F3 66         =6004         DB      01100110B       ;SEGMENT PATTERN FOR DIGIT '4'
06F4 6D         =6005         DB      01101101B       ;SEGMENT PATTERN FOR DIGIT '5'
06F5 7D         =6006         DB      01111101B       ;SEGMENT PATTERN FOR DIGIT '6'
06F6 07         =6007         DB      00000111B       ;SEGMENT PATTERN FOR DIGIT '7'
06F7 7F         =6008         DB      01111111B       ;SEGMENT PATTERN FOR DIGIT '8'
06F8 67         =6009         DB      01100111B       ;SEGMENT PATTERN FOR DIGIT '9'
06F9 77         =6010         DB      01110111B       ;SEGMENT PATTERN FOR DIGIT 'A'
06FA 7C         =6011         DB      01111100B       ;SEGMENT PATTERN FOR DIGIT 'B'
06FB 39         =6012         DB      00111001B       ;SEGMENT PATTERN FOR DIGIT 'C'
06FC 5E         =6013         DB      01011110B       ;SEGMENT PATTERN FOR DIGIT 'D'
06FD 79         =6014         DB      01111001B       ;SEGMENT PATTERN FOR DIGIT 'E'
06FE 71         =6015         DB      01110001B       ;SEGMENT PATTERN FOR DIGIT 'F'
                =6016         SIZECHK
002C            =6019+  SIZE  SET   44
                =6020+;
                =6021+;*****************************************************************
                =6030 ;
                =6031         CODEBLK 12
04F2            =6051+          ORG     1266
                =6055 ;DELAY  SUBROUTINE WAITS FOR THE NUMBER OF COMPLETE
                =6056 ;       DISPLAY SCANS CORRESPONDING TO THE ACC CONTENTS.
                =6057 ;       USED WITH CRUDE HUMAN INTERFACES- AS WHEN OPERATOR SHOULD SEE
                =6058 ;       SOME DISPLAY CHANGE WHILE IT IS CHANGING.
                =6059 DELAY:  MMOV    RDELAY,A
04F2 B93F       =6072+          MOV     R1,#RDELAY
04F4 A1         =6073+          MOV     @R1,A
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

04F5 F4AC      =6077 DELAY1: CALL    TOFFPOL
               =6078         MMOV    A,RDELAY
04F7 B93F      =6087+          MOV     R1,#RDELAY
04F9 F1        =6088+          MOV     A,@R1
04FA 96F5      =6092         JNZ     DELAY1
04FC 83        =6093         RET
               =6094         SIZECHK
000B           =6097+  SIZE  SET   11
               =6098+;
               =6099+;*****************************************************************
               =6108 ;
               =6109         CODEBLK 8
07AF           =6144+          ORG     1967
               =6148 ;KBDPOL POLL STATUS OF KEYBOARD INPUT ROUTINE.
               =6149 ;       RETURN WITH ACC BIT 7 = 0 IF KEYBOARD INPUT HAS BEEN RECEIVED.
07AF BF05      =6150 KBDPOL: MOV     XPCODE,#5
07B1 74D1      =6151         CALL    XPTEST
               =6152         MMOV    A,KBDBUF
07B3 B93B      =6161+          MOV     R1,#KBDBUF
07B5 F1        =6162+          MOV     A,@R1
07B6 83        =6166         RET
               =6167         SIZECHK
0008           =6170+  SIZE  SET   8
               =6171+;
               =6172+;*****************************************************************
               =6181 $EJECT
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

                 6182 $      INCLUDE(.F8:LINK.MOD)
                =6183        CODEBLK 15
07B7             =6218+        ORG     1975
                 =6222 ;EPFET  FETCH DATA BYTE FROM EP INTERNAL RAM ADDRESSED BY SMALO.
                 =6223 EPFET:  MMOV    A,SMALO
07B7 B930        =6232+          MOV     R1,#SMALO
07B9 F1          =6233+          MOV     A,@R1
07BA F4D0        =6237        CALL    EPPASS
07BC 2380        =6238        MOV     A,#10000000B
07BE F4D0        =6239        CALL    EPPASS
07C0 F4D0        =6240        CALL    EPPASS
07C2 83          =6241        RET
                 =6242        SIZECHK
000C             =6245+  SIZE  SET   12
                 =6246+;
                 =6247+;*************************************************************
                 =6256 ;
                 =6257        CODEBLK 15
07C3             =6292+        ORG     1987
                 =6296 ;EPSTOR STORE DATA IN LDATA IN EP INTERNAL RAM AT <SMALO>
07C3 FA          =6297 EPSTOR: MOV     A,LDATA
07C4 F4D0        =6298        CALL    EPPASS
                 =6299        MMOV    A,SMALO
07C6 B930        =6308+          MOV     R1,#SMALO
07C8 F1          =6309+          MOV     A,@R1
07C9 537F        =6313        ANL     A,#01111111B
07CB F4D0        =6314        CALL    EPPASS
07CD F4D0        =6315        CALL    EPPASS
07CF 83          =6316        RET
                 =6317        SIZECHK
000D             =6320+  SIZE  SET   13
                 =6321+;
                 =6322+;*************************************************************
                 =6331 $EJECT
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

                =6332 ;       THE FOLLOWING UTILITIES INVOLVE INTERCHANGES BETWEEN THE MP AND EP.
                =6333 ;
                =6334         CODEBLK 11
07D0            =6369+          ORG     2000
                =6373 ;EPPASS PASSES A SINGLE PARAMETER BYTE TO THE EP THROUGH THE LINK.
                =6374 ;       WRITE THE CONTENTS OF THE ACC TO THE LINK;
                =6375 ;       RELEASE THE EP;
                =6376 ;       READ THE LINK INTO THE ACC;
                =6377 ;       RETURN.
07D0 8A30       =6378 EPPASS: ORL     P2,#00110000B           ;ENABLE LINK WRITES.
07D2 91         =6379         MOVX    @R1,A                   ;WRITE ACC TO LINK.
07D3 99FE       =6380         ANL     P1,#NOT ENGRAM          ;DISABLE BREAKPOINTS.
07D5 8902       =6381         ORL     P1,#ENBLNK              ;SET TO BREAK ON LINK REFERENCE.
07D7 F4DB       =6382         CALL    EPSTEP
07D9 81         =6383         MOVX    A,@R1
07DA 83         =6384         RET
                =6385         SIZECHK
0008            =6388+  SIZE  SET   11
                =6389+;
                =6390+;*****************************************************************
                =6399 ;
                =6400         CODEBLK 25
07DB            =6435+          ORG     2011
                =6439 ;EPSTEP RELEASES EP TO RUN IN PRESENT MODE UNTIL AN ANTICIPATED
                =6440 ;       HARDWARE BREAK OCCURS.
                =6441 ;       (DUE TO SINGLE STEPPING, LINK OPCODE FETCH, OR LINK DATA FETCH.)
                =6442 ;       MUST OCCUR WITHIN A FINITE NUMBER OF CYCLES (<40 MP CYCLES)
                =6443 ;       OR WATCHDOG TIMER WILL ASSUME A COMMUNICATIONS ERROR
                =6444 ;       BETWEEN THE MP AND EP.
07DB F4F4       =6445 EPSTEP: CALL    EPREL
07DD B90A       =6446         MOV     R1,#10
07DF 86F1       =6447 EPSTE1: JNI     EPSTE2
07E1 E9DF       =6448         DJNZ    R1,EPSTE1
07E3 8910       =6449         ORL     P1,#EPRSET
07E5 744F       =6450         CALL    EPBRK
07E7 B8BB       =6451         MOV     R0,#LOW(OV1BAS+OVSIZE)
07E9 746A       =6452         CALL    OVLOAD
07EB 99EF       =6453         ANL     P1,#NOT EPRSET
07ED B80E       =6454         MOV     LDATA,#0EH
07EF 249A       =6455         JMP     PERROR
07F1 744F       =6456 EPSTE2: CALL    EMBRK
07F3 83         =6457         RET
                =6458         SIZECHK
0019            =6461+  SIZE  SET   25
                =6462+;
                =6463+;*****************************************************************
                =6472 ;
                =6473 ;
                =6474 $EJECT
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

