MicroDaSys

POST OFFICE BOX 36051 • LOS ANGELES, CA 90036 U.S.A. • (213) 935-4555

6809 ADAPTER ASSEMBLY INSTRUCTIONS

Follow Carefully

Insert the 40 pin socket in the holes marked 6809 (toward the center of the board) FROM THE UN-PRINTED SIDE. *Is it the right spot?* If so, solder it in place.

Similarly install the 14 pin socket from the un-printed side and solder in place.

Install a wire on the unprinted side of the board between the two separate round holes, and solder.

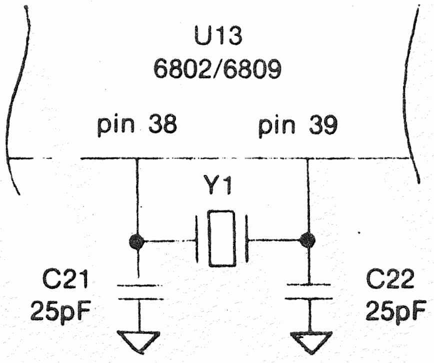
The larger pins of the double-sided carrier will now be soldered to the printed side of the adapter socket board.

Install the carrier from the *printed* side of the board, in the holes marked 6802. *Do not press it all the way in.* Solder from the sides, *being careful not to bridge connections or traces.*

With the 14 pin socket to the right side, install the 74LS32 and the 6809 with pin one to the *upper left.*

OSCILLATOR DAMPING

To improve crystal oscillator response and prevent ringing, connect the two enclosed 25pF capacitators from either side of Y1 to ground, as shown. A convenient ground point is the upper left lead of C4 (with the edge connector toward the bottom of the board as viewed from the component side). Solder both capacitors to this point on the *rear* side of the board and insulate them with tape if necessary.



JUMPER SELECTION

( ) **J1** Cut the existing trace between the middle hole and the upper hole and then jumper the middle hole to the lower hole if your keyboard has a positive active strobe. If using the MicroDaSys keyboard, no alteration is needed.

( ) **J2** Connect the middle hole of each triplet to the lower hole of each triplet.

( ) **J4** For the 2 MHz processor and slower than 250 - 300 ns PROMs, jumper J4 to implement a wait state throughout the top 16K of memory.

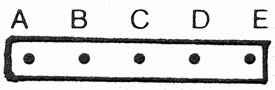
( ) **J5** If an external RAM card is being used from $AOOO to $AFFF, cut J5 and remove U17 and U18. Do not do this until checkout of the board is completed. RAM must be located at $AOOO for MONBUG to function.

( ) **J8** If U19 is a 2716, jumper the following pins:

A to B C to D

If U19 is a 2708, jumper the following pins:

B to C D to E



( ) **J9** Connect the middle hole to the right hole.

( ) Install a small insulated wire jumper from pin 9 of U27 to pin 8 of U24.

For 6809 processors:

( ) Install R17 (2.2K) on the MD690a board. Be sure you are using the **6809** monitor PROM for U19.

( ) **J7** BS (Interrupt Acknowledge) Jumper the middle hole of J7 to the upper hole only.

( ) **J11** DMA/BREQ (D.M.A. Request) If you do not plan to use this input (this is very likely) connect the middle hole of J11 to +5v at pin 20 of U31. Also jumper the left hole to the right hole. If you do plan to use it, it must be pulled up.

( ) **J6, J10** If you intend the use FIRQ (fast interrupt request) make the following change. Cut the trace that goes to the middle hole of the triplets at J10 **on the side toward U26** and jumper this hole to the upper hole of J6. This is the FIRQ, and as with IRQ and NMI it may be jumpered to any of the interrupt lines from the bus. These are the eight holes across the bottom of J10 and also pin 73 of the S100 bus.

CASSETTE CALIBRATION USING MONBUG II

( ) A fully functional processor is assumed. Refer to the User's Manual and execute the following program, recording a 2-3 minute tape.

```
$A100    PO    LDAA #$FF       86 FF
               JSR CASOUT      BD FF62
         P1    LDAA #$CA       86 CA
               JSR CASOUT      BD FF62
               BRA P1          20 F9
```

Play back the tape running this program:

```
$A200    LO    LDX #SCNTOP     8E FO OO
         L1    JSR CASIN       BD FF C3
               STAA X          A7 84
               LEAX 1,X        30 01
               CPX #SCNBOT     8C F4 OO
               BE L1           26 F4
               BRA LO          2O EF
```

Adjust R9 to the middle of the range for which the screen is filled with reverse field J's

( ) Some cassette recorders invert the phase of the playback data. This makes recovery of the valid data from your cassette impossible. AS A LAST RESORT, if it becomes apparent that your casette inverts phase, cut the connection between the middle hole and the right hole of J3 on the back of the board and jumper the middle hole to the left hole. If this does not help, phase inversion was not the problem. Remove the new jumper and re-jumper as before.

- -

□ YES!

ENROLL ME FREE OF CHARGE IN THE MICRODASYS USERS' GROUP. THE USERS' GROUP OFFERS A NEWSLETTER, GENERAL INTEREST SOFTWARE, AND PERIODIC BULLETINS OF INTEREST TO MICRODASYS OWNERS. SOFTWARE CONTRIBUTIONS ARE WELCOMED.

NAME ______________________________

ADDRESS ___________________________

CITY, STATE ________________________

ZIP CODE ___________________________

SEND TO

MICRODASYS
USERS' GROUP EDITOR
P.O. BOX 36051
LOS ANGELES, CA 90036

The MicroDaSys MONBUG II User's Guide

8/21/79

1.0 INTRODUCTION MONBUG II is a simple, interactive monitor with a sophisticated Input/Output structure designed to be upwards expandable for use with disk operating systems and multiple interrupt-driven I/O devices. MONBUG II is designed specifically for use with the MicroDaSys MD-690b 6809 CPU card. The minimal I/O configuration consists of an interrupt-driven keyboard (through PIA located at $F400), memory mapped video board (64x16 located at $F000), and high-speed cassette interface (also through PIA). RSBUG II provides the similar features via an RS-232 serial device, instead of a parallel keyboard and memory-mapped video card. RAM is required from $A000-$A3FF. MONBUG II itself is located at $FC00-$FFFF, although it is position independent with the exception of the interrupt vectors from $FFF2-$FFFF. Control may always be transferred to MONBUG II by hitting RESET.

2.0 COMMANDS MONBUG II has four basic commands. They are:

M - memory examine and change
J - jump to user's program
R - read cassette into RAM
W - write to cassette from memory.

Commands are entered in upper case following the prompt symbol (>). Commands are accompanied by trailing hexadecimal (base 16) numbers. Legal digits are 0-9 and A-F. Hexadecimal numbers are separated by spaces, and command lines are terminated by carriage returns. Leading zeroes may be omitted. Only the last four digits of addresses and the last two digits of data are significant, all else is ignored.

The following characters, obtained by simultaneously depressing the CONTROL key and specified alphabetic operate on command lines:

^H (backspace) - deletes last character entered
^U (cancel line) - deletes current user-typed line.

The all-purpose error message is a question mark (?).

The following terminology is used throughout this documentation:

<addr>   = hexadecimal address.
<return> = carriage return.
<data>   = hexadecimal data.
<space> = space bar.
<comma> = ,
[   ] = contents of brackets are optional.
$ = hexadecimal number follows.
^   = control key depressed sumultaneously.
:   = etc.

Computer-typed information is shown in **bold face**, while user-typed information is not.

2.1 MEMORY EXAMINE AND CHANGE

>M<addr><return>

The <data> at location <addr> is displayed as follows:

<addr> <data><command>

The user may now type one of the following <command>s:

<command>:=
<space> - display next location
<comma> - display previous location
<return> - return to command input mode
<data><command> - enter data at displayed location.

2.2 JUMP

The JUMP command is entered as follows:

>J<addr><return>

Code beginning at location <addr> is executed.

2.3 READ CASSETTE

The READ command is entered as follows:

>R[<addr>]<return>

Data in MONBUG format will be loaded from tape to RAM. If <addr> is specified it will be added to the original location of the data on tape as an offset. The cassette should be in the playback mode before <return> is typed. A checksum error will produce a question mark. A good read will return to the monitor.

2.4 WRITE CASSETTE The WRITE command is entered as follows:

```
>W<addr1><space><addr2><return>
```

Data from RAM starting at address <addr1> and ending at address <addr2> is recorded on cassette.

Recording begins .5 sec after <return> is hit, in case it is desired for the Reader Control line (RC) to start the recorder. Otherwise, the cassette should be running in the record mode before <return> is typed.

3.0 CALLING MONBUG II's I/O ROUTINES The 6809 is an extremely powerful processor, which enables the generation of fully-relocatable object code. This means that absolute addresses are never specified when calling MONBUG II's I/O routines. Instead, all routines are invoked via Software Interrupt number 2 (SWI2). The machine code for SWI2 is $103F. When the processor encounters this instruction, control is transferred to the software interrupt handler in MONBUG II. MONBUG II then looks at the byte following the SWI2 (called the operation postbyte, or "op") to determine the operation desired by the caller. After the specified routine is completed, control returns to the instruction following the operation postbyte (op) except in the case of op2 (see below). MONBUG II currently implements I/O operation codes 0 through 5, although others may easily be added by the user.

The function of the codes are as follows:

op=

0 - Input character from keyboard into A.
1 - Output character from A to video board.
2 - Input a sequence of characters from keyboard.
3 - Convert ASCii hexadecimal string to binary.
4 - Output hexadecimal byte from B to video board.
5 - Output hexadecimal word from D to video board.

A detailed description of each operation follows:

3.0.0

Op0 - Input character from keyboard into A. Whenever interrupts are enabled and a character is typed on the keyboard, that character is immediately read and entered into a dynamically allocated buffer area in RAM called a "circular queue". When a character is needed (i.e. SWI2,op0 is called) it is extracted from the queue. The first characters into the queue are the first extracted. This technique enables the user to type while the system is otherwise occupied (calculating the last digit of pi, for example). Calling SWI2,op0 extracts one character from the queue. If no characters are in the queue, the processor waits until one is available. Whenever the special character control V (^V) is

2.4 WRITE CASSETTE The WRITE command is entered as follows:

```
>W<addr1><space><addr2><return>
```

Data from RAM starting at address <addr1> and ending at address <addr2> is recorded on cassette.

Recording begins .5 sec after <return> is hit, in case it is desired for the Reader Control line (RC) to start the recorder. Otherwise, the cassette should be running in the record mode before <return> is typed.

3.0 CALLING MONBUG II's I/O ROUTINES The 6809 is an extremely powerful processor, which enables the generation of fully-relocatable object code. This means that absolute addresses are never specified when calling MONBUG II's I/O routines. Instead, all routines are invoked via Software Interrupt number 2 (SWI2). The machine code for SWI2 is $103F. When the processor encounters this instruction, control is transferred to the software interrupt handler in MONBUG II. MONBUG II then looks at the byte following the SWI2 (called the operation postbyte, or "op") to determine the operation desired by the caller. After the specified routine is completed, control returns to the instruction following the operation postbyte (op) except in the case of op2 (see below). MONBUG II currently implements I/O operation codes 0 through 5, although others may easily be added by the user.

The function of the codes are as follows:

op=

0	-	Input character from keyboard into A.
1	-	Output character from A to video board.
2	-	Input a sequence of characters from keyboard.
3	-	Convert ASCii hexadecimal string to binary.
4	-	Output hexadecimal byte from B to video board.
5	-	Output hexadecimal word from D to video board.

A detailed description of each operation follows:

3.0.0

Op0 - Input character from keyboard into A. Whenever interrupts are enabled and a character is typed on the keyboard, that character is immediately read and entered into a dynamically allocated buffer area in RAM called a "circular queue". When a character is needed (i.e. SWI2,op0 is called) it is extracted from the queue. The first characters into the queue are the first extracted. This technique enables the user to type while the system is otherwise occupied (calculating the last digit of pi, for example). Calling SWI2,op0 extracts one character from the queue. If no characters are in the queue, the processor waits until one is available. Whenever the special character control V (^V) is typed, all unprocessed characters are cleared from the queue. The I/O status byte (IOSTAT) is located in scratchpad RAM and affects the echoing of characters to the video board. The bits of IOSTAT are numbered from MSB to LSB as B7 through B0. When bit 6 is set, characters typed will be immediately echoed to the screen by the keyboard interrupt service routine. Since the characters are not echoed by op0 itself, unless bit 6 is set, op1 must be called next if the character is to be displayed. Op 0 does not affect any registers except A.

3.0.1

Op1 - Output character from A to video board. The ASCii character in accumulator A is output to the video board. No registers are changed following execution of this routine. The following control characters are recognized by the output routine:

^D - Cursor Down (non-destructive)
^E - Erase to End of Screen
^F - Cursor Left (non-destructive)
^H - Backspace (destructive)
^J - Line Feed
^K - Cursor Up (non-destructive)
^L - Cursor Right (non-destructive)
^M - Carriage Return
^N - Erase to End of Line
^^ - Home and Erase

The console switch (CONSW) is a single byte located in scratchpad memory which affects the way the video output routine interprets control characters. The bits of CONSW are numbered from MSB to LSB as B7 through B0. They are defined as follows:

B7 - If set, ignore all control characters.
B1 - If set, clear screen after home (^^).
B0 - If set, auto line feed after all <return>s.

In addition, the escape key <esc> may be typed, followed by two characters to position the cursor at any x,y location on the screen. Columns (x) are numbered 0 through 63, left to right. Rows (y) are numbered 0 through 15, top to bottom. The coordinates in terms of their character equivalents are as follows:

character	x (column)	y (row)	character	x (column)	y (row)
<space>	0	0	*	10	10
!	1	1	+	11	11
"	2	2	,	12	12
#	3	3	-	13	13
$	4	4	.	14	14
%	5	5	/	15	15
&	6	6	0	16	
'	7	7	1	17	
(	8	8	2	18	
)	9	9	3	19	

char	x	char	x
4	20	J	42
5	21	K	43
6	22	L	44
7	23	M	45
8	24	N	46
9	25	O	47
:	26	P	48
;	27	Q	49
<	28	R	50
=	29	S	51
>	30	T	52
?	31	U	53
@	32	V	54
A	33	W	55
B	34	X	56
C	35	Y	57
D	36	Z	58
E	37	[	59
F	38	\	60
G	39	]	61
H	40	^	62
I	41	_	63

For example, to position the cursor in column 31, row 2 type:

<esc>?"

3.0.2

Op2 - Input a sequence of characters from keyboard. On entry to this routine, the Y register should contain the starting address of the buffer area into which text is to be placed. The calling sequence is as follows:

```
     SWI2              10
                       3F
     op=2              02
  <bufsize-1>        <data>
  <delimeter1>       <del1>
       :            [<  : >]
  <delimeterN>      [<delN>]
       0               00
```

where bufsize is a byte representing the maximum length of characters that can be placed into the buffer. Obviously this cannot be greater than 255. If the number of characters entered exceeds the size of the buffer, the excess will be ignored until a delimeter is reached. Delimeters 1 through N are ASCii characters that, when input from the keyboard, will force a return to the calling routine. As many delimeters as desired may be listed before the zero byte. On return, control is transfered to the opcode following the zero. At this time, the exiting delimeter will be in A, and B will contain the number of characters typed not counting the exiting delimeter. The string typed will be placed into the buffer area terminated by a carriage return ($0D) rather than by the exiting delimeter. CC, X, Y, U and S remain unchanged.

