THE WESTERN DESIGN CENTER, INC.
2166 E. Brown Rd. Mesa, AZ 85213
Ph 480-962-4545 Fx 480-835-6442
www.westerndesigncenter.com

W65C265S Monitor ROM REFERENCE MANUAL

Release 2.00

February 10,1995

Mensch Monitor ROM ($F000)
Reference Manual

Thank you for choosing the state-of-the-art features of the W65C265S microprocessor from Western Design Center. This manual describes the internal ROM monitordeveloped by the Com Log Company, Inc. for the W65C265S microprocessor.

For best results, we recommend that you please carefully read this manual completely before you attempt to use the Mensch Monitor. This manual contains important information on the proper operation of the monitor and it's library of subroutines.

HARDWARE REQUIREMENTS

1. A W65C265S microprocessor from Western Design Center.
   Configuration:
   - The default clock (CLK) for the W65C265S must operate at 32,768 Hz.
   - No external RAM or EPROM or I/O is necessary.
2. This monitor allows a terminal (or computer emulator) to be connected to serial UART port #3. The terminal/interface must be configured as follows:
   - All signals must be converted to TTL levels.
   - Hardware handshaking.
   - 8 bit data.
   - No parity.
   - 9600 Baud (Unless otherwise noted).

NOTICE

The Western Design Center, Inc. has made every attempt to ensure that the information in this manual is complete and accurate. However, WDC assumes no liability for errors, or for any damages that result from the use of this document or the W65C265S microprocessor.

Users of the Mensch Monitor firmware should note that it is both possible and relatively simple to connect *any* microcomputer equipment to external devices, and then to harm or destroy those devices (or anything that they may control). WDC assumes no liability for any connections or use of the W65C265S microprocessor and associated firmware.

Nothing herein shall be construed as a recommendation to use the W65C265S in violation of existing patents or other rights of third parties.

Information in this document is subject to change without notice and does not represent a commitment on the part of The Western Design Center, Inc. or The Com Log Company, Inc. for future products.

Table Of Contents

Introduction 11

W65C265S Internal Hardware 11

Central Processor Unit (CPU) 11
Random Access Memory (RAM) 11
Read Only Memory (ROM) 11
Input/Output (I/O) Ports 11
Clocks 12
Programmable Timers 12
Serial Universal Asynchronous Receiver-Transmitter (UART) Ports 13
Chip Selects 13

Mensch Monitor Configuration 13

External Clocks 14
Terminal/User Console 15
Restrictions On Application Software 15
Memory Map Assumptions 16
Reset Initialization Sequence 17
Low-Power Mode 23

User Console Operation 24

Command Entry 24

Debugging 24

Command Descriptions 25

A, ALTER 26
B, Set BREAKPOINT 26
D, Display Memory 27
F, Fill 28
G, Go 28
H or ?, Help 29
J, Jump to subroutine (JSL / 24-bit) 31
M, Change Memory 32
N, Display/Change Current Date 32
R, Registers 33
S, S28 Record Input 34
T, Display/Change Current Time 34
U, User Command 35
W, S28 Record Output 36
X, Low Power Mode 37
Quick Access Memory Display Commands: <,>,SPACE 37
Quick Access Memory Examine/Change: / 39
Quick Access Register Examine/Change: | 40
***, Reset To External Program** 41

Programming 42
Monitor Features/Functions 42
Interrupts 43
Low-Power Mode 44
Serial I/O Support 44
Time-of-Day Clock Support 45
Programmable Alarm 45
Data Manipulation 45
Custom Commands 47
Monitor Library Subroutines 48
Alter_Memory 49
ASCBIN 50
BACKSPACE 50
BIN2DEC 51
BINASC 51
CONTROL_TONES 52
DO_LOW_POWER_PGM 53
DUMPREGS 54
DumpS28 54
Dump_1_line_to_Output 55
Dump_1_line_to_Screen 56
Dump_It 57
Dump_to_Output 58
Dump_to_Printer 59
Dump_to_Screen 59
Dump_to_Screen_ASCII 60
FILL_Memory 61
GET_3BYTE_ADDRESS 62
Get_Address 62
GET_ALARM_STATUS 63
GET_BYTE_FROM_PC 64
GET_CHR 65
Get_E_Address 65
GET_HEX 65
GET_PUT_CHR 66
Get_S_Address 66
GET_STR 67
HEXIN 67
IFASC 68
ISDECIMAL 69
ISHEX 70
PUT_CHR 71
PUT_STR 71
READ_ALARM 72
READ_DATE 72
READ_TIME 73

RESET 74
RESET_ALARM 74
SBREAK 74
SELECT_COMMON_BAUD_RATE 75
SEND_BYTE_TO_PC 75
SEND_CR 76
SEND_HEX_OUT 76
SEND_SPACE 77
SET_ALARM 77
SET_Breakpoint 77
SET_DATE 79
SET_TIME 79
UPPER_CASE 80
VERSION 81
WR_3_ADDRESS 81
XS28IN 81

Appendices 83
Appendix A - Mensch Monitor Commands 83
Appendix B - Mensch Monitor Subroutines 84

Introduction

The W65C265S chip has firmware to handle interrupts, serial buffering, the real time clock, and the power down mode. It also has a user interface program, called the *monitor*. The monitor acts as a simple operating system, allowing the user to examine registers and memory, load and save programs, and debug applications in RAM.

The monitor is screened into the mask ROM on the W65C265S microprocessor chip. It will run on the W65C265S alone, and needs no ROM or RAM outside the W65C265S chip. It may be used with a variety of different circuits that use the W65C265S but has been designed to be compatible with the development tools from Com Log.

W65C265S Internal Hardware

The W65C265S microprocessor chip comprises a W65C816S CPU surrounded by RAM, ROM, several I/O ports, timers, tone generators,and serial UARTs. There are also built-in special controls for external elements such as chip selects and clocks. All hardware features of the W65C265S chip are described in detail by Western Design Center[1]. This section will merely review the main components.

Central Processor Unit (CPU)

The CPU on the W65C265S chip is essentially the W65C816S microprocessor from Western Design Center[2]. It's instruction set is fully compatible with the W65C816S microprocessor. Several programming instruction books have been published on the W65C816S. Many software tools (i.e. compilers, assemblers, operating systems, etc.) are available to developers from third party sources.

Random Access Memory (RAM)

The W65C265S has 576 bytes of internal RAM. This RAM appears in two places: $00:0000-$00:01FF and $00:DF80-$00:DFBF. The Mensch Monitor in the W65C265S ROM uses the area from $00:01C0 to $00:01FF as a stack, and the area from $00:0000-$00:00B2 as special page zero memory. (Page 0 memory can be used for short addressing and for indirect addressing modes.) IRQ RAM vectors reside in page one ($00:0100-$00:0138) Internal RAM locations at $00:00B3-$00:00FF and $00:0140-$00:01BF are available for user applications.

Read Only Memory (ROM)

The W65C265S has a 8K byte mask ROM from $00:E000 to $00:FFFF. This may be enabled or disabled by hardware or software. The mask ROM is added to the chip as part of the manufacturing process. The user cannot change it. On power-up or reset, the internal mask ROM may be enabled by bringing up the BE/RDY line before the RESB line goes HIGH, and disabled by bringing up the BE/RDY line after RESB line goes HIGH.

Input/Output (I/O) Ports

The W65C265S has a large number of bi-directional I/O ports. Each port has a data register and a data direction register. The data direction register allows the program to select any bit as an input or an output. The data register allows the program to read

---

1 Refer to WDC literature: W65C265S INFORMATION, SPECIFICATION AND DATA SHEET for details.

2 A detailed description may be found in: W65C816S INFORMATION, SPECIFICATION AND DATA SHEET from WDC.

inputs or to write outputs. Both the data direction register and the data register can be read or written; the data direction register will read back the last data written to it, and the data register will read the input lines or the data output, depending on the associated data direction register bits.

If an I/O port is selected as an input, the physical input will act as a light pull up or pull down latch. If the input is high, a 5 microamp pull-up is applied to the input. If the input is low, a 5 microamp pull-down is applied to the input. As the input changes from high to low or low to high, the pull up/down will abruptly change, thus 'pulling' the line quickly through the middle, undefined logic state. Having inputs at HI or LOW reduces standby current.

Another feature of these devices is that they latch input data. If a device connected to the inputs and set to some value is then tri-stated, the original value will stay on the inputs due to the pull up/down latching.

I/O ports 0, 1, 2 and 3 can be selected as either general I/O ports, or as the address and data bus for the W65C816S processor. Typical applications use these ports as the microprocessor bus. The only way they may be used as I/O ports is when a program is masked into the ROM, and no external EPROM or RAM is used.

Clocks

The W65C265S has two clock inputs, called CLK (the default clock) and FCLK (a separate fast clock). On reset, the W65C265S uses the CLK oscillator to run the W65C816S core processor.

The second oscillator, FCLK, is under program control. Software has the option to turn it on or off, and to select it as the input clock for the W65C816S. On reset, this oscillator is turned off.

The ROM monitor assumes that the CLK oscillator is 32768 Hz, and uses it to identify one of the five acceptable frequencies of FCLK. The CLK oscillator is also used to maintain the time-of-day clock.

Programmable Timers

The W65C265S chip has eight programmable timers. Timer T0 is the 'Watchdog' timer. If this timer is never started, it has no effect. Once started, it cannot be stopped by software. This timer counts down, and pulls the system reset line (RESB) when it hits zero. It may be reloaded under program control to prevent it from reaching zero. The purpose of Timer T0 is to provide a way to insure that execution won't get caught in an infinite loop or other error path. A macro or subroutine may be used to reload the timer from a variety of places in the normal program flow. If an error prevents the software from operating properly, then the timer will eventually reach 0. This will reset the W65C265S, hopefully allowing the application to regain control.

Timer T1 is general purpose and uses the CLK input as it's reference. This timer is usually used to provide a source of regular interrupts for the time-of-day clock.

Timer T2 uses a prescaled FCLK (÷16) for reference. The timer T2 also may be used to provide a source of regular interrupts.

The four UARTs rely upon timers T3 and T4 as baud rate generators. Each UART can select which timer (T3 or T4) will drive it. Timer T4 is more sophisticated than T3. First, it may select it's reference source as either FCLK or P60. When T4 is not used as a baud rate generator, it can count pulses on P60/TIN. The output of timer T4 may optionally be output on P61/TOUT to provide a square wave reference.

Timers T5 and T6 are the twin tone generators of the W65C265S. They each use FCLK as their reference and produce digitized sine waves on TG0 and TG1. These were designed with telephone applications in mind. The documentation from Western Design Center explains how the may be used to generate modem tones, DTMF tones, or various other commonly used telephone frequencies. There are numerous other applications for these outputs as well.

The last timer, T7, uses FCLK as a reference and runs continuously when enabled. It can generate normal timer interrupts via the TIFR when so configured. T7 may also be used for pulse width measurement (PWM). An edge interrupt will be generated and the contents of timer T7 will be latched whenever the signal on the PWM input satisfies the selected criteria. This may be configured for positive or negative zero crossings or both.

Serial Universal Asynchronous Receiver-Transmitter (UART) Ports

The W65C265S has four serial ports built into the chip. Enabling a serial port changes pins on the chip from general purpose I/O to receive and transmit data. Each UARTs may select timer T3 or T4 as a baud rate generator. The hardware provides one byte buffering in both directions, parity generation and checking, and interrupts on RX full and TX empty. The nature of the operation of the UART requires that the serial routines use an interrupt for service. The W65C265S's UARTs cannot easily be used in a 'Polled' mode of operation.

Chip Selects

The W65C265S has eight programmable outputs which may be selected as simple outputs or as chip selects. The chip select outputs reduce outside logic and provide selects to external devices when particular address ranges are in use. Enabling particular chip selects may change the way internal RAM is accessed. The chip selects are discussed in detail in the W65C265S literature from Western Design Center.

Mensch Monitor Configuration

The firmware installed from $00:E000 to $00:FFFF provides several functions and utilities consistent with the prior design goals.

Mensch Monitor ROM DESIGN GOALS

- The monitor must be able to be 'shut off'; that is it must exit to another program immediately after RESET, if necessary.
- The monitor must handle the interrupts for serial ports on the W65C265S, and must provide routines such that another program can easily use the serial port via the monitor.
- The monitor must be able to load other programs into RAM, and provide some debugging capabilities. This uses serial port #3.
- The monitor must maintain a time of day clock, and be capable of maintaining that clock on minimum power.
- The monitor must fit in the $00:E000 to $00:FFFF memory space.

These include:

- Power-up & Reset Initialization
- User Console Interface Logic
- Real-time Clock Support
- Library of Utility Subroutines

The W65C265S is a very versatile chip, but this versatility comes at the price of a complex initialization. The monitor firmware will initialize the chip on reset to provide the proper memory mapping of internal and external RAM and registers. The initialization will also set up the interrupts used by the rest of the firmware. The power up sequence is shown in the software listings.

The Mensch Monitor allows the user/developer to access the hardware and applications software. It allows uploading and downloading of programs, memory and register inspection and modifications, and general debugging. Complete descriptions of monitor functions appear elsewhere in this manual.

The firmware interrupt routine on timer T1 provides a complete real-time clock (RTC) implementation in RAM. The real-time clock updates a set of memory locations which can be read or written by any software. The real-time clock implementation also includes a programmable alarm function. The RTC interrupt handler will set a flag if the alarm time matches the current time.

The W65C265S firmware also contains many useful subroutines. A vector table is used to provide easy access from application software. These library routines include buffered serial functions, real-time clock support, and I/O control. Each of these functions and its operation is discussed later in this manual.

External Clocks

There are two clock inputs on the W65C265S chip. The default clock is called CLK. It must be connected to a 32,768 Hz watch crystal (or comparable oscillator) if the Mensch Monitor ROM is to be used. This frequency makes the CLK oscillator an excellent time base reference (32,768 is 2^{15}, which easily divides to 1 second). The crystal is a very low power oscillator, generally consuming less than 20 microamps. Upon reset, the W65C265S defaults to this oscillator to run the W65C816S core processor.

The second oscillator, FCLK (fast clock), is under program control. This oscillator is turned off upon hardware reset of the W65C265S. The FCLK frequency may be selected by the application designer. The W65C265S firmware will check the FCLK speed against the CLK frequency (which is ALWAYS 32 KHz) during initialization. This test allows the firmware to select values for the baud rate and system tick timers.

Acceptable crystal values for FCLK include: 1.8432 MHz, 2.4576 MHz, 3.6864 MHz, 4.9152 MHz and 6.144 MHz. It should be noted that this processor does not have a clock divider built in. This means that the processor and memory cycle times are the same as a single clock time.

The Mensch Monitor firmware turns on the FCLK shortly after reset initialization. The monitor then waits for the FCLK oscillator to stabilize, before selecting the FCLK as the operating frequency for the core W65C816S. In most cases, this selection is not changed - the processor always runs from the FCLK. The one exception is power down mode. In this case, the FCLK is turned off and the W65C816S core is run from the low frequency, low power oscillator.

Terminal/User Console

This monitor checks for a terminal (or computer emulator) connected to serial port #3. If present, the terminal will be used as a console for entering commands to the monitor. The terminal must be configured as follows:

- Hardware handshaking.
- 8 bit data.
- No parity.
- 9600 Baud (Unless otherwise noted).

Signal conditioning and conversion to/from TTL levels to/from RS-232 (or other links) must be performed by external circuitry.[3]

The W65C265S's firmware provides full interrupt control of the serial port, along with software buffering of transmit and receive data. The user can select the size and placement of the software serial buffers. The Mensch Monitor designates two more I/O pins for hardware handshaking of the serial port. These are pins PID0 and PID1. Pin PID0 is used as an output, indicating the serial receive buffer is ready to accept data. Pin PID1 is an interrupt input, and indicates to the monitor that the outside device can or cannot accept data.

Restrictions On Application Software

The W65C265S has eight programmable timers. Timer 1 is used by the monitor for a 1 second time of day interrupt. Timer 2 provides a 1 millisecond interrupt which the monitor uses to maintain the software down counters. Timer 3 is used for the baud rate generator. It can't be used for other purposes when the serial console is present. Timer 0 is the 'Watchdog' timer. It is not used by the Mensch Monitor, and is never enabled. Application software may use this timer. (Be careful in power down modes - it's not reset.) Timers 5 and 6 are used to drive the tone generators. Timers 3 and 7 are not used by the Mensch Monitor in the W65C265S ROM and are available to applications software.

The Mensch Monitor firmware assumes that I/O ports 0, 1, 2 and 3 will be used as the address and data bus for the W65C816S processor. They are used to access external EPROM, RAM and I/O.

[3] A special cable with built-in converter circuitry, for RS-232 to TTL and TTL to RS-232 levels is available from: The Com Log Company, Inc. of Scottsdale, Arizona. Telephone: (602) 248-0769

Memory Map Assumptions

Mensch Monitor Memory Map

Address Range	Function
$00:0000-$00:003F	W65C265S critical pointers.
$00:0040-$00:00BF	Mensch Monitor RAM. (Page #0)
$00:00C0-$00:00FF	Available internal User RAM. (Page #0)
$00:0100-$00:01BF	Available internal User RAM. (Page #1)
$00:01C0-$00:01FF	Mensch Monitor Stack Space in internal RAM.
$00:01C0	Optional ENTRY POINT for *User Check Program* called as subroutine from low power mode.
$00:0200-$00:07FF	External User Memory
$00:0800-$00:0804	Optional semaphore & start-up ENTRY POINT in external memory.
$00:0800-$00:7FFF	External User Memory
$00:8000-$00:8004	Optional semaphore & start-up ENTRY POINT in external memory.
$00:8000-$00:DEFB	External User Memory
$00:DEFC-$00:DEFF	Sentinel for Low-Power Mode and optional JMP to alternate command parser in Bank #0.
$00:DF00-$00:DF7F	External User Memory.
$00:DF80-$00:DFBF	W65C265S internal RAM. (Reserved by Monitor)
$00:DFC0-$00:DFFF	Monitor copies of W65C265 registers, saved by breakpoints.
$00:E000-$00:FFDF	W65C265S Internal ROM, Mensch Monitor firmware.
$00:FFE0-$00:FFFF	Interrupt Vectors
$01:0000-$FF:FFFF	External User Memory

The Mensch Monitor resides in the 8K byte mask ROM from $00:E000 to $00:FFFF. This may be disabled if desired. The monitor executes an initialization sequence after reset occurs. It turns on the external bus, checks locations $00:8000-$00:8002 and jumps to $00:8004 if a 'WDC' is found. This test is performed first using **CS5** and then **CS4** for the $00:800x selection. Using this signalling, a system can be designed to start on the internal ROM, then switch under software control to external ROM or RAM.