                =6475        CODEBLK 9
07F4            =6510+         ORG    2036
                =6514 ;EPREL  RELEASES EP TO RUN IN PRESENT MODE.
                =6515 ;       SEQUENCE IS AS FOLLOWS:
                =6516 ;       PUT MEMORY ARRAY IN EP MODE;
                =6517 ;       RAISE /SSTEP;
                =6518 ;       RETURN.
07F4 99F7       =6519 EPREL:  ANL    P1,#NOT CLRBFF          ;CLEAR BREAK F/F.
07F6 8908       =6520         ORL    P1,#CLRBFF              ;RE-ENABLE BREAK F/F.
07F8 9ABF       =6521         ANL    P2,#NOT 10000000B       ;ENABLE EP CONTROL OF MEM ARRAY
07FA 8904       =6522         ORL    P1,#00000100B           ;FREE EP TO RUN UNTIL BREAK.
07FC 83         =6523         RET
                =6524         SIZECHK
0009            =6527+  SIZE  SET   9
                =6528+;
                =6529+;*************************************************************
                =6530 ;
                =6539 ;
                =6540         CODEBLK 11
034F            =6580+         ORG    847
                =6584 ;EPBRK  REGAIN CONTROL OF MEMORY ARRAY FROM EP.
                =6585 ;       DROP /SSTEP;
                =6586 ;       WAIT 30 USECS.;
                =6587 ;       PUT MEMORY ARRAY IN MP MODE;
                =6588 ;       RETURN.
034F 99FB       =6589 EPBRK:  ANL    P1,#NOT 00000100B       ;FREEZE EMULATION PROCESSOR.
0351 8920       =6590         ORL    P1,#MODOUT              ;SIGNAL EP IS NOT RUNNING USER CODE.
0353 B905       =6591         MOV    R1,#5
0355 E955       =6592         DJNZ   R1,$                    ;DELAY FOR EP TO FINISH INSTRUCTION.
0357 8A40       =6593         ORL    P2,#01000000B           ;SEIZE CONTROL OF MEM ARRAY.
0359 83         =6594         RET
                =6595         SIZECHK
000B            =6598+  SIZE  SET   11
                =6599+;
                =6600+;*************************************************************
                =6609 ;
                =6610 ;
                =6611         CODEBLK 16
035A            =6651+         ORG    858
                =6655 ;OVSWAP OVERLAY SWAP.
                =6656 ;       SWAPS BLOCK OF DATABYTES (USER'S PROGRAM) BETWEEN MP RAM & EP PM.
035A B865       =6657 OVSWAP: MOV    R0,#OVBUF+OVSIZE
035C B917       =6658         MOV    R1,#OVSIZE
035E 2340       =6659         MOV    A,#01000000B
0360 3A         =6660         OUTL   P2,A
0361 C8         =6661 OVSW1:  DEC    R0
0362 C9         =6662         DEC    R1
0363 81         =6663         MOVX   A,@R1
0364 20         =6664         XCH    A,@R0
0365 91         =6665         MOVX   @R1,A
0366 F9         =6666         MOV    A,R1
0367 9661       =6667         JNZ    OVSW1
0369 83         =6668         RET
                =6669         SIZECHK
0010            =6672+  SIZE  SET   16
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

                =6673+;
                =6674+;*****************************************************************
                =6683 ;
                =6684          CODEBLK 14
036A            =6724+           ORG    874
                =6728 ;OVLOAD OVERLAY LOAD.
                =6729 ;        MOVES BLOCK OF DATABYTES (ASSEMBLED SOURCE) FROM PG3 TO EP PM.
                =6730 ;        TOP OF DATA BLOCK LOADED AND BLOCK LENGTH DETERMINED BY R0 AND R1.
036A B917       =6731 OVLOAD:  MOV    R1,#OVSIZE
036C 2340       =6732          MOV    A,#01000000B
036E 3A         =6733          OUTL   P2,A
036F C8         =6734 MML01:   DEC    R0
0370 C9         =6735          DEC    R1
0371 F8         =6736          MOV    A,R0
0372 E3         =6737          MOVP3  A,@A
0373 91         =6738          MOVX   @R1,A
0374 F9         =6739          MOV    A,R1
0375 966F       =6740          JNZ    MML01
0377 83         =6741          RET
                =6742          SIZECHK
000E            =6745+  SIZE   SET    14
                =6746+;
                =6747+;*****************************************************************
                =6756 $EJECT
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

                =6757 ;
                =6758 ;=====================================
                =6759 ;
                =6760 ;       THE REST OF THIS MODULE CONTAINS THE MINI-MONITORS WHICH OVERLAY
                =6761 ;       THE EMULATION PROCESSOR PROGRAM RAM TO GIVE THE
                =6762 ;       MASTER PROCESSOR ACCESS TO INTERNAL REGISTERS AND RAM OF THE EP.
                =6763 ;
                =6764 ;=====================================
                =6765 ;
                =6766         DATABLK 22
0378            =6771+          ORG     888
                =6775 ;
                =6776 ;OV0-   OVERLAY TO BREAK EP EXECUTION AND JUMP TO LOCATION 009H.
                =6777 ;       LOCATION 009H REACHED WITH TOP-OF-STACK = RETURN ADDRESS+2
                =6778 ;       DUE TO FORCED "CALL" DURING WHICH PC WAS INCREMENTED.
                =6779 ;       LOCS 003H & 007H CALL 009H TO SIMULATE SAME CONDITION
                =6780 ;       IF BREAK OCCURS DURING INTERRUPT CYCLE.
                =6781 ;       SOURCE CODE FOR MINI-MONITOR OVERLAYED OVER LOW ORDER PROGRAM RAM.
                =6782 ;
0378            =6783 OV0BAS  EQU     $
0378            =6784 ORG     OV0BAS
0378 1409       =6785         CALL    009H
037A 00         =6786         NOP
                =6787 ;
037B            =6788 ORG     OV0BAS+003H
037B 1409       =6789         CALL    009H
037D 00         =6790         NOP
037E 00         =6791         NOP
                =6792 ;
037F            =6793 ORG     OV0BAS+007H
037F 1409       =6794         CALL    009H
0381 00         =6795         NOP
0382 00         =6796         NOP
0383 00         =6797         NOP
0384 00         =6798         NOP
0385 00         =6799         NOP
0386 00         =6800         NOP
0387 00         =6801         NOP
0388 00         =6802         NOP
0389 00         =6803         NOP
038A 00         =6804         NOP
038B 00         =6805         NOP
                =6806 ;
038C            =6807 ORG     OV0BAS+014H
038C 0409       =6808         JMP     009H
                =6809 ;
                =6810         SIZECHK
0016            =6813+  SIZE  SET   22
                =6814+;
                =6815+;*************************************************************
                =6824 $EJECT
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