The I/O status byte (IOSTAT) is located in scratchpad RAM and affects the echoing of characters to the video board. When bit 6 is set, characters typed will be immediately echoed to the screen by the keyboard interrupt service routine. When bit 7 is set characters input using the buffer input routine (op2) are echoed to the screen. Setting both bits 6 and 7 results in a double echo when the buffer input routine is called. This routine recognizes control U and control H.

3.0.3

Op3 - **Convert ASCii hexadecimal string to binary.** On entry, Y points to an ASCii hexadecimal string terminated by a non-hex character. On return, X contains the binary value of the string, A contains the non-hex terminating character, and the carry will be set if and only if at least one hex character was encountered. Leading zeroes are ignored, and if more than four hex characters are entered, only the last four are significant.

3.0.4

Op4 - **Output hexadecimal byte from B to video board.** The binary value of B is output to the video board as two ASCii hexadecimal characters. A is destroyed. All other registers are unchanged.

3.0.5

Op5 - **Output hexadecimal word from D to video board.** The binary value of D is output to the video board as four ASCii hexadecimal characters. A is destroyed. All other registers are unchanged.

4.0 MONITOR EXPANSION MONBUG II is upward expandable simply by installing additional PROMs with supplemental features. After initialization and before issuing its prompt, MONBUG II checks address $C000 for the expansion code $69. If it is found execution will be transferred to $C003.

4.1 VECTOR REASSIGNMENT Although all interrupt vectors are located at the top of MONBUG II, interrupt service (with the exception of RESET) is "re-vectored" through scratchpad RAM locations to enable the user to change the service routines. The start of these locations is pointed to by the first two bytes of MONBUG II ($FC00). This vector points to the interrupt vector table, which consists of all interrupt vectors in the following order:

SWIV	Software Interrupt
SWI2V	Software Interrupt 2
SWI3V	Software Interrupt 3
IRQV	Interrupt Request
FIRQV	Fast Interrupt Request
NMIV	Non-Maskable Interrupt
SCR	MONBUG II scratchpad RAM pointer

All MONBUG II scratchpad RAM locations are specified relative to the contents of SCR, with locations both above and below this pointer being used.

5.0 SAMPLE PROGRAM The following program demonstrates some of the powerful capabilities of MONBUG II. While the processor is engaged in a relatively large task (in this case counting to 131,072) the user's rapid typing is automatically serviced. However, in this example we have programmed the processor to accept a character from the queue only once every second. The result is a fast input process supporting a slow output process. In short, multi-tasking.

Here is the program:

```
A000    86 02      START   LDA    #2
A002    8E 00 00   FUBAR   LDX    #0
A005    30 1F      FUD     LEAX   -1,X
A007    26 FC              BNE    FUD
A009    4A                 DEC    A
A00A    26 F6              BNE    FUBAR
A00C    10 3F              SWI2
A00E    00                 FCB    0         this calls op 0
A00F    10 3F              SWI2
A011    01                 FCB    1         this calls op 1
A012    20 EC              BRA    START
```

To enter the program type:

```
>MA000<return>
A000 xx 86<space>
A001 xx 02<space>
    :
    :
A013 xx EC<return>
```

Where xx is some random byte from memory. Save the program on cassette by running the cassette in the record mode and typing:

```
>WA000<space>A013<return>
```

When the prompt returns, stop the cassette and execute the program by typing:

```
>JA000<return>
```

Now type anything as fast as you want. The processor will respond to each character typed, but will not immediately display them. It will, however, eventually display them, unless you exceed the 80 character default buffer length.

While this program may not be very useful, it demonstrates some of the versatility which can be achieved with the amazing 6809 micro-processor and MONBUG II.

SUPPLEMENTAL INFORMATION

************************ MEMORY MAP ************************

THE FOLLOWING IS AN ITEMIZATION OF THE
MEMORY USAGE FOR THE 64K ADDRESS SPACE OF THE MD-690B:

$0000-$9FFF	40K USER RAM
$A000-$A3FF	1K SCRATCHPAD RAM (SUPPLIED ON MD-690B)
$A400-$BFFF	7K USER RAM
$C000-$DFFF	8K USER PROM SPACE (ON MD-690B) (MAY BE USED FOR RAM)
$E000-$E3FF	} DUPLICATE OF F800-FFFF
$E400-$E7FF	
$E800-$EFFF	I/O SIGNALS GENERATED HERE INSTEAD OF MEMORY
$F000-$F3FF	MEMORY MAPPED VIDEO GRAPHICS BOARD
$F400-$F403	PIA (KBD, CASSETTE, SPARE I/O PORTS)
$F404-$F7FF	NOT USED
$F800-$FFF7	MONBUG II in 2708 or 2nd ½ of 2716
$FFF8-$FFFF	INTERRUPT AND RESTART SERVICE VECTORS

*********************** PIA BIT USAGE ***********************

PA0-PA7	KEYBOARD DATA
CA1	KEYBOARD STROBE
CA2	READER CONTROL
PB0	CASSETTE OUTPUT DATA
PB1-PB6	AVAILABLE I/O
PB7	CASSETTE INPUT DATA
CB1	CASSETTE TRANSMIT CLOCK
CB2	CASSETTE RECEIVE CLOCK

******* CASSETTE FORMAT *******

WHEN THE COMMAND W IS HIT TWO HEXADECIMAL ADDRESSES MUST BE INPUT. FOR EXAMPLE: >W4C 332E<return>
A CASSETTE TAPE SHOULD BE RUNNING IN THE RECORD MODE BEFORE RETURN IS TYPED. ALL THE DATA WILL BE RECORDED FROM (IN THIS EXAMPLE) $004C TO $3B2E. EACH RECORD EXCEPT THE LAST WILL BE 70 BYTES LONG AND IN THE FOLLOWING FORMAT:

```
         $FF (FOR PLAYBACK SYNCRONIZATION)
         AABCCDDDDDD...DDDDDDDDDDDE
WHERE:    AA IS THE START CODE $EC9D (2 BYTES)
          B  IS THE NUMBER OF BYTES IN THIS RECORD
          CC IS THE STARTING ADDRESS OF THIS RECORD
          D  IS A DATA BYTE (UP TO 64 BYTES PER RECORD)
          E  IS A CHECKSUM.
```

THIS DATA MAY BE RELOADED AT ANY TIME USING THE READ COMMAND.

WHEN R IS TYPED THE PROCESSOR WILL WAIT FOR A CASSETTE TAPE TO BE PLAYED FROM THE TAPE RECORDER THROUGH THE 2400 BAUD CASSETTE INTERFACE. IF THE PROPER FORMAT IS ENCOUNTERED ON THE TAPE, DATA WILL BE LOADED INTO MEMORY. THIS DATA COULD BE A PROGRAM WHICH THE USER COULD THEN EXECUTE. THIS IS HOW BASIC COULD BE LOADED. WHEN THE TAPE IS FINISHED THE PROCESSOR WILL COME OUT OF THE LOAD MODE BECAUSE SPECIAL CHARACTERS ($ECB9) HAVE BEEN RECORDED AT THE END.

ALSO, CONTROL CAN BE REGAINED BY HITTING RESET. REMEMBER-- NO MATTER WHAT HAPPENS, CONTROL WILL ALWAYS BE RETURNED TO THE MONITOR BY HITTING RESET. IF THERE IS AN ERROR ON THE TAPE A QUESTION MARK (?) WILL BE PRINTED ON THE SCREEN AND CONTROL WILL RETURN TO THE MONITOR. USING THE MICRODASYS 2400 BAUD CASSETTE INTERFACE AND MONBUG FORMATTED TAPE, IT REQUIRES ONLY 19 SECONDS TO LOAD 4096 (4K) BYTES OF MEMORY.

************************ SUMMARY ************************

FOR FURTHER INFORMATION ON THE OPERATION OF THE 6800 MICROPROCESSORS IN PARTICULAR, OR MICROCOMPUTERS IN GENERAL, WE RECOMMEND THE FOLLOWING REFERENCES, AVAILABLE FROM MICRODASYS OR YOUR LOCAL COMPUTER STORE:

AN INTRODUCTION TO MICROCOMPUTING, VOLUME 0 - THE BEGINNER'S BOOK
BY ADAM OSBORNE, PUBLISHED BY ADAM OSBORNE & ASSOCIATES, 1977
SOFTCOVER, $7.50

USING THE 6800 MICROPROCESSOR
BY ELMER POE, PUBLISHED BY HOWARD W. SAMS & CO., 1978
SOFTCOVER, $6.95

M6809 MICROPROGRAMMING MANUAL
PUBLISHED BY MOTOROLA INC., 1975
SOFTCOVER

AN INTRODUCTION TO MICROCOMPUTING, VOLUME 1 - BASIC CONCEPTS
BY ADAM OSBORNE, PUBLISHED BY ADAM OSBORNE & ASSOCIATES, 1976
SOFTCOVER, $7.50