An identical set of semaphores ('WDC') may exist in memory at location $00:0800 (with a JMP at $00:0804). This is checked immediately *after* the semaphore at $00:8000 is checked. These semaphore locations allow a variety of configurations for the system developer. These specific addresses chosen for semaphores, correspond to elements in the MENSCH COMPUTER configuration, but do not significantly restrict other designs.

The Mensch Monitor uses locations: $00:01C0-$00:01FF as a stack, and the area from $00:0000-$00:00FF as special page zero memory. This (Page #0) memory may be used for short (8-bit) addressing and for indirect addressing modes. Internal RAM locations $00:00C0-$00:01BF are available for user applications.

Reset Initialization Sequence

The monitor was designed for the internal ROM of the W65C265S. It assumes that the RESET vector is entered from an internal ROM reset and therefore the code is started with the BCR=00.

Reset may be either a triggered reset or a power-up reset. There is no simple way for the firmware to differentiate which reset occurred. However, some semaphores (flags) in memory tell the monitor that certain aspects of the system have already been initialized. These should not be changed by the reset initialization sequence.

There is a checksum associated with the time-of-day clock and baud rate. If the checksum is correct, then the time-of-day clock has been running. If this is the case, the clock value will not be re-initialized.

The following pages describe the steps taken by the Mensch Monitor from power-up (or triggered RESET) to a command prompt. This is intended as an overview only; not a specific, line by line analysis of the code.

STATE #0 - POWER OFF or Any previous powered state, including LOW-POWER MODE.

- RESET OCCURS!
  (This may result from POWER ON or a triggered RESET.)

STATE #1 - "Essential initialization"

- Disable interrupts

- Reset stack

- "Enable External Memory"

  - Set BCR to $01
    (This turns on external address and data lines.)

  - Set PCS7 to $20
    (This turns on external chip select: CS5 from the W65C265S.)

- Check location $00:8000-$00:8002 for the string 'WDC', using CS5.

  If the string is there, transfer control to USER program in EXTERNAL MEMORY.

  (JMP $00:8004)

STATE #2 - "Check External Memory"

- "Enable External Memory"

  - Set PCS7 to $30
    (This turns on external chip selects: CS4 and CS5 from the W65C265S.)

- Check location $00:8000-$00:8002 for the string 'WDC', using CS4.

  If the string is there, transfer control to USER program in EXTERNAL MEMORY.

  (JMP $00:8004)

STATE #3 - "Check External Memory"

- "Enable External Memory"

  - Set PCS7 to $08
    (This turns on external chip select: CS3 from the W65C265S.)

- Check locations $00:0800-$00:0802 for the string 'WDC', using CS3.

  If the string is there, transfer control to USER program in EXTERNAL MEMORY.
  (JMP $00:0804)

STATE #4 - "Miscellaneous Initialization"

- Set PCS7 to $FB
  (This turns on external chip selects: CS0,1,3,4,5,6, and 7)

- Start the fast clock. (Don't use it, just start it.)

- Initialize the RAM interrupt vectors.

- Delay while fast clock becomes stable.

- Switch to fast clock.

- Set Timer 1 for a 1 second interrupt (ToD timer).

- Enable T1 interrupt (but not the I bit yet).

- Set up pointers to the serial buffers in internal RAM.

- Calculate the fast crystal frequency by comparing it to the 32 KHz clock crystal.

- Finally, check the Time-of-Day clock checksum.

  If the clock checksum is valid, proceed to **STATE #6**.

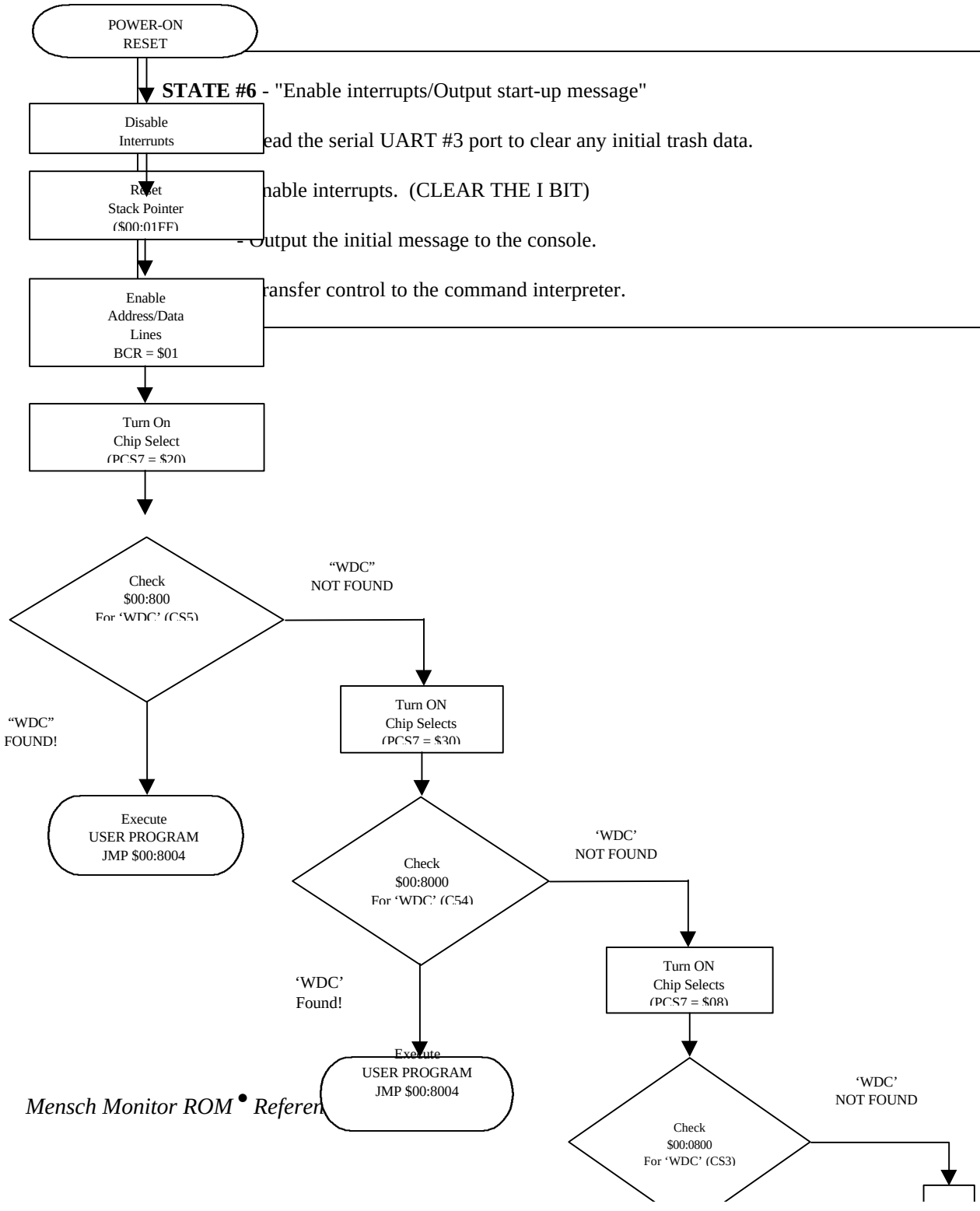
  Else perform additional initialization in **STATE #5**.

STATE #5 - "Initialize Time-of-Day Clock"

- Reset the Time-of-Day clock and reset the baud rate counters for the serial port to the default values.

- Set up control port of serial UARTs.

- Proceed to **STATE #6**.



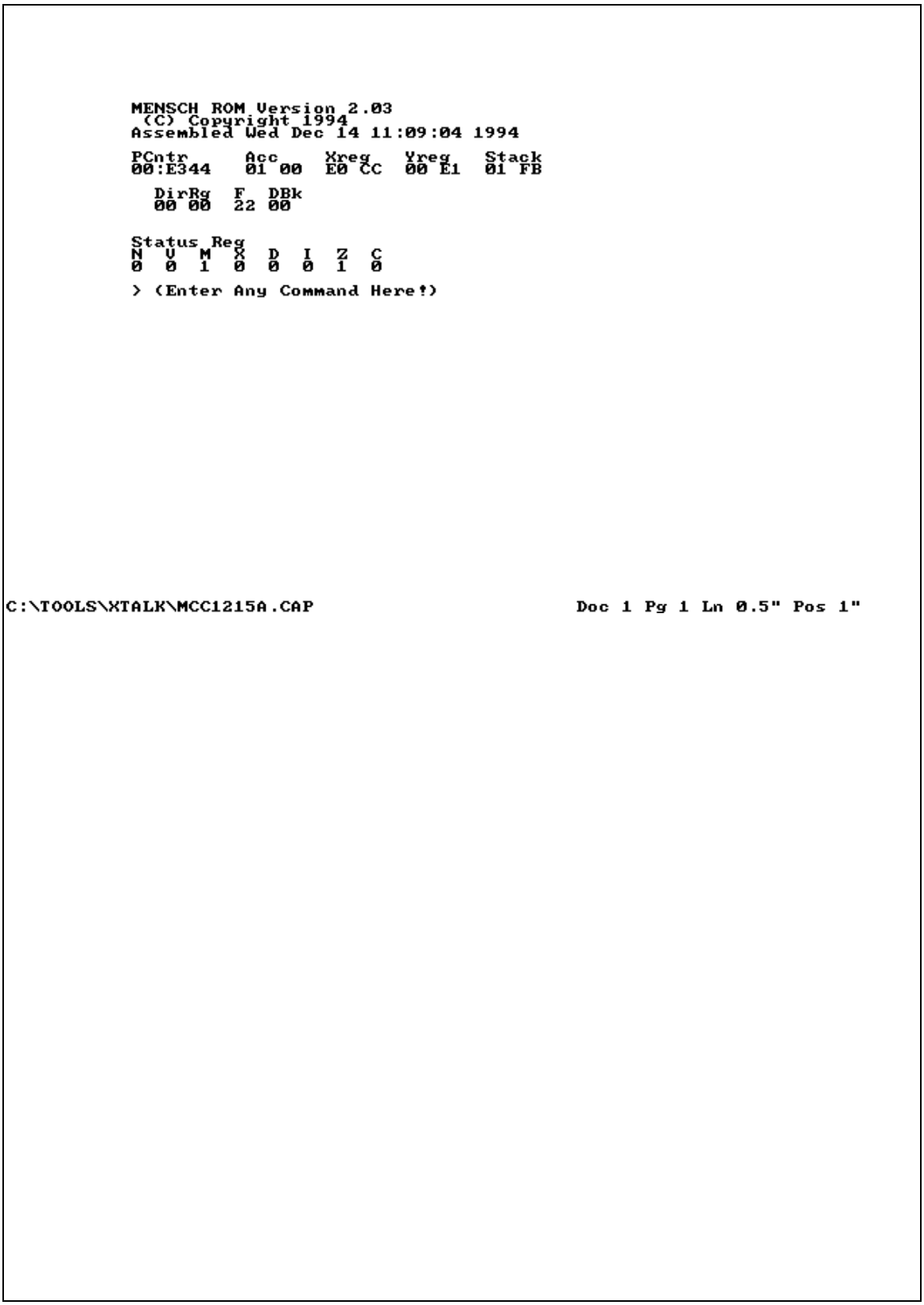
STATE #6 - "Enable interrupts/Output start-up message"

ead the serial UART #3 port to clear any initial trash data.

nable interrupts. (CLEAR THE I BIT)

- Output the initial message to the console.

ransfer control to the command interpreter.


MENSCH ROM Version 2.03
(C) Copyright 1994
Assembled Wed Dec 14 11:09:04 1994
PCntr Acc Xreg Yreg Stack
00:E344 01 00 E0 CC 00 E1 01 FB
DirRg F DBk
00 00 22 00
Status Reg
N V M X D I Z C
0 0 1 0 0 0 1 0
> (Enter Any Command Here!)
C:\TOOLS\XTALK\MCC1215A.CAP
Doc 1 Pg 1 Ln 0.5" Pos 1"

Low-Power Mode

The Mensch Monitor enters low-power mode when a "power-down" interrupt occurs. It accomplishes this by performing the following sequence:

- Shut down all interrupts (except TOD clock).
- Clear any pending interrupts.
- Reset the stack to ($00:)01FF.
- Enable the power down routine.
- Switch to the slow (default) clock and then shut off the fast clock.
- Configure I/O ports: 0,1,2, and 3 to outputs.

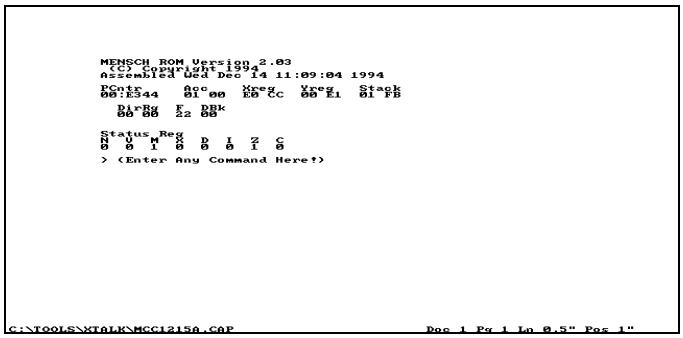
The power down routine will service the time-of-day interrupt. It will then check RAM for a semaphore series at location: $00:DFB3. If there is a valid power down routine in RAM, then locations: $00:DFB3= $55, and $00:DFB4 = $AA. If the semaphore indicates that a low power service routine exists in RAM, the monitor will transfer control via a JSR to location: $00:01C0 once per second.

The RAM routine may do anything appropriate to the low-power mode. Bus operations are not allowed. RAM locations: $00:0000 - $00:01BF retain meaningful data during low-power mode, and RAM locations: $00:01F8-$00:01FF are used for the stack when servicing the time-of-day interrupt. The RAM service routine should not modify these memory locations. Finally, it must return to the monitor, when finished. It must execute an RTS instruction.

If the user's low power routine returns with the OVERFLOW-bit=1, the system will remain in low power mode. If OVERFLOW-bit=0, the system will exit from low power mode via a vector at $00:0130. The reset initialization logic will initialize this vector to contain: "JML RESET", but it may be changed by the user program.

If a physical reset occurs while the system is in low-power mode, the normal reset initialization sequence will be performed.

User Console Operation



The monitor is entered upon power-up, reset, and when a BRK instruction is encountered. When the monitor is ready for a command, a '>' (greater) will appear as a prompt. Upon reset, the monitor sends copyright and version notices, as well as a register display to the terminal. (There are semaphores in memory to enable/disable specific features.)

Command Entry

Commands are entered after the prompt. The Mensch Monitor parses a command as it is entered. Spaces and other separators are automatically generated by the monitor as parsing dictates their appearance. Backspaces are usually allowed. Buffer limitations allow no provision for editing commands. If an error is made in entry, a carriage return (CR) or ENTER will usually cancel the command and return the monitor prompt. If a command has been started, a control-C character will usually cancel the command.

The first command to learn on the Mensch Monitor is the HELP command. Enter a 'H' at the command prompt (no carriage return is needed) and a help menu will be displayed. This menu simply lists all the monitor commands.

Debugging

Code debugging may be accomplished by placing BRK (00) instructions in the code. When the BRK instruction is executed, the CPU registers will be displayed and the monitor prompt will appear. The user may examine or change memory and registers. BRK instructions should be used with care.

WARNING!

Generally, if code goes wild, execution will eventually encounter a BRK instruction ($00) and return to the monitor. Interrupts may however, still be running. A bad interrupt can disable the monitor functions.

Command Descriptions

The following pages describe all of the monitor commands in detail. They are listed here, in summary, along with their functions.

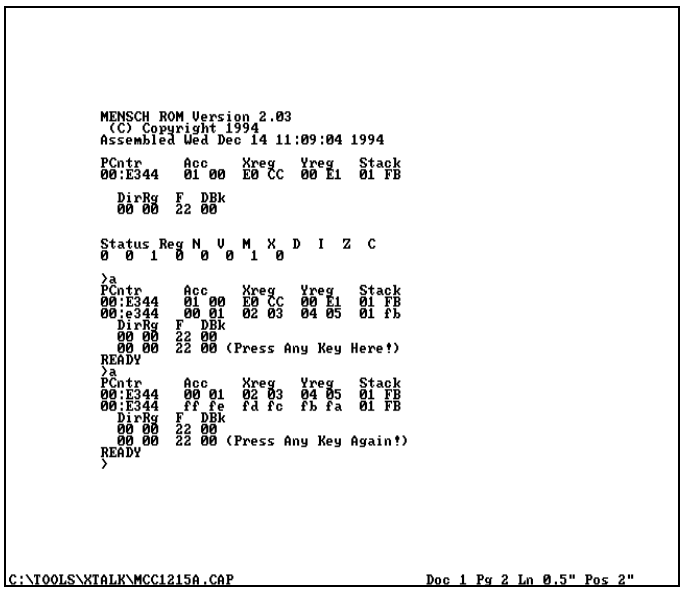
(NOTE: The Mensch Monitor outputs are shown on the following pages in normal font, and user inputs are shown in *italics*.)

Command Summary

Command	Usage
A	ALTER registers.
B	Set BREAKPOINT
D	DISPLAY memory block in hexadecimal.
F	FILL memory block with constant.
G	GO to address (JML execution).
H or ?	Display HELP Menu.
J	Jump to subroutine using 24-bit address. (JSL execution)
M	Examine/Change MEMORY location.
N	Display/Change current DATE.
R	Display REGISTERS.
S	Read 'S28' record format.
T	Examine/Change current TIME.
U	USER command prefix.
W	WRITE block of memory as 'S28' records.
|	Quick access: Examine/change registers.
/	Quick access: Examine/change memory.
>	Quick access: Display NEXT memory location.
<	Quick access: Display PREVIOUS location.
(space)	Quick access: Display current memory location.
^C	Cancel current operation.