                =6825        DATABLK 22
038E            =6830+         ORG     910
                =6834 ;
                =6835 ;OV3-    OVERLAY TO SAVE STATUS DATA AFTER BREAK.
                =6836 ;        ACC, TIMER/COUNTER, PSW (WITH F1), & RAM LOC 0 PASSED SEQUENTIALLY
                =6837 ;        TO MP.
                =6838 ;        SOURCE CODE FOR MINI-MONITOR OVERLAYED OVER LOW ORDER PROGRAM RAM.
                =6839 ;
038E            =6840 OV3BAS   EQU     $
038E            =6841 ORG      OV3BAS
038E 0400       =6842          JMP     000H
0390 00         =6843          NOP
                =6844 ;
0391            =6845 ORG      OV3BAS+003H
0391 83         =6846          RET
0392 00         =6847          NOP
0393 00         =6848          NOP
0394 00         =6849          NOP
                =6850 ;
0395            =6851 ORG      OV3BAS+007H
0395 83         =6852          RET
0396 00         =6853          NOP
                =6854 ;
0397            =6855 ORG      OV3BAS+009H
0397 90         =6856          MOVX    @R0,A
0398 42         =6857          MOV     A,T
0399 90         =6858          MOVX    @R0,A
039A C7         =6859          MOV     A,PSW
039B 7611       =6860          JF1     OV3B1
039D 53F7       =6861          ANL     A,#11110111B
0311            =6862 OV3B1    EQU     $-(LOW OV3BAS)
039F 90         =6863          MOVX    @R0,A
03A0 C5         =6864          SEL     RB0
03A1 F8         =6865          MOV     A,R0
03A2 0409       =6866          JMP     009H
                =6867 ;
                =6868          SIZECHK
0016            =6871+  SIZE   SET   22
                =6872+;
                =6873+;*********************************************************
                =6882 $EJECT
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

                =6883          DATABLK 22
03A4            =6888+           ORG     932
                =6892 ;
                =6893 ;OV1-    OVERLAY 1 TO GIVE MP ACCESS TO EP RAM LOCS. 01H-7FH.
                =6894 ;        SOURCE CODE FOR MINI-MONITOR OVERLAYED OVER LOW ORDER PROGRAM RAM.
                =6895 ;
03A4            =6896 OV1BAS   EQU     $
                =6897 ;
03A4 040R       =6898          JMP     OV1B1
03A6 00         =6899          NOP
                =6900 ;
03A7            =6901 ORG      OV1BAS+003H
03A7 83         =6902          RET
03A8 00         =6903          NOP
03A9 00         =6904          NOP
03AA 00         =6905          NOP
                =6906 ;
03AB            =6907 ORG      OV1BAS+007H
03AB 83         =6908          RET
03AC 00         =6909          NOP
                =6910 ;
03AD            =6911 ORG      OV1BAS+009H
03AD 90         =6912          MOVX    @R0,A
                =6913 ;
000A            =6914 OV1B1    EQU     $-OV1BAS
                =6915 ;
03AE 80         =6916          MOVX    A,@R0
03AF A8         =6917          MOV     R0,A
03B0 80         =6918          MOVX    A,@R0
03B1 F213       =6919          JB7     OV1B2
03B3 28         =6920          XCH     A,R0
03B4 A0         =6921          MOV     @R0,A
03B5 0409       =6922          JMP     009H
                =6923 ;
0313            =6924 OV1B2    EQU     $-LOW OV1BAS
                =6925 ;
03B7 F0         =6926          MOV     A,@R0
03B8 0409       =6927          JMP     009H
                =6928 ;
                =6929          SIZECHK
0016            =6932+  SIZE   SET   22
                =6933+;
                =6934+;*************************************************************
                =6943 $EJECT
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

                =6944         DATABLK 23
03BA            =6949+          ORG     954
                =6953 ;
                =6954 ;OV2--  OVERLAY TO RESTORE EP STATUS SAVED ON BREAK AND RESUME USER'S PROGRAM.
                =6955 ;       SOURCE CODE FOR MINI-MONITOR OVERLAYED OVER LOW ORDER PROGRAM RAM.
                =6956 ;
03BA            =6957 OV2BAS  EQU     $
03BA            =6958 ORG     OV2BAS
03BA 0400       =6959         JMP     000H
03BC 00         =6960         NOP
                =6961 ;
03BD            =6962 ORG     OV2BAS+003H
03BD 83         =6963         RET
03BE 00         =6964         NOP
03BF 00         =6965         NOP
03C0 00         =6966         NOP
                =6967 ;
03C1            =6968 ORG     OV2BAS+007H
03C1 83         =6969         RET
03C2 00         =6970         NOP
                =6971 ;
03C3            =6972 ORG     OV2BAS+009H
03C3 90         =6973         MOVX    @R0,A
                =6974 ;
03C4 80         =6975         MOVX    A,@R0
03C5 A8         =6976         MOV     R0,A
03C6 80         =6977         MOVX    A,@R0
03C7 D7         =6978         MOV     PSW,A
03C8 A5         =6979         CLR     F1
03C9 B5         =6980         CPL     F1
03CA 7213       =6981         JB3     OV2B1
03CC A5         =6982         CLR     F1
                =6983 ;
0313            =6984 OV2B1   EQU     $-LOW OV2BAS
                =6985 ;
03CD 80         =6986         MOVX    A,@R0
03CE 62         =6987         MOV     T,A
03CF 80         =6988         MOVX    A,@R0
03D0 93         =6989         RETR
                =6990         SIZECHK
0017            =6993+  SIZE  SET   23
                =6994+;
                =6995+;*****************************************************************
                =7004 $EJECT
```