THE INFORMATION PRESENTED IN THIS GUIDE IS OF TWO TYPES: BASIC INFORMATION TO GET YOU STARTED AND ADVANCED INFORMATION ABOUT THE INTERNAL OPERATION OF THE MONITOR. IT WILL ENABLE YOU TO USE YOUR MICROCOMPUTER AND BEGIN LEARNING ABOUT ITS OPERATION AND PROGRAMMING. SOON YOU'LL BE WRITING PROGRAMS FAR MORE COMPLEX THAN YOU'VE DREAMED OF. REMEMBER - THE MORE YOU USE YOUR MICROCOMPUTER THE MORE YOU'LL LEARN ABOUT IT. THE CAPABILITIES OF A COMPUTER ARE ENDLESS. QUITE LITERALLY, THEY DEPEND ONLY ON YOUR PROGRAMMING IMAGINATION.

```
00001
00002                          **********************************
00003                          *                                *
00004                          *           MONBUG II            *
00005                          *   INTERACTIVE MONITOR & I/O    *
00006                          *      PACKAGE FOR THE 6809      *
00007                          *                                *
00008                          **********************************
00009                          *
00010                          * WRITTEN BY ROBERT ALKIRE
00011                          * COPYRIGHT - MICRODASYS
00012                          *  6/79  rev  1.0
00013                          *
00014                                 OPT     LLEN=80
00015                          *
00016                          * COMMAND LIST:
00017                          *  M<ADR> = MODIFY MEMORY
00018                          *  J<ADR> = JUMP TO ADR
00019                          *  R<ADR> = READ FROM CASSETTE
00020                          *  W<ADR ADR> = WRITE TO CASSETTE
00021                          *
00022                          * MONITOR EQUATES
00023                          *
00024            F000        A SCRN    EQU    $F000    TOP OF CRT MEMORY
00025            F400        A PIADA   EQU    $F400    KEYBOARD PIA DATA
00026            F401        A PIASA   EQU    $F401    KEYBOARD PIA STATUS
00027            F402        A PIADB   EQU    $F402    CASSETTE PIA DATA
00028            F403        A PIASB   EQU    $F403    CASSETTE PIA STATUS
00029                          * CURSOR CONTROL CHARACTERS
00030            000B        A UP      EQU    $B       CTL-K
00031            0004        A DOWN    EQU    $4       CTL-D
00032            0006        A LEFT    EQU    $6       CTL-F
00033            000C        A RIGHT   EQU    $C       CTL-L
00034            000D        A CR      EQU    $D
00035            000A        A LF      EQU    $A
00036            0009        A TAB     EQU    $9
00037            001B        A ESC     EQU    $1B      ESCAPE
00038            000E        A EEOL    EQU    $E       CTL-N
00039            0005        A FF      EQU    $5       CTL-E
00040            001E        A HOME    EQU    $1E
00041            0015        A CTLU    EQU    $15      ERASE CURRENT LINE
00042            0016        A CTLV    EQU    $16      ERASE QUEUE BUFFER
00043            0008        A BS      EQU    $8       BACK SPACE
00044                          * MEMORY ALLOCATION
00045            A000        A RAM     EQU    $A000    START OF SCRATCH RAM
00046            A3FF        A MAXRAM  EQU    $A3FF    END OF SCRATCH RAM
00047A FC00                           ORG    $FC00
00048                          * RAM INTERUPT VECTORS
00049            A3F2        A SWIV    EQU    MAXRAM-13
00050            A3F4        A SWI2V   EQU    MAXRAM-11
00051            A3F6        A SWI3V   EQU    MAXRAM-9
00052            A3F8        A IRQV    EQU    MAXRAM-7
00053            A3FA        A FIRQV   EQU    MAXRAM-5
00054            A3FC        A NMIV    EQU    MAXRAM-3
00055            A3FE        A SCR     EQU    MAXRAM-1 SCRATCH PAD MEMORY PTR
00056            FFF3        A OFS     EQU    -13      SCRATCH PTR OFFSET
00057            0050        A QLIM    EQU    80       QUEUE SIZE
00058            0048        A BUFLIM  EQU    72       INPUT BUFFER LIMIT
```

```
00059                FFF1     A STACK   EQU     -2+OFS    SYSTEM STACK
00060                FFF3     A IODEV   EQU     0+OFS     I/O DEVICE FLAGS
00061                FFF4     A IOSTAT  EQU     1+OFS     I/O STATUS FLAGS
00062                FFF5     A CONSW   EQU     2+OFS     CONSOLE SWITCHES
00063                FFF6     A CPTR    EQU     3+OFS     CURSOR POINTER  (COUT)
00064                FFF8     A LCHR    EQU     5+OFS     LAST CHARACTER  (COUT)
00065                FFF9     A XYF     EQU     6+OFS     XY FLAG         (COUT)
00066                FFFA     A IQCNT   EQU     7+OFS     INPUT QUEUE COUNTER
00067                FFFB     A IQPTK   EQU     8+OFS     INPUT QUEUE KBD PTR
00068                FFFC     A IQPTQ   EQU     9+OFS     INPUT QUEUE OUT PTR
00069                FFFD     A IQLIM   EQU     10+OFS    INPUT QUEUE LIMIT
00070                FFFE     A IQBEG   EQU     11+OFS    INPUT QUEUE PTR
00071                0000     A QUEUE   EQU     13+OFS    INPUT QUEUE
00072                0050     A INBUF   EQU     13+OFS+QLIM INPUT BUFFER (INBF)
00073                           * SCRATCH PAD LOCATION
00074                A359     A SCRPD   EQU     MAXRAM-27-QLIM-BUFLIM-OFS
00075                           *
00076                           * SCRATCH PAD POINTER
00077                           *
00078A FC00          A3F2     A         FDB     SWIV      POINT TO VECTORS
00079                           *
00080                           * SWI2 COMMAND VECTORS
00081                           *
00082A FC02          FE90     A SWCMD   FDB     INCH      COMMAND 0
00083A FC04          FD75     A         FDB     COUT      COMMAND 1
00084A FC06          FEAD     A         FDB     INBF      COMMAND 2
00085A FC08          FD28     A         FDB     HEXIN     COMMAND 3
00086A FC0A          FD63     A         FDB     OHX8      COMMAND 4
00087A FC0C          FD5B     A         FDB     OHX16     COMMAND 5
00088                           *
00089                           * SWI 2 COMMAND HANDLER
00090                           *
00091A FC0E AE       6A       A SWI2C   LDX     10,S      GET PCR
00092A FC10 E6       80       A         LDB     0,X+      GET COMMAND VALUE
00093A FC12 AF       6A       A         STX     10,S      RESTORE RETURN ADR
00094A FC14 30       8C EB              LEAX    SWCMD,PCR POINT TO SWI2 LIST
00095A FC17 58                          LSLB              MAKE WORD OFFSET
00096A FC18 AE       85       A         LDX     B,X       GET ROUTINE ADR
00097A FC1A EE       68       A         LDU     8,S
00098A FC1C AF       68       A         STX     8,S       GO TO COMMAND ROUTINE
00099A FC1E 35       BF       A         PULS    cc,A,B,DPR,X,Y,PCEXIT SWI2
00100                           *
00101                           *INPUT INTERUPT VECTOR
00102                           *
00103A FC20 B6       F400     A INPUT   LDA     PIADA     GET KYBD DATA FM PIA
00104A FC23 FE       A3FE     A         LDU     SCR       GET SCRATCH POINTER
00105A FC26 84       7F       A         ANDA    #$7F      PARITY OFF
00106A FC28 81       16       A         CMPA    #CTLV     CTRL V?
00107A FC2A 26       07    FC33         BNE     ENTER
00108A FC2C 6F       5A       A         CLR     IQCNT,U   CLEAR CURRENT QUEUE CTR
00109A FC2E 6F       5B       A         CLR     IQPTK,U   RESET KBD QUEUE PTR
00110A FC30 6F       5C       A         CLR     IQPTQ,U   RESET QUEUE PTR
00111A FC32 3B                  OVFLW   RTI
00112                           *
00113A FC33 E6       5A       A ENTER   LDB     IQCNT,U   GET QUEUE COUNT
00114A FC35 E1       5D       A         CMPB    IQLIM,U   TEST IF LIMIT EXCEDED
00115A FC37 24       F9    FC32         BHS     OVFLW     IGNORE EXTRA
00116A FC39 6C       5A       A         INC     IQCNT,U   UPDATE COUNT
```

```
00117A FC3B AE    5E        A        LDX    IQBEG,U  GET BEGINNING OF QUEUE
00118A FC3D E6    5B        A        LDB    IQPTK,U  GET KEYBOARD POINTER
00119A FC3F A7    85        A        STA    B,X      SAVE CHARACTER
00120A FC41 5C                       INCB            NEXT QUEUE LOCATION
00121A FC42 E1    5D        A        CMPB   IQLIM,U  TEST POINTER .GT. LIMIT
00122A FC44 23    01   FC47          BLS    CIRCLE   CIRCULAR QUEUE
00123A FC46 5F                       CLRB            RESET QUEUE POINTER
00124A FC47 E7    5B        A CIRCLE STB    IQPTK,U  SAVE KBD PTR
00125A FC49 E6    54        A        LDB    IOSTAT,U GET I/O STATUS
00126A FC4B 58                       LSLB            TEST BIT 6 SET
00127A FC4C 17    02B1 FF00          LBSR   OUTX2    OUTPUT CHAR IF 6 SET
00128A FC4F 3B                       RTI
00129                         *
00130                         * START OF MAIN ROUTINE
00131                         *
00132A FC50 CE    A359      A RESET  LDU    #SCRPD   SET SCRATCH PTR
00133A FC53 FF    A3FE      A        STU    SCR      SET CURRENT POINTER
00134A FC56 32    51        A        LEAS   STACK,U  SET MONITOR STACK
00135A FC58 30    8C C5              LEAX   INPUT,PCR POINT TO INPUT ROUTINE
00136A FC5B BF    A3F8      A        STX    IRQV     SET INPUT ROUTINE VECTOR
00137A FC5E 30    8C AD              LEAX   SWI2C,PCR POINT TO SWI2 COMMAND
00138A FC61 BF    A3F4      A        STX    SWI2V    SET SWI2 COMMAND PTR
00139A FC64 8E    0000      A        LDX    #0       CLEAR WORKSPACE
00140A FC67 AF    59        A        STX    XYF,U    CLEAR XY FLAG & IQCNT
00141A FC69 AF    5B        A        STX    IQPTK,U  CLEAR IQPTK & IQPTQ
00142A FC6B 8E    8007      A        LDX    #$8007   SET IOSTAT=80, CONSW=7
00143A FC6E AF    54        A        STX    IOSTAT,U
00144A FC70 86    50        A        LDA    #QLIM    GET QUEUE SIZE
00145A FC72 A7    5D        A        STA    IQLIM,U  SET LIMIT
00146A FC74 30    C4        A        LEAX   QUEUE,U  GET START OF QUEUE
00147A FC76 AF    5E        A        STX    IQBEG,U  SET START OF QUEUE
00148A FC78 86    1E        A        LDA    #HOME    OUTPUT CLEAR SCREEN
00149A FC7A 8D    4F   FCCB          BSR    OUT2
00150A FC7C 86    35        A        LDA    #$35     SET FOR KYBD PIA
00151A FC7E B7    F401      A        STA    PIASA
00152A FC81 7C    F402      A        INC    PIADB    FOR CASSETTE
00153A FC84 1C    EF        A        ANDCC  #$EF     CLEAR IRQ FOR KBD
00154A FC86 B6    C000      A        LDA    $C000    TEST PROM
00155A FC89 81    69        A        CMPA   #$69     EXECUTABLE??
00156A FC8B 26    03   FC90          BNE    EXEC
00157A FC8D 7E    C003      A        JMP    $C003    GO EXECUTE AUX. PROM
00158A FC90 FE    A3FE      A EXEC   LDU    SCR      RENEW SCRATCH POINTER
00159A FC93 32    51        A        LEAS   STACK,U  RENEW STACK PTR
00160A FC95 86    0D        A        LDA    #CR      CARRAIGE RETURN
00161A FC97 8D    32   FCCB          BSR    OUT2
00162A FC99 86    3E        A        LDA    #'>
00163A FC9B 8D    2E   FCCB          BSR    OUT2     OUTPUT PROMPT
00164A FC9D 31    C8 50     A        LEAY   INBUF,U  POINT TO INPUT BUFFER
00165A FCA0 17    020A FEAD          LBSR   INBF     GET BUFFER FM KBD
00166A FCA3       48        A        FCB    BUFLIM,CR,0 PARAMETERS FOR INBF
00167A FCA6 A6    A0        A        LDA    0,Y+     GET COMMAND CHARACTER
00168A FCA8 81    4D        A        CMPA   #'M      MODIFY?
00169A FCAA 27    3C   FCE8          BEQ    INSP
00170A FCAC 81    4A        A        CMPA   #'J      JUMP?
00171A FCAE 27    15   FCC5          BEQ    JUMP
00172A FCB0 81    57        A        CMPA   #'W      WRITE?
00173A FCB2 27    21   FCD5          BEQ    CSTOT
00174A FCB4 81    52        A        CMPA   #'R      READ?
```

```
00175A FCB6 27    16    FCCE          BEQ    CSTIN
00176A FCB8 86    3F       A ERROR    LDA    #'?        ERROR MESSAGE
00177A FCBA 8D    0F    FCCB          BSR    OUT2
00178A FCBC 20    D2    FC90 EXEC2    BRA    EXEC
00179A FCBE 8D    68    FD28 HEXCR    BSR    HEXIN      GET HEX VALUE
00180A FCC0 81    0D       A          CMPA   #CR        TEST EOL
00181A FCC2 26    F4    FCB8          BNE    ERROR      ERROR IF NOT
00182A FCC4 39                        RTS
00183A FCC5 8D    F7    FCBE JUMP     BSR    HEXCR      GET JUMP ADDR
00184A FCC7 6E    84       A          JMP    0,X        GO TO ROUTINE
00185A FCC9 86    20       A SPACE    LDA    #$20       SPACE
00186A FCCB 16    00A7 FD75 OUT2      LBRA   COUT
00187                        *
00188A FCCE 8D    EE    FCBE CSTIN    BSR    HEXCR      GET OFFSET ADDR
00189A FCD0 1F    12       A          TFR    X,Y
00190A FCD2 16    02BA FF8F           LBRA   CREAD      GO READ CASSETTE
00191                        *
00192A FCD5 8D    51    FD28 CSTOT    BSR    HEXIN      GET START ADDR
00193A FCD7 24    DF    FCB8          BCC    ERROR
00194A FCD9 81    20       A          CMPA   #$20       TEST SPACE DELIMITER
00195A FCDB 26    DB    FCB8          BNE    ERROR
00196A FCDD 34    10       A          PSHS   X
00197A FCDF 8D    DD    FCBE          BSR    HEXCR      GET END ADR
00198A FCE1 30    01       A          LEAX   1,X
00199A FCE3 34    10       A          PSHS   X
00200A FCE5 16    021D FF05           LBRA   CTOT       GO OUTPUT CASSETTE
00201                        *
00202A FCE8 8D    D4    FCBE INSP     BSR    HEXCR      GET INSPECT ADDR
00203A FCEA 1F    10       A INSP1    TFR    X,D        OUTPUT ADDR
00204A FCEC 8D    6D    FD5B          BSR    OHX16
00205A FCEE 8D    D9    FCC9          BSR    SPACE      OUTPUT SPACE
00206A FCF0 E6    84       A          LDB    0,X        GET MEMORY VALUE
00207A FCF2 8D    6F    FD63          BSR    OHX8       OUTPUT VALUE
00208A FCF4 8D    D3    FCC9          BSR    SPACE      OUTPUTTSPACE
00209A FCF6 31    C8 50    A          LEAY   INBUF,U    POINT TO INPUT BUFFER
00210A FCF9 17    01B1 FEAD           LBSR   INBF       READ BUFFER FM KBD
00211A FCFC       48       A          FCB    BUFLIM,CR,$20,$2C,0
00212A FD01 34    12       A          PSHS   A,X
00213A FD03 8D    23    FD28          BSR    HEXIN      GET VALUE
00214A FD05 24    09    FD10          BCC    INSP2      SKIP IF NO VALUE
00215A FD07 81    0D       A          CMPA   #CR        TEST IF EOLN
00216A FD09 26    AD    FCB8          BNE    ERROR
00217A FD0B 1F    10       A          TFR    X,D        GET VALUE
00218A FD0D E7    F8 01    A          STB    [1,S]      POINTER ON STACK
00219A FD10 35    12       A INSP2    PULS   A,X        RESTORE POINTER
00220A FD12 81    0D       A          CMPA   #CR        TEST RETURN SBCMD
00221A FD14 27    A6    FCBC          BEQ    EXEC2
00222A FD16 81    20       A          CMPA   #$20       TEST SPACE
00223A FD18 27    06    FD20          BEQ    INSP3      NEXT LOCATION
00224A FD1A 81    2C       A          CMPA   #',        TEST COMMA
00225A FD1C 26    9A    FCB8          BNE    ERROR      IMPROPER DELIMITER
00226A FD1E 30    1E       A          LEAX   -2,X       BACK UP ONE
00227A FD20 30    01       A INSP3    LEAX   1,X        AHEAD ONE
