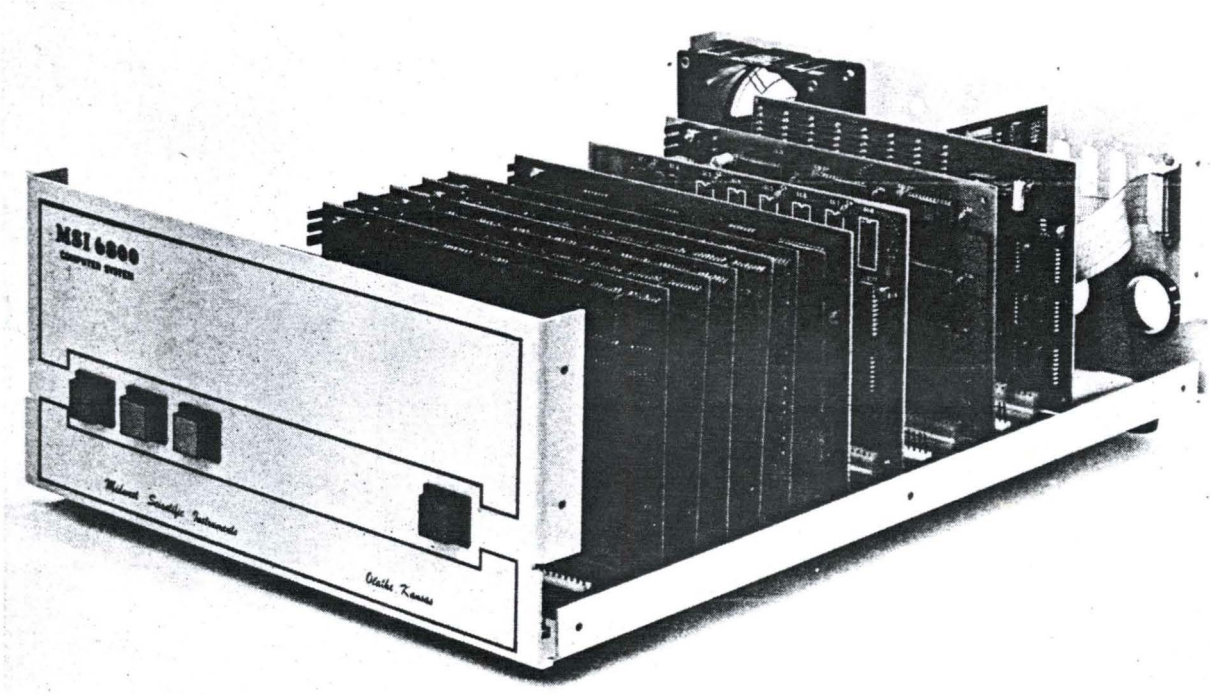


THE MSI 6800 COMPUTER SYSTEM

Midwest Scientific Instruments, Inc.
Olathe, Kansas

TABLE OF CONTENTS

Page No.

UNPACKING & PRELIMINARY CHECK-OUT INSTRUCTIONS

for the

MSI 6800 Computer System & FD-8 Disk Memory System

The following is a list of preliminary check-out instructions which will allow you to get wired and tested MSI 6800 Computer Systems and FD-8 Disk Memory Systems into operation quickly. Refer to the operating manual for the detailed descriptions of the systems.

(1) First of all remove the cover from the computer and remove the foam packing material which is on top of the circuit boards. The serial interface which communicates with the control terminal is positioned in Port 0, that is on the left-hand most I/O position in the rear of the computer. The PIA interface card for the disk memory plugs into I/O Port 7 which is the right-hand most I/O position as viewed from the front of the computer.

(2) If the FD-8 Disk Memory System is used, remove the covers from the disk drives and remove the strapping tape, the cardboard packing blocks, and foam sponge material which is used to secure the drives during shipment. On dual drive systems, the parent drive (drive 0) communicates with drive 1 by means of a wide ribbon cable having 50 conductors. This cable exits from the rear of drive 0, passes thru the opening in the rear of drive 1, and plugs into the edge connector on the back of the disk drive itself. Be sure that the blue stripe on the edge of the ribbon cable enters drive 1 on the same side that it exited from drive 0. That is, if the stripe was on the left side as it exited drive 0 make sure that it is also on the left side as it enters drive 1. The disk drives are connected in parallel. Following the steps above will insure that the cable is plugged in with proper orientation.

(3) The power transformer in the computer has a split primary which is capable of being hooked up in either 110 vac or 220 vac configuration. The primary is also tapped so as to permit proper bus voltages to be obtained under conditions of heavy load in the computer or low AC line voltage. On equipment destined for Europe and other foreign locations using 220 vac 50 cycle current, the units are always shipped from our factory properly configured for 220 volt operation even though the standard American line plugs are installed. These line cord connectors may have to be changed in order to adapt to standard European 220 volt AC plugs.

(4) Disk drive 0 communicates with the PIA-1 interface card which is plugged into the I/O Port 7 of the computer. Pass the ribbon cable thru one of the large holes in the rear of the chassis and plug into the 16 Pin sockets on top of the interface card. When plugged in properly, the cable should exit away from the top of the interface card.

(5) Please note that all strappable options on the serial interface card for the control terminal are available on the DB-25

connector as well as on the interface card itself. You may examine the schematic diagram of the interface card in order to determine the correct strapping options. The card is generally provided strapped for either a 1200 baud CRT terminal or 110 baud current loop configuration. Connections to industry standard CRT terminals should be made as shown in the table below:

Computer Connector	Terminal DB-25 Connector	Signal Identification
2	3	Data from Computer to CRT
3	2	Data from CRT to Computer
7	7	Ground (LOGIC Common)

(6) After all equipment has been unpacked and properly interconnected, power may be applied to the system. An asterisk should appear on the CRT terminal which is the PROMPT for the MSIBUG Monitor. Pressing the "reset" button on the front panel of the computer should cause the asterisk to appear again each time "reset" is pressed. The MSIBUG Monitor accepts only uppercase commands from the keyboard of the terminal. Typing an "R" on the CRT keyboard should cause a register dump to the screen showing contents of various CPU registers and is a good test of proper communication between the computer and the terminal.

(7) At this point a diskette which is labeled SDSK-B or SDSK-CB should be inserted into the drive 0 with the label facing upward. Insert the disk all the way until a click is heard which will allow the disk drive door to then be closed until it locks securely. On the keyboard type GEC00 in order to bootstrap the disk operating system. The system should respond by saying DOS ready and will then obey various commands which are issued from keyboard such as "Files". Typing Files 1 will cause the disk files from drive No. 1 to be printed. The system can then load the BASIC interpreter by typing BASIC. After system responds with "MSI READY" a basic program can be entered and saved. Refer to the Disk Manual at this point for more complete discussion of the Disk Operating System Commands and Software.

(8) The system diskette should be regarded as a "Master Copy" and should not be written on. Do not place a "write enable tab" on this diskette under any circumstances but keep it in a protected place and use it only to generate working copies using the "copy" routine. Please note that before a new blank diskette is used it should be placed in one of the drives and initialized as described in the manual. This formats each sector of the diskette properly before it is used to generate a new copy or to store files on it.

DESCRIPTION OF MONITOR COMMANDS:

The MSIBUG Monitor is compatible with the Motorola Mikbug Monitor. The memory examine and change function is similar, type M0000 to examine the memory content of memory location 0000. Typing a slash (/) advances the locations while typing a period advances in the

reverse direction. Deposit a new instruction into a given memory location by first typing space followed by the new instruction. Typing two spaces or a carriage return returns the monitor software to the asterisk. Typing an "N" while in the memory examine and change function allows a new address to be entered.

Both punch and load routines can be carried out thru either Port 0 or Port 1, the Port number must enter following the "P" command which is then followed by the beginning and ending addresses of the memory area to be punched. An optional echo on the control terminal is also available. The Port number must also be entered following an "L" command to specify either Port 0 or Port 1.

Typing a "T" followed by beginning and ending memory addresses allows a desired area of memory to be listed on the terminal in instruction format.

A Control "S" is used to stop the output character routine, typing any other character allows it to resume once again. Typing a Control "D" returns the software to the monitor. Control "E" turns off the echo.

A checksum routine is available, type "C" followed by the beginning and ending addresses of the memory block on which the checksum is to be calculated.

The 'go to user program function" is carried out by typing "G XXXX" which specifies the beginning address at which program execution is to begin.

THE MSI 6800 COMPUTER SYSTEM

GENERAL DESCRIPTION

The MSI 6800 Computer is a high quality and well engineered microcomputer system, based upon the Motorola MC-6800 Microprocessor Chip. The outstanding engineering features, of the MSI 6800 Computer System, make it suitable for use in commercial or industrial applications where high reliability is necessary.

The MSI 6800 System overcomes all of the engineering disadvantages of early 6800 based microcomputer systems, which centered around the Motorola monitor routines. In the MSI system, we have introduced the new MSI-BUG Monitor which offers many improvements and additions to earlier monitor programs. The interface addresses have been located at the top of memory so as to permit a full 56K of user RAM memory if desired. Serial ACIA interfaces have also been used, which eliminate the disadvantages of the modified PIA interface.

The Power Supply Section of the MSI 6800 Computer is designed to deliver a 18 amps minimum at +8 V.D.C. on the unregulated bus, which permits full memory, as well as EPROM boards and other accessories, to be used in the system without any concern for power supply capacity. The power transformer has a split primary, which permits a series operation from 230 V.A.C. power, as well as parallel operation from 115 V.A.C. power. In addition, each of the primary windings is tapped, to deal with low line voltage conditions, or heavy DC loads in the computer, which keep the unregulated voltages on the bus from dropping below acceptable limits.

The CPU Board in the MSI system has many outstanding features. First of all, it contains space for 4K of EPROM memory, 1K of which is occupied by the MSI-BUG Monitor routines. The additional 3K of memory may be occupied by user specified 2708 EPROM software. This makes it very convenient for the CPU Board to be used in OEM or industrial control applications where a stand-alone card, containing both RAM as well as PROM, may be desirable. The CPU Board contains 128 bytes of RAM memory, which is used for the MSI-BUG routines, and is also available for use by the user. A number of strappable options are provided on the CPU Board to allow the user to select DMA refresh/grant, slow memory lines, clocks, any desired baud rate, and others. A complete description of strappable options is provided later in this manual. The CPU Board uses a 6875 clock driver, which provides the two-phase non-overlaping clock for the CPU, separate from the baud rate generator clock, which then permits the user to run the system at 2 MHz if desired. A separate 14411 baud rate generator provides all of the

standard baud rate clocks which are available to the mother board.

A separate restart vector PROM is used, providing four different sets of restart vectors, which may be selected easily with strapping options. One set of vectors goes to the MSI-BUG Monitor routines on RESET, another to MSI PROM DOS, another to an FD-8 DISK BOOTSTRAP routine, and the last is completely specified by the user. This feature allows one to configure his system easily to an exact and specific circumstance or application.

The Serial Interface Board, which is used with the MSI 6800 Computer System, contains all the baud rate selections, and clock option straps on the DB-25 connector in order to facilitate quick changing of terminals. Primary and Secondary RS-232 outputs, teletype current loop inputs/outputs, reader control, as well as REQUEST TO SEND and CLEAR TO SEND functions are available. A second Serial Interface Card, using the MC-6852 Synchronous Interface Adapter Chip, will soon be available. This card will permit synchronous data to be transmitted serially at high data rates. Both the Synchronous and the Asynchronous Serial Interfaces are designed to work with data sets, modems, or acoustic couplers, using standard RS-232C format.

This manual includes separate sections for each of the circuit boards which make up the MSI 6800 Computer System. Each section contains a complete schematic diagram, parts list, an assembly drawing showing the lay out and orientation of all components, as well as assembly instructions and theory of operation. Selection of strappable options for each board is also included.

We have tried to make this manual as complete as possible, in order to make system assembly and use convenient for you. We are always interrested in receiving information from our users concerning ways in which we might improve our products, or make them easier for the user to handle. We always appreciate hearing from you concerning your own applications and your own needs for additional products. We believe that you have purchased the finest 6800 based computer system available today, and MSI stands behind it all the way. We wish you good luck with your system and hope that you are satisfied. Please don't hesitate to let us know if there is anything more we can do to serve you.

MIDWEST SCIENTIFIC INSTRUMENTS, INC.

MSI 6800 MEMORY UTILIZATION

The MSI 6800 Computer System is capable of directly addressing 65536 bytes of memory. With the new MSI 16K memory boards, a bank select feature is available, which permits more than 65K of memory to be utilized, only one bank of which may be active at any one time. The bank select is available under software control.

The table included shows the areas of memory which have been utilized by MSI for various functions. In the MSI System, the RESTART and INTERRUPT vectors reside at the very top of memory, in locations $FFF8 thru $FFFF.

A block of addresses from $F400 to $F7FF have been reserved for input/output interfaces and other specialized memory control or multi-user functions. The CPU Board, in the MSI System, normally expects addresses above $E000 to reside on the CPU Board, and everything below $E000 somewhere else on the mother board bus. However, the address block $F400 to $F7FF is an exception and the CPU board recognizes this block of addresses as residing on the interface bus which is of course off the CPU Board. The user may change the CPU strapping if it becomes necessary to alter the on-board/off-board addressing scheme.

The MSI-BUG Monitor RAM area is located from $F000 to $F07F. This RAM area can also be strapped to a different location by the user for specialized applications.

The 4K EPROM memory, which normally resides from $E000 to $EFFF, may be strapped to other high order addresses if desired. Normally, the MSI-BUG Monitor software routines reside from $E000 to $E3FF. The MSI Extended Monitor, containing memory check and other useful programs, resides in PROM residing from $E400 thru $E7FF. The PROM area from $E800 to $EBFF has not been used by MSI, at this time, and is recommended as a suitable location for user PROMS. The top EPROM location, from $EC00 thru $EFFF has been reserved for the MSI PROM DISK DRIVER routines and DISK BOOTSTRAP.

Memory locations between $D000 and $DFFF are used by the MSI Interpretive Debugger Routines, which can be placed on EPROMS if desired. Also, the MSI MULTI-USER SYSTEM will use memory within this block.

Memory addresses $C000 thru $CFFF have been reserved for EPROMS as well, and are used by MSI MULTI-USER OPERATING SYSTEM. One set of the selectable RESTART vector routines directs the system to the area of $C000 thru $C00B, in order that the user might place his own jump routines on PROMS in order to bring the system up in any desired operating system or software.

All the memory below $C000 is available for user RAM

area. Any of the MSI RAM-68 memory boards may be strapped, using the switch selectable address block, in order to operate in this region. The RAM-68 RAM board also operates at higher addresses if desired by simply selecting the appropriate beginning address on the board.

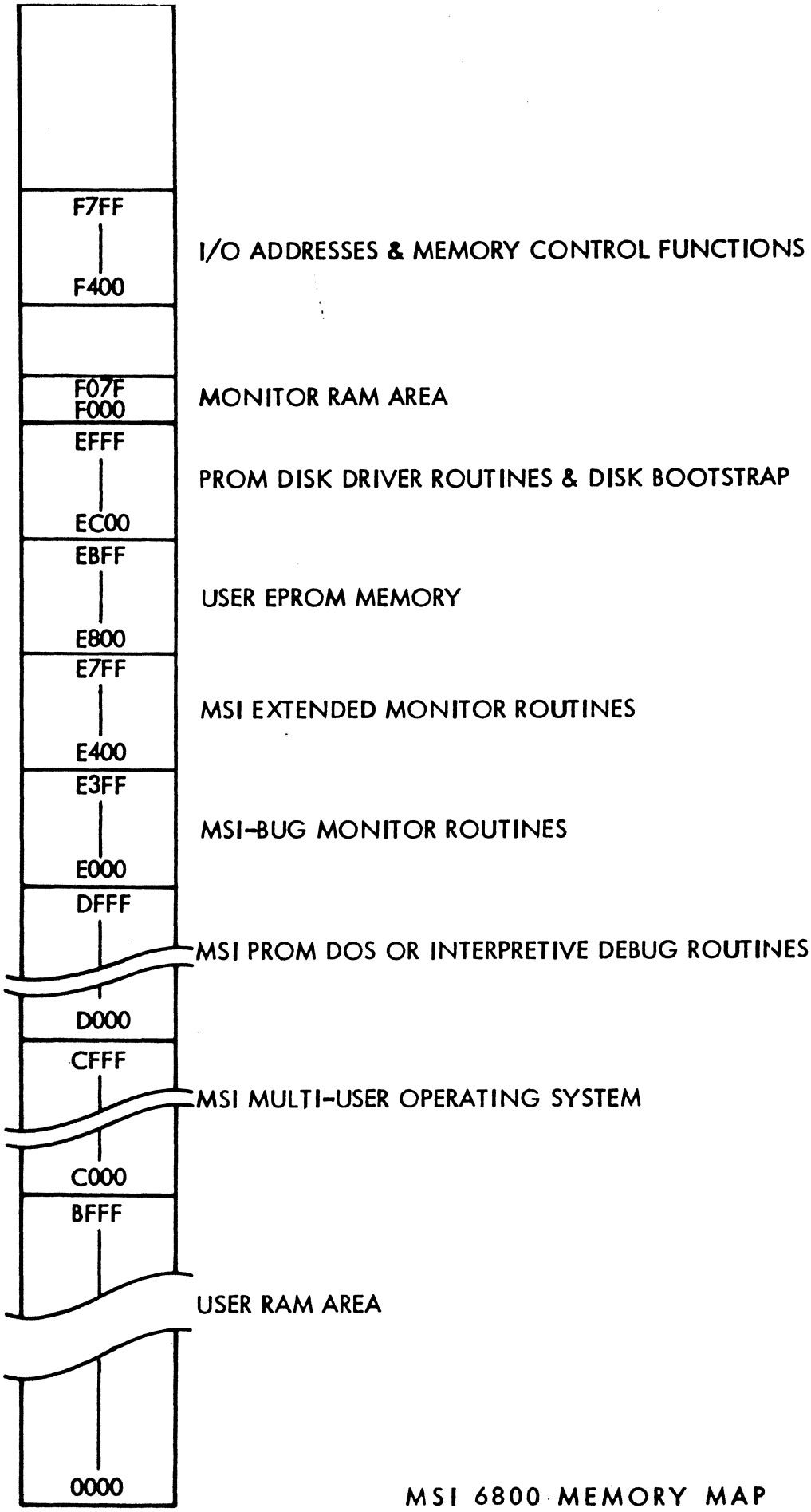
MSI WARRANTY POLICY

MSI warrants all equipment and materials to be free from defective workmanship and material for a period of 90 days beyond the date of purchase from either MSI directly or an authorized MSI dealer. Activation of product warranty must be by the return of the warranty registration card.

During the warranty period, any products purchased as wired and tested units will be repaired or replaced, at MSI's sole option, free of charge, when shipped to MSI prepaid, accompanied by a complete written description of the defect and a return authorization number. MSI accepts no responsibility for equipment returned freight collect, without a return authorization number, or without a written description of the defect.

During the warranty period, any products purchased in kit form will be repaired or replaced, at MSI's sole option, free of any charge for parts. However, labor charges for the repair will be assessed on a time required basis. In no case, will repair charges exceeding $100.00 be made without prior notification of the customer.

Upon completion of repairs, the unit will be returned to the customer freight collect.