```
LOC  OBJ          LINE       SOURCE STATEMENT

                  7005 ;
                  7006         CODEBLK 11
03D1              7046+          ORG     977
03D1 8A80         7050 XPTEST: ORL     P2,#80H
03D3 0A           7051         IN      A,P2
03D4 9A7F         7052         ANL     P2,#(NOT 80H)
03D6 F2D9         7053         JB7     $+3
03D8 83           7054         RET
03D9 F5           7055         SEL     MB1
03DA 0400         7056         JMP     800H
                  7057         SIZECHK
000B              7060+  SIZE  SET   11
                  7061+;
                  7062+;*************************************************************
                  7071 ;
                  7072         CODEBLK 13
03DC              7112+          ORG     988
03DC 28432931     7116         DB      '(C)1979 INTEL'
03E0 39373920
03E4 494E5445
03E8 4C
                  7117         SIZECHK
000D              7120+  SIZE  SET   13
                  7121+;
                  7122+;*************************************************************
                  7131 ;
                  7132 ;
                  7133         RSOURCE
0100              7135+          PGSIZE  SET   ORGPG0-000H    ;BYTES USED ON PAGE 0
00FD              7136+          PGSIZE  SET   ORGPG1-100H    ;BYTES USED ON PAGE 1
0100              7137+          PGSIZE  SET   ORGPG2-200H    ;BYTES USED ON PAGE 2
00E9              7138+          PGSIZE  SET   ORGPG3-300H    ;BYTES USED ON PAGE 3
00FD              7139+          PGSIZE  SET   ORGPG4-400H    ;BYTES USED ON PAGE 4
00F1              7140+          PGSIZE  SET   ORGPG5-500H    ;BYTES USED ON PAGE 5
00FF              7141+          PGSIZE  SET   ORGPG6-600H    ;BYTES USED ON PAGE 6
00FD              7142+          PGSIZE  SET   ORGPG7-700H    ;BYTES USED ON PAGE 7
                  7143+$EJECT
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

                7145 ;*********************************************************
                7146 ;
                7147 ;       FILL ALL UNUSED MEMORY LOCATIONS WITH NOP OPCODES
                7148 ;
                7149 ;*********************************************************
                7150 ;
                7151 $GEN
                7158 ;
01FD            7160           ORG   ORGPG1
                7161         REPT (200H - ORGPG1)
-               7162           DB    0
                7163         ENDM
01FD 00         7164+          DB    0
01FE 00         7165+          DB    0
01FF 00         7166+          DB    0
                7168 ;
                7175 ;
03E9            7177           ORG   ORGPG3
                7178         REPT (400H - ORGPG3)
-               7179           DB    0
                7180         ENDM
03E9 00         7181+          DB    0
03EA 00         7182+          DB    0
03EB 00         7183+          DB    0
03EC 00         7184+          DB    0
03ED 00         7185+          DB    0
03EE 00         7186+          DB    0
03EF 00         7187+          DB    0
03F0 00         7188+          DB    0
03F1 00         7189+          DB    0
03F2 00         7190+          DB    0
03F3 00         7191+          DB    0
03F4 00         7192+          DB    0
03F5 00         7193+          DB    0
03F6 00         7194+          DB    0
03F7 00         7195+          DB    0
03F8 00         7196+          DB    0
03F9 00         7197+          DB    0
03FA 00         7198+          DB    0
03FB 00         7199+          DB    0
03FC 00         7200+          DB    0
03FD 00         7201+          DB    0
03FE 00         7202+          DB    0
03FF 00         7203+          DB    0
                7205 ;
04FD            7207           ORG   ORGPG4
                7208         REPT (500H - ORGPG4)
-               7209           DB    0
                7210         ENDM
04FD 00         7211+          DB    0
04FE 00         7212+          DB    0
04FF 00         7213+          DB    0
                7215 ;
05FF            7217           ORG   ORGPG5
                7218         REPT (600H - ORGPG5)
```

```
 LOC  OBJ        LINE       SOURCE STATEMENT

 -               7219          DB   0
                 7220        ENDM
 05FF 00         7221+         DB   0
                 7223 ;
 06FF            7225          ORG  ORGPG6
                 7226        REPT (700H - ORGPG6)
 -               7227          DB   0
                 7228        ENDM
 06FF 00         7229+         DB   0
                 7231 ;
 07FD            7233          ORG  ORGPG7
                 7234        REPT (800H - ORGPG7)
 ..              7235          DB   0
                 7236        ENDM
 07FD 00         7237+         DB   0
 07FE 00         7238+         DB   0
 07FF 00         7239+         DB   0
                 7241 ;
                 7242 $EJECT
```

```
TTYOUT  539#  4428   4430
TYPE   1176#  1429   1579  1585  1748  1771  1777  1822  2448  2550  3011  3072  4768  4966  5171
UPDAD1 2265#  2558   3371
UPDADR 2195   2248#
VERSNO 1050#
WBRK   1522#  1928
WDISP  2010   2030   2269  2274  2560  3373  5948#
WDISP1 5988   5991#
XPCODE  837#  1410   1539  2318  5799  5949  6150
XPTEST 1411   1540   2319  5800  5950  6151  7050#
ZERO    684#  1570   1586  1778  2494  3428  3860  5667

CROSS REFERENCE COMPLETE
```

```
BRKFIL 2433  2437#
BRKNXT 2459# 2499
BUFCNT 1266# 3446  3519  3534  3970  3990  4046
BUFLEN  662# 1329  3875
BYTEI1 3554  3628#
BYTEIN 3432  3451  3470  3489  3525  3627#
BYTEO  3995  4010  4025  4028  4039  4087  4154#
CGO    2986  3002#
CGONB  3019  3038#
CGOPAT 3022  3025#
CGOSS  3021  3034#
CGOTRR 3023  3033#
CGOWB  3020  3026#
CHARCR 3393# 4119
CHARIN 3417  3549  3699  3749#
CHARLF 3394# 4121
CHARNO  598# 1313  1349  1370  5648  5871  5887  5990
CHARO  3891  3896  3977  3980  4031  4037  4120  4122  4392#
CHKERR 3577  3598#
CHKSUM  803# 3428  3566  3573  3664  3665  3860  4074  4081  4155  4156
CI0    4531# 4531  4532
CI1    4533# 4533  4534  4536
CI2    4537# 4585
CI3    4539  4542#
CI4    4541  4545#
CIN    3749  4529#
CKSMOK 3551  3578#
CLEAR  1974  5870#
CLRBFF  534# 6519  6520
CMDINT 1836  1842  1845  1855#
CMPMAS 3871  4673  5343#
CMPRET 5395#
CNTRLZ 3395# 3418  3420  3895
CNTTBL 3077  3080#
CNTTRR 3083  3084  3091#
CO1    4427# 4475
CO2    4427  4430#
CO3    4429  4433#
CODEBL  199# 1398  1526  1945  2155  2232  2298  2365  2516  2649  2677  3146  3397  3615  3683  3733
       3769  3919  4104  4139  4178  4232  4358  4493  4618  4693  4722  4915  5136  5223  5262  5307
       5412  5759  5838  5907  6031  6109  6183  6257  6334  6400  6475  6540  6611  6684  7006  7072
COMCBR 2407  2432#
COMFIL 1428  1432  2426  4639#
COMGOR 2429  2992#
COMSBR 2406  2436#
COMSIZ 1596  1899#
CTAB   1557  1898#
CURDIG  928# 5478  5484  5647  5648
DATABL  244# 1332  1875  6766  6825  6883  6944
DATO   4029  4033#
DATO1  4034# 4060
DBLANK 2215  5872# 5874
DBPNT  2036  2085#
DBRK   1519# 1924
DCB    2045  2100#
DDABRK 2053  2122#
DDAMEM 2049  2116#
```