00228A FD22 86    0D       A          LDA    #CR        CARRAIGE RETURN
00229A FD24 8D    A5    FCCB          BSR    OUT2       OUTPUT IT
00230A FD26 20    C2    FCEA          BRA    INSP1      CONTINUE INSPECT
00231                        *
00232                        * HEXADECIMAL INPUT ROUTINE
```

```
00233                            * Y POINTS TO CHARACTER IN HEX
00234                            * EXIT: A CONTAINS DELIMITER, X CONTAINS VALUE
00235                            * CY SET INDICATES AT LEAST ONE VALID HEX
00236                            * CHARACTER ENCOUNTERED
00237                            *
00238A FD28 4F                   HEXIN  CLRA                  RESET CARRY
00239A FD29 34   01        A            PSHS   CC
00240A FD2B 8E   0000      A            LDX    #0             CLEAR FOR HEX VALUE
00241A FD2E A6   A0        A     HEXI   LDA    0,Y+           GET CHARACTER
00242A FD30 81   30        A            CMPA   #'0            < 0
00243A FD32 25   25   FD59              BLO    HXDL           EXIT
00244A FD34 81   46        A            CMPA   #'F            > F
00245A FD36 22   21   FD59              BHI    HXDL           EXIT
00246A FD38 81   39        A            CMPA   #'9            <= 9
00247A FD3A 23   06   FD42              BLS    HMSK           SKIP ALPHA MASK
00248A FD3C 81   41        A            CMPA   #'A            < A
00249A FD3E 25   19   FD59              BLO    HXDL           EXIT
00250A FD40 8B   09        A            ADDA   #9             ALPHA OFFSET
00251A FD42 84   0F        A     HMSK   ANDA   #$F            MASK LOWER BITS
00252A FD44 1E   10        A            EXG    X,D            DO 4 PLACE 16 BIT SHIFT
00253A FD46 58                          LSLB
00254A FD47 49                          ROLA
00255A FD48 58                          LSLB
00256A FD49 49                          ROLA
00257A FD4A 58                          LSLB
00258A FD4B 49                          ROLA
00259A FD4C 58                          LSLB
00260A FD4D 49                          ROLA
00261A FD4E 1E   10        A            EXG    X,D
00262A FD50 30   86        A            LEAX   A,X            ADD CURRENT NUMBER
00263A FD52 35   01        A            PULS   CC             SET CARRY
00264A FD54 43                          COMA                  INDICATE NOT NIL
00265A FD55 34   01        A            PSHS   CC
00266A FD57 20   D5   FD2E              BRA    HEXI           CONTINUE
00267A FD59 35   81        A     HXDL   PULS   CC,PC
00268                            *
00269                            * OUTPUT HEXADECIMAL ROUTINE
00270                            * OUTPUTS VALUE IN D
00271                            *
00272A FD5B 34   04        A     OHX16  PSHS   B
00273A FD5D 1F   89        A            TFR    A,B            OUTPUT UPPER BYTE
00274A FD5F 8D   02   FD63              BSR    OHX8           OUTPUT 8 BITS
00275A FD61 35   04        A            PULS   B              RESTORE LOWER BYTE
00276A FD63 1F   98        A     OHX8   TFR    B,A            OUTPUT UPPER NYBBLE
00277A FD65 44                          LSRA                  MOVE
00278A FD66 44                          LSRA                  UPPER NYBBLE
00279A FD67 44                          LSRA                  TO
00280A FD68 44                          LSRA                  LOWER NYBBLE
00281A FD69 8D   02   FD6D              BSR    OHX
00282A FD6B 1F   98        A            TFR    B,A            OUTPUT LOWER NYBBLE
00283A FD6D 84   0F        A     OHX    ANDA   #$F            MASK LOWER NYBBLE
00284A FD6F 8B   90        A            ADDA   #$90           CONVERT BINARY TO ASCII
00285A FD71 19                          DAA
00286A FD72 89   40        A            ADCA   #$40
00287A FD74 19                          DAA
00288                            *
00289                            * MAIN OUTPUT ROUTINE
00290                            *
```

```
00291                           * OUTPUT CHARACTER IN A
00292                           *
00293A FD75 34   77         A COUT    PSHS    CC,A,B,X,Y,U
00294A FD77 FE   A3FE       A         LDU     SCR         GET SCRATCH POINTER
00295A FD7A EC   56         A         LDD     CPTR,U      GET CURSOR POINTER
00296A FD7C 84   03         A         ANDA    #3          SO IT'S ALWAYS ON SCREEN
00297A FD7E 8B   F0         A         ADDA    #SCRN/256
00298A FD80 1F   01         A         TFR     D,X
00299A FD82 E6   58         A         LDB     LCHR,U      GET THE LAST CHARACTER
00300A FD84 E7   84         A         STB     0,X         REPLACE CURSOR
00301A FD86 A6   61         A         LDA     1,S         GET CHARACTER
00302A FD88 E6   55         A         LDB     CONSW,U     GET CRT SWITCHES
00303A FD8A 102B 00D3 FE61            LBMI    DSPLY
00304A FD8E 84   7F         A         ANDA    #$7F        PARITY OFF
00305A FD90 6D   59         A         TST     XYF,U       TEST IF XY COORDINATES
00306A FD92 2A   04    FD98           BPL     NSETX       NOT X
00307A FD94 A7   59         A STXYF   STA     XYF,U       SET XY COOR.  FLAG
00308A FD96 20   31    FDC9           BRA     VFIN3
00309A FD98 27   1F    FDB9 NSETX     BEQ     MVUP        NOT Y EITHER
00310A FD9A 80   20         A         SUBA    #$20        ADJUST BIAS
00311A FD9C 2B   17    FDB5           BMI     XYER
00312A FD9E 81   10         A         CMPA    #16         SCREEN HEIGHT
00313A FDA0 24   13    FDB5           BHS     XYER
00314A FDA2 E6   59         A         LDB     XYF,U       GET X COOR.
00315A FDA4 C0   20         A         SUBB    #$20        ADJUST BIAS
00316A FDA6 2B   0D    FDB5           BMI     XYER
00317A FDA8 C1   40         A         CMPB    #64         SCREEN WIDTH
00318A FDAA 24   09    FDB5           BHS     XYER        OUT OF BOUNDS
00319A FDAC 8E   F000       A         LDX     #SCRN       POINT TO CRT
00320A FDAF 3A                        ABX                 COOR:=64*Y+X
00321A FDB0 C6   40         A         LDB     #64
00322A FDB2 3D                        MUL
00323A FDB3 30   8B         A         LEAX    D,X
00324A FDB5 6F   59         A XYER    CLR     XYF,U       RESET XY FLAG
00325A FDB7 20   10    FDC9           BRA     VFIN3
00326A FDB9 81   0B         A MVUP    CMPA    #UP
00327A FDBB 26   0E    FDCB           BNE     MVDN
00328A FDBD 30   88 C0      A         LEAX    -64,X       MOVE CURSOR UP
00329A FDC0 8C   F000       A         CMPX    #SCRN       TOP OF SCREEN
00330A FDC3 24   6B    FE30           BHS     VFIN2
00331A FDC5 30   89 0400 A            LEAX    1024,X      GO TO BOTTOM
00332A FDC9 20   65    FE30 VFIN3     BRA     VFIN2
00333A FDCB 81   04         A MVDN    CMPA    #DOWN       TEST IF DOWN
00334A FDCD 26   0E    FDDD           BNE     MVRT
00335A FDCF 30   88 40      A         LEAX    64,X        MOVE CURSOR DOWN
00336A FDD2 8C   F400       A         CMPX    #SCRN+1024 BOTTOM OF SCREEN
00337A FDD5 25   59    FE30           BLO     VFIN2
00338A FDD7 30   89 FC00 A            LEAX    -1024,X     GO TO TOP OF SCREEN
00339A FDDB 20   53    FE30           BRA     VFIN2
00340A FDDD 81   0C         A MVRT    CMPA    #RIGHT      TEST IF RIGHT
00341A FDDF 26   0D    FDEE           BNE     MVLT
00342A FDE1 30   01         A         LEAX    1,X         MOVE CURSOR RIGHT
00343A FDE3 1F   10         A         TFR     X,D
00344A FDE5 C4   3F         A         ANDB    #$3F
00345A FDE7 26   47    FE30           BNE     VFIN2       IF NOT EOLN
00346A FDE9 30   88 C0      A         LEAX    -64,X       MOVE TO BEGINNING
00347A FDEC 20   42    FE30           BRA     VFIN2
00348A FDEE 81   06         A MVLT    CMPA    #LEFT       TEST IF LEFT
```

```
00349A FDF0 26   0B   FDFD         BNE    LINEF
00350A FDF2 30   1F       A        LEAX   -1,X      MOVE CURSOR LEFT
00351A FDF4 1F   10       A        TFR    X,D
00352A FDF6 53                     COMB
00353A FDF7 C4   3F       A        ANDB   #$3F      TEST BEGINNING OF LINE
00354A FDF9 26   35   FE30         BNE    VFIN2
00355A FDFB 20   04   FE01         BRA    LNEF      TO END OF LINE
00356A FDFD 81   0A       A LINEF  CMPA   #LF       TEST IF LINE FEED
00357A FDFF 26   05   FE06         BNE    CARET
00358A FE01 30   88 40    A LNEF   LEAX   64,X
00359A FE04 20   5F   FE65         BRA    SCRL
00360A FE06 81   0D       A CARET  CMPA   #CR       TEST CARRAIGE RETURN
00361A FE08 26   0B   FE15         BNE    SETXY
00362A FE0A 1E   10       A        EXG    X,D
00363A FE0C C4   C0       A        ANDB   #$C0      CLEAR LOWER 6 BITS
00364A FE0E 1E   10       A        EXG    X,D       OF CURSOR POINTER
00365A FE10 56                     RORB             TEST AUTO LF
00366A FE11 25   EE   FE01         BCS    LNEF      DO LINE FEED
00367A FE13 20   1B   FE30         BRA    VFIN2
00368A FE15 81   1B       A SETXY  CMPA   #ESC      SET XY?
00369A FE17 26   05   FE1E         BNE    EREOL
00370A FE19 86   80       A        LDA    #$80      SET MSB
00371A FE1B 16   FF76 FD94         LBRA   STXYF
00372A FE1E 81   0E       A EREOL  CMPA   #EEOL     TEST IF ERASE END OF LINE
00373A FE20 26   10   FE32         BNE    EREOS
00374A FE22 34   10       A        PSHS   X         SAVE CURSOR POINTER
00375A FE24 86   20       A EREL   LDA    #$20      SPACE
00376A FE26 A7   80       A        STA    0,X+      CLEAR LINE
00377A FE28 1F   10       A        TFR    X,D       TEST END OF LINE
00378A FE2A C4   3F       A        ANDB   #$3F
00379A FE2C 26   F6   FE24         BNE    EREL
00380A FE2E 35   10       A EREX   PULS   X         RESTORE POINTER
00381A FE30 20   52   FE84 VFIN2   BRA    VFIN
00382A FE32 81   05       A EREOS  CMPA   #FF
00383A FE34 27   0B   FE41         BEQ    ERASE
00384A FE36 81   1E       A        CMPA   #HOME     HOME UP?
00385A FE38 26   14   FE4E         BNE    BUP
00386A FE3A 8E   F000     A        LDX    #SCRN     TOP OF SCREEN
00387A FE3D C5   02       A        BITB   #2        TEST IF HOME & CLEAR
00388A FE3F 27   43   FE84         BEQ    VFIN
00389A FE41 34   10       A ERASE  PSHS   X
00390A FE43 86   20       A        LDA    #$20      SPACE CHARACTER
00391A FE45 A7   80       A CLRSC  STA    0,X+      CLEAR SCREEN
00392A FE47 8C   F400     A        CMPX   #SCRN+1024 TEST END OF SCREEN
00393A FE4A 25   F9   FE45         BLO    CLRSC
00394A FE4C 20   E0   FE2E         BRA    EREX
00395A FE4E 81   08       A BUP    CMPA   #BS       TEST IF BACKSPACE
00396A FE50 26   0B   FE5D         BNE    NCONT
00397A FE52 8C   F000     A        CMPX   #SCRN     TEST IF BEGINNING SCREEN
00398A FE55 23   2D   FE84         BLS    VFIN
00399A FE57 86   20       A        LDA    #$20
00400A FE59 A7   82       A        STA    0,-X      BACK UP AND DESTROY
00401A FE5B 20   27   FE84         BRA    VFIN
00402A FE5D 81   20       A NCONT  CMPA   #$20      TEST IF CONTROL
00403A FE5F 25   23   FE84         BLO    VFIN      SKIP IF TRUE
00404A FE61 A6   61       A DSPLY  LDA    1,S       GET CHARACTER
00405A FE63 A7   80       A        STA    0,X+      ONTO THE SCREEN
00406A FE65 8C   F400     A SCRL   CMPX   #SCRN+1024 TEST END OF SCREEN
```

```
00407A FE68 25    1A    FE84        BLO    VFIN
00408A FE6A 1F    12       A        TFR    X,Y        SAVE CURRENT SCREEN PTR
00409A FE6C 8E    F000     A        LDX    #SCRN      POINT TO TOP OF SCREEN
00410A FE6F A6    88 40    A SCRLM  LDA    64,X       CHAR FM NEXT LINE DOWN
00411A FE72 A7    80       A        STA    0,X+       MOVE UP ONE LINE
00412A FE74 8C    F3C0     A        CMPX   #SCRN+960 TEST END OF SCREEN
00413A FE77 25    F6    FE6F        BLO    SCRLM      LOOP TIL ALL MOVED UP
00414A FE79 CC    2040     A        LDD    #$20*256+64 SPACE BOTTOM LINE OUT
00415A FE7C A7    80       A CLRLN  STA    0,X+       CLEAR CRT MEMORY
00416A FE7E 5A                      DECB              COUNT 64 LOCATIONS
00417A FE7F 26    FB    FE7C        BNE    CLRLN
00418A FE81 30    A8 C0    A        LEAX   -64,Y      BACK UP LAST PTR ONE LINE
00419A FE84 AF    56       A VFIN   STX    CPTR,U     SET NEW CURSOR POINTER
00420A FE86 A6    84       A        LDA    0,X        GET LAST CHARACTER
00421A FE88 A7    58       A        STA    LCHR,U     SAVE LAST CHARACTER
00422A FE8A 86    5F       A        LDA    #$5F       UNDERLINE FOR CURSOR
00423A FE8C A7    84       A        STA    0,X        SET CURSOR
00424A FE8E 35    F7       A        PULS   CC,A,B,X,Y,U,PC
00425                          *
00426                          * INPUT KEYBOARD ROUTINE
00427                          * INPUTS CHARACTER IN ACC A
00428                          *
00429A FE90 34    55       A INCH   PSHS   CC,B,X,U
00430A FE92 1C    EF       A        ANDCC  #$EF       SET KYBD INTERUPT
00431A FE94 FE    A3FE     A        LDU    SCR        GET SCRATCH POINTER
00432A FE97 6D    5A       A INCH2  TST    IQCNT,U    TEST CHARACTERS IN QUEUE
00433A FE99 27    FC    FE97        BEQ    INCH2      LOOP UNTIL CHARACTER
00434A FE9B 6A    5A       A        DEC    IQCNT,U
00435A FE9D E6    5C       A        LDB    IQPTQ,U    GET QUEUE POINTER
00436A FE9F AE    5E       A        LDX    IQBEG,U    GET START OF QUEUE
00437A FEA1 A6    85       A        LDA    B,X        GET CHARACTER
00438A FEA3 5C                      INCB
00439A FEA4 E1    5D       A        CMPB   IQLIM,U    TEST IF END OF QUEUE
00440A FEA6 23    01    FEA9        BLS    CIRCE
00441A FEA8 5F                      CLRB              CIRCULAR QUEUE
00442A FEA9 E7    5C       A CIRCE  STB    IQPTQ,U    SET NEW QUEUE PTR
00443A FEAB 35    D5       A        PULS   CC,B,X,U,PC
00444                          *
00445                          * INPUT BUFFER ROUTINE
00446                          * ENTER Y POINTS TO BUFFER
00447                          *        BSR INBF
00448                          *        FCB BUFFER LIMIT-1
00449                          *        DELIMITERS FOLLOWED BY A ZERO
00450                          *
00451A FEAD 34    77       A INBF   PSHS   CC,A,B,X,Y,U
00452A FEAF FE    A3FE     A        LDU    SCR        GET SCRATCH POINTER
00453A FEB2 AE    69       A        LDX    9,S        GET RETURN ADDR
00454A FEB4 5F               INBF1  CLRB              BYTE COUNTER
00455A FEB5 8D    D9    FE90 INBF2  BSR    INCH       GET CHARACTER
00456A FEB7 81    08       A        CMPA   #BS        TEST IF BACK SPACE
00457A FEB9 27    33    FEEE        BEQ    BACK
00458A FEBB 81    15       A        CMPA   #CTLU      TEST IF CONTROL U
00459A FEBD 27    35    FEF4        BEQ    NEWLN
00460A FEBF 34    12       A        PSHS   A,X
00461A FEC1 30    01       A TDL    LEAX   1,X        PAST COUNT
00462A FEC3 A6    84       A        LDA    0,X        TEST END OF LIST
00463A FEC5 27    16    FEDD        BEQ    INBF4      EXIT IF END
00464A FEC7 A1    E4       A        CMPA   0,S        TEST IF DELIMITER
```

```
00465A FEC9 26   F6   FEC1        BNE    TDL       NO KEEP LOOKING