MSI 6800 MEMORY MAP

GENERAL TROUBLE SHOOTING AND CHECKOUT PROCEDURES

MSI 6800 COMPUTER SYSTEM

INTRODUCTION

The biggest asset for trouble shooting a microcomputer system is to have a working system, into which individual circuit boards may be substituted for checkout. Of course, this is not always possible even though it is the most desirable approach, especially when checking out a new system which may have more than one problem. The purpose of this section is to provide a logical trouble shooting procedure and some suggestions which we have found to be helpful in our own debugging experience.

TEST EQUIPMENT NEEDED

The MSI Extender Board, Model EXT-1, is recommended to place the board under test into a convienient position during checkout. A high quality D.C. oscilloscope, having dual trace and triggered sweep, is almost essential. In addition, a volt-ohm-milliammeter (VOM) or digital voltmeter/ohmmeter is a must. A desoldering station is very valuable when I.C. packages must be removed from a PCB. The usual hand tools and soldering iron used for construction may be necessary.

TROUBLE SHOOTING PROCEDURE

For the procedures described below, it is assumed that the system is inoperative and does not respond with the asterisk on powerup or reset. If the system does respond correctly with the asterisk, but does not execute system software correctly, then proceed to the memory checkout section of this manual.

( ) First check the fuses on the power supply PCB, as well as the AC fuse on the rear panel of the computer, to ascertain that they are good. It is wise to check fuses with an ohmmeter since occasionally a bad fuse escapes visual detection.

( ) With power applied to the system, check the mother board power busses for correct voltages with respect to mother board ground (logic common). Chassis ground should not be connected to mother board ground. These voltages should also be examined with an oscilloscope in order to be sure that they are free of ripple and any oscillation. These measurements should be made again with system circuit boards installed in order to be sure that the voltages are correct under actual load conditions.

( ) Check the output of the voltage regulators on each of the circuit boards to be sure that the outputs are at the correct levels and are free from any ripple or oscillation.

( ) Using the oscilloscope, check each IC package, on every circuit board, to be sure that Vcc (+5V usually) and ground are actually reaching each chip as expected.

( ) Check the strappable options on each circuit board to be sure that the proper options have been selected as desired.

( ) Recheck all of the circuit boards for obvious defects or faults which may have been inadvertantly created during assembly. Solder bridges between close pads and runs are the most common fault even among experienced assemblers. On the CPU board, the EPROM sockets are the most vulnerable area due to the high density of pads and runs on this portion of the board.

( ) Using an ohmmeter, check between any two pins of the mother board bus to be sure that no shorts exist. Also check between any two pins of the interface bus, on the Interface Adapter board, for any shorts.

( ) On the CPU board, check for shorts between any two pins of the CPU chip itself, as well as any two pins of any given EPROM socket.

If the above procedures fail to reveal the problem, then further examination of signals, using the oscilloscope, will be necessary.

( ) First check the CPU board to be sure that the system clock and the baud rate generator clock are oscillating as expected. Also check phase 1 and phase 2 clock signals on the CPU chip, as well as on the mother board bus, to be sure that they are present. Be sure that the clocks are oscillating at the correct frequencies, rather than on a harmonic, and that the crystals have not been accidentally reversed.

( ) Next examine the R/W and VMA signals on the mother board to see that they toggle, as expected, immediately following a system reset.

( ) Examine the HALT signal on the mother board to be sure that it is high (not in the HALT state).

( ) Examine the baud rate clock lines on the mother board, as well as on the Interface Adapter Board, to be sure that they are reaching the Serial Interface Board as expected.

( ) Examine the Serial Interface Board to be sure that the correct baud rate selection straps have been installed and that other desired options have been selected as desired.

Be sure the frequency is correct for each of the clock signals (16X the bit rate).

( ) Be certain that the control terminal is set to the correct baud rate and is functioning properly. Connect pins 2 & 3 of the RS-232 interface together on the terminal. With the terminal in full duplex, characters should echo on the screen as they are typed. This procedure verifies that the terminal is transmitting and receiving properly.

( ) Refer to the RS-232 or Current Loop connection diagrams and be sure that the terminal is connected properly.

( ) Using the oscilloscope, examine the transmitted data line, or the Tx data output pin on the 6850 ACIA, immediately following a reset function to see if the asterisk character is being transmitted.

( ) While holding the RESET button in, check address lines to see if address $FFFE is present. Upon release, address line A0 should also go high then all address lines change as the RESTART vector address is applied. By triggering the scope on address line A0, the data word $D0 should be seen on the data lines while RESET is depressed, followed by $E0 after release of the RESET button on the positive transition of address line A0. IC 6 should be enabled (pin 15 low) while the RESET button is depressed. Only one PROM should be enabled during this time.

( ) Using the oscilloscope, examine the I/O select line of I/O PORT #1 immediately following a reset function. This line should toggle continuously as the CPU transmits the asterisk prompt character and then waits in a continuous loop for an input character command.

( ) Examine all of the bus signals, expecially the address and data lines, to see that they toggle correctly between +5 V. and ground and are not somewhere in between. An in between condition could indicate a short to another output line, the lack of a proper ground condition on an IC package, or a defective IC chip. A short is the most likely.

( ) If a short is suspected, then look for another line having the identical pattern. If a defective IC is suspected, then look at the inputs for that chip to see if they look normal.

( ) On the CPU board, check to see if the data bus driver packages get enabled.

( ) Check the processor RESET line and the bus RESET line to be sure that they are high and only go low during a system reset function.

Problems on the CPU board are the most difficult to

find and correct. The most common problem here is the receipt of incorrect data by the CPU chip as a result of an addressing problem on the PROM chips. If the CPU reads defective data, then its behavior is completely unpredictable.

MEMORY CHECKOUT PROCEDURES

INTRODUCTION

The execution of the following memory diagnostic program is essential following assembly of any memory board. Don't be misled by the various memory diagnostic programs, which are in circulation for 6800 systems, since most of these programs are very inadequate and fail to reveal memory problems in many cases. The program presented herein is the best that we have seen and will detect almost any memory problem which we have encountered thus far. The code can be relocated easily and can be used on EPROM which is highly recommended for quick availability. This program is a part of the MSI Extended Monitor EPROM.

To execute the program, place the beginning address of the memory area to be tested in memory locations $F002-$F003 (Monitor RAM area). The ending memory location+1 is placed in memory locations $F004-$F005. Then execute the program at its beginning address. Be careful not to test the memory area which contains the MEMORY TEST program itself, or it will be wiped out.

If memory location $F020 contains a $00, then the program will print a "@" after each 256 passes through memory. If $F020 is not $00, then a "+" will be printed following each pass through memory. This is the most desireable for a quick check.

When memory defects are detected, the bad address, expected data, and actual data read back are printed on the terminal.

TYPES OF MEMORY PROBLEMS

If a single bit remains set, or fails to set, within a 1 K segment of memory, then a single defective 2102 memory chip almost certainly is at fault. Refer to the schematics on the RAM-68 memory board in order to locate the bad chip.

If a single bit remains set, or fails to set, throughout all addresses on the memory board, then look for a more general problem with that particular data line or data bus driver package.

If the memory fails in a repetitive pattern through the board then look for a memory addressing problem, such as a shorted address line or a defective address buffer package.

Memory failures, not caused by bad memory chips, usually manifest themselves as one or more bits responding to multiple addresses. Locating such faults is relatively simple if the memory test first detects a failing location.

First zero memory as shown in example 1. Then examine the defective location and verify that the contents are actually zero. Write an $FF into the defective location(s) then check to see what other locations were simultaneously altered.

Consider the following possibilities:

1. Data going high changes the address by raising an address line high that should have remained low (data line shorted to address line).

2. Two address lines shorted. All bits in the address less that those shorted will have common data. Cause the processor to execute a $9D instruction and observe the address lines on the 2102 chips to see if they toggle in the correct relationship to each other.

3. Data bits alternately high and low usually indicate that the data line is actually displaying one bit of the address (data line and address line shorted together).

```
00610                    *
00620                    * MIDWEST SCIENTIFIC INSTRUMENTS
00630                    *
00640                          OPT    O
00650        F002      FIRST   EQU    $F002
00660        F004      LAST    EQU    $F004
00670        F022      SEED    EQU    $F022
00680        F024      COUNT   EQU    $F024
00690        F026      GROUP   EQU    $F026
00700        F028      SAVEX   EQU    $F028
00710 2000                     ORG    $2000
00720 2000 7F F022     TOP     CLR    SEED
00730 2003 CE 0000             LDX    #0000
00740 2006 FF F026             STX    GROUP
00750 2009 8D 02       TRICKY  BSR    START    *THIS ALLOWS BRA MORE THAN 12
00760 200B 20 FC               BRA    TRICKY   *BECAUSE THE ADDRESS IS ON
00770 200D FE F026     START   LDX    GROUP    *THE STACK
00780 2010 FF F024             STX    COUNT
00790 2013 FE F002     START2  LDX    FIRST    *THIS PUTS DATA INTO RAM
00800 2016 B6 F025     SET     LDA A  COUNT+1
00810 2019 B8 F024             EOR A  COUNT
00820 201C B8 F022             EOR A  SEED
00830 201F A7 00               STA A  0,X
00840 2021 7C F025             INC    COUNT+1
00850 2024 26 03               BNE    COMP1
00860 2026 7C F024             INC    COUNT
00870 2029 08          COMP1   INX
00880 202A BC F004             CPX    LAST     *STAA BEFORE CPX MEANS LAST
00890 202D 26 E7               BNE    SET      *NOT TESTED
00900 202F FE F026             LDX    GROUP    *RE-INITIALIZE COUNT
00910 2032 FF F024             STX    COUNT
00920 2035 FE F002     CHECK   LDX    FIRST    *POINT TO FIRST LOCATION
00930 2038 B6 F025     TEST    LDA A  COUNT+1
00940 203B B8 F024             EOR A  COUNT
00950 203E B8 F022             EOR A  SEED
00960 2041 16                  TAB             *SAVE THE EXPECTED DATA
00970 2042 A8 00               EOR A  0,X
00980 2044 27 21               BEQ    CONTIN   *IF THEY ARE THE SAME THEY
00990 2046 37                  PSH B           *CANCEL
01000 2047 36                  PSH A
01010 2048 FF F028             STX    SAVEX
01020 204B CE E17D             LDX    #$E17D
01030 204E BD E07E             JSR    $E07E
01040 2051 CE F028             LDX    #SAVEX
01050 2054 BD E0C8             JSR    $E0C8    *PRINT ADDRESS
01060 2057 30                  TSX             *POINT TO FAILED BITS
01070 2058 BD E0CA             JSR    $E2CA    *OUTPUT 2HEX AND SPACE
01080 205B BD E0CA             JSR    $E0CA    *AND AGAIN
01090 205E 32                  PUL A
01100 205F 32                  PUL A
01110 2060 FE F028             LDX    SAVEX    *POINT TO ACTUAL DATA
01120 2063 BD E0CA             JSR    $E0CA
01130 2066 09                  DEX             *PREPARE FOR CONTIN
01140 2067 7C F025     CONTIN  INC    COUNT+1
```

```
01150 206A 26 23          BNE    COMP2
01160 206C 7C F024        INC    COUNT
01170 206F 08      COMP2  INX
01180 2070 BC F004        CPX    LAST
01190 2073 26 C3          BNE    TEST      *GET NEW DATA
01200 2075 7C F022 NEXT   INC    SEED      *NEW SEED FOR NEXT PASS
01210 2078 26 0D          BNE    LOOP
01220 207A FE F026        LDX    GROUP
01230 207D 08             INX               *NEXT GROUP
01240 207E FF F026        STX    GROUP
01250 2081 CE F026        LDX    #GROUP
01260 2084 7E E4C8        JMP    $E4C8
01270 2087 7D F022 LOOP   TST    $F020
01280 208A 26 01          BNE    P1         *PRINT '+'?
01290 208C 39             RTS               *NO
01300 208D 96 2B   P1     LDA A  #$2B       *YES
01310 208F 7E E1D1 PRINT  JMP    $E1D1
01320                     END

TOTAL ERRORS 00000

ENTER PASS : 1P,2P,2L,2T
```

```
00010                          NAM     MEMEXS
00020                          OPT     O
00030 0100                     ORG     $100
00040       E0D0     MON       EQU     $E0D0
00050       F002     MEMSTR    EQU     $F002
00060       F004     MEMEND    EQU     $F004
00070                *
00080                * EXAMPLE 1 FILLS MEMORY WITH ZEROS
00090                * STARTING AT THE ADDRESS IN MEMSTR
00100                * ($F002 & $F003), UP TO THE ADDRESS
00110                * IN MEMEND ($F004 & $F005).
00120                *
00130 0100 86 00    EXAM1   LDA A   #$00
00140 0102 FE F002          LDX     MEMSTR
00150 0105 A7 00    FILL    STA A   0,X
00160 0107 08               INX
00170 0108 BC F004          CPX     MEMEND
00180 010B 26 F8            BNE     FILL
00190 010D 7E E0D0          JMP     MON         GO TO MONITOR
00200                *
00210                * EXAMPLE 2 CHECKS MEMORY FOR ZEROS
00220                * FROM THE ADDRESS IN MEMSTR UP TO THE
00230                * ADDRESS IN MEMEND. WHEN A NON-ZERO
00240                * BYTE IS FOUND, THE SWI INSTRUCTION
00250                * CAUSES THE REGISTERS TO BE PRINTED.
00260                * THE INDEX REGISTER HAS THE ADDRESS
00270                * OF THE NON-ZERO BYTE.
00280                *
00290 0110 FE F002 FXAM2   LDX     MEMSTR
00300 0113 09              DEX
00310 0114 86 00   COMP    LDA A   #$00
00320 0116 08              INX
00330 0117 BC F004 NEXT    CPX     MEMEND
00340 011A 26 03           BNE     CONT
00350 011C 7E E0D0         JMP     MON         GO TO MONITOR
00360 011F A1 00   CONT    CMP A   0,X
00370 0121 27 F1           BEQ     COMP
00380 0123 3F              SWI                 ERROR - DISPLAY REGISTERS
00390                      END

TOTAL ERRORS 00000

ENTER PASS : 1P,2P,2L,2T
```

```
0030 *                  MEMORY TEST PROGRAM
0040 *
0050 * THIS ALGORITHM ORIGINALLY WRITTEN BY CHARLES MC COLLOUGH
0060 * RE-WRITTEN AND EXPANDED BY EDGAR R. ALLEN
0070 * THIS VERSION WRITTEN 1/20/78
0080 *
0090 *       MOST MEMORY TESTS EITHER DEAL WITH A SINGLE
0100 * LOCATION ONLY, WHICH WILL NOT FIND BITS RESPONDING TO
0110 * MORE THAN ONE ADDRESS OR THEY FILL MEMORY WITH A BYTE
0120 * THEN CHANGE BITS IN THE TEST LOCATION THEN VERIFY THAT
0130 * NO OTHERS HAVE BEEN ALTERED. THE FIRST IS TOTALLY
0140 * INADEQUATE AND THE SECOND HAS AN EXPONENTIAL INCREASE
0150 * RATE. THIS MEANS THAT TESTING TWICE AS MUCH MEMORY
0160 * TAKES FOUR TIMES AS LONG. WE FEEL THIS IS UNACCEPTABLE
0170 * WITH TODAYS LARGER MEMORY SIZES.
0180 *       THIS PROGRAM GENERATES A PATTERN DEPENDENT UPON A
0190 * SEED BYTE AND THE PLACEMENT OF THE CURRENT TEST LOC-
0200 * ATION.   THEN THE SAME PATTERN IS GENERATED WITHIN
0210 * THE REGISTERS AND COMPARED TO THE DATA IN MEMORY.
0220 * IF THE EXPECTED AND ACTUAL DATA AGREE THEN THE NEXT
0230 * LOCATION IS EXAMINED. OTHERWISE THE FAILING ADDRESS,
0240 * THE FAILING BITS, THE EXPECTED DATA AND THE ACTUAL
0250 * DATA ARE PRINTED. THE EXPECTED AND ACTUAL DATA MAY AGREE,
0260 * THIS PROBABLY MEANS THAT THE MEMORY IS TOO SLOW. THIS
0270 * MAKES THE PRINT OF THE FAILED BITS QUITE HANDY.
0280 *       THE WAY THE DATA IS GENERATED IS BY EXCLUSIVE-
0290 * ORING THE LOW AND HIGH BYTES OF THE COUNTER
0300 * WITH THE SEED. THIS COUNTER IS THEN INCREMENTED FOR
0310 * THE NEXT LOCATION. AFTER ALL LOCATIONS HAVE BEEN
0320 * FILLED THEY ARE COMPARED TO THE SAME DATA GENERATED
0330 * IN THE SAME WAY.  THE SEED IS THEN INCREMENTED AND
0340 * ANOTHER PASS IS MADE. AFTER ALL COMBINATIONS OF THE
0350 * SEED HAVE BEEN TRIED THE COUNTER IN 'SAVEX' IS
0360 * INCREMENTED AND USED AS THE START OF ANOTHER 256 PASSES
0370 * OR ONE GROUP. THIS REMOVES ONE BYTE FROM THE FRONT OF THE
0380 * SERIES AND ADDS A NEW BYTE TO THE BACK.  THIS HAS THE
0390 * EFFECT OF SHIFTING THE PATTERN OVER ONE ADDRESS
0400 * MEANING THE TRANSITIONS FROM ONE STATE TO THE OTHER
0410 * OF EACH BIT ALSO MOVES.  THE TEST WILL DETECT ERRORS
0420 * WHEN THE FAILING BITS ATTEMPT TO MAKE A TRANSITION
0430 * BETWEEN THE TWO, OR MORE, ADDRESSES WHICH AFFECT
0440 * EACH OTHER.
0450 *       THIS PROGRAM USES MSIBUG SCRATCHPAD RAM
0460 * LOCATIONS. THE PROGRAM ITSELF IS FULLY RELOCATABLE.
0470 *       TO RUN THE MEMORY TEST, PUT THE BEGINNING ADDRESS
0480 * TO CHECK IN $F002 & $F003 AND THE END ADDRESS +1
0490 * IN $F004 & $F005. GO AT +0 INTO THE PROGRAM WHICH IS
0500 * $2000 IN THIS EXAMPLE. THE GROUP NUMBER IS PRINTED
0510 * AFTER 256 PASSES THROUGH MEMORY. IF $F020 IS NON-ZERO
0520 * A + IS PRINTED AFTER EACH PASS. WHEN AN ERROR OCCURS,
0530 * THE FAILING ADDRESS, THE FAILING BITS, THE EXPECTED
0540 * DATA AND THE ACTUAL DATA ARE PRINTED. TO START THE
0550 * MEMORY TEST ON A PARTICULAR GROUP, CLEAR THE SEED
0560 * ($F022), PUT THE DESIRED GROUP NUMBER IN $F026 & $F027,
0570 * AND GO AT +9 INTO THE PROGRAM WHICH IN THIS EXAMPLE
0580 * IS $2009.

READY
```

THE INSTRUCTION SET

- 19 LOAD, STORE, ADDRESS
- 8 REGISTER OPERATIONS
- 6 LOGICAL OPERATIONS
- 8 ARITHMETIC OPERATIONS
- 5 SERVICE INTERRUPTS
- 18 BRANCH AND JUMP
- 8 MISCELLANEOUS

D1322

M6800 PROGRAM
72 INSTRUCTIONS
7 ADDRESSING MODES

D632

DATA HANDLING INSTRUCTIONS
(Data Movement)