```
PSEGLO 000C   RDELAY 003F   RECDON 02CC   RECTYP 0042   REGC   0044   REORG  0005   RERKOR 0198   RIN1   0011
ROTCNT 0003   ROTPAT 0002   RSOURC 0012   SCAN3  077C   SCAN5  0788   SCAN8  079D   SEGMAP 0046   SING   001A
SIZE   000D   SIZECH 0011   SMAHI  0031   SMALO  0030   STRCOM 001D   STRGOC 002C   STRMEM 0026   STRTMP 0040
STRUTL 0019   STSAVE 0500   TCRLFO 01D2   TIINT  074E   TIKET1 07A8   TOFPOL 07AC   TTYOUT 0040   TYPE   0037
UPDAD1 017C   UPDADR 0178   VERSNO 0029   WBRK   0016   WDISP  06D8   WDISP1 06EE   XPCODE 0007   XPTEST 03D1
ZERO   0000

ASSEMBLY COMPLETE,  NO ERRORS
```

ISIS-II ASSEMBLER SYMBOL CROSS REFERENCE, V2.1 PAGE 1

```
?A      105#  1614  1629  1637  1650  1658  1721  1787  1806  1818  1977  1985  1999  2177  2388  2444
        2546  2586  2719  2788  2796  2843  2857  2865  2898  2910  2928  2943  2958  2973  3007  3068
        3096  3207  3215  3226  3234  3245  3253  3264  3272  3288  3302  3310  3323  3331  3350  3358
        3434  3442  3453  3461  3472  3480  3491  3499  3515  3562  3583  3637  3958  3966  3986  4001
        4016  4070  4393  4401  4592  4764  4788  4803  4819  4841  4859  4875  4962  4986  5001  5017
        5042  5095  5167  5180  5196  5348  5360  5374  5386  5456  5464  5546  5580  5592  5600  5692
        5704  5712  5726  5807  5956  6060  6068  6083  6157  6228  6304
?ASAVE  1235#  5460  5466  5722  5728
?B      1280#  4413  4413  4413  4419  4459  4469  4569  4579
?B0PNT  117#  746  754#  763  771#  780  788#  797  805#  814  822#  831  839#
?B0R2   110#  754
?B0R3   111#  771
?B0R4   112#  788
?B0R5   113#  805
?B0R6   114#  822
?B0R7   115#  839
?B1PNT  126#  859  867#  880  888#  901  909#  922  930#  943  951#
?B1R2   119#  867
?B1R3   120#  888
?B1R4   121#  909
?B1R5   122#  930
?B1R6   123#  951
?B1R7   124#
?BCODE  1163#  1561  1561  1561  1567  1610  1616  2390
?BINOP  415#  1817  2387  2439  2909  3632  5179  5359  5385
?BITSO  4217#  4411
?BUFCN  1262#  3438  3444  3511  3517  3532  3542  3962  3968  3982  3988  4044  4054
?BUFLE  649#
?CHARN  585#  5876
?CHKSU  791#  3426  3426  3426  3558  3564  3571  3571  3858  3858  3858  4066  4072  4079  4079
?CONST  104#  585  586  590  594  601  602  606  610  617  618  622  626  633  634  638
        642  649  650  654  658  671  672  676  680  686  687  691  695  701  702  706
        710  716  717  721  725  4217  4218  4222  4226
?CURDI  912#
?DEBNC  617#
?DSPTI  1037#  3092  3098
?DSPTM  888#
?EMAHI  1136#  5388
?EMALO  1127#  5362
?EPACC  965#  2969  2975  3211  3217
?EPPCH  1010#  2761  2777  2912  3354  3360
?EPPCL  1001#  2734  2750  2806  2814  2819  3327  3333
?EPPSW  974#  2792  2798  2839  2845  2894  2900  2939  2945  3249  3255  3284  3290
?EPR0   992#  2924  2930  3268  3274  4855  4861  5060  5082
?EPTIM  983#  2954  2960  3230  3236
?FORM1  295#  1615  1634  1655  1688  1695  1722  1745  1788  1807  1819  1982  2000  2013  2178  2389
        2441  2445  2547  2587  2720  2735  2742  2762  2769  2793  2811  2818  2844  2862  2877  2899
        2911  2929  2944  2959  2974  3008  3069  3097  3212  3231  3250  3269  3289  3307  3328  3355
        3439  3458  3477  3496  3516  3531  3563  3584  3634  3638  3807  3814  3834  3841  3963  3987
        4002  4017  4043  4071  4263  4270  4290  4297  4322  4398  4439  4458  4550  4568  4593  4645
        4652  4765  4789  4804  4820  4842  4860  4876  4963  4987  5002  5018  5043  5061  5068  5096
        5168  5181  5197  5349  5361  5375  5387  5461  5584  5547  5581  5597  5616  5623  5693  5709
        5727  5808  5957  5971  6065  6084  6158  6229  6305
?FORM2  319#  1638  1659  1692  1702  1986  2739  2749  2765  2776  2797  2815  2825  2866  3216  3235
        3254  3273  3311  3332  3359  3443  3462  3481  3500  3811  3821  3838  3848  3967  4267  4277
        4294  4304  4402  4649  4659  5065  5081  5465  5601  5620  5636  5713  6069
?FORM3  339#  1755  2023  2452  2887  3541  3651  4053  4332  4449  4468  4560  4578  5520  5981
```