00466A FECB 35   12      A        PULS   A,X
00467A FECD ED   61      A        STD    1,S       SET DELIMITER
00468A FECF 8D   2D   FEFE        BSR    OUTX      OUTPUT DELIMITER
00469A FED1 86   0D      A        LDA    #CR       SET BUFFER DELIMITER TO CR
00470A FED3 A7   A5      A        STA    B,Y       SAVE DELIMITER
00471A FED5 6D   80      A FRT    TST    0,X+      FIND END OF DELIMITERS
00472A FED7 26   FC   FED5        BNE    FRT
00473A FED9 AF   69      A        STX    9,S       SET RETURN ADDR
00474A FEDB 35   F7      A        PULS   CC,A,B,X,Y,U,PC
00475A FEDD 35   12      A INBF4  PULS   A,X
00476A FEDF E1   84      A        CMPB   0,X       TEST IF LIMIT EXCEDED
00477A FEE1 24   D2   FEB5        BHS    INBF2     IF SO IGNORE
00478A FEE3 81   20      A        CMPA   #$20      NO CONTROLS IN BUFFER
00479A FEE5 25   CE   FEB5        BLO    INBF2
00480A FEE7 A7   A5      A        STA    B,Y       SET CHARACTER
00481A FEE9 5C                    INCB             UPDATE COUNTER
00482A FEEA 8D   12   FEFE INBF3  BSR    OUTX      OUTPUT CHARACTER
00483A FEEC 20   C7   FEB5        BRA    INBF2
00484A FEEE 5D             BACK   TSTB             TEST BEGINNING OF LINE
00485A FEEF 27   C4   FEB5        BEQ    INBF2
00486A FEF1 5A                    DECB             BACK UP
00487A FEF2 20   F6   FEEA        BRA    INBF3
00488A FEF4 5D             NEWLN  TSTB             TEST IF BEGINNING
00489A FEF5 27   BD   FEB4        BEQ    INBF1     YES CONTINUE
00490A FEF7 86   08      A        LDA    #BS       BACK UP TO BEGINNING
00491A FEF9 8D   03   FEFE        BSR    OUTX      OUTPUT BACKSPACE
00492A FEFB 5A                    DECB             COUNT OF # CHARACTERS
00493A FEFC 20   F6   FEF4        BRA    NEWLN
00494A FEFE 6D   54      A OUTX   TST    IOSTAT,U  TEST STATUS
00495A FF00 102B FE71 FD75 OUTX2  LBMI   COUT
00496A FF04 39                    RTS
00497                      *
00498                      * CASSETTE OUTPUT ROUTINE
00499                      *
00500A FF05 86   3C      A CTOT   LDA    #$3C      KBD OFF, RC ON
00501A FF07 B7   F401    A        STA    PIASA     SET PIA STATUS
00502A FF0A 4F                    CLRA             WAIT LOOP AFTER RC ON
00503A FF0B 5F                    CLRB
00504A FF0C 83   0001    A WCWT   SUBD   #1
00505A FF0F 26   FB   FF0C        BNE    WCWT
00506A FF11 86   FF      A PUNCH  LDA    #$FF      FORCE SYNC
00507A FF13 8D   4D   FF62        BSR    CASOUT
00508A FF15 8D   49   FF60        BSR    SYCASO    OUTPUT TRAILER
00509A FF17 86   9D      A        LDA    #$9D      OUTPUT RECORD MARK
00510A FF19 8D   47   FF62        BSR    CASOUT
00511A FF1B EC   E4      A        LDD    0,S
00512A FF1D A3   62      A        SUBD   2,S       END-BEGIN
00513A FF1F 1083 0040    A        CMPD   #64       .GT.64
00514A FF23 25   02   FF27        BLO    STMAX
00515A FF25 C6   40      A        LDB    #64       SET MAX BYTES PER RECORD
00516A FF27 1F   98      A STMAX  TFR    B,A       OUTPUT COUNT
00517A FF29 8D   37   FF62        BSR    CASOUT
00518A FF2B A6   62      A        LDA    2,S       OUTPUT HIGH ADDR
00519A FF2D 8D   33   FF62        BSR    CASOUT
00520A FF2F A6   63      A        LDA    3,S       OUTPUT LOW ADDR
00521A FF31 8D   2F   FF62        BSR    CASOUT
00522A FF33 AE   62      A        LDX    2,S       GET ADDR
```

```
00523A FF35 6F   E2        A        CLR    0,-S      SET UP CHECKSUM
00524A FF37 A6   84        A WCAS   LDA    0,X       GET VALUE
00525A FF39 8D   27   FF62          BSR    CASOUT    OUTPUT TO CASSETTE
00526A FF3B A6   80        A        LDA    0,X+
00527A FF3D AB   E4        A        ADDA   0,S       UPDATE CHECKSUM
00528A FF3F A7   E4        A        STA    0,S
00529A FF41 5A                      DECB             COUNT # BYTES
00530A FF42 26   F3   FF37          BNE    WCAS
00531A FF44 35   02        A        PULS   A         GET CHECKSUM
00532A FF46 8D   1A   FF62          BSR    CASOUT
00533A FF48 AF   62        A        STX    2,S       SAVE NEW POINTER
00534A FF4A AC   E4        A        CMPX   0,S       TEST IF DONE
00535A FF4C 26   C3   FF11          BNE    PUNCH     NO CONTINUE
00536A FF4E 8D   10   FF60          BSR    SYCASO    OUTPUT TRAILER
00537A FF50 86   B9        A        LDA    #$B9      END OF FILE
00538A FF52 8D   0E   FF62          BSR    CASOUT
00539A FF54 86   35        A CEXIT  LDA    #$35      RC OFF, KEYBOARD ON
00540A FF56 B7   F401      A        STA    PIASA
00541A FF59 1025 FD5B FCB8          LBCS   ERROR
00542A FF5D 16   FD30 FC90          LBRA   EXEC
00543                        *
00544A FF60 86   EC        A SYCASO LDA    #$EC      SYNC CHARACTER
00545                        *
00546A FF62 34   14        A CASOUT PSHS   B,X
00547A FF64 C6   05        A        LDB    #5        WRITE CLOCK ENABLE
00548A FF66 8D   17   FF7F          BSR    LD8
00549A FF68 13                      SYNC             WAIT FOR CLOCK EDGE
00550A FF69 6F   02        A        CLR    2,X       START BIT
00551A FF6B 13               COTLP  SYNC             WAIT FOR ANOTHER EDGE
00552A FF6C 46                      RORA
00553A FF6D 69   02        A        ROL    2,X       OUTPUT BIT
00554A FF6F 5A                      DECB             8 TIMES
00555A FF70 26   F9   FF6B          BNE    COTLP
00556A FF72 13                      SYNC
00557A FF73 1A   01        A CAST   ORCC   #$01      SET CARRY
00558A FF75 69   02        A        ROL    2,X
00559A FF77 C6   04        A        LDB    #4        TURN CLOCK OFF
00560A FF79 E7   03        A        STB    3,X
00561A FF7B 1C   EE        A        ANDCC  #$EE      ENABLE KEYBOARD INTERUPT
00562A FF7D 35   94        A        PULS   B,X,PC
00563                        *
00564                        *
00565A FF7F 1A   10        A LD8    ORCC   #$10      DISABLE KEYBOARD INTERUPT
00566A FF81 8E   F400      A        LDX    #PIADA    POINT TO PIA
00567A FF84 E7   03        A        STB    3,X       SET CLOCK INTERUPT
00568A FF86 E6   01        A        LDB    1,X
00569A FF88 C4   FC        A        ANDB   #$FC      CLEAR KEYBOARD
00570A FF8A E7   01        A        STB    1,X
00571A FF8C C6   08        A        LDB    #8
00572A FF8E 39                      RTS
00573                        *
00574                        *
00575                        *
00576A FF8F 8D   32   FFC3 CREAD    BSR    CASIN     GET SYNC
00577A FF91 81   EC        A        CMPA   #$EC      TEST IF SYNC
00578A FF93 26   FA   FF8F          BNE    CREAD     NO LOOP
00579A FF95 8D   2C   FFC3          BSR    CASIN
00580A FF97 81   B9        A        CMPA   #$B9      END OF FILE?
```

```
00581A FF99 27    B9    FF54           BEQ     CEXIT
00582A FF9B 81    9D       A           CMPA    #$9D       VALID RECORD MARKER
00583A FF9D 26    F0    FF8F           BNE     CREAD
00584A FF9F 8D    22    FFC3           BSR     CASIN      READ HEADER
00585A FFA1 34    02       A           PSHS    A
00586A FFA3 8D    1E    FFC3           BSR     CASIN
00587A FFA5 1F    89       A           TFR     A,B
00588A FFA7 8D    1A    FFC3           BSR     CASIN      GET ADDRESS
00589A FFA9 1E    89       A           EXG     A,B        CORRECT ADDRESS
00590A FFAB 30    AB       A           LEAX    D,Y        CALCULATE OFFSET
00591A FFAD 5F                         CLRB
00592A FFAE 8D    13    FFC3 RLOAD     BSR     CASIN      GET DATA
00593A FFB0 6A    E4       A           DEC     0,S        COUNT
00594A FFB2 2B    06    FFBA           BMI     CKSM       GET CKSM WHEN DONE
00595A FFB4 A7    84       A           STA     0,X        SAVE DATA
00596A FFB6 EB    80       A           ADDB    0,X+       UPDATE CHECKSUM
00597A FFB8 20    F4    FFAE           BRA     RLOAD
00598A FFBA 34    02       A CKSM      PSHS    A          CHECKSUM TO STACK
00599A FFBC E1    E1       A           CMPB    0,S++      TEST CHECKSUM
00600A FFBE 27    CF    FF8F           BEQ     CREAD
00601A FFC0 43                         COMA               SET CARRY FOR CHECKSUM ERROR
00602A FFC1 20    91    FF54           BRA     CEXIT
00603                        *
00604                        * CASSETTE INPUT
00605                        *
00606A FFC3 34    14       A CASIN     PSHS    B,X
00607A FFC5 C6    0C       A           LDB     #$C        SET READ CLOCK INTERUPT
00608A FFC7 8D    B6    FF7F           BSR     LD8
00609A FFC9 13               CIN2      SYNC               WAIT FOR CLOCK
00610A FFCA 6D    02       A           TST     2,X        START BIT?
00611A FFCC 2B    FB    FFC9           BMI     CIN2
00612A FFCE 13               CIN3      SYNC               WAIT FOR NEXT BIT
00613A FFCF 69    02       A           ROL     2,X        GET BIT
00614A FFD1 46                         RORA               INTO A
00615A FFD2 5A                         DECB               DO 8 TIMES
00616A FFD3 26    F9    FFCE           BNE     CIN3
00617A FFD5 20    9C    FF73           BRA     CAST
00618                        *
00619                        * INDIRECT VECTORS FOR 6809
00620                        *
00621A FFD7 6E    9F A3F2 A VSWI       JMP     [SWIV]
00622A FFDB 6E    9F A3F4 A VSWI2      JMP     [SWI2V]
00623A FFDF 6E    9F A3F6 A VSWI3      JMP     [SWI3V]
00624A FFE3 6E    9F A3F8 A VIRQ       JMP     [IRQV]
00625A FFE7 6E    9F A3FA A VFIRQ      JMP     [FIRQV]
00626A FFEB 6E    9F A3FC A VNMI       JMP     [NMIV]
00627                        * VECTORS
00628A FFF2                            ORG     $FFF2
00629A FFF2       FFDF     A           FDB     VSWI3
00630A FFF4       FFDB     A           FDB     VSWI2
00631A FFF6       FFE7     A           FDB     VFIRQ
00632A FFF8       FFE3     A           FDB     VIRQ
00633A FFFA       FFD7     A           FDB     VSWI
00634A FFFC       FFEB     A           FDB     VNMI
00635A FFFE       FC50     A           FDB     RESET
00636                                  END
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000
```

```
    FEEE BACK    00457 00484*
    0008 BS      00043*00395 00456 00490
    0048 BUFLIM  00058*00074 00166 00211
    FE4E BUP     00385 00395*
    FE06 CARET   00357 00360*
    FFC3 CASIN   00576 00579 00584 00586 00588 00592 00606*
    FF62 CASOUT  00507 00510 00517 00519 00521 00525 00532 00538 00546*
    FF73 CAST    00557*00617
    FF54 CEXIT   00539*00581 00602
    FFC9 CIN2    00609*00611
    FFCE CIN3    00612*00616
    FEA9 CIRCE   00440 00442*
    FC47 CIRCLE  00122 00124*
    FFBA CKSM    00594 00598*
    FE7C CLRLN   00415*00417
    FE45 CLRSC   00391*00393
    FFF5 CONSW   00062*00302
    FF6B COTLP   00551*00555
    FD75 COUT    00083 00186 00293*00495
    FFF6 CPTR    00063*00295 00419
    000D CR      00034*00160 00166 00180 00211 00215 00220 00228 00360 00469
    FF8F CREAD   00190 00576*00578 00583 00600
    FCCE CSTIN   00175 00188*
    FCD5 CSTOT   00173 00192*
    0015 CTLU    00041*00458
    0016 CTLV    00042*00106
    FF05 CTOT    00200 00500*
    0004 DOWN    00031*00333
    FE61 DSPLY   00303 00404*
    000E EEOL    00038*00372
    FC33 ENTER   00107 00113*
    FE41 ERASE   00383 00389*
    FE24 EREL    00375*00379
    FE1E EREOL   00369 00372*
    FE32 EREOS   00373 00382*
    FE2E EREX    00380*00394
    FCB8 ERROR   00176*00181 00193 00195 00216 00225 00541
    001B ESC     00037*00368
    FC90 EXEC    00156 00158*00178 00542
    FCBC EXEC2   00178*00221
    0005 FF      00039*00382
    A3FA FIRQV   00053*00625
    FED5 FRT     00471*00472
    FCBE HEXCR   00179*00183 00188 00197 00202
    FD2E HEXI    00241*00266
    FD28 HEXIN   00085 00179 00192 00213 00238*
    FD42 HMSK    00247 00251*
    001E HOME    00040*00148 00384
    FD59 HXDL    00243 00245 00249 00267*
    FEAD INBF    00084 00165 00210 00451*
    FEB4 INBF1   00454*00489
    FEB5 INBF2   00455*00477 00479 00483 00485
    FEEA INBF3   00482*00487
    FEDD INBF4   00463 00475*
    0050 INBUF   00072*00164 00209
    FE90 INCH    00082 00429*00455
    FE97 INCH2   00432*00433
    FC20 INPUT   00103*00135
```

```
   FCE8 INSP    00169 00202*
   FCEA INSP1   00203*00230
   FD10 INSP2   00214 00219*
   FD20 INSP3   00223 00227*
   FFF3 IODEV   00060*
   FFF4 IOSTAT  00061*00125 00143 00494
   FFFE IQBEG   00070*00117 00147 00436
   FFFA IQCNT   00066*00108 00113 00116 00432 00434
   FFFD IQLIM   00069*00114 00121 00145 00439
   FFFB IQPTK   00067*00109 00118 00124 00141
   FFFC IQPTQ   00068*00110 00435 00442
   A3F8 IRQV    00052*00136 00624
   FCC5 JUMP    00171 00183*
   FFF8 LCHR    00064*00299 00421
   FF7F LD8     00548 00565*00608
   0006 LEFT    00032*00348
   000A LF      00035*00356
   FDFD LINEF   00349 00356*
   FE01 LNEF    00355 00358*00366
   A3FF MAXRAM  00046*00049 00050 00051 00052 00053 00054 00055 00074
   FDCB MVDN    00327 00333*
   FDEE MVLT    00341 00348*
   FDDD MVRT    00334 00340*
   FDB9 MVUP    00309 00326*
   FE5D NCONT   00396 00402*
   FEF4 NEWLN   00459 00488*00493
   A3FC NMIV    00054*00626
   FD98 NSETX   00306 00309*
   FFF3 OFS     00056*00059 00060 00061 00062 00063 00064 00065 00066 00067
                00068 00069 00070 00071 00072 00074
   FD6D OHX     00281 00283*
   FD5B OHX16   00087 00204 00272*
   FD63 OHX8    00086 00207 00274 00276*
   FCCB OUT2    00149 00161 00163 00177 00186*00229
   FEFE OUTX    00468 00482 00491 00494*
   FF00 OUTX2   00127 00495*
   FC32 OVFLW   00111*00115
   F400 PIADA   00025*00103 00566
   F402 PIADB   00027*00152
   F401 PIASA   00026*00151 00501 00540
   F403 PIASB   00028*
   FF11 PUNCH   00506*00535
   0050 QLIM    00057*00072 00074 00144
   0000 QUEUE   00071*00146
   A000 RAM     00045*
   FC50 RESET   00132*00635
   000C RIGHT   00033*00340
   FFAE RLOAD   00592*00597
   A3FE SCR     00055*00104 00133 00158 00294 00431 00452
   FE65 SCRL    00359 00406*
   FE6F SCRLM   00410*00413
   F000 SCRN    00024*00297 00319 00329 00336 00386 00392 00397 00406 00409
                00412
   A359 SCRPD   00074*00132
   FE15 SETXY   00361 00368*
   FCC9 SPACE   00185*00205 00208
   FFF1 STACK   00059*00134 00159
   FF27 STMAX   00514 00516*
```

```
PAGE  014  MICRODAS.SA:1

    FD94 STXYF   00307*00371
    FC02 SWCMD   00082*00094
    FC0E SWI2C   00091*00137
    A3F4 SWI2V   00050*00138 00622
    A3F6 SWI3V   00051*00623
    A3F2 SWIV    00049*00078 00621
    FF60 SYCASO  00508 00536 00544*
    0009 TAB     00036*
    FEC1 TDL     00461*00465
    000B UP      00030*00326
    FE84 VFIN    00381 00388 00398 00401 00403 00407 00419*
    FE30 VFIN2   00330 00332 00337 00339 00345 00347 00354 00367 00381*
    FDC9 VFIN3   00308 00325 00332*
    FFE7 VFIRQ   00625*00631
    FFE3 VIRQ    00624*00632
    FFEB VNMI    00626*00634
    FFD7 VSWI    00621*00633
    FFDB VSWI2   00622*00630
    FFDF VSWI3   00623*00629
    FF37 WCAS    00524*00530
    FF0C WCWT    00504*00505
    FDB5 XYER    00311 00313 00316 00318 00324*
    FFF9 XYF     00065*00140 00305 00307 00314 00324
```