FUNCTION	MNEMONIC	OPERATION
LOAD ACMLTR	LDAA	M → A
	LDAB	M → B
PUSH DATA	PSHA	A → M_{SP}, SP − 1 → SP
	PSHB	B → M_{SP}, SP − 1 → SP
PULL DATA	PULA	SP + 1 → SP, M_{SP} → A
	PULB	SP + 1 → SP, M_{SP} → B
STORE ACMLTR	STAA	A → M
	STAB	B → M
TRANSFER ACMLTRS	TAB	A → B
	TBA	B → A

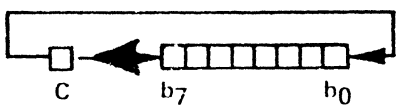
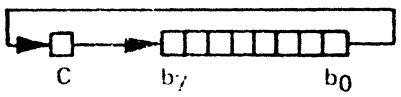
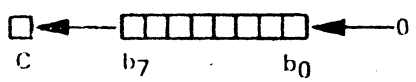
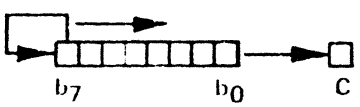
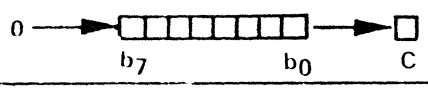
D1489

DATA HANDLING INSTRUCTIONS
(ALTER DATA)

FUNCTION	MNEMONIC	OPERATION
CLEAR	CLR	00 → M
	CLRA	00 → A
	CLRB	00 → B
DECREMENT	DEC	M – 1 → M
	DECA	A – 1 → A
	DECB	B – 1 → B
INCREMENT	INC	M + 1 → M
	INCA	A + 1 → A
	INCB	B + 1 → B
COMPLEMENT, 2'S (NEGATE)	NEG	00 – M → M
	NEGA	00 – A → A
	NEGB	00 – B → B
COMPLEMENT, 1'S	COM	$\overline{M}$ → M
	COMA	$\overline{A}$ → A
	COMB	$\overline{B}$ → B

D1487

DATA HANDLING INSTRUCTIONS
(SHIFT AND ROTATE)

FUNCTION	MNEMONIC		OPERATION
ROTATE LEFT	ROL	M	
	ROLA	A	
	ROLB	B	
ROTATE RIGHT	ROR	M	
	RORA	A	
	RORB	B	
SHIFT LEFT, ARITHMETIC	ASL	M	
	ASLA	A	
	ASLB	B	
SHIFT RIGHT, ARITHMETIC	ASR	M	
	ASRA	A	
	ASRB	B	
SHIFT RIGHT, LOGIC	LSR	M	
	LSRA	A	
	LSRB	B	

D1488-1

ARITHMETIC INSTRUCTIONS

FUNCTION	MNEMONIC	OPERATION
ADD	ADDA ADDB	A + M → A B + M → B
ADD ACCUMULATORS	ABA	A + B → A
ADD WITH CARRY	ADCA ADCB	A + M + C → A B + M + C → B
COMPLEMENT, 2'S (NEGATE)	NEG NEGA NEGB	00 − M → M 00 − A → A 00 − B → B
DECIMAL ADJUST, A	DAA	CONVERTS BINARY ADD. OF BCD CHARACTERS INTO BCD FORMAT
SUBTRACT	SUBA SUBB	A − M → A B − M → B
SUBTRACT ACCUMULATORS	SBA	A − B → A
SUBTRACT WITH CARRY	SBCA SBCB	A − M − C → A B − M − C → B

D1482

LOGIC INSTRUCTIONS

FUNCTION	MNEMONIC	OPERATION
AND	ANDA ANDB	A • M → A B • M → B
COMPLEMENT, 1'S	COM COMA COMB	$\overline{M}$ → M $\overline{A}$ → A $\overline{B}$ → B
EXCLUSIVE OR	EORA EORB	A ⊕ M → A B ⊕ M → B
OR, INCLUSIVE	ORA ORB	A + M → A B + M → B

D1483

JUMP AND BRANCH INSTRUCTIONS

FUNCTION	MNEMONIC	BRANCH TEST
BRANCH ALWAYS	BRA	NONE
BRANCH IF CARRY CLEAR	BCC	C = 0
BRANCH IF CARRY SET	BCS	C = 1
BRANCH IF = ZERO	BEQ	Z = 1
BRANCH IF $\geq$ ZERO	BGE	N $\oplus$ V = 0
BRANCH IF > ZERO	BGT	Z + (N $\oplus$ V) = 0
BRANCH IF HIGHER	BHI	C + Z = 1
BRANCH IF $\leq$ ZERO	BLE	Z + (N $\oplus$ V) = 1
BRANCH IF LOWER OR SAME	BLS	C + Z = 1
BRANCH IF < ZERO	BLT	N $\oplus$ V = 1
BRANCH IF MINUS	BMI	N = 1
BRANCH IF NOT EQUAL ZERO	BNE	Z = 0

D1485

JUMP AND BRANCH INSTRUCTIONS

FUNCTION	MNEMONIC	BRANCH TEST
BRANCH IF OVERFLOW CLEAR	BVC	V = 0
BRANCH IF OVERFLOW SET	BVS	V = 1
BRANCH IF PLUS	BPL	N = 0
BRANCH TO SUBROUTINE	BSR	
JUMP	JMP	
JUMP TO SUBROUTINE	JSR	
NO OPERATION	NOP	ADVANCES PROG. CNTR. ONLY
RETURN FROM SUBROUTINE	RTS	

D1486

DATA TEST INSTRUCTIONS

FUNCTION	MNEMONIC	TEST
BIT TEST	BITA	A • M
	BIT B	B • M
COMPARE	CMPA	A – M
	CMPB	B – M
	CBA	A – B
TEST, ZERO OR MINUS	TST	M – 00
	TSTA	A – 00
	TSTB	B – 00

D1484

CONDITION CODE REGISTER INSTRUCTIONS

FUNCTION	MNEMONIC	OPERATION
CLEAR CARRY	CLC	0 → C
CLEAR INTERRUPT MASK	CLI	0 → I
CLEAR OVERFLOW	CLV	0 → V
SET CARRY	SEC	1 → C
SET INTERRUPT MASK	SEI	1 → I
SET OVERFLOW	SEV	1 → V
ACMLTR A → CCR	TAP	A → CCR
CCR → ACMLTR A	TPA	CCR → A

D1481

INDEX REGISTER AND STACK POINTER INSTRUCTIONS

FUNCTION	MNEMONIC	OPERATION
COMPARE INDEX REG	CPX	$X_H - M, X_L - (M + 1)$
DECREMENT INDEX REG	DEX	$X - 1 \rightarrow X$
DECREMENT STACK PNTR	DES	$SP - 1 \rightarrow SP$
INCREMENT INDEX REG	INX	$X + 1 \rightarrow X$
INCREMENT STACK PNTR	INS	$SP + 1 \rightarrow SP$
LOAD INDEX REG	LDX	$M \rightarrow X_H, (M + 1) \rightarrow X_L$
LOAD STACK PNTR	LDS	$M \rightarrow SP_H, (M + 1) \rightarrow SP_L$
STORE INDEX REG	STX	$X_H \rightarrow M, X_L \rightarrow (M + 1)$
STORE STACK PNTR	STS	$SP_H \rightarrow M, SP_L \rightarrow (M + 1)$
INDX REG → STACK PNTR	TXS	$X - 1 \rightarrow SP$
STACK PNTR → INDX REG	TSX	$SP + 1 \rightarrow X$

D1480

INPUT/OUTPUT INSTRUCTIONS

NONE!

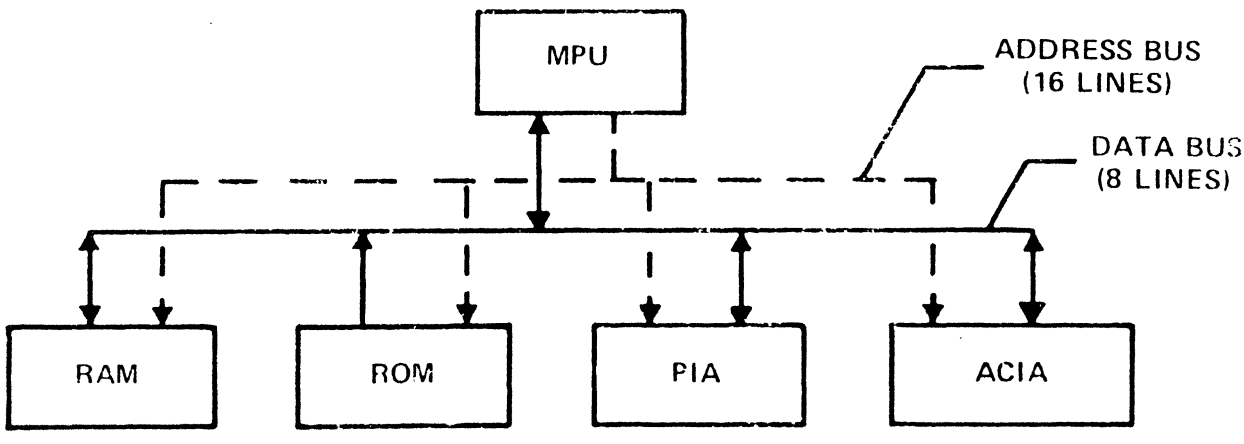
D1449A

INTRODUCTION

The Motorola M6800 Microcomputer System of standard LSI (Large Scale Integration) devices permits the systems designer to configure and connect a total system with a minimum amount of time and effort. The MC6800 Microprocessing Unit (MPU) forms the nucleus of the system. LSI modules available which may be used to configure a total system in conjunction with the MC6800 MPU, include: 1) MC6810 Random Access Memory (RAM); 2) MC6816 Read Only Memory (ROM); 3) MC6820 Peripheral Interface Adapter (PIA), and 4) MC6850 Asynchronous Communications Interface Adapter (ACIA).

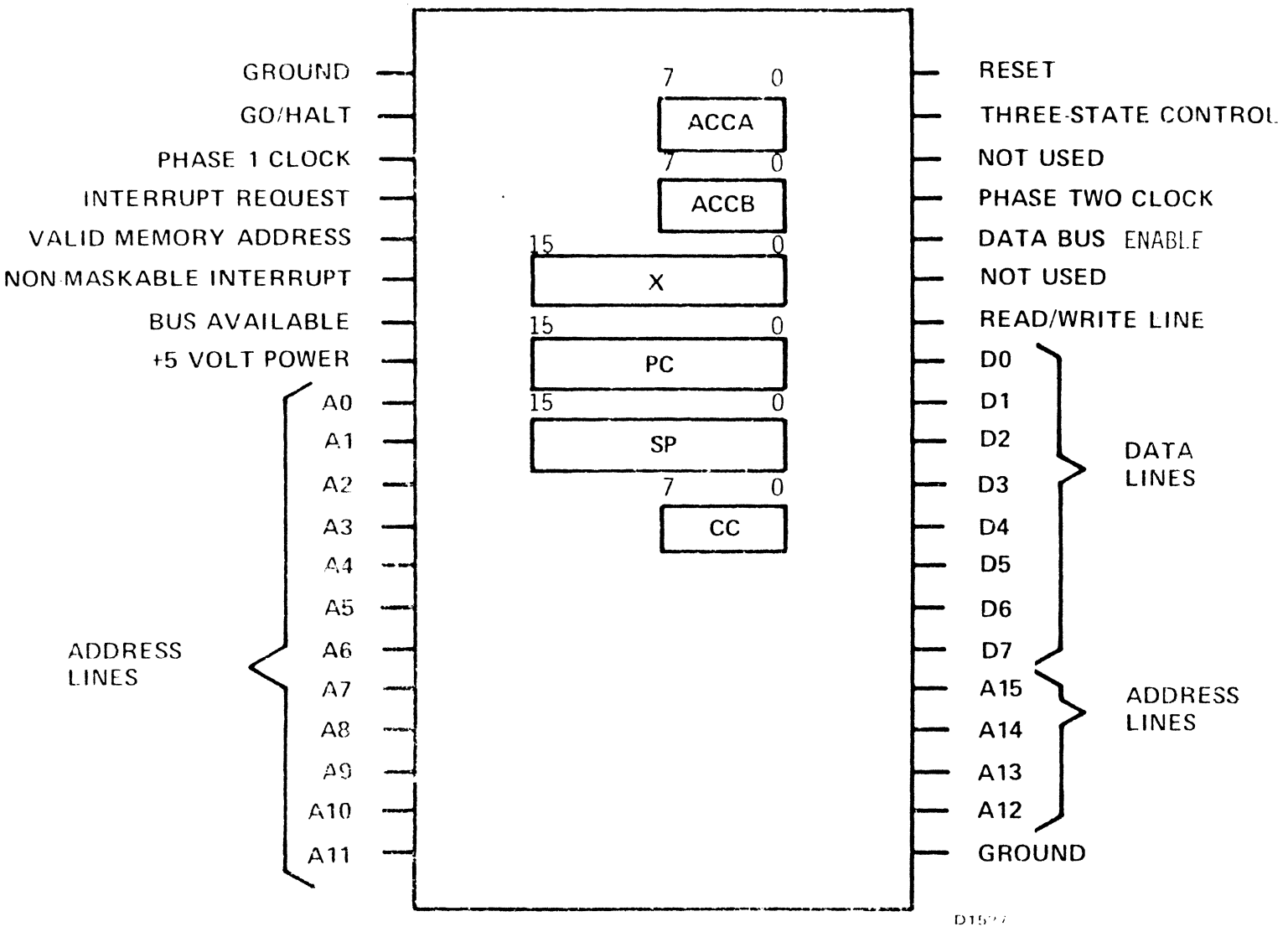
The MPU communicates with the rest of the system via a 16 bit address bus and an 8 bit data bus. The 16 bit address bus provides the MPU the capability of addressing up to 64K. The 8 bit data bus is bi-directional in that data is transferred both into the MPU or out of the MPU over the same bus. A read/write (R/W) line is provided to allow the MPU to control the direction of data transfer. Since the same bus is used for both data into the MPU or out of the MPU, a separate 8 line bus is saved.

Other features of the M6800 system include a single +5 volt supply, operation at clock rates from 100 kilohertz to 1 megahertz, plus hardware and software interrupt capability.



Microprocessing Unit (MC6800)

The nucleus of the M6800 Microcomputer Family is the microprocessing unit (MPU). The MPU is enclosed in a 40 pin package as shown below:



Features included in the MPU are:

1. Two accumulators (ACCA and ACCB)
2. One index register (X)
3. One program counter register (PC)
4. One stack pointer register (SP)

5. One condition code register (CC)
6. 72 instructions
7. Five addressing modes
8. System clock range of 100 kilohertz to 1 megahertz
9. Program interrupt capability

Accumulators

The MPU contains 2 accumulators designated ACCA and ACCB. Each accumulator is 8 bits (one byte) long and is used to hold operands and data from the arithmetic logic unit. Instructions which involve one or both accumulators are:

ABA - Add accumulator A to accumulator B
ADC - Add with carry
ADD - Add without carry
AND - Logical AND
ASL - Arithmetic shift left
ASR - Arithmetic shift right
BIT - Bit test
CBA - Compare accumulators
CLR - Clear
CMP - Compare
COM - Complement
DAA - Decimal adjust ACCA
DEC - Decrement
EOR - Exclusive OR

INC - Increment

LDA - Load accumulator

LSR - Logical shift right

NEA - Negate

ORA - Inclusive OR

PSH - Push data onto stack

PUL - Pull data from stack

ROL - Rotate left

ROR - Rotate right

RTI - Return from interrupt

SBA - Subtract accumulators

SBC - Subtract with carry

STA - Store accumulator

SUB - Subtract

SWI - Software interrupt

TAB - Transfer from accumulator A to accumulator B

TAP - Transfer from accumulator A to processor condition codes register

TBA - Transfer from accumulator B to accumulator A

TPA - Transfer from processor condition codes register to accumulator A

TST - Test

WAI - Wait for interrupt

Index Register

The index register (X) is a 16 bit (2 byte) register which is primarily used to store a memory address in the Indexed mode of memory addressing. The index register may be decremented, incremented and stored. Instructions which involve the index register are:

CPX - Compare index register

DEX - Decrement index register

INX - Increment index register

LDX - Load index register

RTI - Return from interrupt

STX - Store index register

SWI - Software interrupt

TSX - Transfer stack pointer to index register

TXS - Transfer index register to stack pointer

WAI - Wait for interrupt

Program Counter

The program counter (PC) is a 16 bit register that contains the address of the next byte to be fetched from memory. When the current value of the program counter is placed on the address buss, the program counter will be incremented automatically.

Stack Pointer

The Stack Pointer (SP) is a 16 bit (2 byte) register that contains a beginning address, normally in RAM, where the status of the MPU registers may be stored when the MPU has other functions to perform, such as during an interrupt or during a Branch to Subroutine (BTS). The address in the stack pointer is the starting address of sequential memory locations in RAM where MPU status registers will be stored. The status of the MPU will be stored in the RAM as follows:

Stack Pointer Address	:	contents of PCL
Stack Pointer Address-1	:	contents of PCH
Stack Pointer Address-2	:	contents of IXL
Stack Pointer Address-3	:	contents of IXH
Stack Pointer Address-4	:	contents of ACCA

Stack Pointer Address-5 : Contents of ACCB

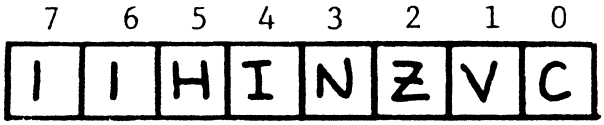
Stack Pointer Address-6 : Contents of CC

After the status of each register is stored on the stack, the Stack Pointer will be decremented. When the stack is unloaded (status of registers restored), the status of the last register on the stack will be the first register that is restored.

Condition Code Register (CC)

The condition code register is an 8 bit register. Each individual bit may get set or get cleared from execution of an instruction. To see how each instruction effects the condition code register, refer to the M6800 programming manual. The primary use of this register is execution of the conditional branch instruction. Bit 6 and 7 are not used and remain at logic "1."

CONDITION CODE REGISTER



BIT NO.	FUNCTION
0	C (Carry-Borrow Test)
1	V (Overflow Test)
2	Z (Zero Test)
3	N (Negative Test)
4	I (Interrupt Mask Test)
5	H (Half Carry Test)

Carry-Borrow: For addition, the carry-borrow condition code (C) in the zero bit position, represents a carry. This bit gets set (C=1) to indicate a carry, and is reset (C=0) if there is no carry.

For subtraction, the C bit is set (C=1) to indicate a borrow and is reset (C=0) to indicate there was no borrow.

Overflow: The V bit (bit 1) of the condition code register is set (V=1) when two's complement overflow results from an arithmetic operation, and is reset (V=0) if two's complement overflow does not occur.

Zero: The Z bit (bit 2) of the condition code register is set (Z=1) if the result of an arithmetic operation is zero, and is reset (Z=0) if the result is not zero.

Negative: The N bit (bit 3) of the condition code register is set (N=1) if bit 7 of an arithmetic operation is set (equal to 1). This indicates that the two's complement number, represented by the bit pattern of the result, is negative. The N bit is reset (N=0) if bit 7 of the arithmetic result is equal to 0.

Interrupt Mask: If this I bit (bit 4) is set (I=1), the MPU cannot respond to an interrupt request from any peripheral device.