```
?FORM4  356#
?FORM5  388#  1560  1576  1611  1630  1651  1684  1718  1768  1784  1803  1978  1996  2174  2250  2331
       2466   2484  2543  2583  2716  2731  2758  2789  2807  2840  2858  2895  2925  2940  2955  2970
       3004   3065  3093  3208  3227  3246  3265  3285  3303  3324  3351  3425  3435  3454  3473  3492
       3512   3559  3580  3803  3830  3857  3959  3983  3998  4013  4067  4259  4286  4394  4412  4589
       4641   4761  4785  4800  4816  4838  4856  4872  4959  4983  4998  5014  5039  5057  5092  5164
       5193   5345  5371  5457  5543  5577  5593  5612  5657  5675  5689  5705  5723  5804  5877  5953
       6061   6080  6154  6225  6301
?H     1298#  4262  4278  4323  4333
?HBITH 1028#  4258  4266  4271
?HBITL 1019#  4285  4293  4298
?HEXBU 1324#
?HREGA 1055#
?HREGB 1064#
?HREGC 1073#
?HREGD 1082#
?HREGE 1091#
?HREGF 1100#
?ITMP   774#  1687  1703  1710  1710  1717  1723  1730  1730
?KBDBU 1208#  2332  2332  2332  2338  5615  5637  5803  5809  6153  6159
?KEY    757#  2582  2588  2595  2595
?KEYFL  933#
?KEYLO 1217#  5542  5548  5658  5658  5658  5664
?LASTK  891#  5611  5619  5624  5631  5631  5676  5676  5676
?LDATA  740#  2810  2826  2833  2833  3633  3639  3646  3646  3652  3658  3658  4644  4660  4667  4667
       5056   5064  5069  5076  5076
?LENGT 1333#  1388  1399#  1442  1527#  1866  1876#  1936  1946#  2146  2156#  2223  2233#  2289  2299#  2356
       2366#  2507  2517#  2640  2650#  2667  2678#  3137  3147#  3383  3398#  3606  3616#  3674  3684#  3723
       3734#  3758  3770#  3909  3920#  4095  4105#  4130  4140#  4169  4179#  4206  4233#  4348  4359#  4484
       4494#  4608  4619#  4684  4694#  4713  4723#  4906  4916#  5127  5137#  5213  5224#  5253  5263#  5298
       5308#  5402  5413#  5750  5760#  5829  5839#  5898  5908#  6022  6032#  6100  6110#  6173  6184#  6248
       6258#  6323  6335#  6391  6401#  6464  6476#  6530  6541#  6601  6612#  6675  6685#  6748  6767#  6816
       6826#  6874  6884#  6935  6945#  6996  7007#  7063  7073#  7123
?MEMHI 1154#  3806  3822  3997  4003
?MEMLO 1145#  3833  3849  4012  4018  4640  4648  4653
?MINDX  156#   967   971#  971   976   980#  980   985   989#  989   994   998#  998  1003  1007#  1007
       1012  1016#  1016  1021  1025#  1025  1030  1034#  1034  1039  1043#  1043  1048  1052#  1052  1057
       1061#  1061  1066  1070#  1070  1075  1079#  1079  1084  1088#  1088  1093  1097#  1097  1102  1106#
       1106  1111  1115#  1115  1120  1124#  1124  1129  1133#  1133  1138  1142#  1142  1147  1151#  1151
       1156  1160#  1160  1165  1169#  1169  1174  1178#  1178  1183  1187#  1187  1192  1196#  1196  1201
       1205#  1205  1210  1214#  1214  1219  1223#  1223  1228  1232#  1232  1237  1241#  1241  1246  1250#
       1250  1255  1259#  1259  1264  1268#  1268  1273  1277#  1277  1282  1286#  1286  1291  1295#  1295
       1300  1304#  1304  1309  1313#  1313  1317  1321#  1321  1325  1329#  1329
?MSAVE  158#   587   603   619   635   651   673   688   703   718   742   759   776   793   810   827
        851   872   893   914   935   967   976   985   994  1003  1012  1021  1030  1039  1048  1057
       1066  1075  1084  1093  1102  1111  1120  1129  1138  1147  1156  1165  1174  1183  1192  1201
       1210  1219  1228  1237  1246  1255  1264  1273  1282  1291  1300  1309  1317  1325  4219
?NCOLS  601#
?NEG1   716#  2330  5674
?NEXTP 1199#  2251  2251  2251  2257  5878  5878  5878  5884  5952  5958  5972  5982
?NREPT 1226#  5576  5582  5596  5602
?NUMCO 1181#  1654  1660  2173  2179  2467  2407  2467  2473  2715  2721
?OPTIO 1190#  1633  1639  1683  1691  1696  1783  1789  1802  1808
?OVBUF 1316#
?OVSIZ  633#
?PLUS1  686#  2465
?PLUS3  701#  2249
```

```
?R1      99#  4289  4305  4312  4312
?RAM    103#   965   966   974   975   983   984   992   993  1001  1002  1010  1011  1019  1020  1028
       1029  1037  1038  1046  1047  1055  1056  1064  1065  1073  1074  1082  1083  1091  1092  1100
       1101  1109  1110  1118  1119  1127  1128  1136  1137  1145  1146  1154  1155  1163  1164  1172
       1173  1181  1182  1190  1191  1199  1200  1208  1209  1217  1218  1226  1227  1235  1236  1244
       1245  1253  1254  1262  1263  1271  1272  1280  1281  1289  1290  1298  1299
?RB0    101#   740   741   745   757   758   762   774   775   779   791   792   796   808   809   813
        825   826   830
?RB1    102#   849   850   854   858   870   871   875   879   891   892   896   900   912   913   917
        921   933   934   938   942
?RDELA 1244#  5688  5694  5708  5714  6064  6070  6079  6085
?RECTY 1271#  3495  3501  3579  3585
?REGC  1289#  4397  4403  4440  4450  4551  4561  4508  4594
?ROTCN  870#
?ROTPA  849#  5505  5512  5512  5521  5527  5527
?RSAVE  144#   591   595   607   611   623   627   639   643   655   659   677   681   692   696   707
        711   722   726   746   763   780   797   814   831   855   859   876   880   897   901   918
        922   939   943  4223  4227
?SEGMA 1308#
?SIZE   255#  1383  1437  1861  1931  2141  2218  2284  2351  2502  2635  2662  3132  3378  3601  3669
       3718  3753  3904  4090  4125  4164  4201  4343  4479  4603  4679  4708  4901  5122  5208  5248
       5293  5397  5745  5824  5893  6017  6095  6168  6243  6318  6386  6459  6525  6596  6670  6743
       6811  6869  6930  6991  7058  7118
?SMAHI 1118#  2405  2485  2485  2491  2757  2765  2770  3457  3463  3802  3810  3815  4784  4790  4982
       4988  5182  5370  5376
?SMALO 1109#  2730  2738  2743  2861  2867  2878  2888  3306  3312  3476  3482  3829  3837  3842  4799
       4805  4815  4821  4837  4843  4871  4877  4997  5003  5013  5019  5038  5044  5091  5097  5192
       5198  5344  5350  6224  6230  6300  6306
?START 1339#  1383  1383  1391  1405#  1437  1437  1445  1533#  1861  1861  1869  1882#  1931  1931  1939
       1957#  2141  2141  2149  2162#  2218  2218  2226  2244#  2284  2284  2292  2310#  2351  2351  2359
       2382#  2502  2502  2510  2533#  2635  2635  2643  2656#  2662  2662  2670  2699#  3132  3132  3140
       3173#  3378  3378  3386  3414#  3601  3601  3609  3622#  3669  3669  3677  3695#  3718  3718  3726
       3745#  3753  3753  3761  3796#  3904  3904  3912  3951#  4090  4090  4098  4116#  4125  4125  4133
       4151#  4164  4164  4172  4190#  4201  4201  4209  4254#  4343  4343  4351  4385#  4479  4479  4487
       4525#  4603  4603  4611  4635#  4679  4679  4687  4700#  4708  4708  4716  4754#  4901  4901  4909
       4952#  5122  5122  5130  5150#  5208  5208  5216  5235#  5248  5248  5256  5279#  5293  5293  5301
       5334#  5397  5397  5405  5449#  5745  5745  5753  5791#  5824  5824  5832  5865#  5893  5893  5901
       5939#  6017  6017  6025  6053#  6095  6095  6103  6146#  6168  6168  6176  6220#  6243  6243  6251
       6294#  6318  6318  6326  6371#  6386  6386  6394  6437#  6459  6459  6467  6512#  6525  6525  6533
       6582#  6596  6596  6604  6653#  6670  6670  6678  6726#  6743  6743  6751  6773#  6811  6811  6819
       6832#  6869  6869  6877  6890#  6930  6930  6938  6951#  6991  6991  6999  7048#  7058  7058  7066
       7114#  7118  7118  7126
?STRTM 1253#  1981  1987  1995  2001  2014  2024
?TYPE  1172#  1577  1577  1577  1583  1746  1756  1769  1769  1769  1775  1820  2440  2446  2453  2542
       2548  3003  3009  3064  3070  4760  4766  4958  4964  5163  5169
?UNARY  459#  1744  2012  2876  3530  4042  4321  4438  4457  4549  4567  5503  5970
?VERSN 1046#
?XPCOD  825#
?ZERO   671#  1559  1575  1767  2483  3424  3856  5656
AFETCH 4704  4759#
ASAVE  1239#  5468  5730
ASCERR 3704  3711  3715#
B      1284#  4415  4421  4461  4529  4571
BCODE  1167#  1563  1569  1598  1618  2392
BITSO  4230#  4422
BRKEND 2462  2500#
BRKERR 3080  3088#
```