MOTOROLA

SEMICONDUCTORS

3501 ED BLUESTEIN BLVD., AUSTIN, TEXAS 78721

MC6809
(1.0 MHz)
MC68A09
(1.5 MHz)
MC68B09
(2.0 MHz)

Advance Information

8-BIT MICROPROCESSING UNIT

The MC6809 is a revolutionary high performance 8-bit microprocessor which supports modern programming techniques such as position independence, reentrancy, and modular programming.

This third-generation addition to the MC6800 family has major architectural improvements which include additional registers, instructions and addressing modes.

The basic instructions of any computer are greatly enhanced by the presence of powerful addressing modes. The MC6809 has the most complete set of addressing modes available on any microprocessor today.

The MC6809 has hardware and software features which make it an ideal processor for higher level language execution or standard controller applications.

MC6800 COMPATIBLE

- Hardware - Interfaces with All M6800 Peripherals
- Software - Upward Source Code Compatible Instruction Set and Addressing Modes

ARCHITECTURAL FEATURES

- Two 16-bit Index Registers
- Two 16-bit Indexable Stack Pointers
- Two 8-bit Accumulators can be concatenated to form one 16-bit Accumulator
- Direct Page Register allows Direct Addressing throughout memory map

HARDWARE FEATURES

- On chip oscillator (4 X fo XTAL)
- $\overline{\text{DMA/BREQ}}$ allows DMA operation or memory refresh
- Fast Interrupt Request Input stacks only Condition Code Register and Program Counter
- MRDY Input extends data access times for use with slow memory
- Interrupt Acknowledge output allows vectoring by devices
- SYNC Acknowledge output allows for synchronization to external event.
  - Single Bus-cycle RESET
  - Single 5-volt operation
  - NMI blocked after RESET until after first load of Stack Pointer
  - Early address valid allows use with slower memories

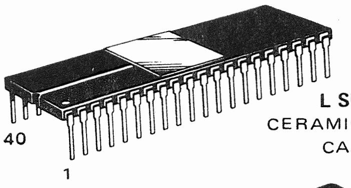
SOFTWARE FEATURES

- 10 Addressing Modes
  - 6800 Upward compatible Addressing Modes
  - Direct Addressing anywhere in memory map
  - Long Relative Branches
  - Program Counter Relative
  - Indirection
  - Expanded Index Addressing
    - 0,5,8,16-bit constant offsets
    - 8, 16-bit accumulator offsets
    - Auto-increment/decrement
- Improved Stack Manipulation
- 1464 Instructions with unique addressing modes
- 8 x 8 unsigned multiply
- 16-bit arithmetic
- Transfer/Exchange all registers
- Push/Pull **any** or **all** registers
- Load Effective Address

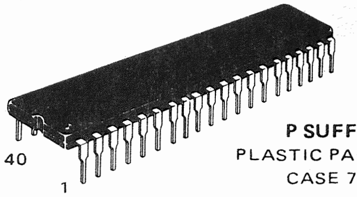
This is advance information and specifications are subject to change without notice.

MOS
(N-CHANNEL, SILICON-GATE)

8-BIT MICROPROCESSING UNIT

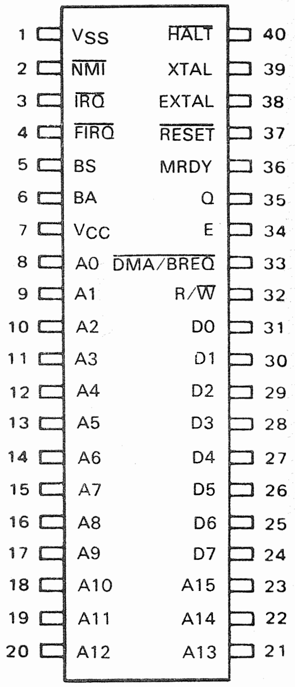


L SUFFIX
CERAMIC PACKAGE
CASE 715



P SUFFIX
PLASTIC PACKAGE
CASE 711

PIN ASSIGNMENT



Pin	Signal	Signal	Pin
1	V_{SS}	$\overline{\text{HALT}}$	40
2	$\overline{\text{NMI}}$	XTAL	39
3	$\overline{\text{IRQ}}$	EXTAL	38
4	$\overline{\text{FIRQ}}$	$\overline{\text{RESET}}$	37
5	BS	MRDY	36
6	BA	Q	35
7	V_{CC}	E	34
8	A0	$\overline{\text{DMA/BREQ}}$	33
9	A1	R/$\overline{\text{W}}$	32
10	A2	D0	31
11	A3	D1	30
12	A4	D2	29
13	A5	D3	28
14	A6	D4	27
15	A7	D5	26
16	A8	D6	25
17	A9	D7	24
18	A10	A15	23
19	A11	A14	22
20	A12	A13	21

MAXIMUM RATINGS

Rating	Symbol	Value	Unit
Supply Voltage	V_{CC}	-0.3 to +7.0	Vdc
Input Voltage	V_{in}	-0.3 to +7.0	Vdc
Operating Temperature Range	T_A	0 to +70	°C
Storage Temperature Range	T_{stg}	-55 to +150	°C
Thermal Resistance	θ_{JA}	70	°C/W

This device contains circuitry to protect the inputs against damage due to high static voltages or electric fields; however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum rated voltages to this high impedance circuit.

ELECTRICAL CHARACTERISTICS (V_{CC} = 5.0 V ±5%, V_{SS} = 0, T_A = 0 to 70°C unless otherwise noted.)

Characteristic		Symbol	Min	Typ	Max	Unit
Input High Voltage	Logic, EXtal $\overline{\text{RESET}}$	V_{IH}	V_{SS} + 2.0 V_{SS} + 4.0	— —	V_{DD} V_{DD}	Vdc
Input Low Voltage	Logic, EXtal, $\overline{\text{RESET}}$	V_{IL}	V_{SS} - 0.3	—	V_{SS} + 0.8	Vdc
Input Leakage Current (V_{in} = 0 to 5.25 V, V_{CC} = max)	Logic	I_{in}	—	1.0	2.5	µAdc
Output High Voltage (I_{Load} = -205 µAdc, V_{CC} = min) (I_{Load} = -145 µAdc, V_{CC} = min) (I_{Load} = -100 µAdc, V_{CC} = min)	 D0-D7 A0-A15, R/$\overline{\text{W}}$, Q, E BA, BS	V_{OH}	 V_{SS} + 2.4 V_{SS} + 2.4 V_{SS} + 2.4	 — —	 — —	Vdc
Output Low Voltage (I_{Load} = 2.0 mAdc, V_{CC} = min)		V_{OL}	—	—	V_{SS} +0.5	Vdc
Power Dissipation		P_D	—	—	1.0	W
Capacitance # (V_{in} = 0, T_A = 25°C, f = 1.0 MHz)	 D0-D7 Logic Inputs, EXtal A0-A15, R/W	C_{in} C_{out}	 — — —	 10 7 —	 15 10 12	pF
Frequency of Operation (Crystal or External Input)	MC6809 MC68A09 MC68B09	f f_{XTAL} f_{XTAL}	— — —	— — —	4 6 8	MHz
Three-State (Off State) Input Current (V_{in} = 0.4 to 2.4 V, V_{CC} = max)	D0-D7 A0-A15, R/W	I_{TSI}	— —	2.0 —	10 100	µAdc

READ/WRITE TIMING (Reference Figures 1 and 2)

Characteristic	Symbol	MC6809			MC68A09			MC68B09			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Cycle Time	t_{CYC}	1000	—	—	667	—	—	500	—	—	ns
Total Up Time	t_{UT}	975	—	—	640	—	—	480	—	—	ns
Peripheral Read Access Time t_{ac} = (t_{AD} = t_{DSR})	t_{ACC}	695	—	—	440	—	—	320	—	—	ns
Data Setup Time (Read)	t_{DSR}	80	—	—	60	—	—	40	—	—	ns
Input Data Hold Time	t_{DHR}	10	—	—	10	—	—	10	—	—	ns
Output Data Hold Time	t_{DHW}	30	—	—	30	—	—	30	—	—	ns
Address Hold Time (Address, R/$\overline{\text{W}}$)	t_{AH}	30	—	—	30	—	—	30	—	—	ns
Address Delay	t_{AD}	—	—	200	—	—	140	—	—	110	ns
Data Delay Time (Write)	t_{DDW}	—	—	225	—	—	180	—	—	145	ns
E_{low} to Q_{high} Time	t_{AVS}	—	—	250	—	—	165	—	—	125	ns
Address Valid to Q_{high}	t_{AQ}	25	—	—	25	—	—	15	—	—	ns
Processor Clock Low	t_{PWEL}	450	—	—	295	—	—	210	—	—	ns
Processor Clock High	t_{PWEH}	450	—	—	280	—	—	220	—	—	ns
MRDY Set Up Time	t_{PCSR}	60	—	—	60	—	—	60	—	—	ns
Interrupts Set Up Time	t_{PCS}	200	—	—	140	—	—	110	—	—	ns
$\overline{\text{HALT}}$ Set Up Time	t_{PCSH}	200	—	—	140	—	—	110	—	—	ns
$\overline{\text{RESET}}$ Set Up Time	t_{PCSR}	200	—	—	140	—	—	110	—	—	ns
$\overline{\text{DMA/BREQ}}$ Set Up Time	t_{PCSD}	125	—	—	125	—	—	125	—	—	ns
Crystal Osc Start Time	t_{rc}	100	—	—	100	—	—	100	—	—	ms
E Rise and Fall Time	t_{ER}, t_{EF}	5	—	25	5	—	25	5	—	20	ns
Processor Control Rise/Fall	t_{PCR}, t_{PLF}	—	—	100	—	—	100	—	—	100	ns
Q Rise and Fall Time	t_{QR}, t_{QF}	5	—	25	5	—	25	5	—	20	ns
Q Clock High	t_{PWQH}	450	—	—	280	—	—	220	—	—	ns