Half-Carry: The half carry bit H (bit 5) of the condition code register is set (H=1) during execution of any of the instructions ABA,ADC, or ADD, if there is a carry from bit position 3 to bit position 4. The half carry is reset (H=0) during these operations, if there is no carry from bit position 4.

MPU Signal Descriptions

1. READ/WRITE (R/W): This output line is used to signal all devices external to the MPU that the MPU is in a read state (R/W=High) or a write state (R/W=Low). The normal standby state of this line when no external devices are being accessed is a high state. This line is three-state. When three-state goes high, this line enters the high impedance mode.

2. VALID MEMORY ADDRESS (VMA): This output line, (when in the high state) tells all devices external to the MPU that there is a valid address in the address bus. For RAM's and ROM's, this line should be ANDed with $\phi 2$ clock and used as one of the enables. For PIA's, this line should be ANDed with one of the PIA address lines. This signal is not three-state.

3. DATA BUS ENABLE(DBE): This signal will enable the data bus drives when in the high state. This input is normally the phase 2 ($\phi 2$) clock. During the high state, it will permit data to be output during a write cycle. During an MPU read cycle, the data bus drives will be disabled internally.

4. INTERRUPT REQUEST(IRQ): This input from the PIA's requests that an interrupt sequence be generated within the machine. The processor will wait until it completes the current instruction that is being executed before it recognizes the request. At that time, if the interrupt mask bit in the Condition Code Register is not set (interrupt masked), the machine will begin an interrupt sequence. The Index Register, Program Counter, Accumulators, and Condition Code Register are stored away on the stack. Next the MPU will respond to the interrupt request by setting the interrupt mask bit high so that no further interrupts may occur. At the end of the cycle, a 16-bit address will be loaded that points to a vectoring address which is located in memory locations n-6 and n-7 where n is the highest ROM address. An address loaded at these locations causes the MPU to branch to an interrupt routine in memory.

5. Phase One (Ø 1)& Phase (Ø2)Clocks: These two pins are used for a two phase non-overlapping clock that runs at the V_{DD} voltage level. These clocks run at a rate up to 1 megahertz.

6. Restart (RES): RESTART (RES)--This input is used to start the MPU from a power down condition, resulting from a power failure or an initial start-up of the processor. If a positive edge is detected on the input, this will signal the MPU to begin the restart sequence. This will restart the MPU and start execution of a routine to initialize the processor. All the higher order address lines will be forced high. For the restart, the last 2 memory locations in the last ROM (n&n-1) will be accessed, whereby an address is stored which is the address to be loaded in the program counter which tells the processor where program execution is to begin.

7. NON-MASKABLE INTERRUPT(NMI): This input requests that a non-mask-interrupt sequence be generated within the processor. As with the Interrupt Request signal, the processor will complete the current instruction that is being executed before it recognizes the NMI signal. The interrupt mask bit in the Condition Code Register has no effect on NMI.
The Index Register, Program Counter, Accumulators, and Condition Code Register are stored away on the stack. At the end of the cycle, a 16-bit address will be loaded that points to a vectoring address which is located in memory locations n-2 and n-3. An address loaded at these locations causes the MPU to branch to an non-maskable interrupt routine in memory.

8. Go/Halt(G/H): When this input is in the high state, the machine will fetch the instruction

addressed by the program counter and start execution. When low all activity in the machine will be halted. This input is level sensitive. In the halt mode, the machine will stop at the end of an instruction. Bus Available will be at a logic "1" level. Valid Memory Address will be at a logic "0" and all other three-state lines will be in the three-state mode.

The halt line must go low with the leading edge of phase one to insure single instruction operation. If the halt line does not go low with the leading edge of phase one, one or two instruction operations may result, depending on when the halt line goes low relative to the phasing of the clock.

9. BUS AVAILABLE (BA): The Bus Available signal will normally be in the low state. When activated, it will go to the high state indicating that the MPU has stopped and that the address bus is available. This will occur if the GO/HALT line is in the Halt (low) mode or the MPU is in a "Wait" state as the result of some instruction, such as the WAI instruction.

10. THREE-STATE CONTROL (TSC): This input causes all of the address lines and the Read/Write line to go into the off or high impedance state. The Valid Memory address and Bus Available signals will be forced low. The data bus is not affected by TSC and has its own enable (Data Bus Enable). In DMA applications, the Three-State Control line should be brought high on the leading edge of the Phase One Clock. The $\phi 1$ clock must be held in the high state for this function to operate properly. The address bus will then be available for other devices to directly address memory. Since the MPU is a dynamic device, it must be refreshed periodically or destruction of data will occur.

11. ADDRESS BUS (A0/A15): Sixteen pins are used for the address bus. The outputs are three-state bus drivers capable of driving one standard TTL load and 130pf at 1 Megahertz.

When the output is turned off, it is essentially an open circuit. This permits the MPU to be used in DMA applications.

12. DATA BUS (DO/D7): Eight pins are used for the data bus. It is bi-directional, transferring data to and from the memory and peripheral devices. It also has three-state output buffers capable of driving one standard TTL load and 130pf at 1 Megahertz.

Microprocessor Instruction Set--Alphabetic Sequence

ABA	Add Accumulators
ADC	Add with Carry
ADD	Add
AND	Logical And
ASL	Arithmetic Shift Left
ASR	Arithmetic Shift Right
BCC	Branch if Carry Clear
BCS	Branch if Carry Set
BEQ	Branch if Equal to Zero
BGE	Branch if Greater or Equal Zero
BGT	Branch if Greater than Zero
BHI	Branch if Higher
BIT	Bit Test
BLE	Branch if Less or Equal
BLS	Branch if Lower of Same
BLT	Branch if Less than Zero
BMI	Branch if Minus
BNE	Branch if Not Equal to Zero
BPL	Branch if Plus
BRA	Branch Always
BSR	Branch to Subroutine
BVC	Branch if Overflow Clear
BVS	Branch if Overflow Set
CBA	Compare Accumulators
CLC	Clear Carry
CLI	Clear Interrupt Mask
CLR	Clear
CLV	Clear Overflow
CMP	Compare
COM	Complement
CPX	Compare Index Register
DAA	Decimal Adjust
DEC	Decrement
DES	Decrement Stack Pointer
DEX	Decrement Index Register
EOR	Exclusive OR
INC	Increment
INS	Increment Stack Pointer
INX	Increment Index Register
JMP	Jump
JSR	Jump to Subroutine
LDA	Load Accumulator
LDS	Load Stack Pointer
LDX	Load Index Register
LSR	Logical Shift Right
NEG	Negate
NOP	No Operation
ORA	Inclusive OR Accumulator
PSH	Push Data
PUL	Pull Data
ROL	Rotate Left
ROR	Rotate Right
RTI	Return from Interrupt
RTS	Return from Subroutine
SBA	Subtract Accumulators
SBC	Subtract with Carry
SEC	Set Carry
SEI	Set Interrupt Mask
SEV	Set Overflow
STA	Store Accumulator
STS	Store Stack Register
STX	Store Index Register
SUB	Subtract
SWI	Software Interrupt
TAB	Transfer Accumulators
TAP	Transfer Accumulators to Condition Code Reg.
TBA	Transfer Accumulators
TPA	Transfer Condition Code Reg. to Accumulator
TST	Test
TSX	Transfer Stack Pointer to Index Register
TXS	Transfer Index Register to Stack Pointer
WAI	Wait for Interrupt

Hardware Interrupts

What happens when the MPU gets a hardware interrupt? After it has been determined that the interrupt is not non-maskable, the MPU checks the status of the mask bit (bit 4 of the condition code register). If the mask bit is set, the main program continues until a CLI (clears bit 4 of condition code register) instruction is executed, after which time the MPU will honor an interrupt by going to the stack pointer (SP) register and fetch an address which will be the 1st address in RAM where the status of the MPU registers will be stored during servicing of the interrupt.

SP	:	contents of program counter low
SP-1	:	contents of program counter high
SP-2	:	contents of index register low
SP-3	:	contents of index register high
SP-4	:	contents of accumulator A
SP-5	:	contents of accumulator B
SP-6	:	contents of condition code register

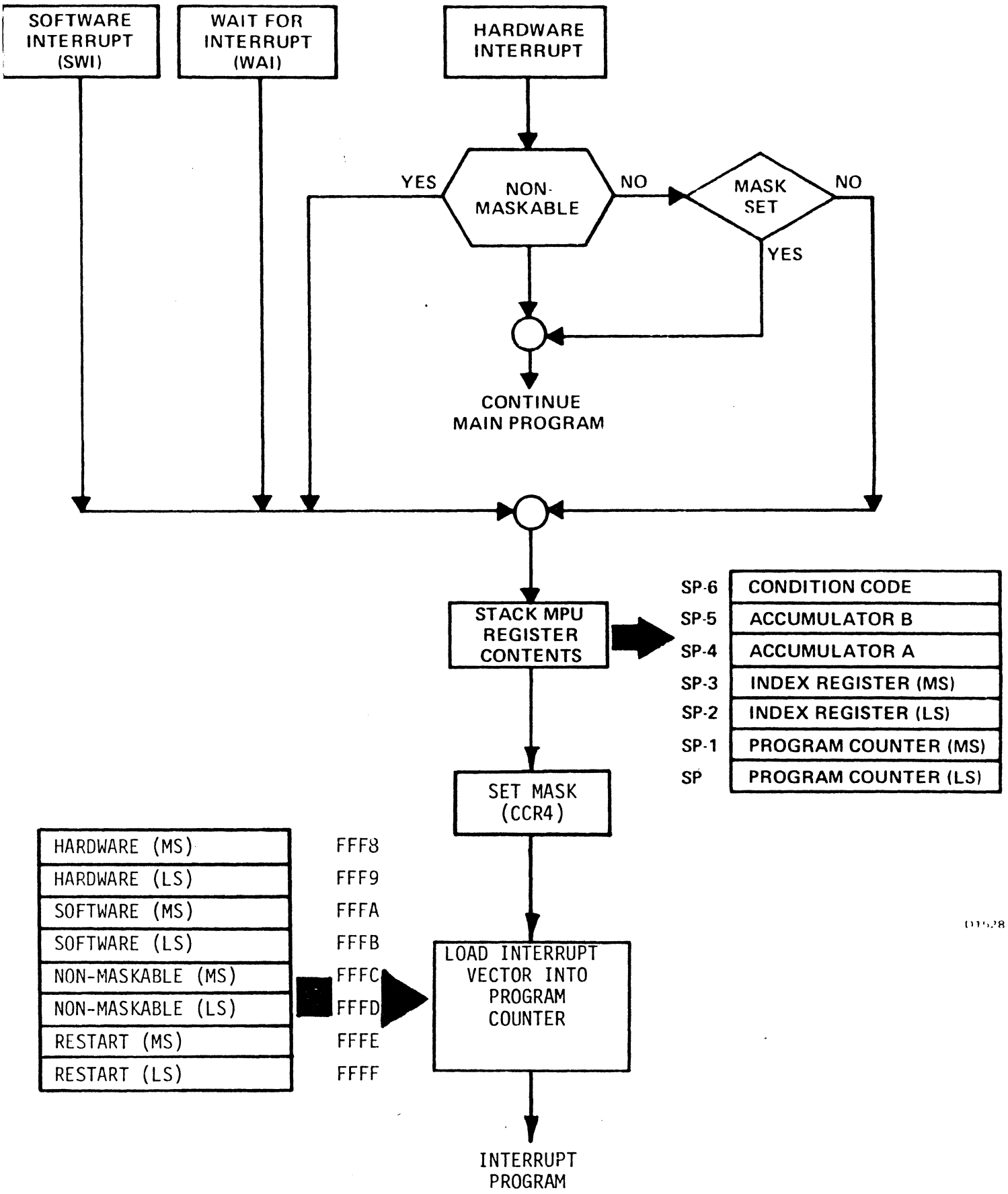
The address in the stack pointer register is determined by the programmer.

After the contents of the MPU registers have been stored in the stack, the mask bit is set thus preventing any further interrupts from interfering with the MPU until the program executes a CLI instruction. Next the MPU hardware automatically looks at addresses FFF8(MS) & FFF9 (LS) for the address of the poling routine to find out where the interrupt came from and what action to take.

After the interrupt has been serviced and an RTI instruction is executed, the stack, which contains the status of the registers before the interrupt, is unloaded in reverse order, i.e. the condition code register is loaded first, then accumulator B is restored, etc. When the registers have been restored to their status before the interrupt, the processor continues as though nothing happened.

The total story of interrupts is shown on the next two pages in the form of flow charts.

INTERRUPT FLOW CHART



SUMMARY OF MPU OPERATION

The MPU requires a two phase symmetrical, TTL compatible, non-overlapping clock. During the first phase of the clock ($\emptyset_1$ high) an address will be placed on the address bus by the MPU. During the second phase of the clock ($\emptyset_2$ high), the bidirectional data bus will be active. The first byte of an instruction enters the MPU and is transferred into an internal instruction register and decoded by the MPU. The MPU will then contain the information needed to read in an additional one or two bytes of program is necessary. Once the entire instruction is read into the MPU (one, two or three bytes) the instruction is then executed. The MPU then reads in the next sequential byte in the program and places it again in the instruction register. The program will sequentially be executed in this manner unless a branch or jump instruction changes the value of the program counter. If this occurs, the next instruction to be executed is determined by the new program counter value.

If an interrupt or reset occurs during this process, the program counter value will also be changed. The new program counter value is determined by the highest eight memory locations that are reserved for reset and interrupt vectors.

In the case of interrupt, the stack pointer is used to store the contents of the internal registers necessary to return to the program location prior to the interrupt. This happens when the interrupt program exits by an RTI (Return from interrupt instruction). Similarly, the stack pointer is used to store the program counter value when a JSR (Jump to Subroutine) or BSR (Branch to Subroutine) instruction occurs. The program counter returns to its original value when an RTS (Return

SUMMARY OF MPU OPERATION (Continued)

from Subroutine) instruction occurs. The stack pointer value is set by an LDS (Load Stack Pointer) instruction.

RESET SEQUENCE

1. While $\overline{\text{HALT}}$ is high, RESET goes low for at least eight cycles of $\emptyset_1$, $\emptyset_2$ during which all internal registers are cleared and interrupt bit (I) in CC is set.
2. Data at FFFE loads into PCH.
3. Data at FFFF loads into PCL.
4. PC contents go out on ADRS bus during $\emptyset_1$.
5. Contents of cell addressed enters instruction register during $\emptyset_2$ and is decoded as first instruction.
6. If two or more byte instruction, additional bytes enter MPU for execution. If not, go to next step.
7. After execution, step 5 is repeated for subsequent instructions.

$\overline{\text{IRQ}}$ SEQUENCE

1. If bit "I" in condition code register is not set (I = 0) and $\overline{\text{IRQ}}$ goes low for at least one $\emptyset_2$ cycle, the $\overline{\text{IRQ}}$ sequence will be entered.
2. After completion of the current instruction, internal registers PC, X, A, B and CC will be stored in RAM at the address indicated by the stack pointer in descending locations (7 bytes in all).
3. The $\overline{\text{IRQ}}$ mask (bit I = 1) is set.
4. Data at FFF8 gets loaded into PCH.
5. Data at FFF9 gets loaded into PCL.
6. PC contents go out on address bus during $\emptyset_1$.
7. Contents of call addressed enters instruction register during $\emptyset_2$ and is decoded as first instruction of interrupt routine.
8. If it is a more than 1 byte instruction, additional bytes enter MPU for execution. If not, go to next step.
9. After execution, step 5 is repeated for subsequent instructions. This loop is repeated until the instruction "RTI" is executed.

$\overline{\text{NMI}}$ SEQUENCE

1. If $\overline{\text{NMI}}$ goes low for at least one $\emptyset_2$ cycle, the MPU will wait for completion of current instruction.
2. The internal registers PC, X, A, B and CC will then be stored in RAM at the address indicated by the stack pointer in descending locations (7 bytes in all).
3. The $\overline{\text{IRQ}}$ (bit I = 1) mask is set.
4. Data at FFFC is loaded into PCH.
5. Data at FFFD is loaded into PCL.
6. PC contents go out on ADRS bus during $\emptyset_1$.
7. Contents of cell addressed enters instruction register during $\emptyset_2$ and is decoded as first instruction of NMI subroutine.
8. If two or more byte instruction, additional bytes enter MPU for execution. If not, go to next step.
9. After execution, Step 5 is repeated for subsequent instructions. This loop is repeated until the instruction "RTI" is executed.

RTI EXECUTION

1. The contents of the stack are loaded back into the MPU. (unwinds)
2. The contents of the PC go out on the address bus to fetch the first byte of the next instruction.

SWI INSTRUCTION

1. Contents of the MPU registers PC, 1X, ACCA, ACCB and CC are stored in RAM at the address indicated by the stack pointer in descending location (7 bytes in all).
2. The $\overline{IRQ}$ mask (bit I = 1) is set.
3. Data at FFFA gets loaded into PCH.
4. Data at FFFB gets loaded into PCL.
5. PC contents go out on address bus during $\emptyset_1$.
6. Contents of cell addressed enters instruction register during $\emptyset_2$ and is decoded as first instruction of SWI subroutine.
7. If it is a more than one byte instruction, additional bytes enter MPU for execution. If not, go to next step.
8. After execution, Step 6 is repeated for subsequent instructions. This loop is repeated until the instruction "RTI" is executed.

Number Systems

Everyone is quite familiar with the base 10 number system i.e. 0, 1, 2, 3, 4, 5, 6, 7, 8, & 9, since this is the system we all use day to day. Let us review a typical number, say 2743, and see what it really means. The least significant digit (LSD) is 3 and the most significant digit (MSD) is 2. Since we are talking about a base 10 number, the number 2743 really is $3\times10^{0} + 4\times10^{1} + 7\times10^{2} + 2\times10^{3} = 3\times1 + 4\times10 + 7\times100 + 2\times1000$

$= 3+40 + 700 + 2000$

$= 2743.$

In digital computers, base 10 numbers are represented in binary form, i.e. 1's & 0's. Lets take a base 10 number and convert it to a binary (base 2) number. A method of doing this is known as "repeated division by 2". The base 10 number of 47 is converted to binary as shown below:

$2\overline{)47}$ R=1

$2\overline{)23}$ R=1

$2\overline{)11}$ R=1

$2\overline{)5}$ R=1

$2\overline{)2}$ R=0

$2\overline{)1}$ R=1

101111_2

Converting 101111_2 back to our base 10 number is done in the same manner as above.

$$101111_2 = 1x2^0 + 1x2^1 + 1x2^2 + 1x2^3 + 0x2^4 + 1x2^5$$
$$= 1x1 + 1x2 + 1x4 + 1x8 + 0 + 1x32$$
$$= 1 + 2 + 4 + 8 + 0 + 32$$
$$= 47_{10}$$

In general, converting from a number in any base to a number in base 10 is accomplished as follows:

$(A_0 B^0 + A_1 B^1 + A_2 B^2 + A_3 B^3 + A_4 B^4 - - - - - - A_m B^m)$

where B is the base of the number system and A is the particular digit in the original number corresponding to its position to the left of the decimal point. On the example just completed, (101111). $A_0 = 1$, $A_1 = 1$, $A_2 = 1$, $A_3 = 1$, $A_4 = 0$, & $A_5 = 1$ and B =2 (base 2).