A, ALTER

This command will display the registers and allow the user to change them.



An executing program will re-enter the Mensch Monitor when it encounters a BRK instruction. The monitor will copy all the registers inside the processor into RAM. These copies of the registers may be viewed using the 'A' or 'R' commands. The 'R' command only allows viewing. It is triggered automatically before the user prompt is displayed. The 'A' command permits the user to enter new values.

When a program is started via the 'J' or 'G' commands, the RAM versions of the registers are copied back into the microprocessor before control is transferred. Changing the registers via the 'A' command only changes the RAM copy. This will have no effect until a program resumes.

The Mensch Monitor will output the current values of the registers to the console: **PCntr**=Program Counter, **Acc**=Accumulator, **Xreg**=X register, **Yreg**=Y register, **Stack**=Stack Pointer, **DirRg**=Direct register, **F**=Flag register, and **DBk**=Data Bank register. The cursor must be placed under each entry, in order, for the user to change the contents. The monitor will insert spaces automatically between fields. A space entered from the console will cause the current field to be skipped, leaving it unchanged.

B, Set BREAKPOINT

This monitor command allows the user to set a breakpoint at a specific location. Basically, this involves storing a **BRK** instruction ($00) at the target location.

```
READY
>f
Enter Lowest Address    BB:AAAA 00:5000
Enter Highest Address   BB:AAAA 00:5fff
Enter Byte in HEX ff
READY
>d
Enter Lowest Address    BB:AAAA 00:5000
Enter Highest Address   BB:AAAA 00:501f
Address 0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F

00:5000 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00:5010 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

READY
>b
Enter Address   BB:AAAA 00:5013
READY
>d
Enter Lowest Address    BB:AAAA 00:5000
Enter Highest Address   BB:AAAA 00:501f
Address 0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F

00:5000 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00:5010 FF FF FF 00 FF FF FF FF FF FF FF FF FF FF FF FF

READY
>
```

C:\DTP\CAPTURE\MCC1221.CAP Doc 1 Pg 8 Ln 0.5" Pos 1"

When execution resumes, the program may attempt to execute the target instruction. The **BRK** instruction will be executed instead, and control will return to the ROM monitor.

D, Display Memory

This monitor command displays a range of memory, in hex and ASCII. Parsing is performed as the command is entered. The user enters a '*D*', followed by the starting (*Lowest)* address, followed by the ending (*Highest*) address. No spaces, colons, or ENTER keys are required. The monitor echoes these automatically.

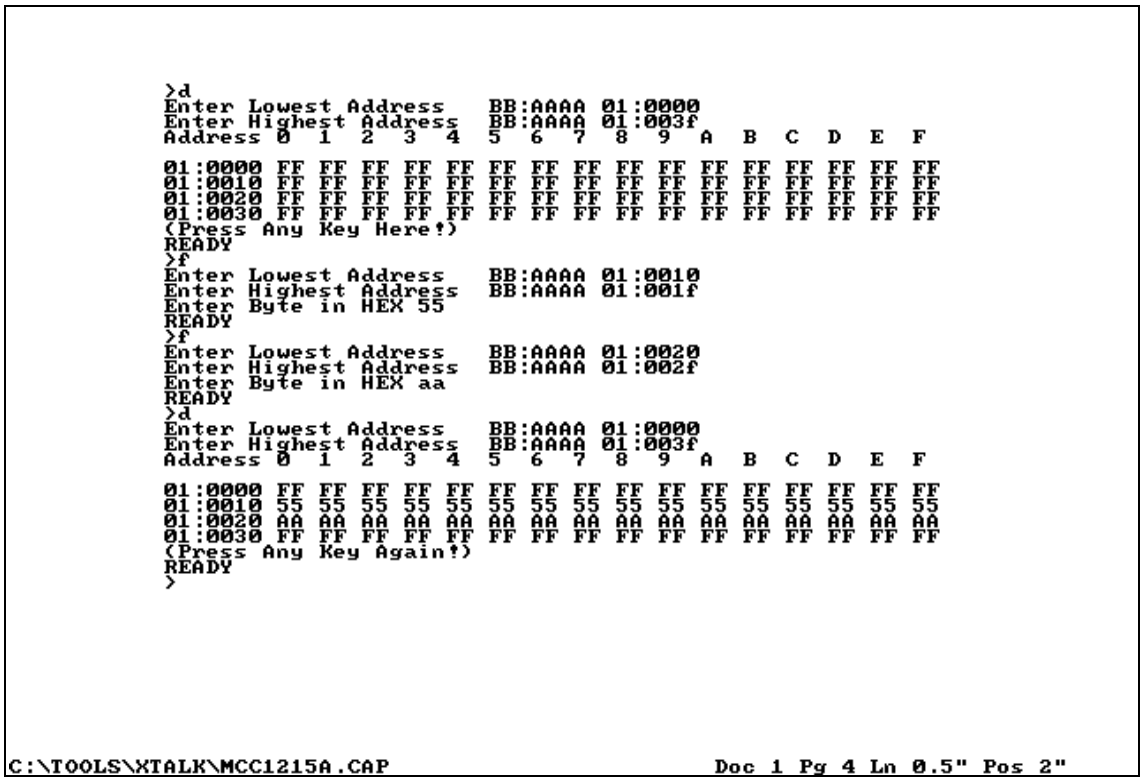
```
>d
Enter Lowest Address    BB:AAAA 00:0000
Enter Highest Address   BB:AAAA 00:00ff
Address 0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F

00:0000 00 00 00 00 39 01 0A 00 00 00 00 00 43 01 0A 00
00:0010 00 00 00 00 4D 01 0A 00 08 00 08 00 57 01 0A 00
00:0020 00 00 00 00 61 01 0A 00 00 00 00 00 6B 01 0A 00
00:0030 00 00 00 00 75 01 0A 00 09 00 04 00 7F 01 0A 00
00:0040 00 00 00 14 00 00 00 66 00 00 00 80 10 08 00 00
00:0050 00 00 00 10 8E 01 00 09 00 00 AF 00 00 5D 00 00
00:0060 00 00 00 FF 00 00 5F 01 00 01 00 00 00 00 00 00
00:0070 F0 46 00 70 F1 7A F1 84 F1 BE F1 BE F1 BE F1 BE
00:0080 F1 A2 FF 00 00 00 04 00 00 00 00 00 00 00 00 00
00:0090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00:00A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0A
00:00B0 00 3C 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00:00C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00:00D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00:00E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00:00F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
(Press Any Key Here!)
READY
>d
Enter Lowest Address    BB:AAAA 00:0015
Enter Highest Address   BB:AAAA 00:004b
Address 5  6  7  8  9  A  B  C  D  E  F  0  1  2  3  4

00:0015 01 0A 00 08 00 08 00 57 01 0A 00 00 00 00 00 61
00:0025 01 0A 00 00 00 00 00 6B 01 0A 00 00 00 00 00 75
00:0035 01 0A 00 01 00 06 00 7F 01 0A 00 00 00 00 14 00
00:0045 00 00 62 00 00 00 80 10 08 00 00 00 00 00 10 8E
(Press Any Key Here!)
READY
>
```

C:\TOOLS\XTALK\MCC1215A.CAP Doc 1 Pg 3 Ln 0.5" Pos 2"

Please note that whenever memory is displayed using the 'D' command, the display will stop without a command prompt. The user may/must press any key to return to the command processor.

F, Fill

This command is used to fill a block of memory with a particular value. Parsing is performed as the command is entered. The user enters a '*F*', followed by the starting address, followed by the ending address, and finally, the fill value in hexadecimal. No spaces, colons, or ENTER keys are required. The monitor echoes these automatically.

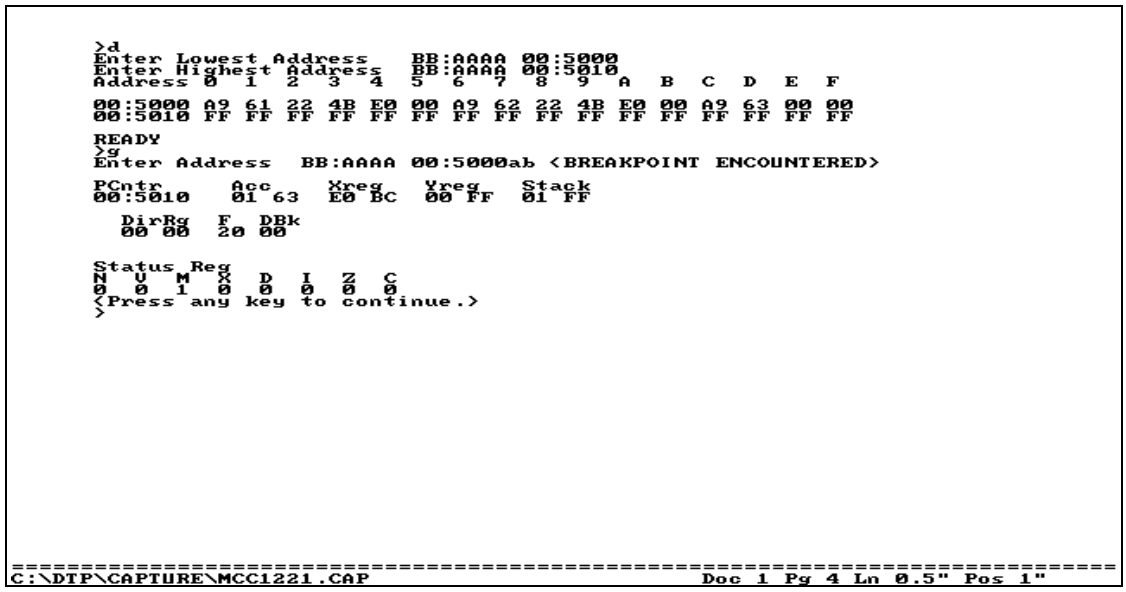


Please note that whenever memory is filled using the 'F' command, the display will stop without a command prompt. The user may/must press any key to return to the command processor.

G, Go

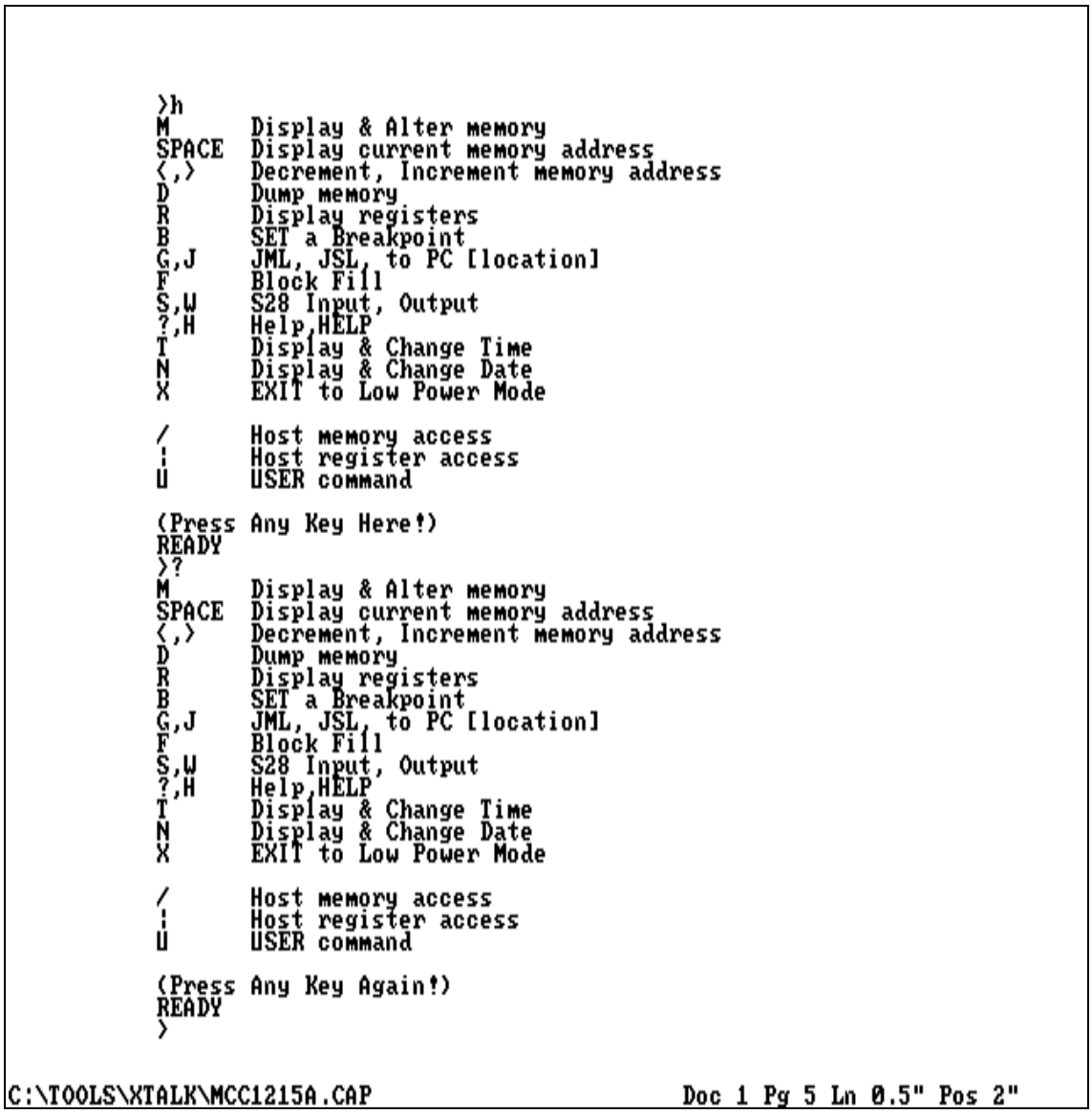
This command executes a program in memory. It transfers execution control via a long jump ("JML") instruction. The user enters a 'G', followed by a starting address. No spaces, colons, or ENTER keys are typed by the operator if an address is specified. The monitor echoes these automatically. Pressing the **ENTER** key in response to the address prompt will cause execution to begin from the address specified by the monitor's copy of the program counter.

The registers used to restart the processor are the RAM copies made when the program was stopped. These can be modified before resuming execution. (See the **A, Alter** command for more details.)



H or ?, Help

The HELP command lists all the commands available via the monitor. It is a single character ('H','h', or '?') entered at the prompt. No <ENTER> or <space> is necessary.


>h
M Display & Alter memory
SPACE Display current memory address
<,> Decrement, Increment memory address
D Dump memory
R Display registers
B SET a Breakpoint
G,J JML, JSL, to PC [location]
F Block Fill
S,W S28 Input, Output
?,H Help,HELP
T Display & Change Time
N Display & Change Date
X EXIT to Low Power Mode
/ Host memory access
! Host register access
U USER command
(Press Any Key Here!)
READY
>?
M Display & Alter memory
SPACE Display current memory address
<,> Decrement, Increment memory address
D Dump memory
R Display registers
B SET a Breakpoint
G,J JML, JSL, to PC [location]
F Block Fill
S,W S28 Input, Output
?,H Help,HELP
T Display & Change Time
N Display & Change Date
X EXIT to Low Power Mode
/ Host memory access
! Host register access
U USER command
(Press Any Key Again!)
READY
>
C:\TOOLS\XTALK\MCC1215A.CAP Doc 1 Pg 5 Ln 0.5" Pos 2"

J, Jump to subroutine (JSL / 24-bit)

This command is similar to the '*G*' command, excepting that the full twenty-four bit address of the **BRK** handler is put on the stack before starting the program. If the program ends with an **RTL**, it will return cleanly to the monitor. Upon return, the program counter register will contain the address of the subroutine. Other register values may be changed because of action by the subroutine.

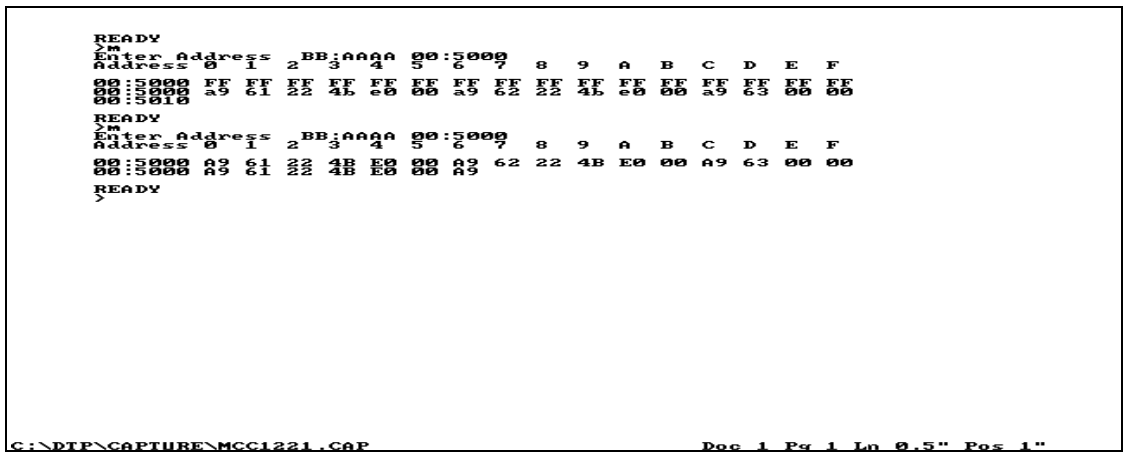
Example:

```
MENSCH ROM Version 2.03
 (C) Copyright 1994
Assembled Wed Dec 14 11:09:04 1994

PCntr      Acc      Xreg   Yreg    Stack
00:E3A7  01 0D   00 30  00 32   01 FF

  DirRg    F   DBk
  00 00   30   00


Status Reg
N  V  M  X  D  I  Z  C
0   0  1  1  0  0  0  0

>J
Enter Address  BB:AAAA 01:0050 {This assumes a valid subroutine at $01:0050}
          {RTL Encountered in subroutine.}
PCntr      Acc      Xreg   Yreg    Stack
01:0050  00 00   00 00  00 00   01 FF

  DirRg    F   DBk
  00 00   00   00

Status Reg
N  V  M  X  D  I  Z  C
0   0  0  0  0  0  0  0
>
```

The example assumes no arguments are passed in registers. If the target subroutine required such arguments, the registers would need to be initialized. Excepting the **SP** (which is bumped for the return address) and the **PC**, the registers are copied from the monitor's RAM into the processor before the jump takes place. (See *A, Alter* for more details.)