```
 LOC  OBJ        LINE       SOURCE STATEMENT

                 7243        END

USER SYMBOLS
?A      0004   ?ASAVE 0002   ?B      0002   ?B0PNT 0008   ?B0R2  0003   ?B0R3  0004   ?B0R4  0005   ?B0R5  0006
?B0R6   0007   ?B0R7  0008   ?B1PNT 0007   ?B1R2  0003   ?B1R3  0004   ?B1R4  0005   ?B1R5  0006   ?B1R6  0007
?B1R7   0008   ?BCODE 0002   ?BINOP 0022   ?BITS0 0003   ?BUFCN 0002   ?BUFLE 0003   ?CHARN 0003   ?CHKSU 0000
?CONST  0003   ?CURDI 0001   ?DEBNC 0003   ?DSPTI 0002   ?DSPTM 0000   ?EMAHI 0002   ?EMALO 0002   ?EPACC 0002
?EPPCH  0002   ?EPPCL 0002   ?EPPSW 0002   ?EPR0  0002   ?EPTIM 0002   ?FORM1 0016   ?FORM2 0018   ?FORM3 001A
?FORM4  001C   ?FORM5 001E   ?H      0002   ?HBITH 0002   ?HBITL 0002   ?HEXDU 0003   ?HREGA 0002   ?HREGB 0002
?HREGC  0002   ?HREGD 0002   ?HREGE 0002   ?HREGF 0002   ?ITMP  0000   ?KBDDU 0002   ?KEY   0000   ?KEYFL 0001
?KEYLO  0002   ?LASTK 0001   ?LDATA 0000   ?LENGT 000D   ?MEMHI 0002   ?MEMLO 0002   ?MINDX 0075   ?MSAVE 0001
?NCOLS  0003   ?NEG1  0003   ?NEXTP 0002   ?NREPT 0002   ?NUMCO 0002   ?OPTIO 0002   ?OVBUF 0003   ?OVSIZ 0003
?PLUS1  0003   ?PLUS3 0003   ?R1    0000   ?RAM   0002   ?RE0   0000   ?RB1   0001   ?RDELA 0002   ?RECIV 0002
?REGC   0002   ?ROTCN 0001   ?ROTPA 0001   ?RSAVE 0000   ?SEGMA 0003   ?SIZE  000E   ?SMAHI 0002   ?SMALO 0002
?START  03DC   ?STRTM 0002   ?TYPE  0002   ?UNARY 002A   ?VERSN 0002   ?XPCOD 0000   ?ZERO  0003   AFETCH 0678
ASAVE   003E   ASCERR 01C9   B       0043   BCODE  0036   BITS0 000B   BRKEND 024D   BRKERR 04A6   BRKFIL 022E
BRKNXT  0234   BUFCNT 0041   BUFLEN 0010   BYTEI1 00F2   BYTEIN 00F0   BYTEO  01DB   CGU    046B   CGUNB  047C
CGOPAT  0476   CGOSS  0480   CGOTRA 0480   CGOWB  0476   CHARCK 000D   CHARIN 01CD   CHARLF 000A   CHARNO 0008
CHARO   05BD   CHKERR 02E1   CHKSUM 0005   CI0    064D   CI1    0651   CI2    0659   CI3    0662   CI4    0665
CIN     0649   CKSMOK 02DB   CLEAR  05F1   CLRBFT 0008   CMDINT 00DA   CMFMAS 05E2   CMPRET 05F0   CNTRLZ 001A
CNTTBL  04A1   CNTTRA 04AA   CO1    05C5   CO2    05CB   CO3    05CF   CODEBL 0006   CUMCBR 0228   COMFIL 02E5
COMGOR  0461   COMSBR 022C   COMSIZ 0003   CTAB   0023   CURDIG 0005   DATABL 000C   DATO   062C   DATO1  062E
DBLANK  05F5   DBPNT  0144   DBRK   0015   DCB    015A   DDABRK 0167   DDAMEM 0161   DEBNCE 0008   DECLNR 0003
DECSM1  021F   DECSMA 02F4   DELAY  04F2   DELAY1 04F5   DERROR 0131   DFILL  014B   DGO    0149   DGPATS 00EF
DGR     015D   DINTRG 0169   DLST   014E   DMOD   0146   DNOBRK 016B   DONE   02E0   DPA    0172   DPRBRK 0165
DPRMEM  015F   DREC   0151   DREL   0154   DRM    0163   DRUN   013E   DSB    0157   DSGNON 0137   DSPACC 06D3
DSPHI   018E   DSPLO  0194   DSPM1  0192   DSPMID 0190   DSPTIM 0028   DSPTMP 0006   DSS    016F   DTR    0175
DWBRK   016D   ELSIF1 00D7   ELSIF2 00E5   EMAHI  0033   EMALO  0032   ENBLNK 0002   ENBRAM 0001   ENDF1  059E
ENDFIL  0596   ENDREC 0641   EOFREC 05AE   EPACC  0020   EPBRK  034F   EPCNT  0441   EPCON1 041F   EPCONT 0415
EPFET   07B7   EPPASS 07D0   EPPCHI 0025   EPPCLO 0024   EPPSW  0021   EPR0   0023   EPREL  07F4   EPRET  04C7
EPRSET  0010   EPRUN  0480   EPRUN1 048A   EPRUN2 0499   EPRUN3 0495   EPRUN4 0482   EPRUN5 04B3   EPRUN6 04BA
EPSSTP  0004   EPSTE1 07DF   EPSTE2 07F1   EPSTEP 07DB   EPSTOR 07C3   EPTIMR 0022   ERROR2 01B6   EXAM0  0250
EXAM1   027B   EXAM2  0281   EXAM3  028A   EXAM4  0293   EXAM5  0275   EXAMIN 024F   EXMON  0080   FDUMP1 0617
FDUMP2  0628   FDUMP3 0636   FDUMP4 0648   FDUMP5 0632   FINDOP 0042   GOTBL  0471   H      0045   HBD1   04D7
HBD2    04D5   HBDLAY 04C9   HBITHI 0027   HBITLO 0026   HDATIN 02B9   HEXASC 01E6   HEXBUF 0065   HEXNIB 01EF
HFDONE  05A7   HFILEO 0572   HRECIN 0297   HRECO  0600   HREGA  002A   HRLGB  002B   HREGC  002C   HREGD  002D
HREGE   002E   HREGF  002F   IMPLEM 0200   INCSMA 01F2   INCW   01F4   INCW1  01FC   INIT   0000   INITLP 000E
INPAD1  00C7   INPADR 00C8   INPKEY 00EC   INVALS 0300   ITMP   0004   JGORES 0226   JMPTBL 0206   JTOFIL 0222
JTOGO   0220   JTOLST 021A   JTOMOD 020F   JTOREC 0211   JTOREL 0216   KBDBUF 003B   KBDI1  06C6   KBDIN  06C2
KBDPOL  07AF   KCLRB  000C   KEY    0003   KEYCLR 0017   KEYDM  0016   KEYEND 0013   KEYFIL 0010   KEYFLG 0006
KEYGO   001E   KEYLOC 003C   KEYLST 001C   KEYMOD 001F   KEYNXT 0012   KEYPAT 0015   KEYPM  001A   KEYREC 0018
KEYREG  001B   KEYREL 0014   KEYTRA 0019   KGORES 001D   KSETB  0006   LASTKY 0004   LDATA  0002   LDBYTE 0582
LFEBR1  06C1   LFEBRK 06B1   LFEDM  0698   LFEINT 06A9   LFEPM  0684   LFER0  06A5   LFEREG 069C   LFETBL 067E