Another base which is very convenient in digital computers is base 8, since base 8 is really a convenient way of representing base 2. Lets illustrate by converting a base 10 number to base 8 & base 2. Let's convert 61 in base 10 to a number in base 8 and a number in base 2. By continuous division:

$8\overline{)61}$ R=5 ↑ 75_8

$8\overline{)7}$ R=7

$2\overline{)61}$ R=1

$2\overline{)30}$ R=0

$2\overline{)15}$ R=1 ↑ 111101_2

$2\overline{)7}$ R=1

$2\overline{)3}$ R=1

$2\overline{)1}$ R=1

First lets prove that 75_8 & 111101_2 are really equal to 61_{10}.

$$75_8 = 5x8^0 + 7x8^1$$
$$= 5x1 + 7x8$$
$$= 5 + 56$$
$$= 61_{10}$$

$$111101_2 = 1x2^0 + 0x2^1 + 1x2^2 + 1x2^3 + 1x2^4 + 1x2^5$$
$$= 1x1 + 0 + 1x4 + 1+8 + 1x16 + 1x32$$
$$= 1 + 0 + 4 + 8 + 16 + 32$$
$$+ 61_{10}$$

Notice that if we take the base 8 number of 75 and convert each digit to base 2, we have the same number as when we converted the base 10 number to base 2. i.e.

Convert 7 to base 2

$2\overline{)7}^{3}$ R=1

$2\overline{)3}^{1}$ R=1 ↑ 111_2

$2\overline{)1}^{0}$ R=1

Convert 5 to base 2

$2\overline{)5}^{2}$ R=1

$2\overline{)2}^{1}$ R=0 ↑ 101_2

$2\overline{)1}^{0}$ R=1

Therefore $75_8 = 111101$ which is the same pattern of 1's & 0's as we got from converting from base 10 to base 2. What this really says that it is easier to convert any base 10 number to base 8 by continuou division, and then convert each digit of the base 8 number to base 2. Let's look at another example. Convert 183_{10} to base 8 & to base 2.

$8\overline{)183}$ R=7

$8\overline{)22}$ R=6 ↑ 267_8

$8\overline{)2}$ R=2

$2\overline{)183}$ R=1

$2\overline{)91}$ R=1

$2\overline{)45}$ R=1 ↑ 10110111_2

$2\overline{)22}$ R=0

$2\overline{)11}$ R=1

$2\overline{)5}$ R=1

$2\overline{)2}$ R=0

$2\overline{)1}$ R=1

$267_8 = 7x8^0 + 6x8^1 + 2x8^2$

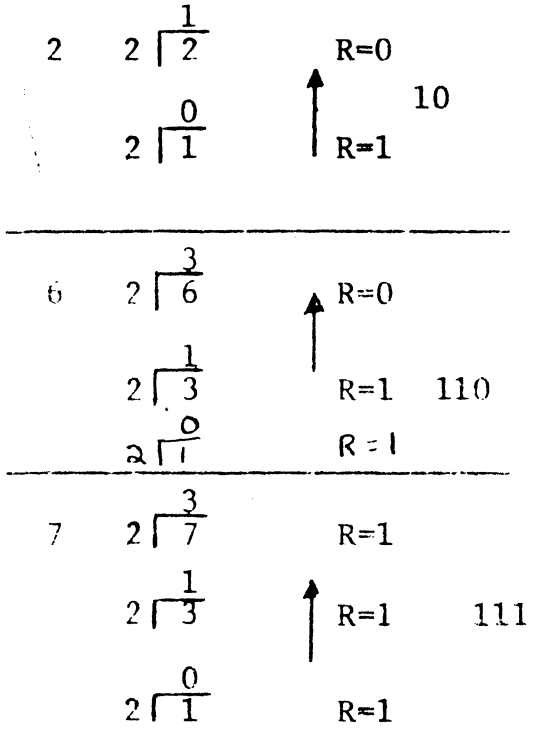
$= 7x1 + 6x8 + 2x64$

$= 7 + 48 + 128$

$= 183$

to convert

267_8 directly to base 2, we convert each base 8 digit separately.



therefore $2_8 = 10_2$, $6_8 = 110_2$, & $7_8 = 111_2$ and

$267_8 = 10110111_2$

Digital computers are designed to use binary numbers in their working registers. The working registers vary in number of bits depending on the manufacturer. The Motorola M6800 micro-processor utilizes, in general, 8 bit words (or registers). This leads to another number base, not yet mentioned, of hexadecimal. Hexadecimal is really a base 16 number system and can be handled in exactly the same manner as base 8 or base 2. In hexadecimal, four bits (in binary) represents one hexadecimal number. Thus , an eight bit register can be represented by a hex number of 2 digits long. To illustrate, lets assume we have the number of 147_8 in an eight bit register. This in binary form is 01100111 . If this bit pattern is divided into 2-four bit words of 0110 & 0111, then in hex, 147_8 can be represented as 67_{16}. To prove both are equal, lets convert both back to their base 10 number.

$$147_8 = 7x8^0 + 4x8^1 + 1x8^2$$
$$= 7x1 + 4x8 + 1x64$$
$$= 7 + 32 + 64$$
$$= 103_{10}$$

$$67_{16} = 7x16^0 + 6x16^1$$
$$= 7x1 + 6x16$$
$$= 7 + 96$$
$$= 103_{10}$$

As you probably have wondered by now, how do we represent these hex (base 16) numbers above 9? Here is the base 16 number conpared with its equivalent base 10 number.

Base 10	Base 16
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	B
12	C

13	D
14	E
15	F

To convert any base 10 number to hex (base 16) you may convert it to base 8 first, then represent the base 8 number with its binary representation. By taking the binary representation of the number and grouping the bits from right to left in groups of four which are then represented in hex per the above table. Or one may convert any base 10 number to hex by our continuous division rule as before. Lets convert 825_{10} to hex.

$$16 \overline{)825}^{51} \quad R=9$$

$$16 \overline{)51}^{3} \quad R=3 \qquad \uparrow \quad 339_{16}$$

$$16 \overline{)3}^{0} \quad R=3$$

therefore $825_{10} = 339_{16}$

to convert 339_{16} back to our base 10 number,

$$\begin{aligned} 339_{16} &= 9\text{x}16^{0} + 3\text{x}16^{1} + 3\text{x}16^{2} \\ &= 9\text{x}1 + 3\text{x}16 + 3\text{x}256 \\ &= 9 + 48 + 768 \\ &= 825_{10} \end{aligned}$$

To show the relationship between hex, binary, and octal, lets convert 825_{10} to octal & to binary and then back to hex.

825_{10} to octal

$8 \overline{)825}$	103	R=1
$8 \overline{)103}$	12	R=7
$8 \overline{)12}$	1	R=4
$8 \overline{)1}$	0	R=1

1471_8

825_{10} to binary

$2 \overline{)825}$	412	R=1
$2 \overline{)412}$	206	R=0
$2 \overline{)206}$	103	R=0
$2 \overline{)103}$	51	R=1
$2 \overline{)51}$	25	R=1
$2 \overline{)25}$	12	R=1
$2 \overline{)12}$	6	R=0
$2 \overline{)6}$	3	R=0
$2 \overline{)3}$	1	R=1
$2 \overline{)1}$	0	R=1

1100111001_2

$825_{10} = 1471_8$

$= 1x8^0 + 7x8^1 + 4x8^2 + 1x8^3$

$= 1x1 + 7x8 + 4x64 + 1x512$

$= 1 + 56 + 256 + 512$

$= 825_{10}$

$825_{10} = 1100111001_2$

$= 1x2^0 + 0x2^1 + 0x2^2 + 1x2^3 + 1x2^4 + 1x2^5 + 0x2^6 + 0x2^7 + 1x2^8 + 1x2^9$

$= 1x1 + 0 + 0 + 1x8 + 1x16 + 1x32 + 0 + 0 + 1x256 + 1x512$

$= 1 + 0 + 0 + 8 + 16 + 32 + 0 + 0 + 256 + 512$

$= 825_{10}$

Or taking 1471_8 and representing each digit by its binary representation, we get 1=001, 4=100, 7=111 & 1 = 001 which when put together equal 001100111001 Notice this is the same bit pattern as when we converted from base 10 to base 2. Now if we group this into three groups of four bits and then convert each group to its hex counterpart, we will have the number of 825_{10} represented in hex. 001100111001 = 0011 0011 1001 = 339_{16} . Notice this agrees with the result when we converted directly to hex fron one base 10 number.

In summary, lets take the situation when an MPU 6800 8 bit register contains all 1's.

$11111111 = 1x2^0+1x2^1+1x2^2+1x2^3+1x2^4+1x2^5+1x2^6+ + 1x2^7$

$= 1x1 +1x2+1x4+1x8+1x16+1x32+1x64+1x128$

$= 1 + 2 + 4 + 8 + 16 + 32 + 64 + 128$

$= 255_{10}$

or

$11111111 = 11\ 111\ 111$

$= 3\ \ 7\ \ 7_8 = 7x8^0 + 7x8^1 + 3x8^2$

$= 7x1 + 7x8 + 3x64$

$= 7 + 56 + 192$

$= 255_{10}$

or

11111111 = 1111 1111

F F$_{16}$ $=15\times16^{0} + 15\times16^{1}$

$=15\times1 + 240$

$=15 + 240$

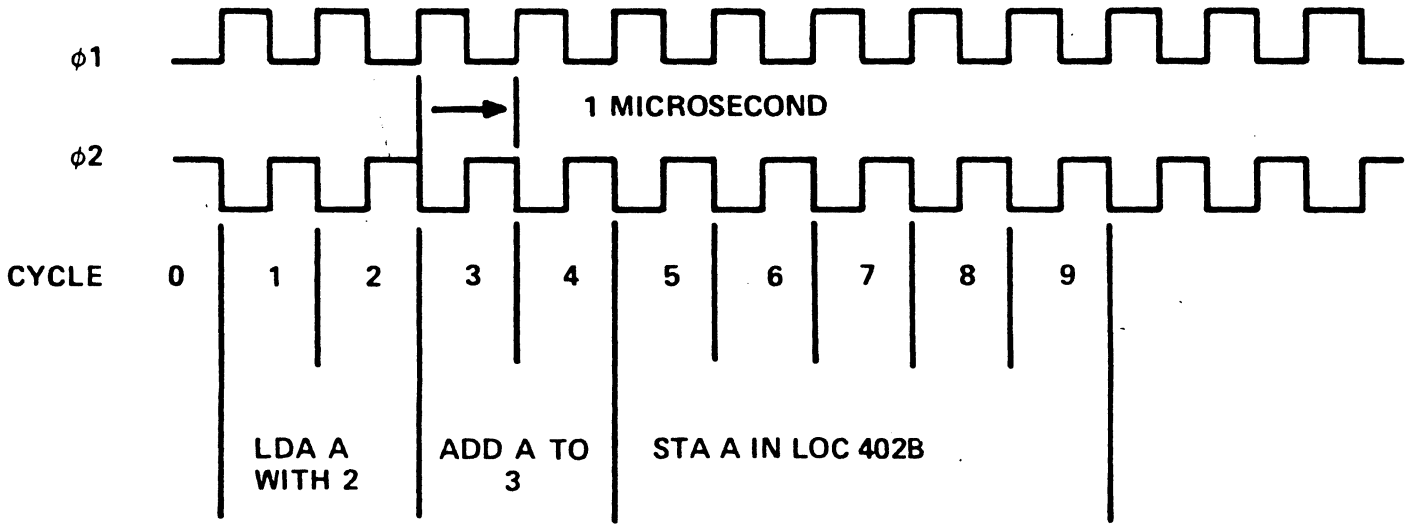
$=255_{10}$

Conversion Chart

Decimal	Octal	hexadecimal	binary
0	0	0	0000 0000
1	1	1	0000 0001
2	2	2	0000 0010
3	3	3	0000 0011
4	4	4	0000 0100
5	5	5	0000 0101
6	6	6	0000 0110
7	7	7	0000 0111
8	10	8	0000 1000
9	11	9	0000 1001
10	12	A	0000 1010
11	13	B	0000 1011
12	14	C	0000 1100
13	15	D	0000 1101
14	16	E	0000 1110
15	17	F	0000 1111
16	20	10	0001 0000
17	21	11	0001 0001
18	22	12	0001 0010
19	23	13	0001 0011
20	24	14	0001 0100
21	25	15	0001 0101

22	26	16	0001 0110
23	27	17	0001 0111
24	30	18	0001 1000
25	31	19	0001 1001
26	32	1A	0001 1010
27	33	1B	0001 1011
28	34	1C	0001 1100
29	35	1D	0001 1101
30	36	1E	0001 1110
31	37	1F	0001 1111
32	40	20	0010 0000
33	41	21	0010 0001
34	42	22	0010 0010
35	43	23	0010 0011
36	44	24	0010 0100
37	45	25	0010 0101
38	46	26	0010 0110
39	47	27	0010 0111
40	50	28	0010 1000

CYCLE BY CYCLE DESCRIPTION OF SAMPLE PROGRAM



ROM ADDRESS	ROM CONTENT	INSTRUCTION
0018	86	LDA A #2
0019	02	
001A	8B	ADD A #3
001B	03	
001C	F6	STA A $402B
001D	40	
001E	2B	

# INDICATES IMMEDIATE MODE OF ADDRESSING

$ INDICATES A HEX NUMBER

NOTE: ADDRESS 402B MUST BE A RAM, PIA, OR ACIA.

DESCRIPTION OF PROGRAM: The A register is loaded with the number 2. Then the number 3 is added to the 2 in the A register with the result of 5 left in the A register. The 5 in the A register is then stored in location 402B.

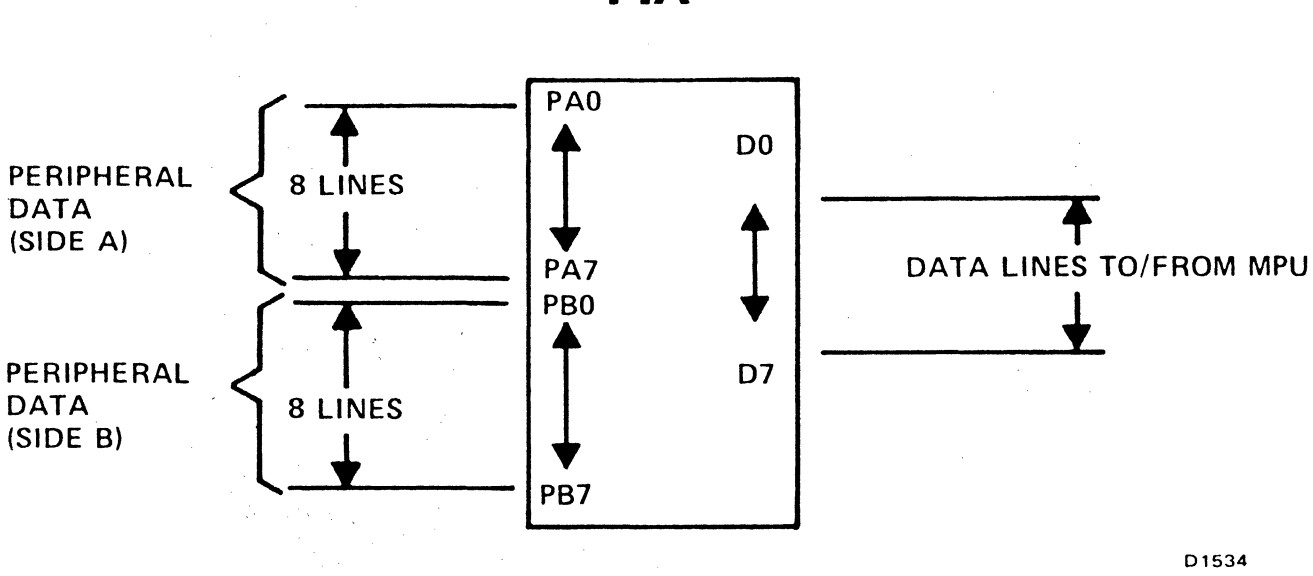
Cycle By Cycle Description of Sample Program

Cycle	Description
0	The program counter is assumed to be set at 0018.
1	The program counter is gated onto the Address Bus (A0-A15) and the read/write (R/W) line is put in a high state corresponding to a read condition. This results in ROM address 0018 be accessed and the contents of this address (86) being loaded into the instruction register (IR). The program counter is then incremented and becomes 0019.
2	The byte "86" in the IR is decoded and interpreted to be a load A immediate (LDA A IMM) instruction. Simultaneously, the program counter is gated onto the Address Bus and the R/W line is set high corresponding to a read condition. This accesses ROM address 0019 with the contents of this address (02) being put on the Data Bus (D0-D7). Since the instruction was decoded to be a LDA A immediate, the "02" is loaded into the A register. The program counter is then incremented and becomes 001A.
3	The sequence in (1) is repeated except ROM address 001A is accessed resulting in 8B being loaded into the instruction register. The program counter is incremented to 001B.
4.	The sequences in (2) is repeated except the instruction is decoded to be an ADD A immediate. Thus, the data "03" is added to the A register giving a result in the A register of "05". The program counter is incremented to 001C.
5	The sequences in (1) is repeated which results in F6 being loaded into the instruction register. The program counter is incremented to 001D.
6	The instruction register is decoded and determined to be a STA A extended. This causes the MPU to interpret the next two sequential locations in memory (D01D & 001E) as a 16 bit address with 001D the most significant and 001E the L.S. half of the address. Simultaneously, the number in ROM address 001D is read by the MPU and saved the program counter is incremented to 001E.
7	The contents of ROM address 001E (2B) is read by the MPU and saved. The MPU now has a full 16 bit address saved of 402B.
8	The extended address of 402B is gated onto the address bus register.
9	Address 402B is accessed and the R/W line is put in a low state, corresponding to a write. The data in the A register is then gated onto the data bus and stored in location 402B.