M, Change Memory

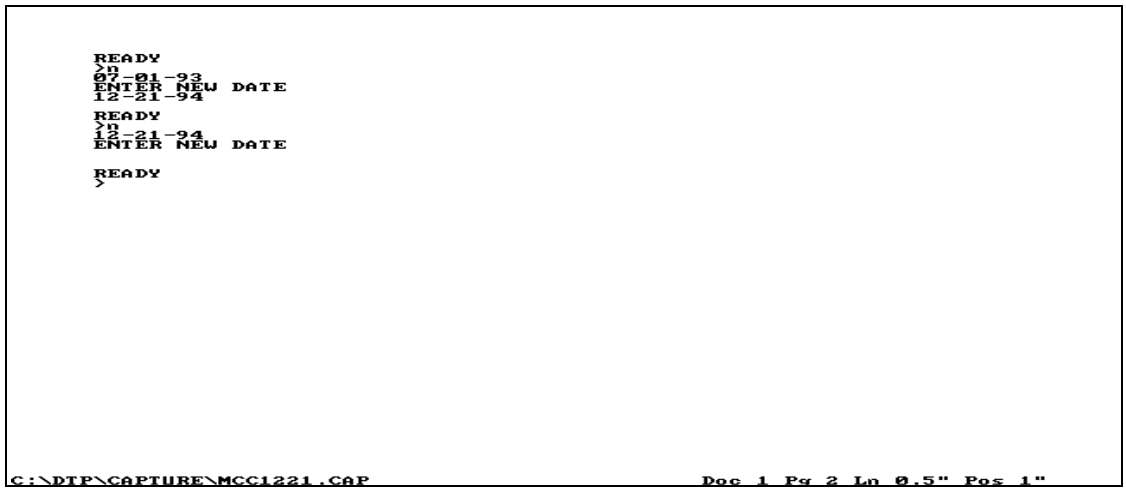
This command is used to change the contents of memory locations, beginning at the specified address.



The user must enter: '*M*' and an *address*. The monitor will respond by printing a legend and the current contents of the first 16 locations. The cursor is then positioned one line below the first memory location. The user may enter a new value for that location. If the user continues typing new values, and they modify consecutive locations. Parsing is performed as the command is typed. The monitor will automatically output spaces between fields. A space may be entered by the user to skip a location. That byte will be left at its original value. The user may terminate this command by pressing: **ENTER** ($0D).

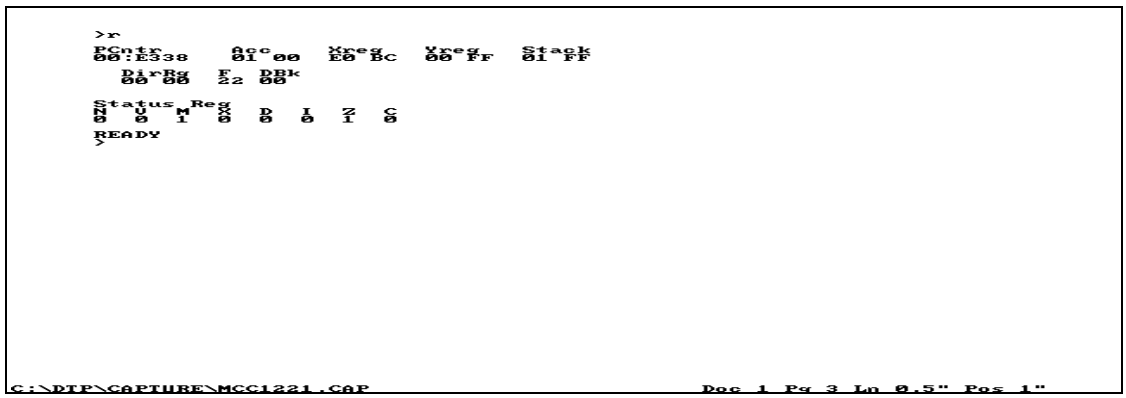
N, Display/Change Current Date

This command displays the current Date and allows the user to enter new values.



R, Registers

This command displays the current RAM copy of the microprocessor registers.



An executing program will re-enter the Mensch Monitor when it encounters a BRK instruction. The monitor will copy all the registers inside the processor into RAM. These copies of the registers may be viewed using the 'A' or 'R' commands. The 'R' command only allows viewing. It is triggered automatically before the user prompt is displayed. The 'A' command permits the user to enter new values.

When a program is started via the 'J' or 'G' commands, the RAM versions of the registers are copied back into the microprocessor before control is transferred. Changing the registers via the 'A' command only changes the RAM copy. This will have no effect until a program resumes.

The Mensch Monitor will output the current values of the registers to the console: **PCntr**=Program Counter, **Acc**=Accumulator, **Xreg**=X register, **Yreg**=Y register, **Stack**=Stack Pointer, **DirRg**=Direct register, **F**=Flag register, and **DBk**=Data Bank register.

S, S28 Record Input

The Mensch Monitor uses Motorola style "S28" records to upload or download programs. This format comprises lines starting with an "S2" or "S8" sequence. The "S2" records contain data to be loaded into memory.

S28 Format:

S2LLHHMMLWDDDDDDDD...CC

where:

S2 are literally the ASCII characters:"S2",
LL is the length of the data+4,
HH is the high byte of the address,
MM is the middle byte of the address,
LW is the low byte of the address,
DD is one byte of data,
DD is the next byte, etc
CC is the checksum (1's complement of the
sum of the length, address, and data bytes.)
NOTE: Convert the bytes from ASCII to hex before performing the addition.

A valid S28 record is:

S2140090004C28BA4C629D0102031589E6F3D002E6AD

The "S8" record is always the last record in a load and does not contain data. It just terminates the load operation. The actual content of the "S8" record is: "S804000000FB".

The user typically will not type 'S' records directly as commands. The records will usually be from a file transfer on the host computer. When the monitor receives an 'S' as the first character of a command, it assumes that an "S28" load is beginning. It initializes it's record counter and checksum, and then proceeds to process the first record. Echo on port #3 will be turned: OFF during the load sequence. This means that the characters of the "S28" records will not be echoed back to the host computer. The user will only see a single period ('.') and 4-digit record number as each record processed. The load operation will continue processing records until one of the following conditions: (1) An ESCAPE character is received, (2) A format or checksum error is detected, (3) The "S8" sequence starts a record.

This "S28" loader performs only minimal error checking on the records. It should be used with caution. Data is written to the specified locations in memory as each "S2" record is processed. Validation of the checksum occurs only after a complete record has been received. The *perceived destination* memory will already have been loaded. Transmission errors during the *address* field of the record may cause loading at incorrect addresses before detection. If transmission errors occur during the *length* field, the system may appear to go into a *lockup* condition.

T, Display/Change Current Time

The T command displays the current time and allows the user to enter new values.

```
READY
>t
12:43:55
ENTER NEW TIME
14:43:00
READY
>t
14:43:02
ENTER NEW TIME

READY
>
```

```
C:\PIP\CAPTURE\MCC1221.CAP                    Doc 1 Pg 2 Ln 0.5" Pos 1"
```

U, User Command

This is a mechanism by which the user can easily add his own commands. When the monitor prompt is present, characters are parsed by the Mensch Monitor as they are entered. If the first character of a command is a 'U' or 'u', the monitor will perform a **JSL USER_CMD**. Additional characters entered after the 'U' must be handled by the user routine. When the user's routine is finished, an **RTL** instruction will return control to the monitor.

The user must supply a routine to handle the characters after the 'U', and then store a jump (**JMP** or **JML** to the user's handler routine at location: **USER_CMD** ($00:012C). The user's command processor should return to the monitor via an **RTL** instruction.

```
MENSCH ROM Version 2.03
 (C) Copyright 1994
Assembled Mon Dec 19 08:52:02 1994

PCntr     Acc    Xreg   Yreg   Stack
00:E338   01 00  E0 BC  00 80  01 FF

  DirRg  F  DBk
  00 00  22 00

Status Reg
N  V  M  X  D  I  Z  C
0  0  1  0  0  0  1  0

>d
Enter Lowest Address    BB:AAAA 01:0400
Enter Highest Address   BB:AAAA 01:041f
Address 0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F

01:0400 22 66 E0 00 A9 01 A2 80 00 22 4E E0 00 22 66 E0
01:0410 00 22 36 E0 00 6B 22 66 E0 00 A9 01 A2 CA 00 22

READY
>m
Enter Address   BB:AAAA 00:012c
Address C  D  E  F  0  1  2  3  4  5  6  7  8  9  A  B

00:012C 5C B0 E0 00 5C B0 E0 00 C2 01 01 01 03 01 01 01
00:012C 5c 00 04 01

READY
>u
Transfer of control to this routine successful, press any key to continue

READY
>
```

```
M:\...\DOCS\W65C265\USERCMD.CAP                Doc 1 Pg 1 Ln 0.5" Pos 1"
```

The subroutine source for the above example is shown later in the **User Commands** section of this manual.

W, S28 Record Output

The Mensch Monitor uses Motorola style "S28" records to upload or download programs. This format comprises lines starting with an "S2" or "S8" sequence. The "S2" records contain data to be loaded into memory. This command is used to output S28 records via the serial port #3.

S28 Format:

S2LLHHMMLWDDDDDDDD...CC

where:

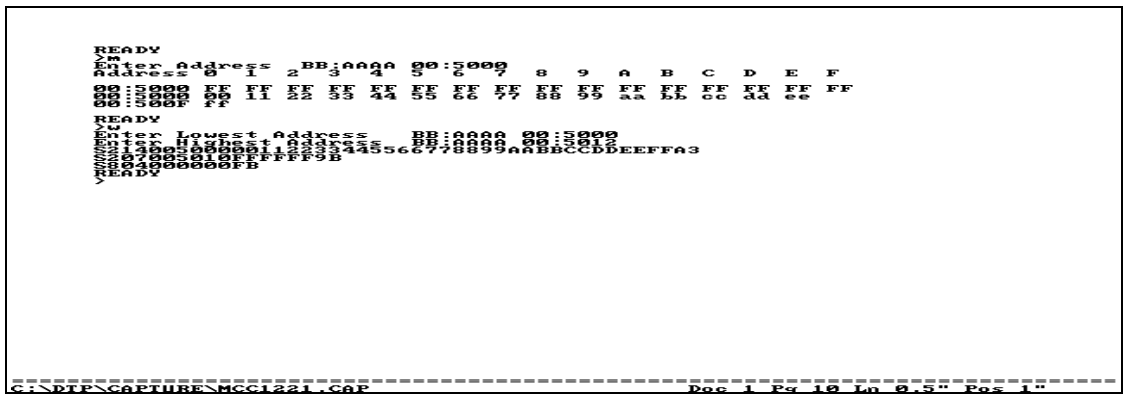
S2 are literally the ASCII characters:"S2",
LL is the length of the data+4,
HH is the high byte of the address,
MM is the middle byte of the address,
LW is the low byte of the address,
DD is one byte of data,
DD is the next byte, etc
CC is the checksum (1's complement of the sum of the length, address, and data bytes.)

NOTE: Convert the bytes from ASCII to hex before performing the addition.

A valid S28 record is:

S2140090004C28BA4C629D0102031589E6F3D002E6AD

The "S8" record is always the last record in a write and does not contain data. It will be used to terminate the load operation. The actual content of the "S8" record is: "S804000000FB".



The first address is the starting address, the second address is the address of the last byte to be sent. This command specifically requires a return, allowing the user to turn on a recording device to record the serial data.

X, Low Power Mode

This command forces the system into low power mode. It will happen immediately. Depending upon context shutdown may occur before the 'X' character is echoed. The system will remain in low-power mode until a physical **Reset** occurs, or the alarm triggers, or until directed to do so by the user's low-power routine.

```
>X      {CPU switches to low-power mode}

{Serial console & external circuitry inactive.}

MENSCH ROM Version 2.03
 (C) Copyright 1994
Assembled Wed Dec 14 11:09:04 1994

PCntr     Acc     Xreg   Yreg    Stack
00:E3A7  01 0D   00 30  00 32   01 FF

  DirRg   F   DBk
  00 00   30   00


Status Reg
N  V  M  X  D  I   Z  C
0   0  1    1  0   0   0  0
>
```

The above example shows how the normal exit from low-power mode will use the RESET initialization sequence. In this case however, the initialization code will not reset the serial ports or Time-of-Day clock. (Refer to the description of the **Reset Initialization Sequence** for more details.)

Quick Access Memory Display Commands: <,>,SPACE

Quick access commands allow "user friendly" software on the host computer to easily access W65C265 resources. All quick commands assume they are interacting with a program. Therefore, echo is turned: OFF until completion of the command, and then restored to it's previous setting. Requests from the host and responses to the host are formatted for minimum overhead, rather than readability. This assumes that the host program will handle all interactions with the user. The SPACE, <, and > commands, allow the user to view the current memory address, one lower, or one higher, respectively. The current address pointer may be initialized via the SLASH (/), M or D commands.

Example:

```
>M 01:1000

  ADDR   00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
01:1000  0A 4C 88 19 33 23 78 55 90 34 44 23 12 82 47 52
01:1000  31 32 33 34 35 36 37 38 39 <Enter>
>D 01:1000 <Space>
  ADDR   00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
01:1000  31 32 33 34 35 36 37 38 39 34 44 23 12 82 47 52
><Space>01:1010 FF

><01:100F 52

><01:100E 47

><01:100D 82

><Space>01:100D 82

>>01:100E 47

>>01:100F 52

>
```

Quick Access Memory Examine/Change: /

The SLASH (/) command allows host computers quick access to memory locations. It may be used by "user friendly" software on the host to examine and change specific locations in memory. The quick commands assume they are interacting with a program. Echo is turned: OFF until completion of the command, and then restored to it's previous setting. Requests from the host and responses to the host are formatted for minimum overhead, rather than readability. The host program should handle formatting, presentation, and interactions with the user. The SLASH (/) command has several forms:

Command	Description
/<*ENTER*>	This returns the current *value* of the address pointer.
/<*SPACE*>	This returns the *data* byte at the current memory location and then increments the address pointer.
/<*dd*><*SPACE*>	This writes the <*dd*> data byte to the memory location specified by the address pointer. It then re-reads the location and returns the contents, as a check for writable memory. Finally, it increments the address pointer.
/<*aaaa*><*SPACE*>	This changes the *value* of the address pointer to *00:*<*aaaa*>. It then reads the location and returns the contents. Finally, it increments the address pointer.
/<*bb*>*:*<*aaaa*><*SPACE*>	This changes the *value* of the address pointer to <*bb*>*:*<*aaaa*>. It then reads the location and returns the contents. Finally, it increments the address pointer.
/<*aaaa*><*dd*>	This changes the *value* of the address pointer to *00:*<*aaaa*>. It then writes the <*dd*> data byte to the memory location specified by the address pointer. Next, it re-reads the location and returns the contents, as a check for writable memory. Finally, it increments the address pointer.
/<*bb*>*:*<*aaaa*><*dd*>	This changes the *value* of the address pointer to <*bb*>*:*<*aaaa*>. It then writes the <*dd*> data byte to that memory location. Next, it re-reads the location and returns the contents, as a check for writable memory. Finally, it increments the address pointer.

Any error in input will result in a NAK ($15) character response.

Quick Access Register Examine/Change:

The PIPE (|) command allows host computers quick access to W65C265 registers. It may be used by "user friendly" software on the host to examine and change specific registers. Any register changes will not take effect until execution is resumed via the G or J commands. This command assumes that it is interacting with a program. Echo is turned: OFF until completion of the command, and then restored to it's previous setting. Requests from the host and responses to the host are formatted for minimum overhead, rather than readability. The host program is responsible for formatting, presentation, and interactions with the user. The PIPE (|) command has several forms:

Command	Description
|<*SPACE*>	This returns the current values of *all* the registers. The individual data fields will be separated by spaces. The fields are returned in the following order: *<PC> <A-Reg> <X-Reg> <Y-Reg> <SP> <DP> <Flags> <DB>* The program counter is 24 bits. All other registers are 16 bits, except the 8-bit Data Bank and Flags/Status registers.
|P<*bb*>*:*<*aaaa*>	This replaces the contents of the **Program Counter** with <*bb*>*:*<*aaaa*>.
|A<*dddd*>	This replaces the contents of the **A-Register** with: <*dddd*>. (NOTE: This is always a 16-bit value, and all 16 bits will be changed. If the A-Register mode was originally configured as 8-bits, then that mode will be restored upon completion of this command.)
|X<*dddd*>	This replaces the contents of the **X-Register** with: <*dddd*>. (NOTE: This is always a 16-bit value, but only 8 bits will be changed if the X-Register mode was originally configured as 8-bits. That mode will be restored upon completion of this command.)
|Y<*dddd*>	This replaces the contents of the **Y-Register** with: <*dddd*>. (NOTE: This is always a 16-bit value, but only 8 bits will be changed if the Y-Register mode was originally configured as 8-bits. That mode will be restored upon completion of this command.)
|S<*aaaa*>	This replaces the contents of the **Stack Pointer** with *00:*<*aaaa*>.
|D<*aaaa*>	This replaces the contents of the **Direct Page** register with: <*aaaa*>.
|F<*ff*>	This replaces the contents of the **Flags/Status** register with: <*ff*>.
|B<*bb*>	This replaces the contents of the **Data Bank** register with <*bb*>.

Normally, the PIPE (|) command will return data or an ACK ($06) character. Any error in input will result in a NAK ($15) character response.