FIGURE 1 — READ DATA FROM MEMORY OR PERIPHERALS

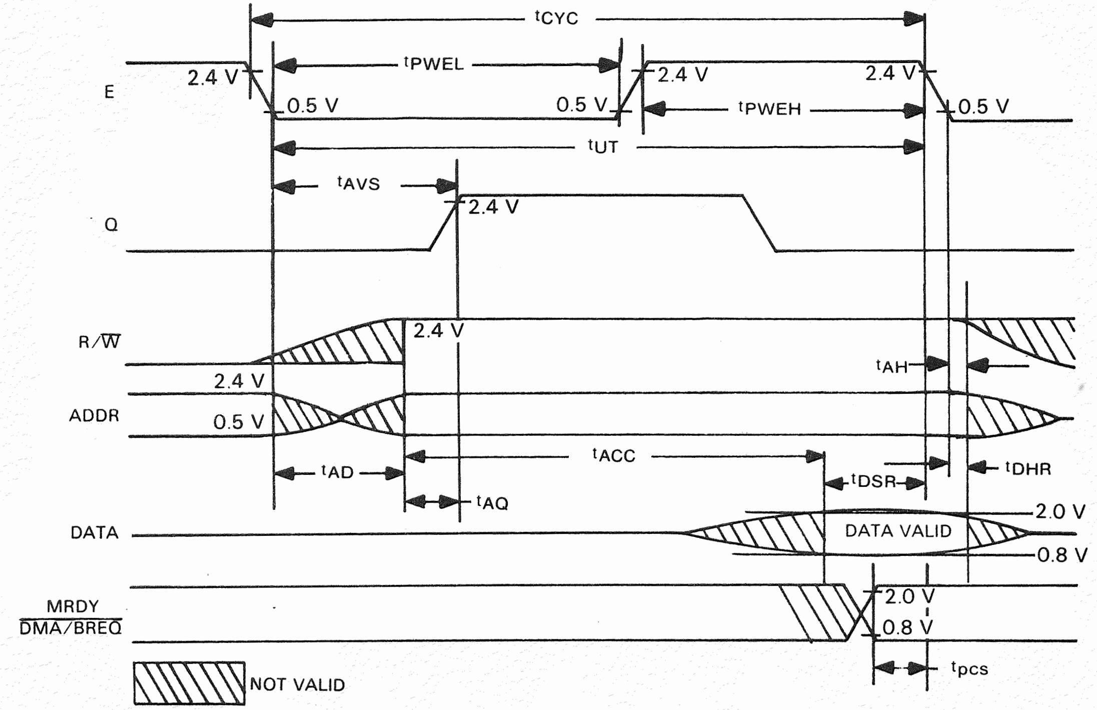


FIGURE 2 — WRITE DATA TO MEMORY OR PERIPHERALS

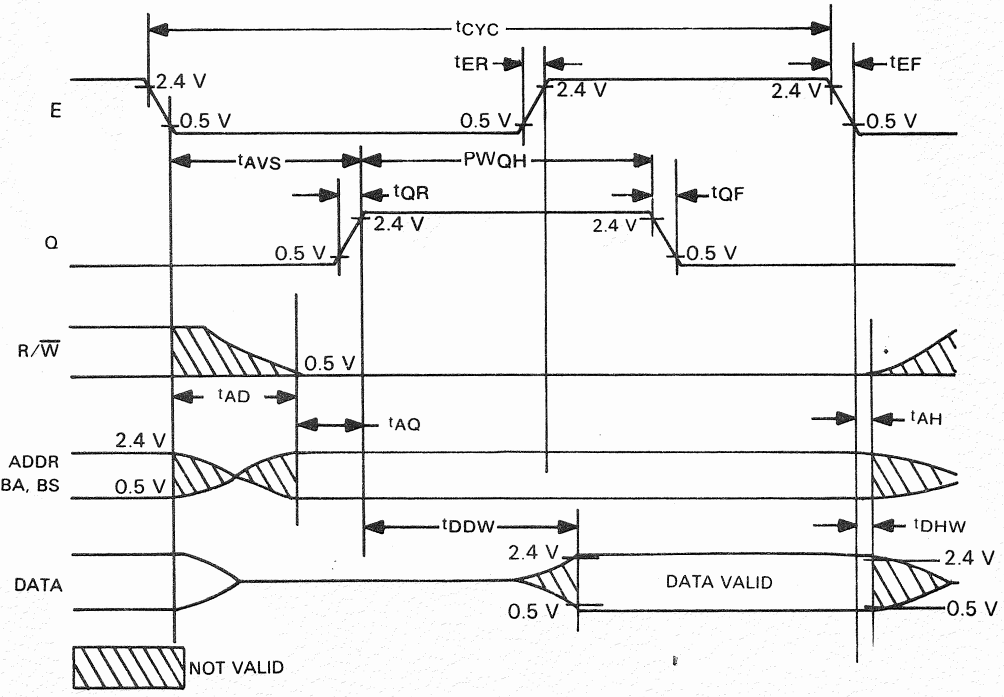


FIGURE 3 — MC6809 EXPANDED BLOCK DIAGRAM

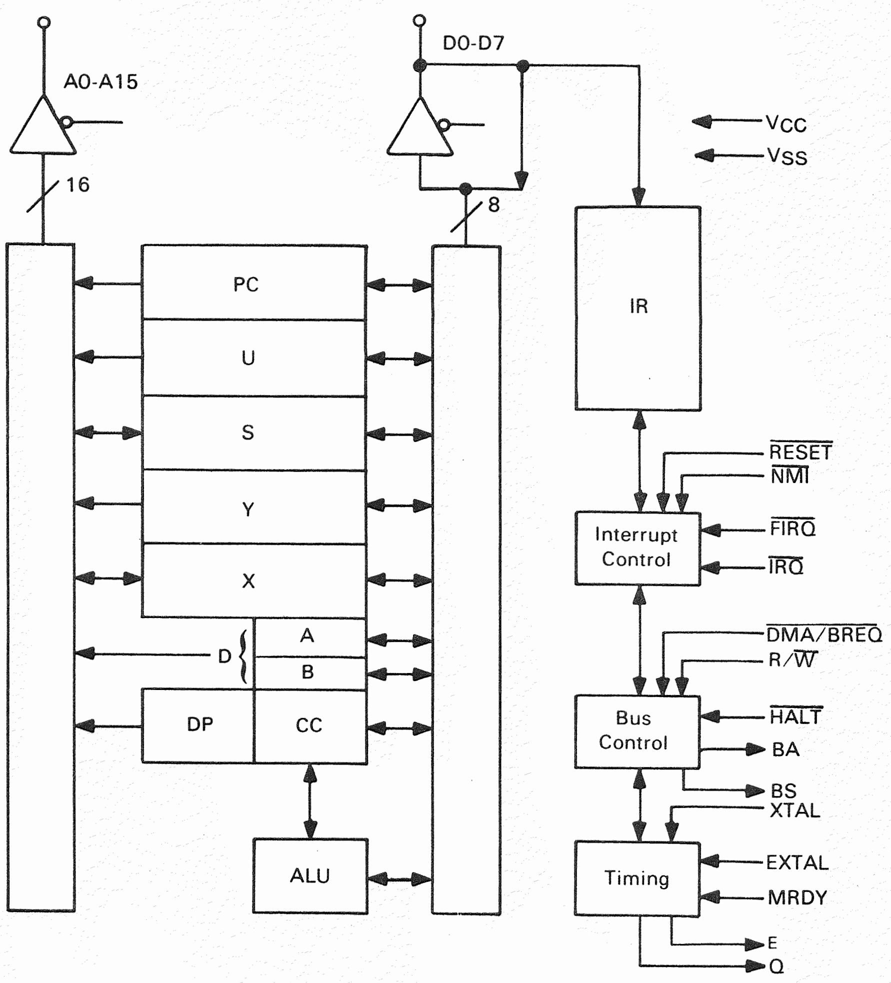
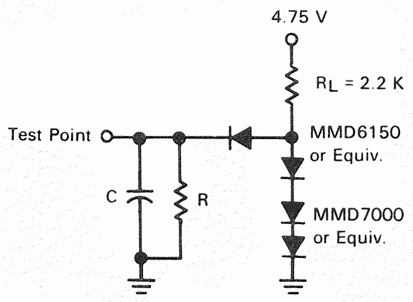


FIGURE 4 — BUS TIMING TEST LOAD



C = 30 pF for BA, BS 130 pF for D0-D7, E, Q 90 pF for A0-A15, R/W	R = 11.7 kΩ for D0-D7 16.5 kΩ for A0-A15, E, Q 24 kΩ for BA, BS

PROGRAMMING MODEL

As shown in Figure 5, the MC6809 adds three registers to the set available in the MC6800. The added registers include a direct page register, the User Stack pointer and a second Index Register.

ACCUMULATORS (A, B, D)

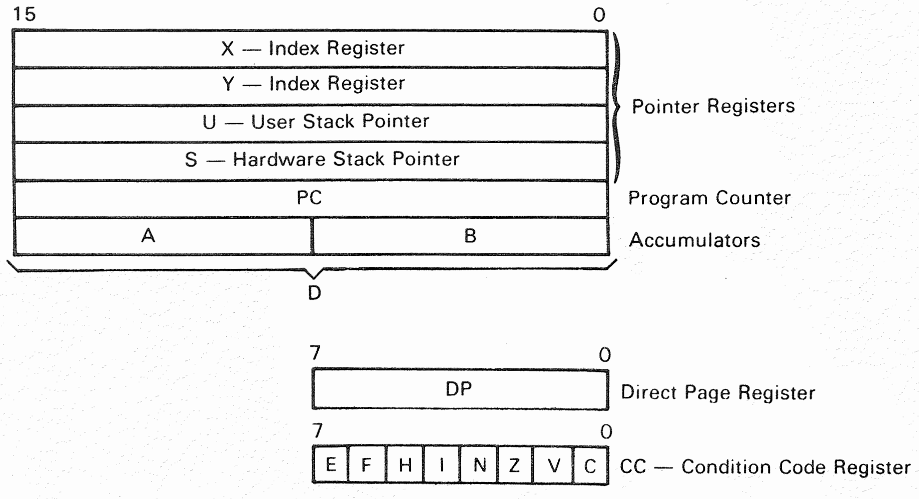
The A and B registers are general purpose accumulators which are used for arithmetic calculations and manipulation of data.

Certain instructions concatenate the A and B registers to form a single 16-bit accumulator. This is referred to as the D register, and is formed with the A register as the most significant byte.

DIRECT PAGE REGISTER (DP)

The Direct Page Register of the MC6809 serves to enhance the Direct Addressing Mode. The content of this register appears at the higher address outputs (A8-A15) during direct Addressing Instruction execution. This allows the direct mode to be used at any place in memory, under program control. To allow 6800 compatibility, all bits of this register are cleared during Processor Reset.

FIGURE 5 — PROGRAMMING MODEL OF THE MICROPROCESSING UNIT



INDEX REGISTERS (X,Y)

The Index Registers are used in indexed mode of addressing. The 16-bit address in this register takes part in the calculation of effective addresses. This address may be used to point to data directly or may be modified by an optional constant or register offset. During some indexed modes, the contents of the index register are incremented and decremented to point to the next item of tabular type data. All four pointer registers (X,Y,U,S) may be used as index registers.

STACK POINTERS (U,S)

The Hardware Stack Pointer (S) is used automatically by the processor during subroutine calls and interrupts. The stack pointers of the MC6809 point to the top of the stack, in contrast to the MC6800 stack pointer, which pointed to the next free location on the stack. The User Stack Pointer (U) is controlled exclusively by the programmer thus allowing arguments to be passed to and from subroutines with ease. Both Stack Pointers have the same indexed mode addressing capabilities as the X and Y registers, but also support **Push** and **Pull** instructions. This allows the MC6809 to be used efficiently as a stack processor, greatly enhancing its ability to support higher level languages and modular programming.

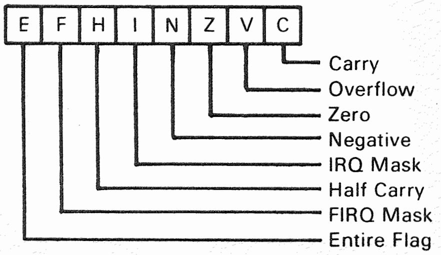
PROGRAM COUNTER

The Program Counter is used by the processor to point to the address of the next instruction to be executed by the processor. Relative Addressing is provided allowing the Program Counter to be used like an index register in some situations.

CONDITION CODE REGISTER

The condition code register defines the State of the Processor at any given time, see Figure 6.

FIGURE 6 — CONDITION CODE REGISTER FORMAT



CONDITION CODE REGISTER DESCRIPTION

BIT 0 (C)

Bit 0 is the Carry Flag, and is usually the carry from the binary ALU. C is also used to represent a 'borrow' from subtract like instructions (CMP, NEG, SUB, SBC). Here the carry flag is the complement of the carry from the binary ALU.

BIT 1 (V)

Bit 1 is the overflow flag, and is set to a one by an operation which causes a signed two's complement arithmetic overflow. This overflow is detected in an operation in which the carry from the MSB in the ALU does not match the carry from the MSB-1.

BIT 2 (Z)

Bit 2 is the zero flag, and is set to a one if the result of the previous operation was identically zero.

BIT 3 (N)

Bit 3 is the negative flag, which contains exactly the value of the MSB of the result of the preceeding operation. Thus, a negative two's-complement result will leave N set to a one.

BIT 4 (I)

Bit 4 is the $\overline{\mathrm{IRQ}}$ mask bit. The processor will not recognize interrupts from the IRQ line if this bit is set to a one. $\overline{\mathrm{NMI}}$, $\overline{\mathrm{FIRQ}}$, $\overline{\mathrm{IRQ}}$, $\overline{\mathrm{RESET}}$, and SWI all set I to a one; SWI2 AND SWI3 do not affect I.

BIT 5 (H)

Bit 5 is the half-carry bit, and is used to indicate a carry from bit 3 in the ALU as a result of an 8-bit addition only (ADC or ADD). This bit is used by the DAA instruction to perform a BCD decimal add adjust operation. The state of this flag is undefined in all subtract-like instructions.

BIT 6 (F)

Bit 6 is the $\overline{\mathrm{FIRQ}}$ mask bit. The processor will not recognize interrupts from the FIRQ line if this bit is a one. $\overline{\mathrm{NMI}}$, $\overline{\mathrm{FIRQ}}$, SWI, and $\overline{\mathrm{RESET}}$ all set F to a one. $\overline{\mathrm{IRQ}}$, SWI2 and SWI3 do not affect F.

BIT 7 (E)

Bit 7 is the entire flag, and when set to a one indicates that the complete machine state (all the registers) was stacked, as opposed to the subset state (PC and CC). The E bit of the stacked CC is used on a return from interrupt (RTI) to determine the extent of the unstacking. Therefore, the current E left in the Condition Code Register represents past action.

MC6809 MPU SIGNAL DESCRIPTION

POWER (V_{SS}, V_{CC})

Two pins are used to supply power to the part: V_{SS} is ground or 0 volts, while V_{CC} is +5.0 V ±5%.

ADDRESS BUS (A0-A15)

Sixteen pins are used to output address information from the MPU onto the Address Bus. When the processor does not require the bus for a data transfer, it will output address FFFF16, R/$\overline{\mathrm{W}}$ = 1, and BS = 0. Addresses are valid on the rising edge of Q (see Figure 1 and 2). All address bus drivers are made high-impedance when output Bus Available (BA) is high. Each pin will drive one Schottky TTL load and typically 90 pF.

DATA BUS (D0-D7)

These eight pins provide communication with the system bi-directional data bus. Each pin will drive one Schottky TTL load and typically 130 pF.

READ/WRITE (R/$\overline{\mathrm{W}}$)

This signal indicates the direction of data transfer on the data bus. A low indicates that the MPU is writing data onto the data bus. R/$\overline{\mathrm{W}}$ is made high impedance when BA is high. R/$\overline{\mathrm{W}}$ is valid on the rising edge of Q, refer to Figure 1 and 2.

$\overline{\mathrm{RESET}}$

A low level on this Schmitt-trigger input for greater than one bus cycle will reset the MPU as shown Figure 7. The Reset vectors are fetched from locations FFFE16 and FFFF16 (Table 1) when Interrupt Acknowledge is true, (BA ∧ BS = 1). During initial power-on, the Reset line should be held low until the clock oscillator is fully operational; see Figure 8.

Because the MC6809 Reset pin has a Schmitt-trigger input with a threshold voltage higher than that of standard peripherals, a simple R/C network may be used to reset the entire system. This higher threshold voltage insures that all peripherals are out of the reset state before the Processor.

$\overline{\mathrm{HALT}}$

A low level on this input pin will cause the MPU to stop running at the end of the present instruction and remain halted indefinitely without loss of data. When Halted, the BA output is driven high indicating the buses are high-impedance. BS is also high which indicates the processor is in the Halt or Bus Grant state. While halted, the MPU will not respond to external real-time requests ($\overline{\mathrm{FIRQ}}$, $\overline{\mathrm{IRQ}}$) although $\overline{\mathrm{DMA/BREQ}}$ will always be accepted, and $\overline{\mathrm{NMI}}$ or $\overline{\mathrm{RESET}}$ will be latched for later response. During the Halt state Q and E continue to run normally. If the MPU is not running ($\overline{\mathrm{RESET}}$, $\overline{\mathrm{DMA/BREQ}}$), a halted state (BA and BS = 1) can be achieved by pulling $\overline{\mathrm{HALT}}$ low while $\overline{\mathrm{RESET}}$ is still low. If $\overline{\mathrm{DMA/BREQ}}$ and $\overline{\mathrm{HALT}}$ are both pulled low, the processor will reach the last cycle of the instruction (by reverse cycle stealing) where the machine will then become halted. See Figure 9.

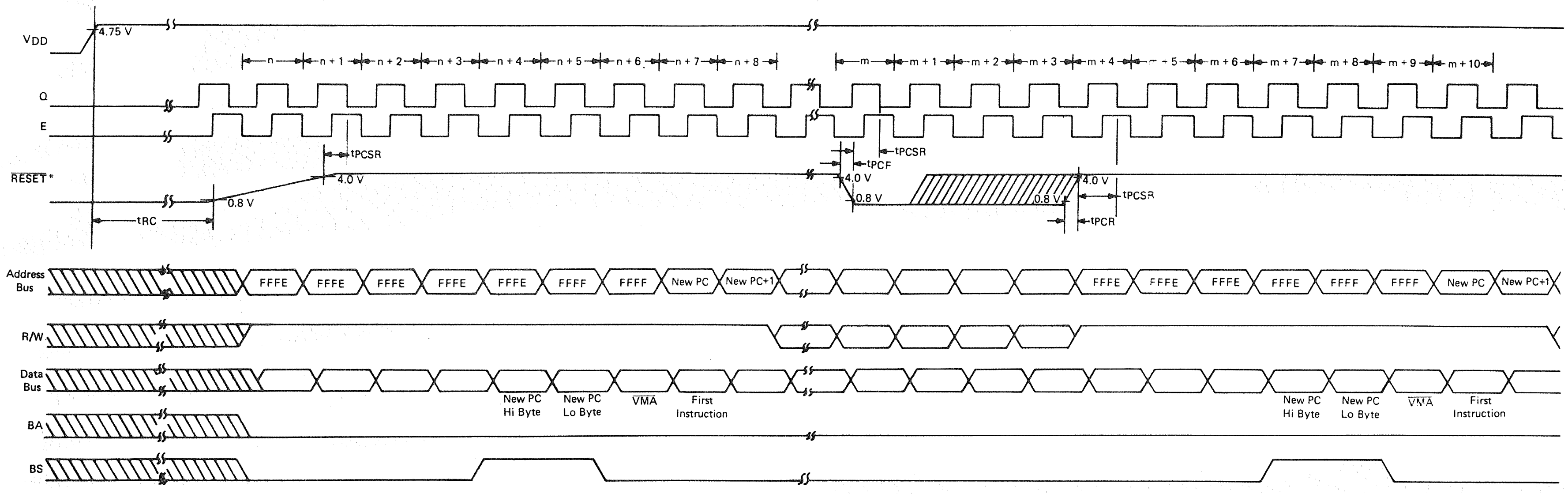
BUS AVAILABLE, BUS STATUS (BA, BS)

The Bus Available output is an indication of an internal control signal which makes the MOS buses of the MPU high-impedance. This signal does not imply that the bus will be available for more than one cycle. When BA goes low, an additional dead cycle will elapse before the MPU acquires the bus.