Peripheral Interface Adapter (PIA) - MC6820

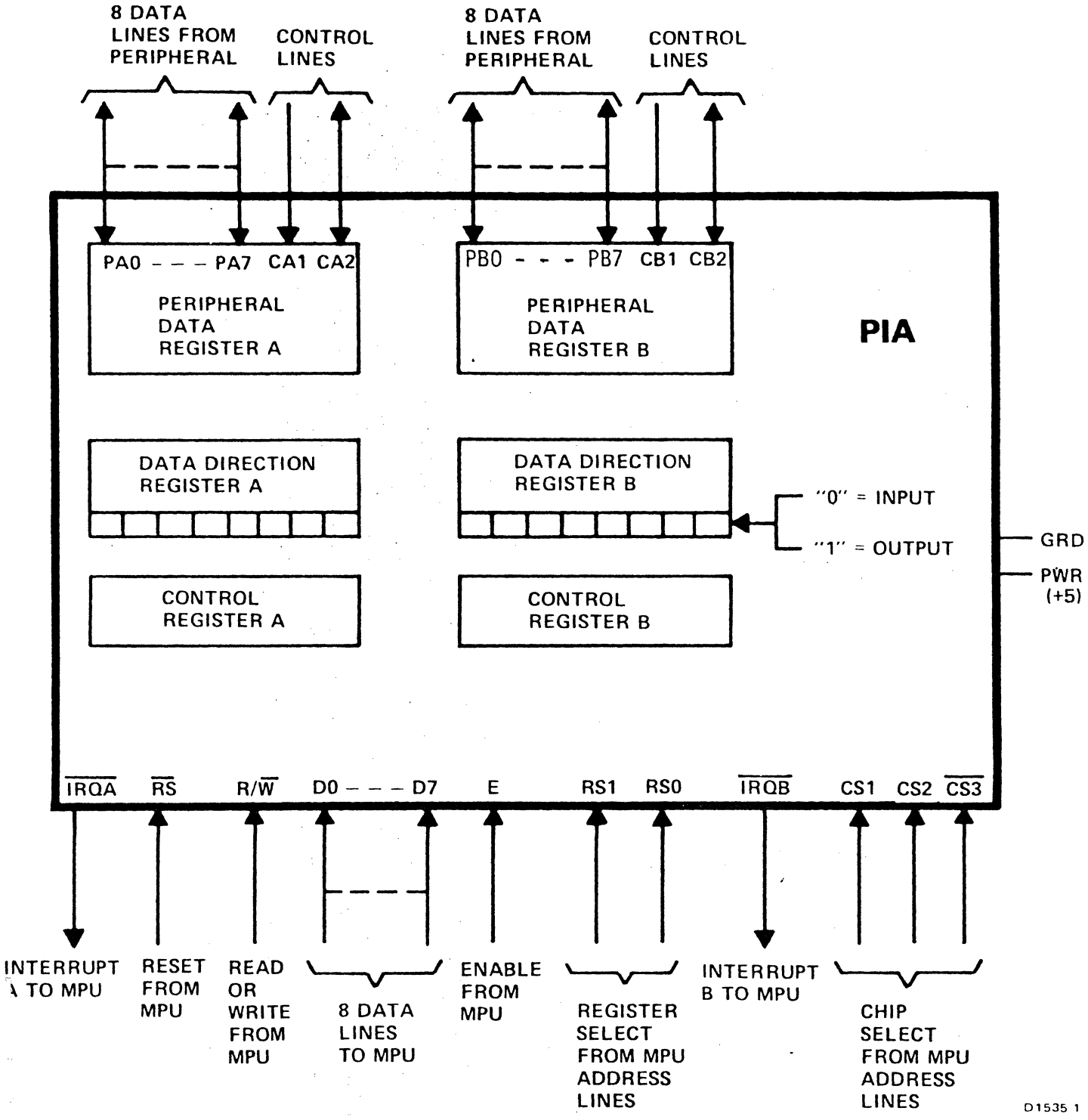
The Peripheral Interface Adapter (PIA) is a means used to interface peripheral equipment with the microprocessing unit (MPU). The PIA communicates with the MPU via an eight bit bi-directional data bus, three chip selects, two register selects, two interrupt request lines, one read/write line, an enable line, and a reset line. These will be discussed in detail later.

Each PIA has two eight bit bi-directional peripheral data buses for interfacing with peripheral equipment as shown in Figure 1.



Each Peripheral data line may be programmed to act as an input or an output. In addition to the two eight bit peripheral data buses, peripheral control lines CA2 and CB2 may be programmed to act as a peripheral data line as will be discussed later.

Each PIA consists of two control registers, two data direction registers, and two peripheral interface registers (peripheral data). The control registers and the data direction registers are used to control the data in and out of the PIA.


8 DATA LINES FROM PERIPHERAL
CONTROL LINES
8 DATA LINES FROM PERIPHERAL
CONTROL LINES
PA0 - - - PA7 CA1 CA2
PERIPHERAL DATA REGISTER A
PB0 - - - PB7 CB1 CB2
PERIPHERAL DATA REGISTER B
PIA
DATA DIRECTION REGISTER A
DATA DIRECTION REGISTER B
"0" = INPUT
"1" = OUTPUT
GRD
PWR (+5)
CONTROL REGISTER A
CONTROL REGISTER B
IRQA RS R/W D0 - - - D7 E RS1 RS0 IRQB CS1 CS2 CS3
INTERRUPT A TO MPU
RESET FROM MPU
READ OR WRITE FROM MPU
8 DATA LINES TO MPU
ENABLE FROM MPU
REGISTER SELECT FROM MPU ADDRESS LINES
INTERRUPT B TO MPU
CHIP SELECT FROM MPU ADDRESS LINES
D1535 1

A. Peripheral Data Lines PAØ thru PA7

Each of these 8 data lines which interface with the outside world can be programmed to act as either an input or an output. This is accomplished by setting a "1" in the corresponding bit in the Data Direction Register (DDR) if the line is to be an output or a "0" in the DDR if the line is to be an input. When the data in the peripheral data lines are read into the MPU by a load instruction, those lines which have been designated as input lines (0 in DDR) will be gated directly to the data bus and into the register selected in the MPU. In the input mode, each line represents a maximum of one standard TTL load.

On the other hand, when an output data instruction (STA A PIA) is executed, data will be transferred via the data bus to the peripheral data register. A "1" output will cause a "high" on the corresponding data line and a "0" output will cause a "low" on the corresponding data line. Data in Peripheral Register A that have been programmed as outputs may be read by an MPU "LDA A from PIA" instruction. If the voltage is above 2 volts for a logic "1" or below .8 volts for a logic "0", the data will agree with that data outputed. However, if these output lines have been loaded such that they do not meet the levels for logic "1", the data read back into the MPU may differ from the data stored in the PIA Peripheral Register A.

B. Peripheral Data Lines PBØ thru PB7

The 8 data lines which interface with the outside world on the B side may also be programmed to act either as an input or as an output. This is also accomplished by setting a "1" in the corresponding bit in the Data Direction Register (DDR) if the line is to be an output or a "0" in the DDR if the line is to be

an input. The output buffers driving these lines have three state capability, allowing them to enter a high impedance state when the peripheral data line is used as an input. Data in Peripheral Register B that have been programmed as outputs may be read by an MPU "LDA A from PIA" instruction even though the lines have been programmed as outputs. If the line has been programmed as an output ("1"), reading the line will indicate a logic "1" due to buffering between the register and the output pin.

C. Data Lines (DØ-D7)

The 8 bi-directional data lines permit transfer of data to/from the PIA and the MPU. The MPU receives data from the outside world from the PIA via these 8 data lines or sends data to the outside world through the PIA's via the 8 data lines. The data bus output drivers are three state devices that remain in the high impedance (off) state except when the MPU performs a PIA read operation.

D. Chip Select Lines (CS1, CS2, $\overline{CS3}$)

These are the lines which are tied to the address lines of the MPU. It is through these lines that a particular PIA is selected (addressed). For selection of a PIA, the CS1 and CS2 lines must be high and the $\overline{CS3}$ must be low. After the chip selects have been addressed, they must be held in that state for the duration of the E (enable) pulse, which is the only timing signal supplied by the MPU to the PIA. This enable pulse (E) is normally the Ø2 clock. One of the address lines should be ANDed with the VMA linewith this output tied to a chip select.

E. Enable Line (E)

The enable pulse (E) is the only timing signal that is supplied to the PIA by the MPU. Timing on all other signals is referenced to the leading or trailing edges of the E pulse.

F. Reset Line (RS)

This line is used to reset all registers in the PIA to a logical zero. This would be used primarily during a reset or power on operation. This line is normally in the high state. The transition of high to low to high resets all registers in the PIA.

G. Read/Write Line (R/W)

This signal is generated by the MPU to control the direction of the data transfers on the Data Bus. A low state on the PIA Read/Write line enables the input buffers and data is transferred from the MPU to the PIA (MPU write) on the E signal if the device has been selected. A high on the Read/Write line sets up the PIA for a transfer of data to the data bus (MPU read). The PIA output buffers are enabled when the proper address & the enable pulse are present thus transferring data to the MPU.

H. Interrupt Request Lines ($\overline{\text{IRQA}}$ & $\overline{\text{IROB}}$)

These lines are used to interrupt the MPU either directly or indirectly through interrupt priority circuitry. These lines are "open source" (no load device on the chip) and are capable of sinking a current of 1.6 ma from an external source. This permits all interrupt request lines to be tied together in a "wired OR" configuration. Interrupts are serviced by a software routine that sequentially reads & tests, on a prioritized basis, the two control registers in each PIA for interrupt flag bits (Bit 6& 7) that are set. Discussion on the control registers & how the flag bits get set will follow. When the MPU reads the Peripheral Data Register, the Interrupt Flag (Bit 6 or Bit 7) is cleared & the Interrupt Request is cleared.

These request lines ($\overline{IRQA}$ & $\overline{IRQB}$) are active low.

I. Interrupt Input Lines (CA1 & CB1)

These lines are input only to the PIA and set the interrupt flag (Bit 7) of the control registers in the PIA. Discussion of these lines in conjunction with the control register will follow.

J. Peripheral Control Line (CA2)

This line can be programmed to act either as an interrupt input or as a peripheral output. As an output, this line is compatible with standard TTL and as an input represents one standard TTL load. The function of this line is programmed with Control Register A (Bits 3,4,&5).

K. Peripheral Control Line (CB2)

This line may also be programmed to act as an interrupt input or as a peripheral output. As an input, this line has greater than 1 megohm input impedence & is compatible with standard TTL. As an output, it is compatible with standard TTL and may also be used as a source of up to 1 millamp at 1.5 volts to directly drive the base of a transister switch. The function of this line is programmed with Control Register B (Bits 3,4, & 5).

CONTROL REGISTER A (CRA)

7	6	5	4	3	2	1	Ø
IRQA1	IRQA2	CA2 CONTROL			DDRA	CA1 CONTROL	

CA1 Control (Bit Ø & 1)

Peripheral control line CA1 is an input only line which may be used to cause an interrupt by setting the interrupt flag IRQA1 (Bit 7) of control register A(CRA). Bits Ø and 1 of CRA are used to determine how the interrupt is to be handled.

After the MPU reads Peripheral Data Register A, the IRQA1 (Bit 7) will be cleared.

Transition of interrupt input line CA1	Status of Bit 1 in CRA	Status of Bit Ø in CRA	IRQA1 (Interrupt flag) Bit 7 of CRA	STATUS OF $\overline{\text{IRQA}}$ LINE (MPU INTERRUPT REQUEST)
⌐‾_	0	0	1	MASKED (Remains High)
⌐‾_	0	1	1	GOES LOW (Processor Interrupted)
_⌐‾	1	0	1	MASKED (Remains High)
_⌐‾	1	1	1	GOES LOW (Processor Interrupted)

(All other combinations of CA1 transition and status of bit 0 and bit 1 will be ignored)

Data Direction Access Control (DDRA)-(Bit 2)

This bit, in conjunction with the register select lines (RSØ & RS1), is used to select either the Peripheral Data Register or the Data Direction Register. To address the A side control register, RS1 is set to a logic "0" and RSØ is set to a logic "1".

RS1	RSØ	CRA(BIT2)	Register Selected
Ø	Ø	1	Peripheral Data Reg.A
Ø	Ø	Ø	Data Dir. Reg. A
Ø	1	–	Control Reg. A

CONTROL REGISTER B (CRB)

7	6	5	4	3	2	1	Ø
IRQB1	IRQB2	CB2 CONTROL			DDRB	CB1 CONTROL	

CB1 Control (Bit Ø & 1)

Peripheral control line CB1 is an input only line which may be used to cause an interrupt by setting the interrupt flag IRQB1 (Bit 7) of control register B (CRB). Bits Ø and 1 of CRB are used to determine how the interrupt is to be handled.

After the MPU reads Peripheral Data Register B, the IRQB1 (Bit 7) will be cleared.

Transition of interrupt input line CB1	Status of Bit 1 in CRB	Status of Bit Ø in CRB	IRQB1 (Interrupt flag)Bit 7 of CRB	Status of $\overline{\text{IRQB}}$ Line(MPU Interrupt Request)
⌐	0	0	1	MASKED (Remains High)
⌐	0	1	1	GOES LOW (Processor Interrupted)
⌐	1	0	1	MASKED (Remains High)
⌐	1	1	1	GOES LOW (Processor Interrupted)

(All other combinations of CB1 transition and status of bit 0 and bit 1 will be ignored)

Data Direction Access Control (DDRB)-Bit 2)

This bit, in conjunction with the register select lines (RSØ & RS1), is used to select either the Peripheral Data Register or the Data Direction Register. To address the B side control register, RS1 is set to a logic "1" and RSØ is set to a logic "1".

RS1	RSØ	CRB(BIT2)	Register Selected
1	Ø	1	Peripheral Data Reg.B
1	Ø	Ø	Data Dir. Reg. B
1	1	—	Control Reg. B

CA2 Control (Bit 3,4, & 5 of CRA)

This line, in addition to generating an interrupt signal, may also be used as an additional output signal. Bits 3,4, & 5 of the control register determine the function of this line.

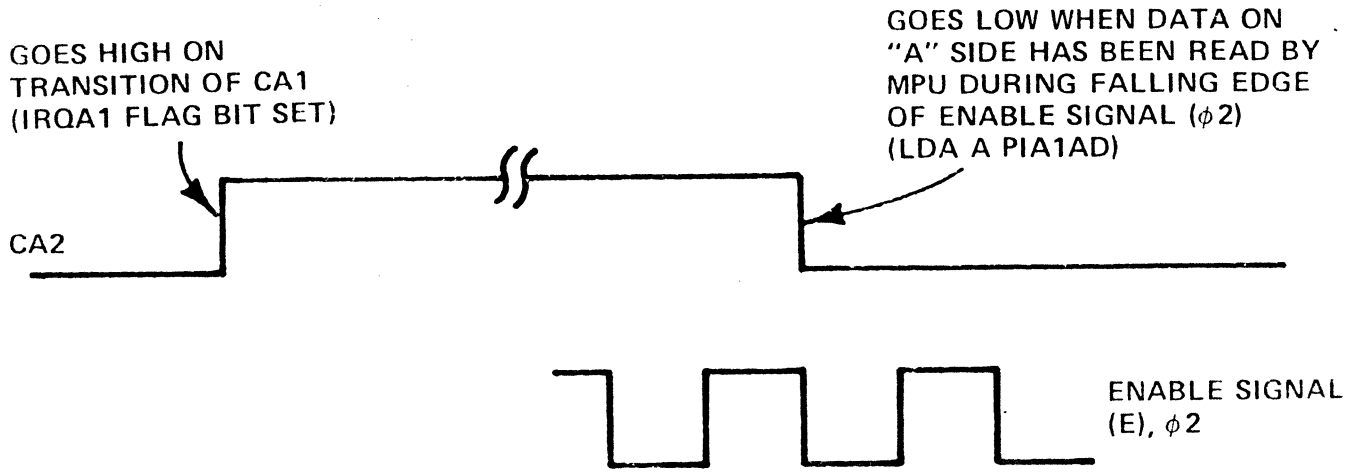
Transition of input CA2	Status of Bit 5 in CRA	Status of Bit 4 in CRA	Status of Bit 3 in CRA	IRQA2 (Interrupt flag)bit 6 of CRA	Status of $\overline{IRQA}$ Line (MPU interrupt request)
⌐	Ø	Ø	Ø	1	MASKED (remains high)
⌐	Ø	Ø	1	1	GOES LOW (Processor interrupted)
⌐	Ø	1	Ø	1	MASKED (remains high)
⌐	Ø	1	1	1	GOES LOW (processor interrupted)

(All other combinations of CA2 transition and status of bit 3 and bit 4 will be ignored)

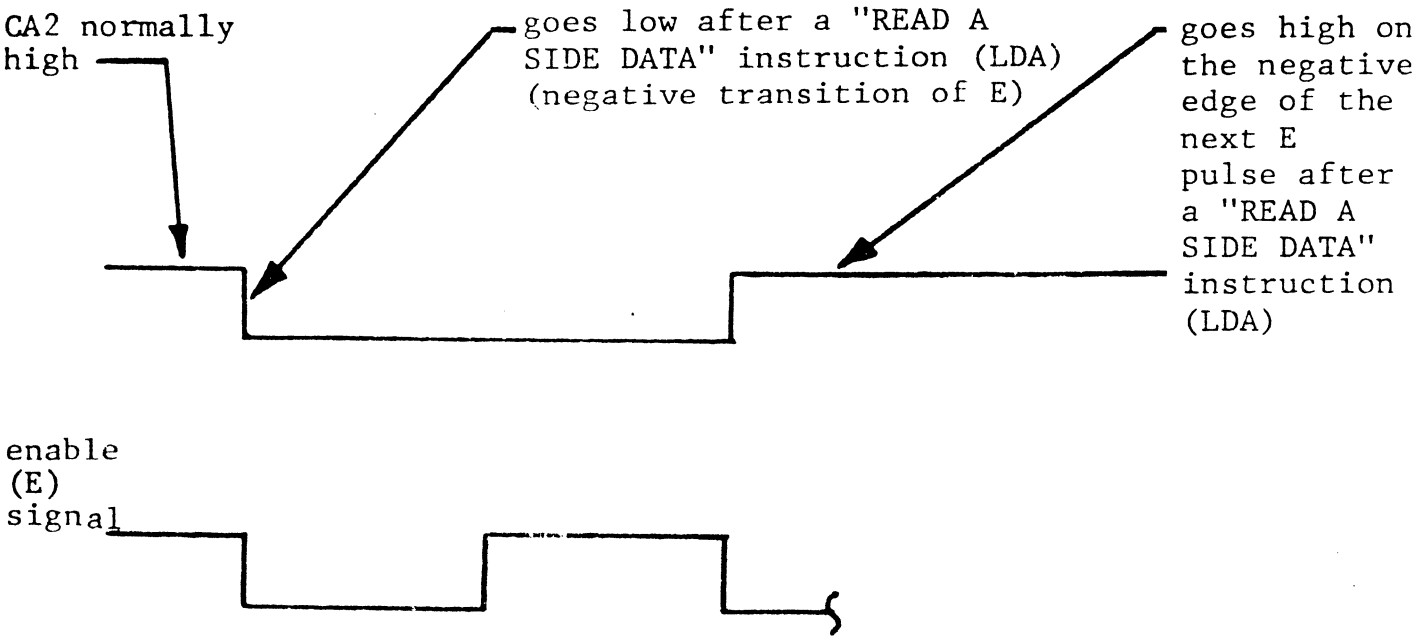
CA2 Used As An Output

If bit 5 of CRA is set to a logic "1", CA2 is designated as an output. The four options utilizing CA2 as an output are shown below. In all four options the IRQA2 flag (bit 6 of CRA remains clear and the $\overline{IRQA}$ interrupt request line remains high.

Bit 5, 4, 3 of CRA = 100 (Handshake Mode)



BIT 5,4, 3 of CRA = 101



BIT 5,4,3 of CRA = 110

CA2 will always be low with Bits 5,4, & 3 equal to 110

Bit 5,4,3 of CRA = 111

CA2 will always be high with Bits 5, 4, and 3 equal to 111.

CB2 Control (Bit 3,4, & 5 of CRB)

This line, in addition to generating an interrupt signal, may also be used as an additional output signal. Bits 3,4, & 5 of the control register determine the function of this line.