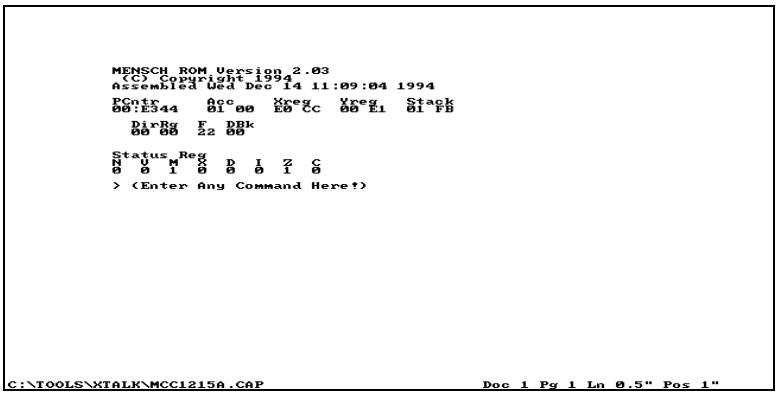
*, Reset To External Program

This command allows developers using the Mensch Monitor in the W65C265S ROM to switch into another *external* program. It has been included in this command set to support the anticipated needs of the Mensch Consumer Computer. The Mensch will have an *extended* monitor or operating system in external EPROM memory. That program will interact with the Mensch Monitor in the W65C265S ROM . It will initialize a jump vector for the asterisk (*) command. Whenever the asterisk (*) command is invoked, control will be transferred to the Mensch Consumer Computer Operating System in external memory.

The Mensch Monitor in the W65C265S ROM will only initialize the jump vector to it's own command processor. Therefore, when the external operating system is not present, the asterisk (*) command does nothing significant. Developers using the W65C265 in configurations other than the Mensch Consumer Computer may either ignore the asterisk (*) command or adapt their own external programs to use it. Refer to **Appendix C - Mensch Monitor Assembly Listing** for specific details regarding the internal operation of this command.

Programming

The monitor is masked into the W65C265S's internal ROM, and resides between $00:E000 and $00:FFFF. This portion of the monitor handles the serial ports and provides a simple command line interface. The command line functions include program upload and download, memory display, alter, and fill. There are additional functions for dealing with the W65C265S's internal registers, the real time clock, and the serial handshaking. The command line interface runs on serial port #3.



This firmware also handles all interrupts (since the vectors are in its screened memory). In most cases, interrupts are re-vectored through RAM to the user's routine. In some cases, the RAM vectors are initialized to point back into the core ROM. The internal ROM has routines for buffering the serial ports, for updating the time of day clock, and so forth. The user may handle these tasks with custom software. The only change necessary involves re-writing the vectors in RAM. All routines in the masked ROM are accessed via a JSL table at $00:E000.

The monitor checks for semaphore sequences in external memory which indicate that the user wants complete control of the PCB and processor. These ares checked immediately after reset. If the 'WDC' pattern is present, the monitor transfers control to the appropriate address before setting any interrupt vectors or internal registers. The user's code can then turn off the internal ROM. All accesses, including accesses to interrupt vectors, may now be pulled from the external memory, typically EPROM. In this case, none of the monitor's routines are accessible to the user.

The semaphore sequence: 'WDC' may appear in three areas. The first area checked is: $00:8000 through $00:8002. If the 'WDC' pattern is found, the monitor will JMP to location $00:8004. This test is actually performed twice. The first time CS4 is ON and CS5 is OFF. The second time, CS4 is OFF and CS5 is ON. The third semaphore series: $00:0800 through $00:0802 will initiate a JMP to location: $00:0804.

Monitor Features/Functions

The W65C265S's hardware and firmware can be used for a variety of applications. There are numerous routines in the firmware designed to handle the hardware and several common interrupt tasks. Most applications could use some of these routines[4].

[4] The subroutines which are described in detail in this manual have ROM vectors at fixed addresses. These vectors will not move as new revisions of the monitor become available. Other subroutines may be found in **Appendix C - Mensch Monitor Assembly Listing**. They may be accessed directly. The user should be aware that such subroutines may be modified, relocated, or removed from later versions of the Mensch Monitor in the W65C265S ROM .

The simplest way to explain how these routines work is by example[5]. The examples shown in this manual are intended to illustrate. They are not necessarily complete. (Error returns are often ignored.) This code was written for clarity rather than speed or size.

Example:

```
        CHIP  65C816
        LONGA OFF
        LONGI ON
        SPACES ON
;
; This program sends the phrase 'Hello, world!' to the serial port.
;
        INCLUDE FIRMWARE.H ;Firmware equates and JMPs

        .ORG        $0800       ;A good test location to start

        JSL   SEND_CR;get to the next line (send CRLF)
        LDA #0                  ;Bank address to accumulator
        LDX #HSTRING            ;16-bit address to X
        JSL   PUT_STR ;Print it
        BRK                     ;return to monitor

HSTRING  .BYTE  'Hello, world!'       ;define the text
          .BYTE  $0D          ;a 'return'
ESTRING  .BYTE  $00           ;define the end of text
```

'Hello, world!'

This (right) is one of the most universal and simplest examples of how to start programming on a new system. The purpose of this example is to try to get the program edited, assembled, linked, into the W65C265S, and run. The exact operation of the program is secondary.

This example was written, assembled, and linked on an IBM AT using 2500AD's W65C816S cross assembler and linker. The result of the assembly and link is a file consisting of the absolute object code in Motorola S28 format.

Interrupts

The W65C265S supports many interrupts, each with its own vector. Definitions of the interrupts are found in the Western Design Center W65C265S Data Specification.

The Mensch Monitor firmware uses some of the interrupts. All interrupts go indirectly through locations in ROM which contain vectors (16-bit addresses) to appropriate interrupt handlers. Many of the interrupt vectors point to routines in the monitor ROM. The UART interrupts are all processed by buffering routines in the ROM.

Several vectors point to ROM JMP(XXXX) instructions, wherein: 'XXXX' is a RAM address. This extra step allows a user-defined RAM vector for the interrupt. The Mensch Monitor start-up initialization sequence will configure all RAM vectors to

[5] The Mensch Monitor source code is included in this manual. There is a cross reference at the end of each portion of the source which can be used to find specific sections.

point to appropriate code in the ROM. These initial settings may be changed by user applications as needed. (Refer to: **Appendix C - Mensch Monitor Assembly Listing** for details regarding specific interrupts.

Interrupts on the W65C265S must be enabled in either the interrupt enable registers, the serial control ports, or the timer control registers. Enabling, disabling, and resetting interrupts is covered in the WDC W65C265S data specification.

Low-Power Mode

The W65C265S uses CMOS circuitry. Power consumption is therefore a function of how many devices are switching and how often they switch. Applications may reduce power consumption by switching to the slower default clock (32 KHz). Further power reductions may be achieved by using instructions such as STP (stop) and WAI (wait) which halt the processor until an interrupt occurs.

Dramatic reductions in power may involve shutting off all circuitry other than the microprocessor. This automatically occurs when incoming power is lost. All devices excepting the W65C265S chip lose power[6]. An interrupt will tell the monitor when main power is going down. This interrupt invokes a routine in the internal ROM of the microprocessor which shuts off the fast clock, the address and data bus, and sets all external lines low. The only external sign of operation will be that the slow clock is still running. All interrupts will be disabled, excepting the time-of-day interrupt. When the time-of-day interrupt occurs, the clock will be updated. The time-of-day clock includes an alarm function which sets a flag when the current time equals the alarm time previously set. The alarm condition will signal monitor to exit from low-power mode. Next, internal RAM will be checked for a flag which indicates that a user's program has been loaded into the on-board RAM. If present, the user's program is run after every time-of-day interrupt (once per second). The on-board RAM limits the user's low-power program size to approximately 100 bytes.

Low-power mode essentially allows the user to write a small program which runs when main power is off. This program could check the I/O ports, looking for special conditions. This user's routine could then decide it is time to re-power the entire system, doing so by raising an I/O bit to turn on external power.

Low-power mode will only work if the W65C265S is running an *internal* program. During low-power mode, the monitor shuts off the external bus and therefore does not have access to the external EPROM or the external RAM. The monitor and it's low-power support routines reside in the internal mask ROM of the W65C265S chip.

Refer to **Appendix C - Mensch Monitor Assembly Listing** for details of the monitor's support code for low-power mode.

Serial I/O Support

The Mensch Monitor provides a user console via the serial UART as a fundamental feature. This allows the user to examine and change memory or registers, load, debug and save programs, and even interact with the Time-of-Day clock. The firmware involved in the implementation of these functions includes many useful subroutines.

There are several serial I/O support subroutines which allow developers to take advantage of the console features in an application program. The developer may use some of these features merely for debugging. Others could become a significant part of the target system.

Complete, detailed descriptions of the vectored serial I/O support subroutines are provided elsewhere in this manual. Refer to: **Monitor Library Subroutines** for more information.

---

[6] This assumes that in the application, the W65C265S chip has some form of battery backup. Otherwise, this section on low-power mode is irrelevant.

Time-of-Day Clock Support

The Time-of-Day clock is probably the most versatile and useful feature of the Mensch Monitor. It provides a "real-world" reference to programs, allows the programmable alarm function to operate, and continues to run even in "low-power" mode.

Several monitor subroutines support the Time-of-Day clock and alarm access by user programs. Most of these subroutines expect or return day, date and time data as a strings.

Descriptions of the specific support subroutines for the Time-of-Day clock are provided in: **Monitor Library Subroutines**. Refer to that section of this manual for more information.

Programmable Alarm

The Time-of-Day clock functions include a programmable alarm feature. The alarm is initialized with day, date and time values. These settings are compared each second to the current value of the Time-of-Day clock. The monitor will set a flag when an acceptable match occurs. This alarm flag will be cleared to zero by the reset initialization sequence. It may set, reset, or checked by user application software[7].

The application software should use the SET_ALARM subroutine to initialize the alarm.

Data Manipulation

Several common data checking and manipulation functions are used by the Mensch Monitor internally. All are available to the application software. Some of these functions convert data from one format to another. Others merely check data using pre-defined criteria. Those routines listed in the table (right) are described in detail elsewhere. These routines all have vectors at the base of the W65C265S internal ROM.

[7] The Mensch Monitor in the W65C265S ROM does not use the programmable alarm feature. This is entirely available to the user application software.

Data Checking/Manipulation Functions

Routine	Vector	Description
ASCBIN	$00:E087	ASCII to Binary conversion (2:1)
BINASC	$00:E08F	Binary to ASCII conversion
BIN2DEC	$00:E08B	Binary to Decimal conversion
HEXIN	$00:E093	Converts ASCII HEX to binary (1:1)
IFASC	$00:E097	Check for ASCII
ISDECIMAL	$00:E09B	Check for ASCII decimal digit.
ISHEX	$00:E09F	Check for ASCII hexadecimal digit.
UPPER_CASE	$00:E0A3	Convert character to Upper-Case ASCII

There are additional subroutines which may be useful to the applications developer, but do not have vectors. These subroutines may be found in **Appendix C - Mensch Monitor Assembly Listing**. They may be accessed directly, but the user should take special care. Some are used by the other functions listed, and could possibly cause conflicts.

Non-vectored subroutines may be modified, relocated, or even deleted in later versions of the Mensch Monitor ROM[8].

[8] The Mensch Monitor ROM is an evolving product. Future changes are expected to maintain the integrity of the vectored functions. Simple, useful internal routines may be relocated as a result of additions or deletions but will probably not be removed. Application software, which uses direct access to these routines, may require re-assembly.

Custom Commands

The system developer may choose to add or replace monitor commands. Commands may be added via the '*U*' command or through an alternate parser.

When the user enters: 'U' from the console, the Mensch Monitor will JSL to a JML indirect through a pointer. The "user command pointer", **USER_CMD** (located @ $00:012C)must be initialized by the application program. Typically, the four locations will contain a long jump (**JML**) to the user's custom command processor or parser logic. The custom command processor/parser should return to the monitor via an RTL instruction upon completion. The monitor will start again and prompt for the next command input.

```
                .ORG    $01:0080

HELLO           DB      "Transfer of control to this routine successful,"
                DB      " press any key to continue",0

                .ORG    $01:0400

USER_CMD_TEST

; Displays message, gets a key and returns

                JSL     SEND_CR

                LDA     #1
                LDX     #HELLO
                JSL     PUT_STR

                JSL     SEND_CR

                JSL     GET_CHR

                RTL
```

The subroutine source shown above was used to generate the code in the '*U* command example. It demonstrates the essential linkage and I/O necessary to implement a custom command processor for use with the ROM monitor.

Monitor Library Subroutines

The W65C265S has a large number of standard subroutines, with fixed JSL vectors, available to the user. The JSL table is in the mask ROM, and starts at address: $00:E000.

Name	Vector
Alter_Memory	$00:E000
BACKSPACE	$00:E003
Reserved For Expansion.	$00:E006
CONTROL_TONES	$00:E009
DO_LOW_POWER_PGM	$00:E00C
DUMPREGS	$00:E00F
DumpS28	$00:E012
Dump_1_line_to_Output	$00:E015
Dump_1_line_to_Screen	$00:E018
Dump_to_Output	$00:E01B
Dump_to_Printer	$00:E01E
Dump_to_Screen	$00:E021
Dump_to_Screen_ASCII	$00:E024
Dump_It	$00:E027
FILL_Memory	$00:E02A
GET_3BYTE_ADDR	$00:E02D
GET_ALARM_STATUS	$00:E030
GET_BYTE_FROM_PC	$00:E033
GET_CHR	$00:E036
GET_HEX	$00:E039
GET_PUT_CHR	$00:E03C
GET_STR	$00:E03F
Get_Address	$00:E042
Get_E_Address	$00:E045
Get_S_Address	$00:E048
PUT_CHR	$00:E04B
PUT_STR	$00:E04E

Name	Vector
READ_ALARM	$00:E051
READ_DATE	$00:E054
READ_TIME	$00:E057
RESET_ALARM	$00:E05A
SBREAK	$00:E05D
SELECT_COMMON_BAUD_RATE	$00:E060
SEND_BYTE_TO_PC	$00:E063
SEND_CR	$00:E066
SEND_SPACE	$00:E069
SEND_HEX_OUT	$00:E06C
SET_ALARM	$00:E06F
SET_Breakpoint	$00:E072
SET_DATE	$00:E075
SET_TIME	$00:E078
VERSION	$00:E07B
WR_3_ADDRESS	$00:E07E
XS28IN	$00:E081
RESET	$00:E084
ASCBIN	$00:E087
BIN2DEC	$00:E08B
BINASC	$00:E08F
HEXIN	$00:E093
IFASC	$00:E097
ISDECIMAL	$00:E09B
ISHEX	$00:E09F
UPPER_CASE	$00:E0A3
Reserved For Expansion.	$00:E0A7

The easiest way to use these subroutines is to set up an equate table, as described in **Appendix B - Mensch Monitor Subroutines**, to define their names. The application may then simply JSL to the appropriate subroutine name as needed. Upon completion, the subroutine will RTL back to the caller. These firmware subroutines reduce the overhead of user applications, and also standardize the way common operations are performed.

Alter_ Memory ($00:E000)

DESCRIPTION:

This is the subroutine invoked by typing the '**M**' command at the monitor prompt. Basically, **Alter_Memory** will request an address and accept input via the console port #3.

The user must enter six ASCII-Hex digits to form a 24-bit address. The input format is:

BB:AAAA

Wherein: "BB" is the bank address. This subroutine will echo a ':' after the 2-digit bank address. "AAAA" is the offset address within the bank.

After a valid address has been entered, **Alter_Memory** prints a single line memory dump starting at the specified address. It then prints a second line repeating the address and positioning under the contents of the first location. The user may input new hexadecimal character data, one byte at a time. This subroutine will automatically position under the next location as values are entered.

Entering a **SPACE** ($20) character skips the current location, without changing it, and positions on the next. The user may terminate this operation at anytime by typing **ENTER** ($0D).

> NOTE: This subroutine would not normally be used by application software on the W65C265. It has been included in this vector table to support anticipated needs of the extended Mensch Computer Operating System in that specific configuration. Developers should consult **Appendix C - Mensch Monitor Assembly Listing** for specific details regarding internal operation of this subroutine in order to determine suitability for other applications and configurations.

EXPECTS:

No input arguments. This is an interactive subroutine which requests parameters from the user as needed.

RETURNS:

The carry-bit will be *clear* if the operation was successfully performed.

ERRORS:

The carry-bit will be *set* if any errors were detected.

ASCBIN ($00:E087)

DESCRIPTION:
This subroutine will convert two ASCII HEX[9] characters into a single binary byte. The ASCII hexadecimal character in register-A corresponds to the least significant nibble of the result. The most significant nibble is defined by the ASCII hexadecimal character in **TEMP**. The resulting binary value will be returned in register-A. The carry-bit will be *clear* upon a normal return from this subroutine. Refer to **Appendix C - Mensch Monitor Assembly Listing** for specific details regarding the internal operation of the **ASCBIN** subroutine.

EXPECTS:
The ASCII hexadecimal character in register-A corresponds to the least significant nibble of the result.

The most significant nibble is defined by the ASCII hexadecimal character in the global variable: **TEMP** ($00:0070).

RETURNS:
The resulting binary value will be returned in register-A.

The carry-bit will be *clear* if the operation was successfully performed.

ERRORS:
The carry-bit will be *set* if either parameter was not an ASCII HEX digit.

BACKSPACE ($00:E003)

DESCRIPTION:
This subroutine will output a **BS** (Backspace = $08) character to console port #3. Refer to **Appendix C - Mensch Monitor Assembly Listing** for specific details regarding the internal operation of the **BACKSPACE** subroutine.

EXPECTS:
No input arguments.

RETURNS:
The carry-bit will always be *clear* upon completion.

ERRORS:

No errors reported.

---

[9] A valid ASCII hexadecimal digit is a numeric character: "0123456789" ($2F < char < $3A) or one of the first six letters: "ABCDEF" ($40 < char < $47) in uppercase or lowercase: "abcdef" ($61 < char < $7A).

BIN2DEC ($00:E08B)

DESCRIPTION:
This subroutine will take a binary value ($00-$63) in register-A and convert it to packed decimal format ($00-$99) also in register-A. Values larger than 99 ($63) will not be properly converted. Refer to **Appendix C - Mensch Monitor Assembly Listing** for specific details regarding the internal operation of the **BIN2DEC** subroutine.