LFETCH  00FC   LFILL  02E9   LFILL1 02F3   LPGSEL 04E1   LSTBR1 0746   LSTBR2 0748   LSTBRK 073D   LSTDM  0721
LSTINT  0734   LSTORE 0700   LSTPM  070C   LSTR0  072F   LSTREG 0726   LSTTBL 070G   M0     0010   M1     0020
MADD    0024   MADDC  0025   MAIN   0029   MAIN2  0033   MAINA  0052   MAINB  0069   MAINB0 009E   MAINB1 00A0
MAINC1  0075   MAIND  0093   MAIND1 0087   MANL   0026   MBLOCK 0002   MDEC   002C   MDJNZ  002D   MEMHI  0035
MEMLO   0034   MERROR 00BC   MINC   002B   MMLO1  036F   MMOV   0020   MODOUT 0020   MORL   0027   MPUSEL 0040
MRL     002E   MRLC   0031   MRR    002F   MRRC   0030   MXCH   0029   MXRL   0028   NCOLS  0004   NEG1   FFFF
NEXTPL  003A   NIBI3  01C2   NIBIN  01B8   NIBIN2 01BA   NIBO   05BB   NOBRK  001B   NOVALS 0023   NREPTS 003D
NUMCON  0038   NXTLOC 0768   OPTAB1 033F   OPTAB2 0346   OPTAB3 0349   OPTION 0039   ORGPG0 0100   ORGPG1 01FD
ORGPG2  0300   ORGPG3 03E9   ORGPG4 04FD   ORGPG5 05FF   ORGPG6 06FF   ORGPG7 07FD   OUTCLR 0102   OUTMSG 0104
OUTUTL  0100   OV0BAS 0378   OV1B1  000A   OV1B2  0313   OV1BAS 03A4   OV2B1  0313   OV2BAS 03BA   OV3B1  0311
OV3BAS  038E   OVBUF  004E   OVLOAD 036A   OVSIZE 0017   OVSW1  0361   OVSWAP 035A   PBRK   0019   PDIGIT 000E
PERROR  019A   PGSIZE 00FD   PINPUT 000B   PLUS1  0001   PLUS3  0003   PRNT1  0117   PRNT2  0108   PSEGHI 000D
```

APPENDIX C
COMMAND SUMMARY

The following is a summary of the commands implemented by the HSE-49 emulator monitor. Within each command group, tokens in each column indicate options the user has when invoking those commands.

Tokens in square brackets indicate dedicated keys on the keyboard (some keys having shared functions); angle brackets enclose hex digit strings used to specify an address or data parameter. Parameters in parentheses are optional, with the effects explained above. The notation used is as follows:

<SMA> — Starting Memory Address for block command,
<EMA> — Ending Memory Address for block command,
<LOC> — LOCation for individual accesses,
<DATA> — DATA byte.

Asterisks (*) indicate the default condition for each command; thus that token is optional and serves to regularize the command syntax.

Program/data entry and verification commands:

[EXAM]	[PROG MEM]* <LOC> [,]	[NEXT]
	[DATA MEM]	[PREV]
	[REGISTER]	[.]
	[HWRE REG]	
	[PROG BRK]	
	[DATA BRK]	

Program/data initialization commands:

[FILL]	[PROG MEM]*	<SMA>	[,] <EMA>	[,] <DATA>	[.]
	[DATA MEM]				
	[REGISTER]				
	[HWRE REG]				
	[PROG BRK]				
	[DATA BRK]				

Intellec® development system or TTY interface commands (for transferring HEX format files):

[UPLOAD]	[PROG MEM]*	<SMA>	[,] <EMA>	[.]
	[DATA MEM]			
	[REGISTER]			
	[HWRE REG]			
	[PROG BRK]			
	[DATA BRK]			

[DNLOAD]	[PROG MEM]*	[.]
	[DATA MEM]	
	[REGISTER]	
	[HWRE REG]	
	[PROG BRK]	
	[DATA BRK]	

Formatted data dump to TTY or CRT:

[LIST]	[PROG MEM]*	<SMA>	[,] <EMA>	[.]
	[DATA MEM]			
	[REGISTER]			
	[HWRE REG]			
	[PROG BRK]			
	[DATA BRK]			

Program execution commands:

[GO]	[NO BREAK]*	(<SMA>) [,]
	[W/ BREAK]	[,]
	[SING STP]	
	[AUTO BRK]	
	[AUTO STP]	

[GO/RST]	[NO BREAK]*	[.]
	[W/ BREAK]	
	[SING STP]	
	[AUTO BRK]	
	[AUTO STP]	

Breakpoint setting and clearing:

[SET BRK]	[PROG MEM]*	<LOC>	([,] <LOC> ...)	[.]
	[DATA MEM]			

[CLR BRK]	[PROG MEM]*	<LOC>	([,] <LOC> ...)	[.]
	[DATA MEM]			

APPENDIX D
ERROR MESSAGES

The following error message codes are used by the monitor software to report an operator or hardware error. Errors may be cleared by pressing [CLR/PREV] or [END/.]. The format used for reporting errors is "Error – .n" where "n" is a hex digit.

Operator Errors

1. Illegal command initiator.
2. Illegal command modifier or parameter digit.
3. Illegal terminator for Examine command.
4. Illegal attempt to clear Error mode.

5-9. Not used.

Hardware Errors

A. ASCII error — non-hex digit encountered in data field of hex format record.

B. Breakpoint error. Break logic activated though breakpoints not enabled.

C. Hex format record checksum error. Note — the checksum will not be verified if the first character of the checksum field is a question mark ("?") rather than a hexidecimal digit. This allows object files to be patched using the ISIS text editor without the necessity of manually recomputing the checksum value.

D. Not used.

E. Execution processor failed to respond to a command or parameter passed to it by the master processor. EP automatically reset. EP internal status may be lost. Program memory not affected.

F. Not used.