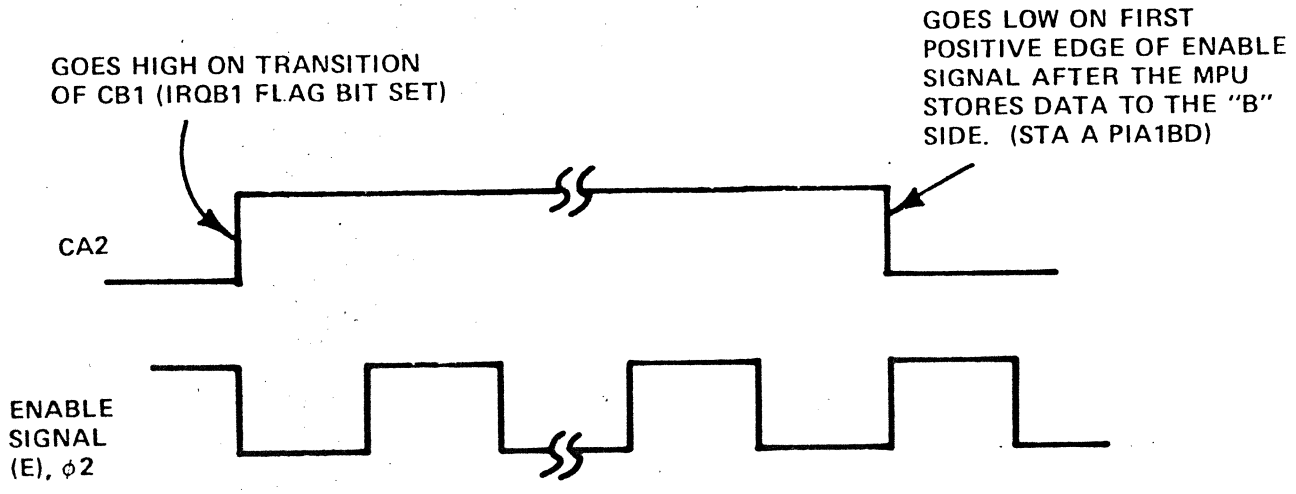
Transition of input CB2	Status of Bit 5 in CRB	Status of Bit 4 in CRB	Status of Bit 3 in CRB	IROB2 (Interrupt flag) bit 6 of CRB	Status of IRQB Line (MPU interrupt request)
⌐	Ø	Ø	Ø	1	MASKED (remains high)
⌐	Ø	Ø	1	1	GOES LOW processor interrupted
⌐	Ø	1	Ø	1	MASKED (remains high)
⌐	Ø	1	1	1	GOES LOW (processor interrupted

(All other combinations of CB2 transition and status of bit 3 and bit 4 will be ignored.)

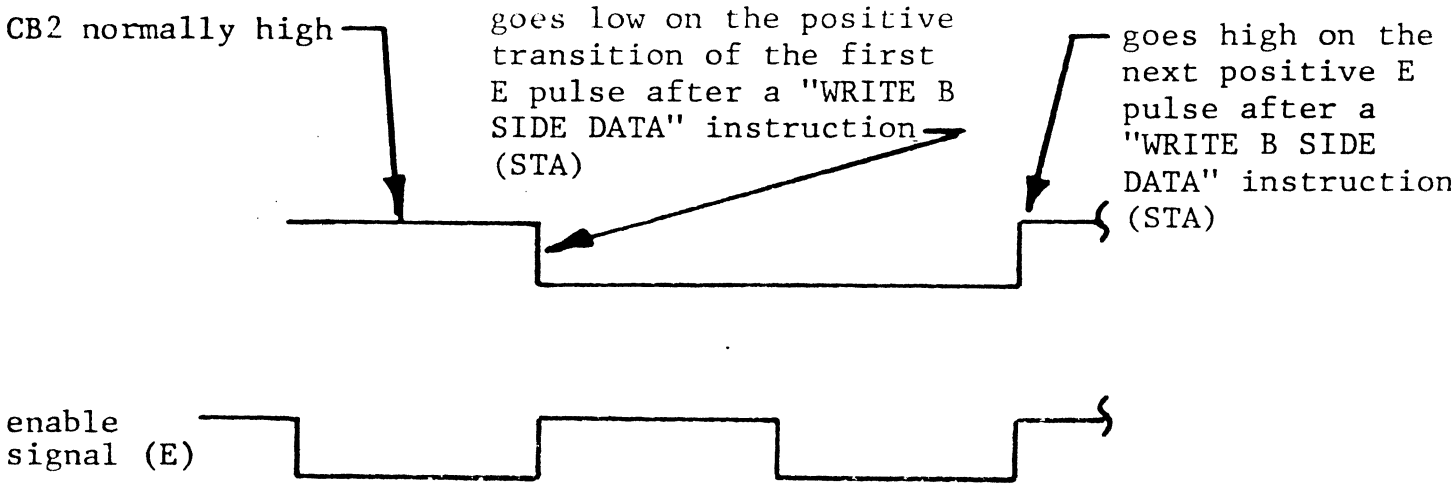
CB2 Used as an Output

If bit 5 of CRB is set to a logic "1", CB2 is designated an an output. The four options utilizing CB2 as an output are shown below. In all four options, the IRQB2 flag (bit 6 of CRB) remains clear and the $\overline{\text{IRQB}}$ interrupt request line remains high

Bit 5, 4, 3 of CRB = 100 (Handshake Mode)



Bit 5,4,3 of CRB = 101



Bit 5,4,3 of CRB = 110

CB2 will always be low with Bits 5,4, & 3 of CRB is equal to 110.

Bit 5,4,3 of CRB = 111

CB2 will always be high when bits 5,4, & 3 of CRB is equal to 111.

SUMMARY OF PIA CONTROL REGISTERS:

a) Register selects RSØ & RS1

If RS1 is set to a logic "0", then "A" side is selected

If RS1 is set to a logic "1", then the "B" side is selected.

If RSØ is set to a logic "0", and CRA (or CRB) Bit 2 is set to a logic "1", the peripheral data register is selected.

If RSØ is set to a logic "0", and CRA (or CRB) Bit 2 is set to a logic "0", then the data direction register is selected.

If RSØ is set to a logic "1", the control register is selected.

b) CA1 or CB1 Interrupt Line

If bit 0 of CRA (or CRB) is set to a logic "0", all interrupts caused by CA1 (or CB1) are disallowed by the PIA.

c) CA2 or CB2 Interrupt Line

If bit 3 of CRA (or CRB) is set to a logic "0", all interrupts caused by CA2 (or CB2) are disallowed by the PIA. If bit 5 of CRA (or CRB) is set to a logic "1", then the CA2 (or CB2) line is used as an output line per previous table.

Summary of Control Registers CRA & CRB

Control Registers CRA & CRB have total control of CA1, CA2, CB1, and CB2 lines. The status of eight bits of the control registers may be read into the MPU. However, the MPU can only write into bit Ø thru bit 5 (6 bits), since bit 6 & bit 7 are set only by CA1, CA2, CB1, or CB2.

Addressing PIA's

Before addressing PIA's, the Data Direction (DDR) must first be loaded with the bit pattern that defines how each line is to function i.e. as an input or an output. A logic "1" in the Data Direction Register defines the corresponding line as an output while a logic "0" defines the corresponding line as an input. Since the DDR and the Peripheral Data Lines have the same address, the control register bit 2 determines which register is being addressed. If bit 2 in the control register is a logic "0", then the DDR is addressed. If bit 2 in the control register is a logic "1", the Peripheral Data Register is addressed. Therefore, it is essential that the DDR be loaded first before setting bit 2 of the control register.

Example: Given a PIA with an address of 4004, 4005, 4006, & 4007. 4004 is the address of the A side Peripheral Interface Register. 4005 is the address of the A side control register. 4006 is the address of the B side Peripheral Interface Register. 4007 is the address of the B side control register. On the A side, bit 0, 1, 2, & 3 will be defined as inputs while bit 4,5,6 & 7 will be used as outputs. On the B side, all lines will be used as outputs.

The program to accomplish the above is as follows

```
PIA1AD = 4004
PIA1AC = 4005
PIA1BD = 4006
PIA1BC = 4007
```

```
1.  LDA  A #% 11110000      (4 inputs, 4 outputs)
2.  STA  A    PIA1AD        (loads A DDR)
3.  LDA  A #% 11111111      (All outputs)
4.  STA  A    PIA1BD        (Loads B DDR)
5.  LDA  A #% 00000100      (sets bit 2)
6.  STA  A    PIA1AC        (Bit 2 set in A contr. reg)
7.  STA  A    PIA1BC        (Bit 2 set in B contr. reg)
```

Statement 2 addresses the DDR since the Control Register (Bit 2) has not been loaded. Statement 6 & 7 loads the control registers with bit 2 set, addressing PIA1AD or PIA1BD accesses the Data Register.

ASYNCHRONOUS COMMUNICATIONS INTERFACE ADAPTER (ACIA) - MC6850

The Asynchronous Communications Interface Adapter (ACIA) is a means used to receive and transmit up to eight bits of data for serial data communications. The ACIA communicates with the MPU via an eight bit bi-directional data bus, three chip select lines, one register select line, one interrupt request line, an enable line, and one read/write line.

The ACIA has four registers which may be addressed by the MPU. The Status Register (SR) and the Receiver Data Register (RDR) are "read only" registers in that the MPU cannot write into two registers. The transmit Data Register (TDR) and the Control Register (CR) are "write only" registers in that the MPU cannot read from these registers.

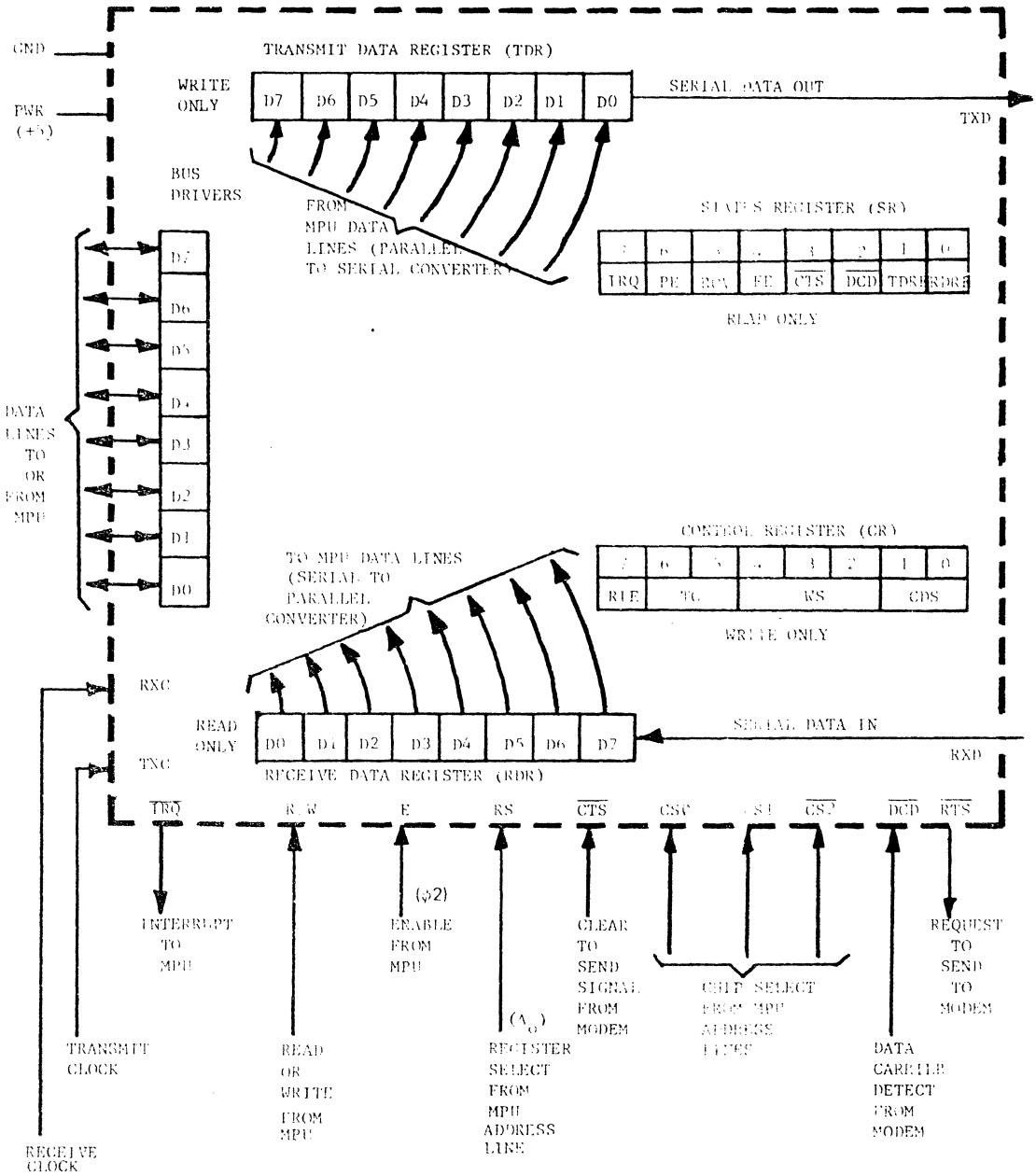
MPU INTERFACE LINES

A. Bi-Directional Data Lines (DØ - D7)

The eight bi-directional data lines permit transfer of data to and from the ACIA and the MPU. The MPU receives data from the outside world from the ACIA via these eight data lines or sends data to the outside world through the ACIA's via the eight data lines. The data bus output drivers are three state devices that remain in the high impedance (off) state except when the MPU performs a ACIA read operation.

B. Chip Select Lines (CSØ, CS1, $\overline{CS2}$)

These are the lines which are tied to the address lines of the MPU. It is through these lines that a particular ACIA is selected (addressed). For selection of an ACIA, the CSØ and CS1 lines must be high and the CS2 must be low. After the chip selects have been addressed, they must be


GND
PWR (+5)
TRANSMIT DATA REGISTER (TDR)
WRITE ONLY
D7 D6 D5 D4 D3 D2 D1 D0
SERIAL DATA OUT
TXD
BUS DRIVERS
FROM MPU DATA LINES (PARALLEL TO SERIAL CONVERTER)
STATUS REGISTER (SR)
7 6 5 4 3 2 1 0
IRQ PE OVRN FE CTS DCD TDRE RDRF
READ ONLY
DATA LINES TO OR FROM MPU
D7 D6 D5 D4 D3 D2 D1 D0
CONTROL REGISTER (CR)
7 6 5 4 3 2 1 0
RIE TC WS CDS
WRITE ONLY
TO MPU DATA LINES (SERIAL TO PARALLEL CONVERTER)
RXC
READ ONLY
D0 D1 D2 D3 D4 D5 D6 D7
SERIAL DATA IN
RXD
TXC
RECEIVE DATA REGISTER (RDR)
IRQ R/W E RS CTS CS0 CS1 CS2 DCD RTS
INTERRUPT TO MPU
(φ2)
ENABLE FROM MPU
CLEAR TO SEND SIGNAL FROM MODEM
CHIP SELECT FROM MPU ADDRESS LINES
REQUEST TO SEND TO MODEM
TRANSMIT CLOCK
READ OR WRITE FROM MPU
(A0)
REGISTER SELECT FROM MPU ADDRESS LINE
DATA CARRIER DETECT FROM MODEM
RECEIVE CLOCK

held in that state for the duration of the E enable pulse, which is the only timing signal supplied by the MPU to the ACIA .

C. Enable Signal (E)

The enable pulse is a high impedance TTL compatible input from the MPU that enables the ACIA input or output buffers and clocks data to or from the ACIA.

D. Read/Write Line (R/W)

The Read/Write line is a high impedance TTL compatible input that is used to control the direction of data flow between the ACIA's eight bit parallel data bus and the MPU. When Read/Write is high (MPU read), the ACIA output driver is turned on and a selected register is read by the MPU. When the Read/Write line is low (MPU write), the ACIA output driver is turned off and the MPU writes into a selected register. Thus, the Read/Write signal, in conjunction with the register select line, is used to select the registers within the ACIA that are read only.

Register Select RS	Read/Write (R/W)	ACIA Register Selected	MPU Read or Write
0	0	Control	Write
0	1	Status	Read
1	0	Transmit Data	Write
1	1	Receive Data	Read

E. Register Select (RS)

The Register Select line is a high impedance TTL compatible input from the MPU that is used to select, in conjunction with the Read/Write line, either the Transmit/Receiver Data Register or the Control/Status register in the ACIA as shown in part D of this section.

F. Interrupt Request Line ($\overline{IRQ}$)

The Interrupt Request Line is a TTL compatible output line to the MPU that is used to interrupt the MPU upon the occurrence of certain events. This line is active in the low state and remains low as long as the course of the interrupt is present and the appropriate interrupt enable within the ACIA is set.

ACIA REGISTERS

A. Status Register (Read Only)

The Status Register can only be read by the MPU. This register is selected when the Register Select (RS) line is low and the Read/Write (R/W) line is high (RS · R/W = 01)

STATUS REGISTERS (SR)

7	6	5	4	3	2	1	Ø
IRQ	PE	ROV	FE	$\overline{CTS}$	$\overline{DCD}$	TDRE	RDRF

Bit Ø - Receiver Data Register Full (RDRF)

"1" - The Receiver Data Register is full. When this bit gets set to a logic "1" indicating the Receiver Data Register is full, the IRQ bit (bit 7) gets set also and remains set until the data is read into the MPU.

"0" - The Receiver Data Register has been read by the MPU. The non-destructive read cycle clears the RDRF bit although the data in the Receiver Data Register is retained. If the $\overline{DCD}$ line goes high indicating loss of carrier, the RDRF bit is clamped at logic "0" indicating the contents of the Receiver Data Register are not current.

Bit 1 - Transmit Data Register Empty (TDRE)

"1" - The Transmit Data Register is empty and new data may be transferred.

- IRQ (bit 7 gets set)

"0" - The Transmit Data Register is full

Bit 2 - Data Carrier Detect ($\overline{DCD}$)

"1" - There is no carrier from the modem. When this bit goes to a logic "1" - the IRQ bit (bit 7) of the status register gets set and remains set until the MPU reads the Status Register and the Receiver Data Register.

"0" - The carrier from the modem is present.

Bit 3 - Clear to Send ($\overline{CTS}$)

"1" - The Clear to Send line from the modem is high, thus inhibiting the Transmit Data Register Empty (TDRE) bit. Modem is not ready for data.

"0" - The Clear to Send line from the modem is low. Modem is ready for data.

Bit 4 - Framing Error (FE)

"1" - Framing error indicates that the received character is improperly framed by the start and stop bit and is detected by the absence of the first stop bit. This error indicates a synchronization error, faulty transmission, or a break condition. This error flag is set or reset during the receiver data transfer time. Therefore, this error indicator is present throughout the time that the associated character is available.

"0" - The received character is properly framed.

Bit 5 - Receiver Overrun (ROV)

"1" - Overrun is an error flag that indicates that one or more characters in the data stream were lost. That is, a character or a number of characters were received but not read from the Receiver Data Register (RDR) prior to subsequent being received. The overrun condition begins at the midpoint of the last bit of the second character received in succession without a read of the RDR having occurred. The Overrun does not occur in the Status Register until the valid character prior to Overrun has been read. Character synchronization is maintained during the Overrun condition. The Overrun indication is reset after the reading of data from the Receive Data Register. Overrun is also reset by the Master Reset.

"0" - No Receiver Data Overrun have occurred.

Bit 6 - Parity Error (PE)

"1" - The parity error flag indicates that the number of highs (ones) in the character does not agree with the preselected odd or even parity. Odd parity is defined to be when the total number of ones is odd. The parity error indication will be present as long as the data character is in the RDR. If no parity is selected, then both the transmitter parity generator output and the receiver parity check results are inhibited.

"0" - No parity error occurred.

Bit 7 - Interrupt Request (IRQ)

"1" - There is an interrupt in the ACIA. This bit being high causes the $\overline{IRQ}$ output line to be low. This will be cleared by reading the Status Register and writing into the Transmit Data Register or reading the Receiving Data Register.

"0" - No interrupt present.