Example:

```
LDA HOURS      ;current HOURS (00-$17)
JSL BIN2DEC    ;make it decimal (00-$23)
JSL SEND_HEX_OUT
...
```

EXPECTS:
Binary value ($00-$63) in register-A.

RETURNS:
Equivalent value in packed decimal format ($00-$99) in register-A.

Values larger than $63 will not be properly converted.

The carry-bit will always be *clear* upon completion.

ERRORS:
No errors reported. The **BIN2DEC** subroutine does not detect any errors or return error codes. If the calling program passes a binary value larger than $63 to this subroutine, the resulting conversion value will be meaningless.

BINASC ($00:E08F)

DESCRIPTION:
This subroutine will convert an 8-bit binary value in register-A into two ASCII HEX characters. Register-A returns the least significant character in ASCII. **TEMP+1** returns the most significant character. Refer to **Appendix C - Mensch Monitor Assembly Listing** for specific details regarding the internal operation of the **BINASC** subroutine.

EXPECTS:
Binary value in 8-bit register-A.

RETURNS:
Least significant character in ASCII in Register-A.

Most significant character in ASCII in the global variable: **TEMP+1** ($00:0071).

ERRORS:
No errors reported.

CONTROL_TONES ($00:E009)

DESCRIPTION:
This subroutine will configure timers: **T5** and **T6**, and gate either or both tone generators to the audio outputs: **TG0** and **TG1**. Configuration values for the timers may need to be calculated for each implementation. The values necessary to produce specific tones are dependent upon the frequency of the fast clock (FCLK)[10]. Refer to **Appendix C - Mensch Monitor Assembly Listing** for specific details regarding the internal operation of the **CONTROL_TONES** subroutine.

EXPECTS:
Control code in 8-bit register-A:

0 = Both tone generators disabled.

1 = Tone generator TG0 is enabled.

2 = Tone generator TG1 is enabled.

3 = Both tone generators enabled.

Other = Invalid

Configuration value for timer: **T5** in 16-bit register-X.

Configuration value for timer: **T6** in 16-bit register-Y.

RETURNS:
No arguments returned.

ERRORS:
The carry-bit normally will return *clear*, but will be *set* if the control code in 8-bit register-A was invalid.

[10] A thorough description of the algorithm is provided in: W65C265S INFORMATION SPECIFICATION AND DATA SHEET. It also includes precalculated tables of values for typical FCLK frequencies and commonly needed tones: DTMF, modems, etc. This document and related literature is available from WDC.

DO_LOW_POWER_PGM ($00:E00C)

DESCRIPTION:
This vector will force the system into low-power mode. Basically, this involves:

1. Reset the *stack pointer* to $00:01FF.

2. Turn OFF all I/O.

3. Shut down all chip selects.

4. Perform *low power mode* maintenance loop.

   - Service interrupts from timer #1, 1/second.
     Update time-of-day clock/calendar and alarm.

   - Execute *User Check Program* subroutine located at: $00:01C0.

Step #4 will repeat, keeping the W65C265 in *low power mode*. This will continue until one of the following events:

- System **RESET** occurs.

- The **Alarm** function times out.

- The *User Check Program* subroutine initiates exit from *low power mode* to begin normal operation again.

Refer to **Appendix C - Mensch Monitor Assembly Listing** for specific details regarding the internal operation of *low power mode*.

EXPECTS:
No input arguments.

RETURNS:
This vector does not return.

ERRORS:
No meaningful errors.

DUMPREGS ($00:E00F)

DESCRIPTION:
This is the subroutine invoked by typing the '**R**' command at the monitor prompt. Basically, **DUMPREGS** will write a formatted display of the register values as they were saved at the most recent monitor prompt. These values were used to initialize the registers prior to the monitor releasing control. This output will be sent to console port #3.

> NOTE: This subroutine would not normally be used by application software on the W65C265. It has been included in this vector table to support anticipated needs of the extended Mensch Computer Operating System in that specific configuration. Developers should consult **Appendix C - Mensch Monitor Assembly Listing** for specific details regarding internal operation of this subroutine in order to determine suitability for other applications and configurations.

EXPECTS:
No input arguments. This is an interactive subroutine which requests parameters from the user as needed.

RETURNS:
The carry-bit will always be *clear* upon completion.

ERRORS:
No errors reported.

DumpS28 ($00:E012)

DESCRIPTION:
This is the subroutine invoked by typing the '**W**' command at the monitor prompt. Basically, **DumpS28** will request *lowest and highest addresses* via the console output port #3. It will accept responses via the console input port #3. Then, it will dump the specified memory block in S28 loader format to console output port #3. The last record written will begin with "S8", indicating that none follow.

> **S28 Format:** S2LLHHMMLWDDDDDDDD...CC
> where: S2 = literally the ASCII characters:"S2",
> LL = length of the data+4,
> HH = high byte of the address,
> MM = middle byte of the address,
> LW = low byte of the address,
> DD = one byte of data, next byte, etc
> CC = checksum (1's complement of the sum of the length, address, and data bytes.)

The user must enter six ASCII-Hex digits to form each 24-bit address. The input format is:

BB:AAAA

Wherein: "BB" is the bank address. This subroutine will echo a ':' after the 2-digit bank address. "AAAA" is the offset address within the bank.

After valid addresses have been entered, **DumpS28** will write S28 records for the entire memory block. If the *lowest address* is greater than the *highest address*, then only one sixteen byte record will be dumped. It will begin with the specified *lowest address*.

> NOTE: This subroutine would not normally be used by application software on the W65C265. It has been included in this vector table to support anticipated needs of the extended Mensch Computer Operating System in that specific configuration. Developers should consult **Appendix C - Mensch Monitor Assembly Listing** for specific details regarding internal operation of this subroutine in order to determine suitability for other applications and configurations.

EXPECTS:

No input arguments. This is an interactive subroutine which requests parameters from the user as needed.

RETURNS:

The carry-bit will be *clear* if the operation was successfully performed.

ERRORS:

The carry-bit will be *set* if invalid address data was entered.

Dump_1_line_to_Output ($00:E015)

DESCRIPTION:

This subroutine will request the *starting address* and accept input via the console port #3.

The user must enter six ASCII-Hex digits to form the 24-bit address. The input format is:

BB:AAAA

Wherein: "BB" is the bank address. This subroutine will echo a ':' after the 2-digit bank address. "AAAA" is the offset address within the bank.

After a valid address has been entered, **Dump_1_line_to_Output** will write a formatted header and one line, sixteen bytes, of memory dump data to console port #3.

NOTE: This subroutine would not normally be used by application software on the W65C265. It has been included in this vector table to support anticipated needs of the extended Mensch Computer Operating System in that specific configuration. Developers should consult **Appendix C - Mensch Monitor Assembly Listing** for specific details regarding internal operation of this subroutine in order to determine suitability for other applications and configurations.

EXPECTS:
No input arguments. This is an interactive subroutine which requests parameters from the user as needed.

RETURNS:
The carry-bit will be *clear* if the operation was successfully performed.

ERRORS:
The carry-bit will be *set* if invalid address data was entered.

Dump_1_line_to_Screen ($00:E018)

DESCRIPTION:
This subroutine will request the *starting address* and accept input via the console port #3.

The user must enter six ASCII-Hex digits to form the 24-bit address. The input format is:
BB:AAAA

Wherein: "BB" is the bank address. This subroutine will echo a ':' after the 2-digit bank address. "AAAA" is the offset address within the bank.

After a valid address has been entered, **Dump_1_line_to_Screen** will write a formatted header and two lines of eight bytes of memory dump data to console port #3.

NOTE: This subroutine would not normally be used by application software on the W65C265. It has been included in this vector table to support anticipated needs of the extended Mensch Computer Operating System in that specific configuration. Developers should consult **Appendix C - Mensch Monitor Assembly Listing** for specific details regarding internal operation of this subroutine in order to determine suitability for other applications and configurations.

EXPECTS:
No input arguments. This is an interactive subroutine which requests parameters from the user as needed.

RETURNS:
The carry-bit will be *clear* if the operation was successfully performed.

ERRORS:
The carry-bit will be *set* if invalid address data was entered.

Dump_It ($00:E027)

DESCRIPTION:
This subroutine will perform a memory dump operation to serial port #3. The memory range and dump configuration must be provided by the caller. It performs no error checking, as it assumes validation will be performed before parameters are passed. Refer to **Appendix C - Mensch Monitor Assembly Listing** for specific details regarding the internal operation of the **Dump_It** subroutine.

> NOTE: This subroutine would not normally be used by application software on the W65C265. It has been included in this vector table to support anticipated needs of the extended Mensch Computer Operating System in that specific configuration. Developers should consult **Appendix C - Mensch Monitor Assembly Listing** for specific details regarding internal operation of this subroutine in order to determine suitability for other applications and configurations.

EXPECTS:
Dump configuration parameter in 8-bit register-A.

Least significant bit:	#0 =	Output S28+byte-count.
	1 =	Format for LCD (40 char/line).
Note: Some possible	2 =	Add spaces between data bytes & HEADER.
combinations	3 =	Add checksum.
are invalid or	4 =	8 bytes not 16 bytes per line.
unpredictable!	5 =	ONE LINE ONLY. *(Not Used by: Dump_It)*
	6 =	ASCII not Hex data.
Most significant bit:	#7 =	*(Not Used.)*

Non-zero number of dump data lines per page in 16-bit register-X. (Note: A header may also be printed.)

The 3-byte starting address must be loaded into the global variable: **TMP0**($00:005D), **TMP0+1**, and **TMP0+2**. The least significant byte (LSB) of the 3-byte address must reside in **TMP0** and the MSB must be in **TMP0+2**.

The 3-byte ending address must be loaded into the global variable: **TMP2**($00:0063),

TMP2+1, and **TMP2+2**. The least significant byte (LSB) of the 3-byte address must reside in **TMP2** and the MSB must be in **TMP2+2**.

RETURNS:
No arguments returned.

ERRORS:
No meaningful errors are detected or reported.

Dump_to_Output ($00:E01B)

DESCRIPTION:
This subroutine will request the *starting and ending addresses* and accept input via the console port #3.

The user must enter six ASCII-Hex digits to form each 24-bit address. The input format is:
BB:AAAA

Wherein: "BB" is the bank address. This subroutine will echo a ':' after the 2-digit bank address. "AAAA" is the offset address within the bank.

After valid addresses have been entered, **Dump_to_Output** will write a **Form-Feed** ($0C) and formatted header line to console port #3. This will be followed by up to sixty lines, of sixteen bytes each, of memory dump data. If the range of memory requires more than sixty lines of dump data, then another form-feed/header will be generated before more dump data is sent. This cycle will repeat until the entire specified block of memory has been dumped.

> NOTE: This subroutine would not normally be used by application software on the W65C265. It has been included in this vector table to support anticipated needs of the extended Mensch Computer Operating System in that specific configuration. Developers should consult **Appendix C - Mensch Monitor Assembly Listing** for specific details regarding internal operation of this subroutine in order to determine suitability for other applications and configurations.

EXPECTS:
No input arguments. This is an interactive subroutine which requests parameters from the user as needed.

RETURNS:
The carry-bit will be *clear* if the operation was successfully performed.

ERRORS:
The carry-bit will be *set* if invalid address data was entered.

Dump_to_Printer ($00:E01E)

DESCRIPTION:
This subroutine will request the *starting and ending addresses* and accept input via the console port #3.

The user must enter six ASCII-Hex digits to form each 24-bit address. The input format is:
BB:AAAA

Wherein: "BB" is the bank address. This subroutine will echo a ':' after the 2-digit bank address. "AAAA" is the offset address within the bank.

After valid addresses have been entered, **Dump_to_Printer** will write a **Form-Feed** ($0C) and formatted header line to console port #3. This will be followed by up to sixty lines, of sixteen bytes each, of memory dump data. If the range of memory requires more than sixty lines of dump data, then another form-feed/header will be generated before more dump data is sent. This cycle will repeat until the entire specified block of memory has been dumped.

NOTE: This subroutine would not normally be used by application software on the W65C265. It has been included in this vector table to support anticipated needs of the extended Mensch Computer Operating System in that specific configuration. Developers should consult **Appendix C - Mensch Monitor Assembly Listing** for specific details regarding internal operation of this subroutine in order to determine suitability for other applications and configurations.

EXPECTS:
No input arguments. This is an interactive subroutine which requests parameters from the user as needed.

RETURNS:
The carry-bit will be *clear* if the operation was successfully performed.

ERRORS:
The carry-bit will be *set* if invalid address data was entered.

Dump_to_Screen ($00:E021)

DESCRIPTION:
This subroutine will request the *starting and ending addresses* and accept input via the console port #3.

The user must enter six ASCII-Hex digits to form each 24-bit address. The input format is:
BB:AAAA

Wherein: "BB" is the bank address. This subroutine will echo a ':' after the 2-digit bank address. "AAAA" is the offset address within the bank.

After valid addresses have been entered, **Dump_to_Screen** will write a formatted header line to console port #3. This will be followed by up to twelve lines, of eight bytes each, of memory dump data. If the range of memory requires more than twelve lines of dump data, then the **Dump_to_Screen** subroutine will pause for input from console port #3. The user may enter any character to acknowledge this pause. Another header will be generated before more dump

> NOTE: This subroutine would not normally be used by application software on the W65C265. It has been included in this vector table to support anticipated needs of the extended Mensch Computer Operating System in that specific configuration. Developers should consult **Appendix C - Mensch Monitor Assembly Listing** for specific details regarding internal operation of this subroutine in order to determine suitability for other applications and configurations.

data is sent. This cycle will repeat until the entire specified block of memory has been dumped.

EXPECTS:
No input arguments. This is an interactive subroutine which requests parameters from the user as needed.

RETURNS:
The carry-bit will be *clear* if the operation was successfully performed.

ERRORS:
The carry-bit will be *set* if invalid address data was entered.

Dump_to_Screen_ASCII ($00:E024)

DESCRIPTION:
This subroutine will request the *starting and ending addresses* and accept input via the console port #3.

The user must enter six ASCII-Hex digits to form each 24-bit address. The input format is:
BB:AAAA

Wherein: "BB" is the bank address. This subroutine will echo a ':' after the 2-digit bank address. "AAAA" is the offset address within the bank.

After valid addresses have been entered, **Dump_to_Screen_ASCII** will write up to twelve lines, of sixteen bytes each, of ASCII dump data. Values which do not translate to printable ASCII characters will appear as an apostrophe ('). If the range of memory requires more than twelve lines of dump data, then the **Dump_to_Screen_ASCII** subroutine will pause for input from console port #3. The user may enter any character to acknowledge this pause. Up to twelve more lines of dump data will be sent. This cycle will repeat until the entire specified block of memory has been dumped.

The final dump will be followed by a pause. Again, the user may enter any character to acknowledge this pause.

NOTE: This subroutine would not normally be used by application software on the W65C265. It has been included in this vector table to support anticipated needs of the extended Mensch Computer Operating System in that specific configuration. Developers should consult **Appendix C - Mensch Monitor Assembly Listing** for specific details regarding internal operation of this subroutine in order to determine suitability for other applications and configurations.

EXPECTS:
No input arguments. This is an interactive subroutine which requests parameters from the user as needed.

RETURNS:
The carry-bit will be *clear* if the operation was successfully performed.

ERRORS:
The carry-bit will be *set* if invalid address data was entered.

FILL_Memory ($00:E02A)

DESCRIPTION:
This is the subroutine invoked by typing the '**F**' command at the monitor prompt. Basically, **FILL_Memory** will request *starting and ending addresses*, a fill constant via the console output port #3. It will accept responses via the console input port #3.

The user must enter six ASCII-Hex digits to form each 24-bit address. The input format is:
BB:AAAA

Wherein: "BB" is the bank address. This subroutine will echo a ':' after the 2-digit bank address. "AAAA" is the offset address within the bank.

After a valid addresses have been entered, **FILL_Memory** will expect a *fill constant* as two hexadecimal characters. It will then write the *fill constant* value to every location in the specified memory block.

NOTE: This subroutine would not normally be used by application software on the W65C265. It has been included in this vector table to support anticipated needs of the extended Mensch Computer Operating System in that specific configuration. Developers should consult **Appendix C - Mensch Monitor Assembly Listing** for specific details regarding internal operation of this subroutine in order to determine suitability for other applications and configurations.

EXPECTS:
No input arguments. This is an interactive subroutine which requests parameters from the user as needed.

RETURNS:
The carry-bit will be *clear* if the operation was successfully performed.

ERRORS:
The carry-bit will be *set* if any errors were detected.

GET_3BYTE_ADDRESS ($00:E02D)

DESCRIPTION:
This subroutine accepts six ASCII-Hex digits, from console port #3, to form a 24-bit address. The input format is:

BB:AAAA

Wherein: "BB" is the bank address. This subroutine will echo a ':' after the 2-digit bank address, to console port #3. "AAAA" is the offset address within the bank.

Refer to **Appendix C - Mensch Monitor Assembly Listing** for specific details regarding the internal operation of the **GET_3BYTE_ADDRESS** subroutine.

EXPECTS:
No input arguments.

RETURNS:
No output arguments.

The 3-byte result is returned in the global variables: **TMP2**($00:0063), **TMP2+1**, and **TMP2+2**. The bytes are ordered such that: **TMP2**=LSB and **TMP2+2**=MSB.

The subroutine will return with the carry-bit=*clear* if a proper 6-digit address has been received.

ERRORS:
The carry-bit will be returned *set* if any non-digit character is detected before all six ASCII-Hex digits have been received.

Get_Address 00:E042)

DESCRIPTION:
This subroutine writes the following prompt string to console output port #3:

"Enter Address: BB:AAAA"

It then performs: **GET_3BYTE_ADDRESS** which accepts six ASCII-Hex digits, from console port #3, to form a 24-bit address. The input format is:

BB:AAAA

Wherein: "BB" is the bank address. This subroutine will echo a ':' after the 2-digit bank address, to console port #3. "AAAA" is the offset address within the bank.

Refer to **Appendix C - Mensch Monitor Assembly Listing** for specific details regarding the internal operation of the **Get_Address** subroutine.

EXPECTS:
No input arguments.

RETURNS:
No output arguments.

The 3-byte result is returned in the global variables: **TMP2**($00:0063), **TMP2+1**, and **TMP2+2**. The bytes are ordered such that: **TMP2**=LSB and **TMP2+2**=MSB.

The subroutine will return with the carry-bit=*clear* if a proper 6-digit address has been received.

ERRORS:
The carry-bit will be returned *set* if any non-digit character is detected before all six ASCII-Hex digits have been received.

GET_ALARM_STATUS ($00:E030)

DESCRIPTION:
This subroutine will retrieve the current status of the system alarm. Refer to **Appendix C - Mensch Monitor Assembly Listing** for specific details regarding the internal operation of the

GET_ALARM_STATUS subroutine.

Note: Calling this subroutine will also automatically reset the alarm, if it has been triggered. Non-destructive testing may be accomplished by accessing the alarm flag directly from page #0. (Refer to the source code listing for specific details.)

EXPECTS:
No input arguments.

RETURNS:
Alarm status returned in 8-bit register-A:

Zero = Alarm has not been set.

Any Non-Zero = Alarm has been set.

The carry-bit will be set if the alarm has been triggered, otherwise it will be clear upon return.

ERRORS:
No meaningful errors.

GET_BYTE_FROM_PC ($00:E033)

DESCRIPTION:
This subroutine will read the next available byte from the PC link serial port #3 input buffer. If the buffer is empty, then the subroutine will return with the carry-bit set. Refer to **Appendix C - Mensch Monitor Assembly Listing** for specific details regarding the internal operation of the **GET_BYTE_FROM_PC** subroutine.

EXPECTS:
No input arguments.

RETURNS:
Received byte from PC link serial port in 8-bit register-A.

ERRORS:
The carry-bit will return *clear* if a received data byte is available in 8-bit register-A. It will be *set* if no received data was available.

GET_CHR ($00:E036)

DESCRIPTION:
This subroutine will get a single character from the console input port #3. Refer to **Appendix C - Mensch Monitor Assembly Listing** for specific details regarding the internal operation of the **GET_CHR** subroutine.

EXPECTS:
No input arguments.

RETURNS:
Always returns with carry-bit *clear*, and received character in 8-bit register-A. All other registers are saved upon entry and restored before returning.

ERRORS:
No meaningful errors.

Get_E_ Address ($00:E045)

DESCRIPTION:
This subroutine writes the following prompt string to console output port #3:

"Enter Highest Address: BB:AAAA"

It then performs: **GET_3BYTE_ADDRESS** which accepts six ASCII-Hex digits, from console port #3, to form a 24-bit address. The input format is:

BB:AAAA

Wherein: "BB" is the bank address. This subroutine will echo a ':' after the 2-digit bank address, to console port #3. "AAAA" is the offset address within the bank.

Refer to **Appendix C - Mensch Monitor Assembly Listing** for specific details regarding the internal operation of the **Get_E_Address** subroutine.

EXPECTS:
No input arguments.

RETURNS:
No output arguments.

The 3-byte result is returned in the global variables: **TMP2**($00:0063), **TMP2+1**, and **TMP2+2**. The bytes are ordered such that: **TMP2**=LSB and **TMP2+2**=MSB.

The subroutine will return with the carry-bit=*clear* if a proper 6-digit address has been received.

ERRORS:
The carry-bit will be returned *set* if any non-digit character is detected before all six ASCII-Hex digits have been received.

GET_HEX ($00:E039)

DESCRIPTION:
This subroutine will get the next character from the console input port #3 into register-A. If it is a **SPACE** ($20) character, then **GET_HEX** will return with the carry-bit=*set*. Otherwise, the subroutine will accept another character.

If either byte is not an ASCII Hex character, then this subroutine will also return with the carry-bit=*set*, but register-A will be cleared to: $00.

If both bytes were ASCII Hex characters, the pair will be evaluated to produce a single binary byte of the value represented by the two hexadecimal digits. The **GET_HEX** subroutine will return with the value in register-A and the carry-bit=*set*.

Refer to **Appendix C - Mensch Monitor Assembly Listing** for specific details regarding the internal operation of the **GET_HEX** subroutine.

EXPECTS:
No input arguments.

RETURNS:
The carry-bit will be *set*, and 8-bit register-A=$20 if the first character was a **SPACE**.

or

The carry-bit will be *set*, and 8-bit register-A=$00 if either input byte was not an ASCII HEX digit.

or

The carry-bit will be *clear* and 8-bit register-A will contain the binary value represented by the two hexadecimal digits input.

ERRORS:
No errors reported.

GET_PUT_CHR ($00:E03C)

DESCRIPTION:
This subroutine inputs and echoes as outputs a character via the console port #3. Refer to **Appendix C - Mensch Monitor Assembly Listing** for specific details regarding the internal operation of the **GET_PUT_CHR** subroutine.

EXPECTS:
No input arguments.

RETURNS:
Always returns normally with carry-bit *clear* and the received character in 8-bit register-A. All other registers are saved upon entry and restored before returning.

ERRORS:
No meaningful errors.

Get_S_ Address ($00:E048)

DESCRIPTION:
This subroutine writes the following prompt string to console output port #3:

"Enter Lowest Address: BB:AAAA "

It then performs: **GET_3BYTE_ADDRESS** which accepts six ASCII-Hex digits, from console port #3, to form a 24-bit address. The input format is:

BB:AAAA

Wherein: "BB" is the bank address. This subroutine will echo a ':' after the 2-digit bank address, to console port #3. "AAAA" is the offset address within the bank.

Refer to **Appendix C - Mensch Monitor Assembly Listing** for specific details regarding the internal operation of the **Get_S_Address** subroutine.

EXPECTS:
No input arguments.

RETURNS:
No output arguments.

The 3-byte result is returned in the global variables: **TMP2**($00:0063), **TMP2+1**, and **TMP2+2**. The bytes are ordered such that: **TMP2**=LSB and **TMP2+2**=MSB.

The subroutine will return with the carry-bit=*clear* if a proper 6-digit address has been received.

ERRORS:
The carry-bit will be returned *set* if any non-digit character is detected before all six ASCII-Hex digits have been received.

GET_STR ($00:E03F)

DESCRIPTION:
This subroutine uses GET_PUT_CHR to receive characters and store them into a specified string buffer. The input string is terminated when an **ENTER** or **ESC** (Escape) character is detected. The completed string is terminated with a **NUL** ($00) character. Refer to **Appendix C - Mensch Monitor Assembly Listing** for specific details regarding the internal operation of the **GET_STR** subroutine.

EXPECTS:
Long pointer to string buffer as follows:

Bank address of string in 8-bit register-A.

Pointer to string in 16-bit register-X.

RETURNS:
No arguments returned.

The received data string, will be in specified string buffer, terminated with a **NULL** ($00) character.

Normally, the **GET_STR** subroutine returns with the carry-bit *clear* when the string entry was ended by the **ENTER** character.

ERRORS:
Exception returns with the carry-bit *set* if an **ESC** (Escape) character was detected. No other errors are detected or reported.

HEXIN ($00:E093)

DESCRIPTION:
This subroutine will convert an ASCII HEX[11] character in register-A into its equivalent binary value. The binary value is returned in the lower nibble of register-A. The carry-bit will be *clear* upon a normal return from this subroutine. Refer to **Appendix C - Mensch Monitor Assembly Listing** for specific details regarding the internal operation of the **HEXIN** subroutine.

---

[11] A valid ASCII hexadecimal digit is a numeric character: "0123456789" ($2F < char < $3A) or one of the first six letters: "ABCDEF" ($40 < char < $47) in uppercase or lowercase: "abcdef" ($61 < char < $7A).

Example:

```
LDA #'d'
JSL HEXIN
BCS NOTHEX3 ;Should never branch
CMP #$0D
BNE FAILED     ;Should never branch
...
LDA #'q'
JSL HEXIN
BCS NOTHEX3 ;Should always branch
...
```

EXPECTS:
The ASCII hexadecimal character in register-A corresponds to the least significant nibble of the result.

RETURNS:
The resulting binary value will be returned in register-A.

The carry-bit will be *clear* if the operation was successfully performed.

ERRORS:
The carry-bit will be *set* if the parameter was not an ASCII HEX digit. (Note: The contents of register-A may be modified even if the conversion is not performed.)

IFASC ($00:E097)

DESCRIPTION:
This subroutine will check the parameter byte in register-A. If the byte is a valid ASCII character[12] it will return with the carry-bit *clear*. The carry-bit will be *set* upon return if the character was not ASCII. Refer to **Appendix C - Mensch Monitor Assembly Listing** for specific details regarding the internal operation of the **IFASC** subroutine.

[12] This manual uses the term: "ASCII" when referring to *visible* characters ($1F < char < $7F) within the American Standard Code for Information Interchange. Special characters and control characters are of course also part of the ASCII character set.

Example:

```
LDA #$F1
JSL IFASC
BCC ASCII_YES ;Should never branch
...
LDA #$41
JSL IFASC
BCS NOT_ASCII ;Should never branch
...
```

EXPECTS:
Input parameter byte in 8-bit register-A.

RETURNS:
The carry-bit will be *clear* if the data byte in register-A corresponds to a visible ASCII character.

If the parameter byte passed in register-A is not a valid ASCII character, the **IFASC** subroutine will return with the carry-bit *set* in the status register.

ERRORS:
No errors reported.

ISDECIMAL ($00:E09B)

DESCRIPTION:
This subroutine will check the parameter byte in register-A. If the byte is a valid ASCII decimal digit ($30-$39), it will return with the carry-bit *clear* in the status register. If the carry-bit is *set* upon return, the character was not an ASCII decimal digit. Refer to **Appendix C - Mensch Monitor Assembly Listing** for specific details regarding the internal operation of the **ISDECIMAL** subroutine.

Example:

```
LDA #$FA
JSL ISDECIMAL
BCC DEC_YES ;Should never branch
...
LDA #'7'
JSL ISDECIMAL
BCS NOT_DEC ;Should never branch
...
```

EXPECTS:

Input parameter byte in 8-bit register-A.

RETURNS:
The carry-bit will be *clear* if the data byte in register-A corresponds to an ASCII decimal character.

If the parameter byte passed in register-A is not a valid ASCII decimal character, the **ISDECIMAL** subroutine will return with the carry-bit *set* in the status register.

ERRORS:
No errors reported.

ISHEX $00:E09F)

DESCRIPTION:
This subroutine will check the parameter byte in register-A. If the byte is a valid ASCII Hexadecimal digit ($30-$39,$41-$46, and $61-$66), it will return with the carry-bit *clear* in the status register. If the carry-bit is *set* upon return, the character was not an ASCII Hexadecimal digit. Refer to **Appendix C - Mensch Monitor Assembly Listing** for specific details regarding the internal operation of the **ISHEX** subroutine.

Ex**ample:**

```
LDA #$FA
JSL ISHEX
BCC HEX_YES ;Should never branch
...
LDA #'C'
JSL ISHEX
BCS NOT_HEX ;Should never branch
...
```

EXPECTS:
Input parameter byte in 8-bit register-A.

RETURNS:
The carry-bit will be *clear* if the data byte in register-A corresponds to an ASCII Hexadecimal character. ASCII Hexadecimal characters in the range: $61-$66 will be converted to upper case. Therefore, an input parameter of: $63 will be returned as: $43 in register-A.

If the parameter byte passed in register-A is not a valid ASCII Hexadecimal character, the **ISHEX** subroutine will return with the carry-bit *set* in the status register.

ERRORS:
No errors reported.

PUT_CHR ($00:E04B)

DESCRIPTION:
This subroutine will output a character from register-A to the serial console output port #3. Refer to **Appendix C - Mensch Monitor Assembly Listing** for specific details regarding the internal operation of the **PUT_CHR** subroutine.

EXPECTS:
Character to be output in 8-bit register-A.

RETURNS:
All registers are saved upon entry and restored before returning.

ERRORS:
No errors reported.

PUT_STR ($00:E04E)

DESCRIPTION:
This subroutine will output a string to the serial console output port #3. Refer to **Appendix C - Mensch Monitor Assembly Listing** for specific details regarding the internal operation of the **PUT_STR** subroutine.

EXPECTS:
Long pointer to the string as follows:

Bank address of string in 8-bit register-A.

Offset address of string in 16-bit register-X.

The string must be terminated with either (1) a null character, or (2) the most significant bit of the last character set.

The maximum string input size is limited to 640 characters.

RETURNS:
No arguments returned. All registers are saved upon entry and restored before returning.

ERRORS:
No errors reported.

READ_ALARM ($00:E051)

DESCRIPTION:
This subroutine will read the current system alarm setting as a null-terminated text string into a user-specified buffer. Refer to **Appendix C - Mensch Monitor Assembly Listing** for specific details regarding the internal operation of the **READ_ALARM** subroutine.

EXPECTS:
A pointer in 16-bit register-X to a nine (9) character buffer, located in memory bank #0.

RETURNS:
Null terminated time string in specified buffer.

The string format is: HH:MM:SS<*null*>
wherein: HH = Hours code (0-23 possible) in ASCII digits.

"00" = Midnight
"12" = Noon
"23" = 11 PM

MM = Minutes code (0-59 possible) in ASCII digits.

SS = Seconds code (0-59 possible) in ASCII digits.

ERRORS:
No meaningful errors.

READ_DATE ($00:E054)

DESCRIPTION:
This subroutine will read the current system date as a null-terminated text string into a user-specified buffer. Refer to **Appendix C - Mensch Monitor Assembly Listing** for specific details regarding the internal operation of the **READ_DATE** subroutine.

EXPECTS:
A pointer in 16-bit register-X to a nine (9) character buffer, located in memory bank #0.

RETURNS:
Null terminated date string in specified buffer.

The string format is: MM-DD-YY<*null*>
wherein: MM = Month code (1-12 possible) in ASCII digits.

"01" = January
"02" = February
"03" = March
"04" = April
"05" = May
"06" = June
"07" = July
"08" = August
"09" = September
"10" = October
"11" = November
"12" = December

DD = Day of month code (1-31 possible) in ASCII digits.

YY = Year code (last two digits: "94"=1994) in ASCII.

ERRORS:
No meaningful errors.

READ_TIME ($00:E057)

DESCRIPTION:
This subroutine will read the current system time as a null-terminated text string into a user-specified buffer. Refer to **Appendix C - Mensch Monitor Assembly Listing** for specific details regarding the internal operation of the **READ_TIME** subroutine.

EXPECTS:
A pointer in 16-bit register-X to a nine (9) character buffer, located in memory bank #0.

RETURNS:
Null terminated time string in specified buffer.

The string format is: HH:MM:SS<*null*>
wherein: HH = Hours code (0-23 possible) in ASCII digits.

"00" = Midnight
"12" = Noon
"23" = 11 PM

MM = Minutes code (0-59 possible) in ASCII digits.

SS = Seconds code (0-59 possible) in ASCII digits.

ERRORS:
No meaningful errors.

RESET ($00:E084)

DESCRIPTION:
This subroutine will invoke the master start-up vector in ROM, effectively resetting the entire W65C265 system. Refer to **Appendix C - Mensch Monitor Assembly Listing** for specific details regarding the processing and internal operations associated with the **RESET** library vector.

EXPECTS:
No input arguments.

RETURNS:
This vector does not return.

ERRORS:
No errors reported.

RESET_ALARM ($00:E05A)

DESCRIPTION:
This subroutine will reset all alarm flags to a "don't care" condition, effectively canceling any alarm setup or active alarm. Refer to **Appendix C - Mensch Monitor Assembly Listing** for specific details regarding the internal operation of the **RESET_ALARM** subroutine.

EXPECTS:
No input arguments.

RETURNS:
No arguments returned.

ERRORS:
No meaningful errors are detected.

SBREAK ($00:E05D)

DESCRIPTION:
This subroutine will call the "software break" routine in ROM. It will save and print the processor's registers, and transfer control to the ROM monitor's command processor. Refer to **Appendix C - Mensch Monitor Assembly Listing** for specific details regarding the processing and internal operations of the **SBREAK** subroutine.

EXPECTS:
No input arguments.

RETURNS:
This vector does not return.

ERRORS:
No errors reported.

SELECT_COMMON_BAUD_RATE ($00:E060)

DESCRIPTION:
This subroutine allows the program to reconfigure and enable the common baud rate generator (T4) which drives the serial ports reserved for the keyboard, printer, and PC link. Changing this baud rate will affect all three ports. If used incorrectly, it can disable communication with the serial console on port #3. Refer to **Appendix C - Mensch Monitor Assembly Listing** for specific details regarding the internal operation of the **SELECT_COMMON_BAUD_RATE** subroutine.

EXPECTS:
Baud rate selection code in 8-bit register-A:

0 =	110 Baud
1 =	150 Baud
2 =	300 Baud
3 =	600 Baud
4 =	1200 Baud
5 =	1800 Baud
6 =	2400 Baud
7 =	4800 Baud
8 =	9600 Baud
9 =	14400 Baud
A =	19200 Baud
B =	38400 Baud
C =	57600 Baud
D =	115000 Baud

RETURNS:
No arguments returned.

ERRORS:
Carry-bit (Clear=OK / Set=Unacceptable selection code.)

SEND_BYTE_TO_PC ($00:E063)

DESCRIPTION:
This subroutine will queue one byte from 8-bit register-A to be sent to the serial PC link port #3. The carry-bit will be set if the serial PC link port #3 cannot accept data, otherwise it will be clear upon return. Refer to **Appendix C - Mensch Monitor Assembly Listing** for specific details regarding the internal operation of the **SEND_BYTE_TO_PC** subroutine.

EXPECTS:
Output byte in 8-bit register-A.

RETURNS:
No arguments returned.
ERRORS:
Carry-bit (*Clear*=OK / *Set*=Serial PC link port cannot accept data.)

SEND_CR ($00:E066)

DESCRIPTION:
This subroutine will output a **CR** (Carriage-Return/Enter = $0D) character to the serial console output port #3. Refer to **Appendix C - Mensch Monitor Assembly Listing** for specific details regarding the internal operation of the **SEND_CR** subroutine.

EXPECTS:
No input arguments.

RETURNS:
The carry-bit will always be *clear* upon completion.

ERRORS:
No errors reported.