B. Control Register (Write Only)

The Control Register can only be written into by the MPU. This register is selected when the Register Select (RS) line and the Read/Write line are both low (RS · R/W = 00)

CONTROL REGISTER (CR)

7	6	5	4	3	2	1	Ø
RIE	Transmitter Control		Word Select			Counter Divide	

Receiver Interrupt Enable

Bit Ø and 1 - Counter Divide Select Bits (CDS)

CR1	CRO	FUNCTION
0	0	÷ 1
0	1	÷ 16
1	0	÷ 64
1	1	Master Reset

Bit 2, 3, 4 - Word Select Bits (WS)

CR4	CR3	CR2	FUNCTION
0	0	0	7 Bit + EP + 2SB
0	0	1	7 Bit + OP + 2SB
0	1	0	7 Bit + EP + 1SB
0	1	1	7 Bit + OP + 1SB
1	0	0	8 Bit + 2SB
1	0	1	8 Bit + 1SB
1	1	0	8 Bit + EP + 1SB
1	1	1	8 Bit + OP + 1SB

EP - Even
OP - Odd
SB - Stop bits

Bit 6 and 5 - Transmitter Control (TC)

CR6	CR5	FUNCTION
0	0	$\overline{RTS}$ = Low, Transmitting Interrupt Disabled (TID)
0	1	$\overline{RTS}$ = Low, Transmitting Enabled (TID)
1	0	$\overline{RTS}$ = High, Transmitting Interrupt Disabled (TID)
1	1	$\overline{RTS}$ = Low, Transmitting Interrupt Disabled (TID) and transmits a Break level on the transmit data output.

Bit 7 - Receiver Interrupt Enable

"1" - Enables interrupts caused by

a) Receiver Data Register Full going high

b) A low to high transition on the Data Carrier Detect signal line

"0" - Cleared by selecting the Receiver Data Register or by resetting the Receiver Interrupt Enable Bit.

CLOCK INPUTS

Separate high impedance TTL compatible inputs are provided for clocking of transmitted and received data. Clock frequencies of 1, 16, or 64 times the data rate may be selected.

A. Transmit Clock (TXC)

The transmit clock input is used for the clocking of transmitted data. The transmitter initiates data on the negative transition of the clock.

B. Receive Clock (RXC)

The Receive Clock input is used for synchronization of received data. The receiver strokes the data on the positive transition of the clock. (In the : 1 mode, the clock and data must be synchronized externally).

MODEM CONTROL

The ACIA includes several functions that permit limited control of a data modem. The functions included are Clear-to-Send, Request-to-Send and Data Carrier Detect.

A. Clear-to-Send ($\overline{CTS}$)

This high impedance TTL compatible input provides automatic control of the transmitting end of a communications link via the modem's "clear-to-send" active low output.

B. Request-to-Send ($\overline{RTS}$)

The Request-to-Send output enables the MPU to control a modem via the data bus. The active state is low.

C. Data Carrier Detected ($\overline{DCD}$)

This high impedance TTL compatible input provides automatic control of the receiving end of a communication link by means of the modem "Data-Carrier-Detect" or "Received-Line-Signal Detect" output. The $\overline{DCD}$ input inhibits and initializes the receiver section of the ACIA when high. A low to high transition of the Data Carrier Detect initiates an interrupt to the MPU to indicate the occurrence of a loss of carrier.

RECEIVED DATA LINE (RX)

The Received Data line is a high impedance TTL compatible input through which data is received in a serial NRZ (Non Return to Zero) format. Synchronization with a clock for detection of data is accomplished internally when clock rates of 16 or 64 times the bit rate are used. Data rates are in the range of 0 to 500Kbps when external synchronization is utilized.

TRANSMITTED DATA LINES (TX)

The Transmit Data Output line transfers serial NRZ data to a modem or other peripheral at the same range of rates as the received data.

MSI 6800 COMPUTER CHASSIS, MODEL CH-1

GENERAL DESCRIPTION

The MSI 6800 Computer Chassis, Model CH-1, is designed to house the MSI Mother Board, Power Supply, and other components which form the complete MSI 6800 Computer System. The chassis kit itself consists of the computer cabinet with lid, front panel switches, mother board mounting hardware, fuse holder, line cord, and wiring.

ASSEMBLY PROCEDURE

Refer to the Chassis Wiring Diagram, Drawing No. 100029, for the general locations of subassemblies within the computer chassis.

( ) Mount the four vinyl feet, ITEM NO. 22, on the bottom side of the chassis, using four 6-32 x 3/4 inch BHM screws, #4 lockwashers, and nuts. The holes for the feet are located approximately 1-1/2 inches from the edge of the chassis.

( ) Mount the two lug terminal strip, ITEM NO. 3, in the left rear corner of the chassis, next to the vinyl foot. Use a 6-32 x 3/4 inch BHM screw, #6 lockwasher, and nut. Place an additional #6 lockwasher between the lug and the chassis to insure a good electrical ground connection to the chassis.

In the following text, the term "attach" means to connect without soldering, while the term "solder" means to connect and solder. Generally, the instruction to "solder" is given only after all wires have been attached to a particular terminal.

( ) Install the fuse holder in the rear of the chassis. The fuse holder mounts in the small diameter hole next to the two square cut-outs on the rear of the chassis. Make sure that the rubber washer supplied with the fuse holder is on the outside of the chassis when the fuse holder is mounted.

( ) Mount the four plastic 1/2 inch spacers in the bottom of the chassis which are used to mount the power supply. Use 6-32 x 1 inch BHM screws, inserted from the bottom side of the chassis. After the screw is inserted, place the 1/2 inch spacer over the screw followed by a NO. 6 lock washer and hexnut.

( ) Remove approximately three inches of the outer insulation from the A.C. power cord. Install the power cord in the rear of the chassis next to the fuse holder. Use the Heyco strain relief bushing, ITEM NO. 22, to secure the power cord in place.

( ) Install the red, latching push button power switch,

with three terminals, in the right-hand front panel position of the chassis.

( ) Install the three additional push button switches in the three holes on the left side of the front panel in the following left to right order: green (RESET), amber (NMI), white (IRQ).

( ) If the optional fan is used, prepare the fan for mounting on the inside rear of the chassis. Use two lengths of 20 gauge wire, one 4 inches and the other 24 inches long. Solder each to one of the terminals on the fan. Cover each terminal on the fan with a 1 inch length of #14 PVC tubing. Lay the fan assembly aside temporarily.

( ) Install the three 1 1/2 inch, plastic, hole bushings in the three large holes on the rear panel of the chassis. The bushings insert from the outside of the chassis.

( ) Solder the green wire of the AC power cord to the grounded terminal of the two lug terminal strip in the left rear corner of the chassis. The grounded lug is the one which is attached to the bolt that secures the terminal strip to the bottom of the chassis.

( ) Solder the black wire of the AC power cord to the end terminal of the fuse holder.

( ) Attach the white wire of the AC power cord to the unused, ungrounded, terminal of the two lug terminal strip in the left rear corner of the chassis.

( ) Attach the 4 inch lead from the fan to the ungrounded terminal of the two lug terminal strip at the rear of the chassis which also terminates the white wire of the AC power cord.

( ) Using 20 gauge wire, cut two lengths each 24 inches long. Solder one of these wires to the unused terminal of the fuse holder. Solder the remaining wire to the two lug terminal strip which terminates the white wire of the AC power cord and one wire from the fan.

( ) Using 6-32 x 3/4 inch BHM screws, #6 lockwashers, and hexnuts, mount the fan on the inside of the chassis rear panel, with the fanguard on the outside. Orient the fan with the wires next to the bottom and outside edge of the chassis with air flow toward the outside of the chassis.

Now, you should have three lengths of 20 gauge wire approximately 24 inches long, one end of each yet to be attached. One of these wires coming from the fan, one from the fuse holder, and one from the two lug terminal strip terminating the white wire of the AC power cord. Lay these three wires along the outer edge of the chassis and direct them towards the front of the chassis where the switches are

located. Instructions for attaching these three wires will be described later.

( ) Using 20 gauge wire, cut three lengths of wire each 15 inches long. Solder one wire to each of the three terminal lugs on the power switch. Note that the lugs on the power switch are numbered one, two and three. Cover each lug with a one inch length of #5 PVC tubing.

( ) Cut four lengths of 24 gauge wire, two 7 inches long, one 9 inches long, and one 11 inches long. Solder a 7 inch wire to Terminal No. 1 of the white IRQ switch. Solder a 9 inch wire to Terminal No. 1 of the amber NMI switch. Solder an 11 inch wire to Terminal No. 1 of the green RESET switch. Attach the remaining 7 inch wire to the unused terminal of the white IRQ switch. Each of the group of three switches should now have wires soldered to Terminal No. 1 and the white IRQ switch should have a wire on the unnumbered terminal.

( ) Using 24 gauge wire, cut two lengths each four inches long. Solder one of these wires to the unnumbered terminal of the white IRQ switch, which already has a 7 inch length of 24 gauge wire attached to it. Attach the other end of this short length of wire to the unnumbered, unused terminal of the center switch. Using the remaining four inch length of wire, solder one end to the unnumbered terminal of the center switch and the other end to the unnumbered terminal of the green RESET switch. You should now have a continuous connection between each of the unnumbered terminals of the three switches.

( ) Before continuing with the chassis wiring procedure, the mother board must be assembled and ready to install in the chassis. The power supply PC board, transformer, and 10 lug terminal strip, which is attached to the primary of the transformer, must be assembled and installed into proper position on the chassis.

( ) Refer to the Chassis Wiring Diagram, Drawing No. 100029, and recheck the transformer primary leads to make sure that they are properly attached to the 10 lug terminal strip.

( ) Attach the wire from the fan to lug No. 12 of the 10 lug terminal strip. This lug should also have an orange wire from the transformer attached to it.

( ) Attach the wire from the fuse holder at the rear of the chassis, to lug No. 19 of the 10 lug terminal strip. This is the last lug on the terminal strip.

( ) Attach the wire from the two lug terminal strip, at the rear of the chassis, to lug No. 10 of the 10 lug terminal stip. This lug should also have a brown wire from the transformer attached to it.

( ) Solder the wire from Terminal No. 3 of the power switch to lug No. 19 of the 10 lug terminal strip. This pin should also have the wire from the fuse holder attached to it.

( ) Attach the wire from Terminal No. 2 of the power switch to lug No. 14 of the 10 lug terminal switch. This lug should also have a green wire from the transformer attached to it.

( ) Attach the remaining wire from Terminal No. 1 from the power switch to lug 18 of the 10 lug terminal strip.

( ) Refer to the Power Supply Schematic Diagram, Drawing No. 100000, and solder jumpers JU-1, JU-2, JU-3, and JU-4 into place, according to the A.C. line voltage to be used. Jumpers JU-1, JU-2, and JU-3 only are used for 115 V.A.C. operation.

The final step in wiring the computer chassis is to attach the wires from the front panel switches, as well as the leads from the power supply PC board, to the mother board. The mother board has two locations for attaching external leads to it, one located on the end of the mother board and the other in the center of the mother board. The power supply leads are attached to the locations in the center of the board. The front panel switches are attached at the locations provided on the end of the mother board. When the mother board is correctly mounted in the chassis, the connections for the front panel switches will be at the front of the chassis. This orients the mother board so that Pin 50 is next to the right hand edge of the chassis and Pin 1 of the mother board is nearer the center of the chassis.

( ) Solder the black 16 gauge wire, coming from Pad 1 of the power supply PCB to the ground bus of the mother board. These wires should be inserted from the bottom side of the mother board and soldered on the top side.

( ) Solder the red 16 gauge wire, coming from Pad 2 of the power supply PCB to the +8 volt bus of the mother board.

( ) Solder the yellow 18 gauge wire, coming from Pad 3 of the power supply PCB to the +12 volt bus of the mother board.

( ) Solder the green 18 gauge wire, coming from Pad 4 of the power supply PCB to the -12 volt bus of the mother board.

The front panel switches must now be attached to the mother board as follows:

( ) Solder the wire from the unmarked terminals of the three front panel switches to the ground bus on the front of the mother board.

( ) Solder the wire from Terminal No. 1 of the white IRQ switch to the Pad identified as IRQ on the mother board.

( ) Solder the wire coming from Terminal No. 1 of the amber NMI switch to the pad on the mother board identified as NMI

( ) Solder the wire from Terminal No. 1 of the green RESET switch to the pad on the mother board identified as M RST.

( ) Install the mother board on the chassis using seven 6-32 x 3/4 inch BHM screws, 1/4 inch plastic spacers, #6 lockwashers, and hexnuts. Refer to the Chassis Wiring Diagram, Drawing No. 100029, for the correct locations of the mounting screws.

This concludes the assembly of the computer chassis.

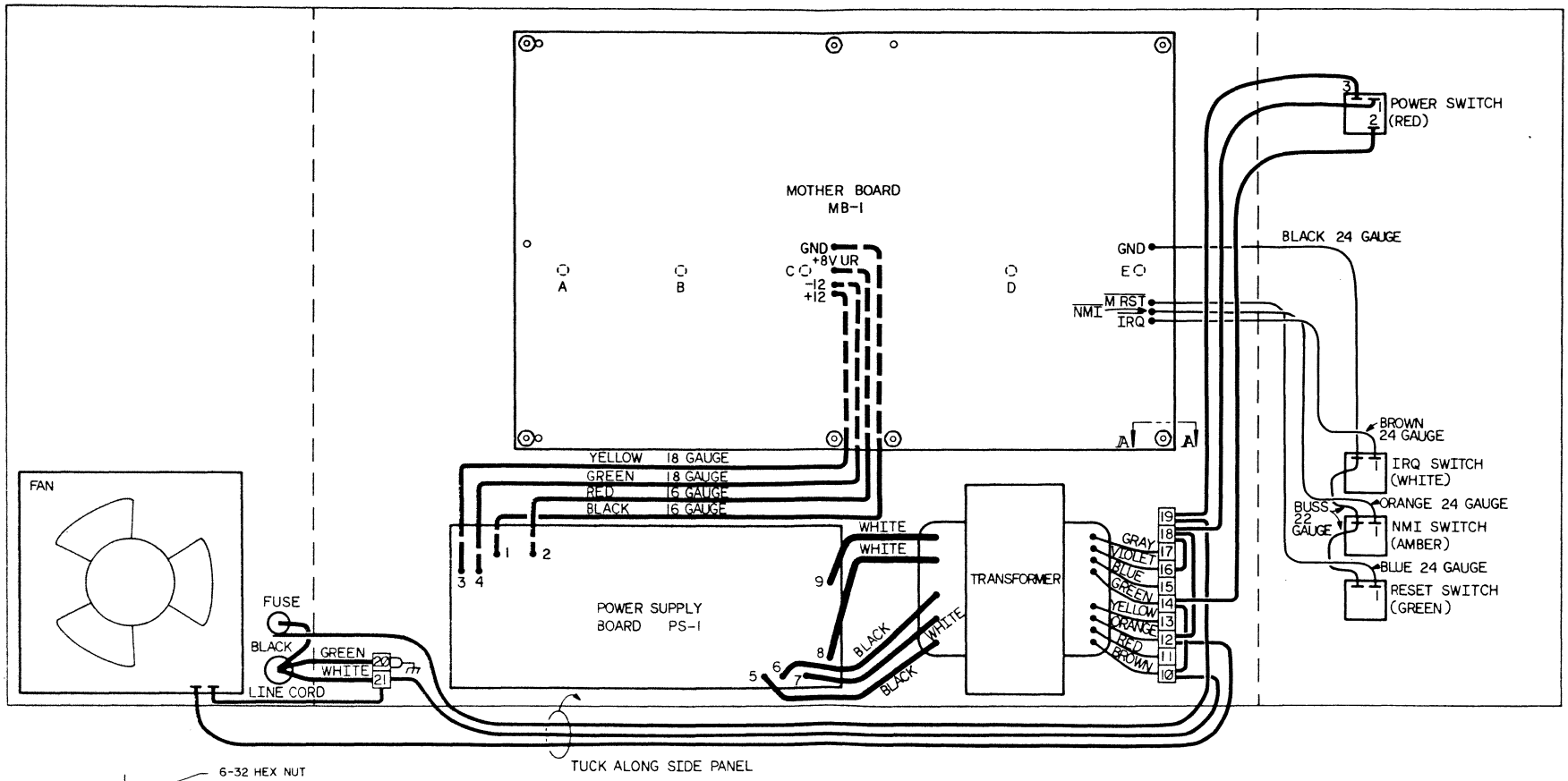
MSI 6800 COMPUTER KIT CHASSIS AND HARDWARE

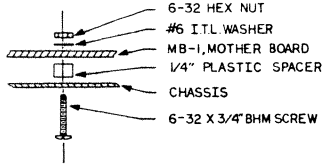
PARTS LIST

ITEM NO.	QUANTITY	DESCRIPTION	MSI PART NO.
1	1	CHASSIS, MSI 6800 Computer	1845
2	1	COVER, MSI 6800 Computer	1842
3	1	TERMINAL STRIP,2Lug/1Grounded Lug	932
4	1	TERMINAL STRIP,10Lug	803
5	1	FUSE, 2Amp Slow Blow	810
6	1	SWITCH, Latching Push Button, Red	933
7	1	SWITCH, Push Button, White	934
8	1	SWITCH, Push Button, Amber	934
9	1	SWITCH, Push Button, Green	934
10	6	SCREW, 4-40 x 3/8", B.H.M.	716
11	3	SCREW, 6-32 x 3/8", B.H.M.	723
12	4	SCREW, 6-32 x 1/2", B.H.M.	724
13	11	SCREW, 6-32 x 3/4", B.H.M.	725
14	18	NUT, 6-32, Hex	721
15	6	WASHER, #4, Flat	740
16	4	WASHER, #6, Flat	741
17	22	WASHER, #6, I.T.L.	745
18	7	SPACER, Plastic, 1/4"	751
19	1	FUSE HOLDER, Panel Mounting	805
20	3	HOLE BUSHING, Plastic, 1 1/2"	927
21	1	BUSHING, Strain Relief, Heyco #6P4	775
22	4	FEET, Vinyl	783
23	3in.	TUBING, P.V.C.,#5	890
24	1	LINE CORD, Beldon	816
25	15in.	WIRE, 16 gauge, Red	1102
26	15in.	WIRE, 16 gauge, Black	1102
27	15in.	WIRE, 18 gauge, Green	1101
28	15in.	WIRE, 18 gauge, Yellow	1101
29	13in.	WIRE, 20 gauge, Orange	1104
30	5ft.	WIRE, 24 gauge, (Blue)	1113

The following parts are optional:

31	1	FAN, 4 1/2"	830
32	1	FAN GUARD, Large	828
33	4	SCREW, 6-32 x 3/4", B.H.M.	725
34	4	WASHER, #6, I.T.L.	745
35	4	NUT, 6-32, Hex	721
36	2in.	TUBING, P.V.C., #14	885





DETAIL A-A

MTG. HARDWARE ASSEMBLY FOR MB-1
TYPICAL 7 PLACES

TOP VIEW

UNFOLDED CHASSIS

1. UNMARKED WIRE IS ORANGE, 20 GAUGE.

NOTES: UNLESS OTHERWISE SPECIFIED

DRWN K Osborn DATE 1-11-78	Midwest Scientific Instruments Olathe, Kansas	
CHK KO / DL DATE 1-12-78	TITLE MSI 6800 COMPUTER CHASSIS WIRING DIAGRAM	
APPD [illegible] DATE 1-17-78		
MATERIAL	SIZE D / MSI PART NO. REF 1845 / DRWG NO. 100029	REV 0
	SCALE 1/1	SHEET 1 OF 1