SEND_HEX_OUT ($00:E06C)

DESCRIPTION:
This subroutine will output an 8-bit value as two ASCII HEX digits to the serial console output port #3. Refer to **Appendix C - Mensch Monitor Assembly Listing** for specific details regarding the internal operation of the **SEND_HEX_OUT** subroutine.

EXPECTS:
Value to be output in 8-bit register-A.

RETURNS:
The carry-bit will always be *clear* upon completion.

ERRORS:
No errors reported.

SEND_SPACE ($00:E069)

DESCRIPTION:
This subroutine will output a **SPACE** ($20) character to serial console output port #3. Refer to **Appendix C - Mensch Monitor Assembly Listing** for specific details regarding the internal operation of the **SEND_SPACE** subroutine.

EXPECTS:
No input arguments.

RETURNS:
The carry-bit will always be *clear* upon completion.

ERRORS:
No errors reported.

SET_ALARM ($00:E06F)

DESCRIPTION:
This subroutine will set the system alarm time from a null-terminated text string in a user-specified buffer. Refer to **Appendix C - Mensch Monitor Assembly Listing** for specific details regarding the internal operation of the **SET_ALARM** subroutine.

EXPECTS:
A pointer in 16-bit register-X to a nine (9) character buffer, located in memory bank #0.

The string format is: HH:MM:SS<*null*>
wherein: HH = Hours code (0-23 possible) in ASCII digits.

"00" = Midnight
"12" = Noon
"23" = 11 PM

MM = Minutes code (0-59 possible) in ASCII digits.

SS = Seconds code (0-59 possible) in ASCII digits.

Note: This subroutine requires a fixed-size, fixed-format input string. Therefore, the <*null*> terminator character is acceptable, but not really necessary.

RETURNS:
No arguments returned.

ERRORS:
Carry-bit (Clear=OK / Set=Error detected)

RELATED:
The "User Alarm Wakeup Subroutine Vector" (UALRMIRQ=$00:0134) allows application programs to associate a subroutine with the alarm timeout condition. This vector should be initialized prior to setting the alarm. When the alarm times out, all essential overhead will be handled by the ROM monitor before transferring control to the vector. Refer to **Appendix C - Mensch Monitor Assembly Listing** for specific details regarding the operation of this feature.

SET_Breakpoint ($00:E072)

DESCRIPTION:
This is the subroutine invoked by typing the '**B**' command at the monitor prompt. The '**B**' monitor command allows the user to set a breakpoint at a specific location. **SET_Breakpoint** will request an address and accept input via the console port #3.

The user must enter six ASCII-Hex digits to form a 24-bit address. The input format is:
BB:AAAA

Wherein: "BB" is the bank address. This subroutine will echo a ':' after the 2-digit bank address. "AAAA" is the offset address within the bank.

After a valid address has been entered, **SET_Breakpoint** will store a **BRK** instruction ($00) at the target location. When program execution reaches that address, the **BRK** instruction will transfer control to the ROM monitor.

> NOTE: This subroutine would not normally be used by application software on the W65C265. It has been included in this vector table to support anticipated needs of the extended Mensch Computer Operating System in that specific configuration. Developers should consult **Appendix C - Mensch Monitor Assembly Listing** for specific details regarding internal operation of this subroutine in order to determine suitability for other applications and configurations.

EXPECTS:
No input arguments. This is an interactive subroutine which requests `parameters from the user as needed.

RETURNS:
The carry-bit will be *clear* if the operation was successfully performed.

ERRORS:
The carry-bit will be *set* if invalid address data was entered.

SET_DATE ($00:E075)

DESCRIPTION:
This subroutine will set the current system date from a null-terminated text string provided in a user-specified buffer. Refer to **Appendix C - Mensch Monitor Assembly Listing** for specific details regarding the internal operation of the **SET_DATE** subroutine.

EXPECTS:
A pointer in 16-bit register-X to a nine (9) character buffer, located in memory bank #0.

The string format must be: MM-DD-YY<*null*> or
MM/DD/YY<*null*>
wherein: MM = Month code (1-12 possible) in ASCII digits.

"01" = January
"02" = February
"03" = March
"04" = April
"05" = May
"06" = June
"07" = July
"08" = August
"09" = September
"10" = October
"11" = November
"12" = December

DD = Day of month code (1-31 possible) in ASCII digits.

YY = Year code (last two digits: "94"=1994) in ASCII.

Note: This subroutine requires a fixed-size, fixed-format input string. Therefore, the <*null*> terminator character is acceptable, but not really necessary.

RETURNS:
No arguments returned.

ERRORS:
Carry-bit (*Clear*=OK / *Set*=Error detected)

SET_TIME ($00:E078)

DESCRIPTION:
This subroutine will set the system time from a null-terminated text string in a user-specified buffer. Refer to **Appendix C - Mensch Monitor Assembly Listing** for specific details regarding the internal operation of the **SET_TIME** subroutine.

EXPECTS:
A pointer in 16-bit register-X to a nine (9) character buffer, located in memory bank #0.

The string format is: HH:MM:SS<*null*>
wherein: HH = Hours code (0-23 possible) in ASCII digits.

"00" = Midnight
"12" = Noon
"23" = 11 PM

MM = Minutes code (0-59 possible) in ASCII digits.

SS = Seconds code (0-59 possible) in ASCII digits.

Note: This subroutine requires a fixed-size, fixed-format input string. Therefore, the <*null*> terminator character is acceptable, but not really necessary.

RETURNS:
No arguments returned.

ERRORS:
Carry-bit (*Clear*=OK / *Set*=Error detected)

UPPER_CASE ($00:E0A3)

DESCRIPTION:
This subroutine will convert a lower-case ASCII (a-z) character into an upper-case ASCII (A--Z) character. The character to be converted must be passed in register-A. The converted result will be returned in register-A. If the byte is not a lower case ASCII character, it will return unchanged. Refer to **Appendix C - Mensch Monitor Assembly Listing** for specific details regarding the internal operation of the **UPPER_CASE** subroutine.

Example:

```
        LDA #'b'
        JSL UPPER_CASE
        CMP #'B'
        BNE FAILED    ;Should    never
branch
        ...
```

EXPECTS:
Character to be converted in register-A.

RETURNS:
Converted upper-case character is returned in register-A. If the byte was not a lower case ASCII character, it will be returned unchanged.

ERRORS:
No errors reported.

VERSION ($00:E07B)

DESCRIPTION:
This subroutine will return info on the current ROM version. Refer to **Appendix C - Mensch Monitor Assembly Listing** for specific details regarding the internal operation of the **VERSION** subroutine.

EXPECTS:
No input arguments.

RETURNS:
ROM version information:

Register-X = Pointer to 4-character string representing the version.
(Example: "2.01")

Register-Y = Pointer to formatted ASCII string representing the last assembly date. (Example: "SAT DEC 3 12:16:05 1994")

Register-A = 0 (No particular significance.)

ERRORS:
No errors reported.

WR_3_ADDRESS ($00:E07E)

DESCRIPTION:
This subroutine will write a 3-byte address to the serial console output port #3. The 3-byte address to be sent must be loaded into global variable: **TMP0**($00:005D). Refer to **Appendix C - Mensch Monitor Assembly Listing** for specific details regarding the internal operation of the **WR_3_ADDRESS** subroutine.

EXPECTS:
No input arguments in registers. The 3-byte address to be sent must be loaded into global variable: **TMP0**($00:005D), **TMP0+1** and **TMP0+2**. The least significant byte (LSB) of the 3-byte address must reside in **TMP0** and the MSB must be in **TMP0+2**.

RETURNS:
No arguments returned.

ERRORS:
No errors detected or reported.

XS28IN ($00:E081)

DESCRIPTION:
This subroutine will read S28 records from the serial console input port #3, and place them into memory. It will wait (i.e. *not return*) until input is provided.

An **ESC** (Escape) character from serial console input port #3 will cancel the load operation and cause the subroutine to return with the carry-bit=*set*. Other characters from multiple inputs

may cause load errors. This subroutine will begin computing a checksum when the start of an S28 record is detected.

Each time a record is processed, this subroutine outputs a period ('.') and 4-digit record number to the serial console output port #3. A '?' is returned if the checksum does not agree. This is for user feedback only. After receiving the final record a carry-bit=*clear* will be returned if no errors had been encountered. Likewise, if errors occurred, a carry-bit=*set* will be returned.

This operation cycle will continue until an error occurs or the "S8" end record is received. Refer to **Appendix C - Mensch Monitor Assembly Listing** for specific details regarding the internal operation of the **XS28IN** subroutine.

EXPECTS:
No input arguments. The initial load address for each block of data is the first part of each S28 record.

RETURNS:
The carry-bit normally will return *clear* if the load operation completes without incident. No register arguments are returned, and no registers have been saved.
ERRORS:
The carry-bit will be *set* if **ESC** (Escape) was detected or if checksum errors occurred. Each S28 record is loaded into memory as it is processed. A checksum can only reflect the integrity of the data. When a checksum error is detected, it means that the memory has already been loaded with contaminated data. This subroutine is used by the 'S' command. Refer to that description for comments regarding error detection.

WARNING:
The address fields of the S28 records are not filtered in any way. ALL POSSIBLE ADDRESSES ARE ACCEPTABLE! The user is responsible for assuring that records do not overwrite critical locations which may disrupt proper operation of the monitor.

Appendices

Appendix A – Mensch Monitor Commands

Mensch Monitor Commands

Time-of-Day Clock Commands

T, t Reads current time of day clock, displays the results, and accepts a new time value

N, n Reads the current date, displays the date, and accepts a new date value.

Memory Commands

D, d Displays memory from low address to high address as ASCII Hex in labelled columns.

> Increment the current address and display the contents. (The M and D commands set the current address pointer.)

< Decrement the current address and display the contents.

SPACE Uses the current address pointer and displays the contents.

/ Quick access to memory.
/<**ENTER**> --- Examine current pointer.
/<**SPACE**> --- Examine current location.
/<Data><**SPACE**> --- Change current contents.
/<Address><**SPACE**> --- Examine memory.
/<Address><Data> --- Change address pointer & memory.

M, m Alter Memory address and locations. The first 6 characters entered after the 'M' set the current address pointer. Each pair of characters entered after the address changes the byte at the current address. A space entered after a byte change increments the current address pointer. A CR will end the command, and can be entered after the initial address or any number of byte changes.

F, f Fill memory from start address to end address with value.

Register Commands

R, r Display processor registers.
(PCntr, Acc, Xreg, Yreg, Stack, DirReg, F, DBk)

A, a Alter registers.
(PCntr, Acc, Xreg, Yreg, Stack, DirReg, F, DBk)
A space skips to the next register. A carriage-return ends the command.

¦ Quick access to registers.
¦ <**ENTER**> --- Examine all.
¦ <Reg><Data> --- Change register.

Execution Commands

G, g Begin execution from specified address.

J, j Execute a "JSL" instruction to the specified address.

I/O Commands

S, s Start of a data record in Motorola S28 format. When this command is received, data is not echoed until an "S8" record is received. This command is used to load programs and data to specific locations.

W, w Output data in Motorola S28 format. This command outputs 16 byte records (the last record may be less) from lowest address to highest address.

Miscellaneous Commands

U, u JSR to user command processor, via the UCMDPTR vector: $00:012C. This is a hook to allow additional commands to be added to the monitor by the user.

? of H, h HELP – Lists available commands.

X, x Switches the system to low power mode.

Appendix B – Mensch Monitor Subroutine

'E000_265 .ASM – Entry points to the monitor ROM'

Alter _Memory	EQU	$00:E000	; Support Subroutine For 'M' Command.
BACKSPACE	EQU	$00:E003	; Outputs a <**BACKSPACE**> ($08) Character.
Reserved	EQU	$00:E006	;
CONTROL_TONES	EQU	$00:E009	; Configures tone generators: TG0 & TG1.
DO_LOW_POWER_PGM	EQU	$00:E00C	; Vector to force *LOW POWER MODE* on system.
DUMPREGS	EQU	$00:E00F	; Support Subroutine For 'R' Command.
Dump528	EQU	$00:E012	; Support Subroutine For 'R' Command
Dump_1_line_to_Output	EQU	$00:E015	; Requests/Accepts *Starting Address* & dumps 8 bytes.
Dump_1_line_to_Screen	EQU	$00:E018	; Requests/Accepts *Starting Address* & dumps 8 bytes.
Dump_to_Output	EQU	$00:E01B	; Dumps specified range of memory data to port #3.
Dump_to_Printer	EQU	$00:E01E	; Dumps specified range of memory data to port #3.
Dump_to_Screen	EQU	$00:E021	; Dumps specified range of memory data to port #3.
Dump_to_Screen_ASCII	EQU	$00:E024	; Dumps memory block in ASCII format.
Dump_It	EQU	$00:E027	; Custom dump support.
FILL_Memory	EQU	$00:E02A	; Support Subroutine For 'F' Command.
GET_3BYTE_ADDR	EQU	$00:E02D	; Accepts 3-byte address as six ASCII Hex digits.
GET_ALARM_STATUS	EQU	$00:E030	; Retrieves & resets current system alarm status.
GET_BYTE_FROM_PC	EQU	$00:E033	; Reads next available input from serial port #3.
GET_CHR	EQU	$00:E036	; Accepts on character from serial port #3.
GET_HEX	EQU	$00:E039	; Accepts two ASCII Hex digits via serial port #3.
GET_PUT_CHR	EQU	$00:E03C	; Accepts & echoes one character via serial port #3.
GET_STR	EQU	$00:E03F	; Uses **GET_PUT_CHR** to build string buffer.
Get_Address	EQU	$00:E042	; Requests/Accepts 3-byte address as ASCII Hex digits.
Get_E_Address	EQU	$00:E045	; Requests/Accepts *HIGHEST ADDRESS* via serial port #3.
Get_S_Address	EQU	$00:E048	; Requests/Accepts *LOWEST ADDRESS* via serial port #3.
PUT_CHR	EQU	$00:E04B	; Output one character to serial console port #3.
PUT_STR	EQU	$00:E04E	; Output specified string of characters to serial port #3.
READ_ALARM	EQU	$00:E051	; Reads current system *Alarm* setting.
READ_DATE	EQU	$00:E054	; Reads current system *Date* setting.
READ_TIME	EQU	$00:E057	; Reads current system *Time* setting.
RESET_ALARM	EQU	$00:E05A	; Resets the system *Alarm* function.
SBREAK	EQU	$00:E05D	; Invokes the software breakpoint logic.
SELECT_COMMON_BAUD_RATE	EQU	$00:E060	; Configures baud rate generator for serial port #3.
SEND_BYTE_TO_PC	EQU	$00:E063	; Outputs byte to serial port #3.
SEND_CR	EQU	$00:E066	; Outputs ASCII **ENTER** ($0D) character.
SEND_SPACE	EQU	$00:E069	; Outputs ASCII **SPACE** ($20) character.
SEND_HEX_OUT	EQU	$00:E06C	; Outputs byte as two ASCII Hex characters.
SET_ALARM	EQU	$00:E06F	; Set the System Alarm time from specified string.
SET_Breakpoint	EQU	$00:E072	; Sets a breakpoint (**BRK**) instruction at address.
SET_DATE	EQU	$00:E075	; Sets the System Time-Of-Day Clock: Date
SET_TIME	EQU	$00:E078	; Sets the System Time-Of-Day Clock: Time
VERSION	EQU	$00:E07B	; Gets firmware (W65C265 ROM) version info.
WR_3_ADDRESS	EQU	$00:E07E	; Outputs a 3-byte address as ASCII Hex characters
XS28IN	EQU	$00:E081	; Accepts & loads Motorola type "S28" records.
RESET	EQU	$00:E084	; Invokes the Master Start-Up vector to Reset system.
ASCBIN	EQU	$00:E087	; Converts two ASCII Hex characters to binary byte.
BIN2DEC	EQU	$00:E08B	; Converts binary byte to packed BCD digits.
BINASC	EQU	$00:E08F	; Converts binary byte to ASCII Hexadecimal characters.
HEXIN	EQU	$00:E093	; Converts ASCII Hexadecimal character to Binary byte.
IFASC	EQU	$00:E097	; Checks for displayable ASCII character.
ISDECIMAL	EQU	$00:E09B	; Checks character for ASCII Decimal Digit.
ISHEX	EQU	$00:E09F	; Checks character for ASCII Hexadecimal Digit
UPPER_CASE	EQU	$00:E0A3	; Converts lower-case ASCII alpha chars to upper-case.
Reserved	EQU	$00:E0A7	
	.End		

‘Internal RAM Locations Used By Library Subroutines For Data Transfer’

```
; Working Variables

TMP0        EQU     $00:005D    ; Used by:  Dump_It
                                            WR_3_Address

TMP2        EQU     $00:0063    ; Used by:  Dump_It
                                            GET_3BYTE_ADDRESS
                                            Get_Address
                                            Get_E_Address
                                            Get_S_Address

TEMP        EQU     $00:0070    ; Used by:  ASCBIN

; TEMP+1     EQU     $00:0071    ; Used by:  BINASC

; Vectors
UBRK        EQU     $00:0100    ; USER -- BREAK VECTOR
UNMI        EQU     $00:0104    ; USER -- NMI VECTOR
UNIRQ       EQU     $00:0108    ; USER -- IRQ VECTOR
COPIRQ      EQU     $00:010C    ; USER -- CO-PROCESSOR IRQ
IABORT      EQU     $00:0110    ; USER -- ABORT POINTER
PIBIRQ      EQU     $00:0114    ; PERIPHERAL INETERFACE IRQ
EDGEIRQS    EQU     $00:0118    ; ALL EDGE IRQS
UNIRQT7     EQU     $00:011C    ; USER -- TIMER 7 IRQ
UNIRQT2     EQU     $00:0120    ; USER -- TIMER 2 IRQ
UNIRQT1     EQU     $00:0124    ; USER -- TIMER 1 IRQ
UNIRQT0     EQU     $00:0128    ; USER -- TIMER 0 IRQ
USER_CMD    EQU     $00:012C    ; USER -- COMMAND PROCESSOR
URESTART    EQU     $00:0130    ; USER -- POWER UP RESTART
VECTOR
UALRMIRQ    EQU     $00:0134    ; USER -- ALARM WAKEUPCALL

            .END
```