The Bus Status output signal, when decoded with BA, represents the MPU state (valid with leading edge of Q):

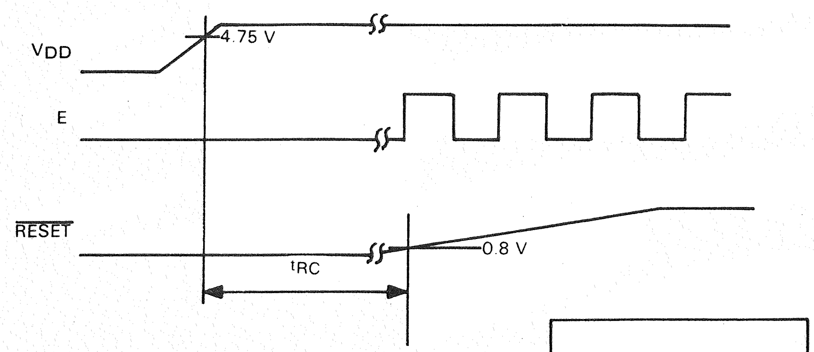
MPU State		
BA	BS	
0	0	Normal (Running)
0	1	Interrupt Acknowledge
1	0	SYNC Acknowledge
1	1	HALT or Bus Grant

FIGURE 7 — RESET TIMING



*Note: Parts with date codes prefixed by 7F will come out of Reset one cycle sooner than shown.

FIGURE 8 — CRYSTAL CONNECTIONS AND OSCILLATOR START UP

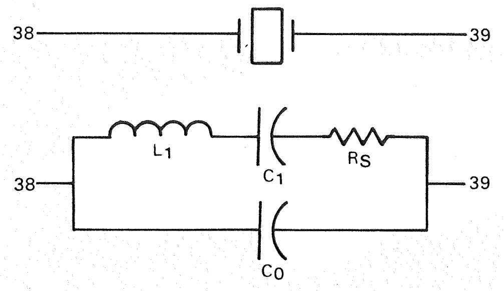


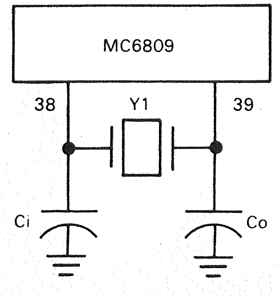
6809 Crystal Parameters*

	3.58 MHz	4.00 MHz	6.0 MHz	8.0 MHz
RS	60 Ω	50 Ω	30-50 Ω	20-40 Ω
Co	3.5 pF	6.5 pF	4-6 pF	4-6 pF
C1	.015 pF	.025 pF	.01-.02 pF	.01-.02 pF
Cin, Cout	25 pF	25 pF	25 pF	25 pF
Q	40 K	30 K	≈20 K	≈20 K

All Parameters Are ±10%

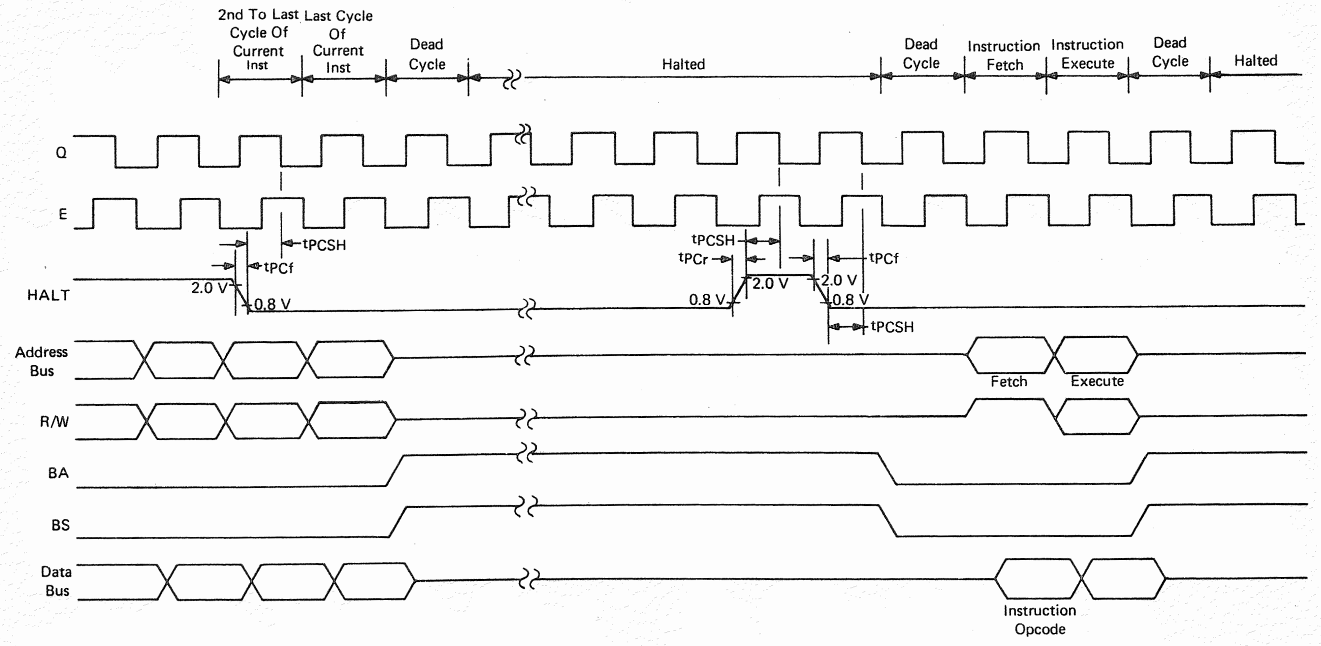
*Note: These are representative AT-cut crystal parameters only. Crystals of other types of cut that work may also be used.





Y1	Cin	Cout
8 MHz	18 pF	18 pF
6 MHz	20 pF	20 pF
4 MHz	24 pF	24 pF

FIGURE 9 — $\overline{\text{HALT}}$ AND SINGLE INSTRUCTION EXECUTION FOR SYSTEM DEBUG



Interrupt Acknowledge is indicated during both cycles of a hardware-vector-fetch ($\overline{\text{RESET}}$, $\overline{\text{NMI}}$, $\overline{\text{FIRQ}}$, $\overline{\text{IRQ}}$, SWI, SWI2, SWI3). This signal, plus decoding of the lower 4 address lines can provide the user with an indication of which interrupt level is being serviced and allow vectoring by device, (see Table 1).

Sync Acknowledge is indicated while the MPU is waiting for external synchronization on an interrupt line.

Halt/Bus Grant is true when the MC6809 is in a Halt or Bus Grant condition.

TABLE 1: MEMORY MAP FOR INTERRUPT VECTORS

Memory Map For Vector Location		Interrupt Vector Description
MS	LS	
FFFE	FFFF	$\overline{\text{RESET}}$
FFFC	FFFD	$\overline{\text{NMI}}$
FFFA	FFFB	SWI
FFF8	FFF9	$\overline{\text{IRQ}}$
FFF6	FFF7	$\overline{\text{FIRQ}}$
FFF4	FFF5	SWI2
FFF2	FFF3	SWI3
FFF0	FFF1	Reserved

***NOTE:** $\overline{\text{NMI}}$, $\overline{\text{FIRQ}}$ and $\overline{\text{IRQ}}$ requests are latched by the falling edge of every Q except during cycle stealing operations (e.g., DMA) where only $\overline{\text{NMI}}$ is latched. From this point, a delay of at least one bus cycle will occur before the interrupt is serviced by the MPU.

NON MASKABLE INTERRUPT ($\overline{\text{NMI}}$)

A negative **edge** on this input requests that a non-maskable interrupt sequence be generated. A non-maskable interrupt cannot be inhibited by the program, and also has a higher priority than $\overline{\text{FIRQ}}$, $\overline{\text{IRQ}}$ or software interrupts. During recognition of an $\overline{\text{NMI}}$, the entire machine state **is** saved on the hardware stack. After reset, an NMI will not be recognized until the first program load of the Hardware Stack Pointer (S). The pulse width of NMI low must be at least one E cycle. If the $\overline{\text{NMI}}$ input does not meet the minimum set up with respect to Q, the interrupt will not be recognized until the next cycle. See Figure 10.

FAST-INTERRUPT REQUEST ($\overline{\text{FIRQ}}$)

A low level on this input pin will initiate a fast interrupt sequence, provided its mask bit (F) in the CC is clear. This sequence has priority over the standard Interrupt Request ($\overline{\text{IRQ}}$), and is fast in the sense that it stacks only the contents of the condition code register and the program counter. The interrupt service routine should clear the source of the interrupt before doing an RTI. See Figure 11.

INTERRUPT REQUEST ($\overline{\text{IRQ}}$)

A low **level** input on this pin will initiate an Interrupt Request sequence provided the mask bit (I) in the CC is clear. Since $\overline{\text{IRQ}}$ stacks the entire machine state it provides a slower response to interrupts than $\overline{\text{FIRQ}}$. $\overline{\text{IRQ}}$ also has a lower priority than $\overline{\text{FIRQ}}$. Again, the interrupt service rouine should clear the source of the interrupt before doing an RTI. See Figure 10.

FIGURE 10 — $\overline{\text{IRQ}}$ AND $\overline{\text{NMI}}$ INTERRUPT TIMING

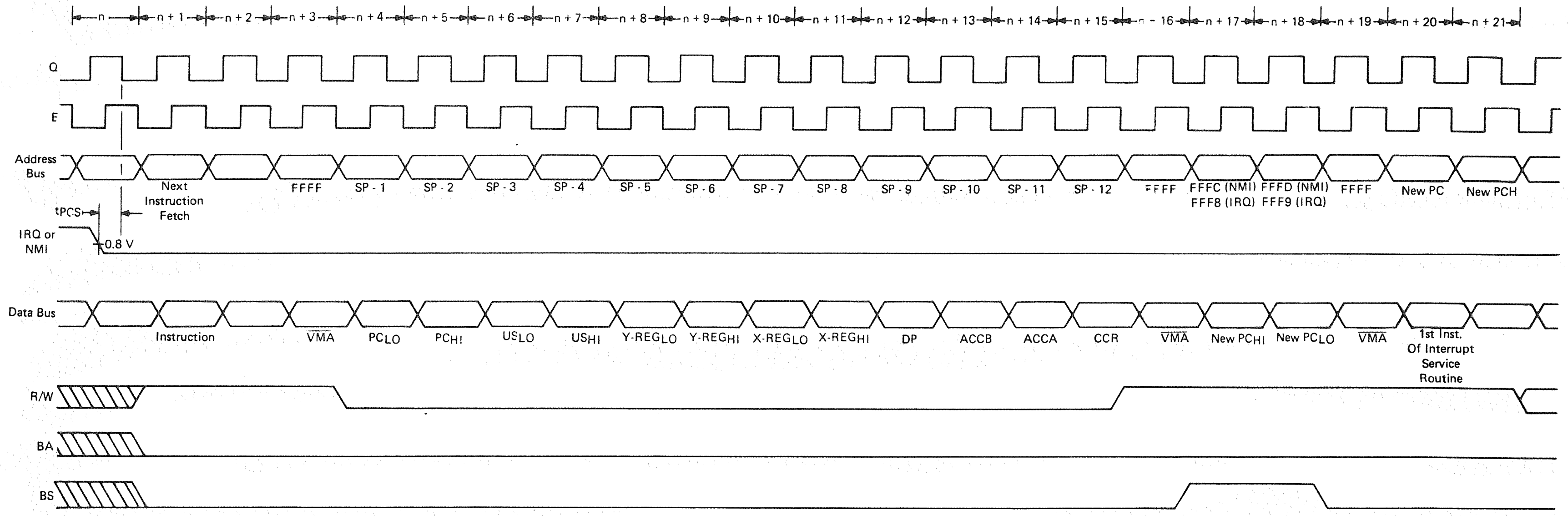
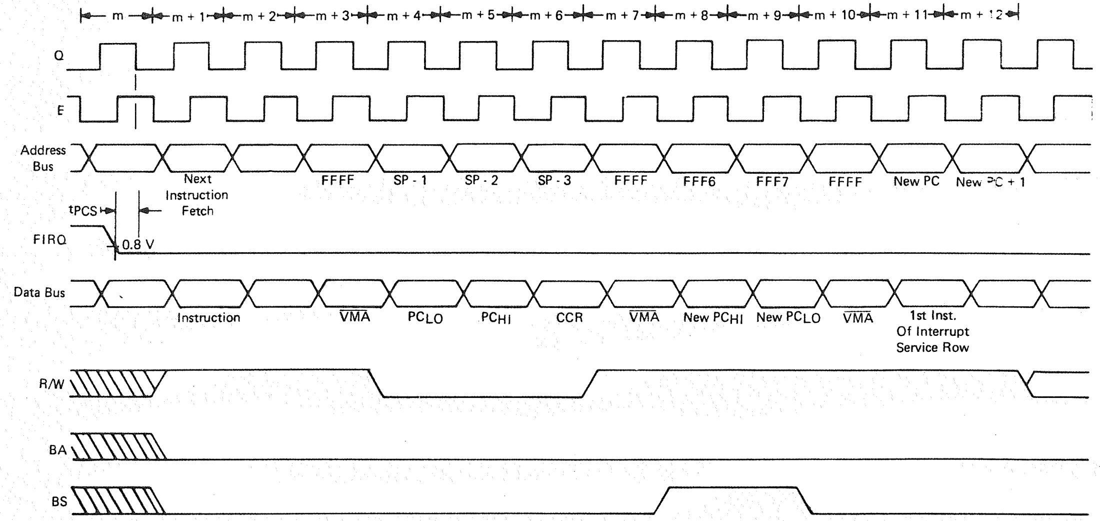


FIGURE 11 — $\overline{\text{FIRQ}}$ INTERRUPT TIMING



XTAL, EXTAL

These input pins are used to connect the on-chip oscillator to an external parallel-resonant crystal. Alternately, the pin EXTAL may be used as a TTL level input for external timing by grounding XTAL. The crystal or external frequency is 4 times the bus frequency, see Figure 8. Proper RF layout techniques should be observed in the layout of printed circuit boards.

E, Q

E is similar to the MC6800 bus timing signal $\phi 2$; Q is a quadrature clock signal which leads E. Q has no parallel on the MC6800. Addresses from the MPU will be valid with the leading edge of Q. Data is latched on the falling edge of E. Timing for E and Q is shown in Figure 12.

MRDY

This input control signal allows stretching of E to extend data-access time. When MRDY is high, E will be in normal operation. When MRDY is low, E may be stretched integral multiples of quarter (¼) bus cycles, thus allowing interface to slow memories as shown in Figure 13. A maximum stretch is 10 microseconds. During non-valid memory accesses ($\overline{\text{VMA}}$ cycles), MRDY has no effect on stretching E. This inhibits slowing the processor speed during "don't care" bus accesses.

$\overline{\text{DMA/BREQ}}$

The $\overline{\text{DMA/BREQ}}$ input provides a method of suspending execution and acquiring the MPU bus for another use as shown in Figure 14. Typical uses include DMA and dynamic memory refresh.

Transition of $\overline{\text{DMA/BREQ}}$ should occur during Q. A low level on this pin will stop instruction execution at the end of the current cycle. The MPU will acknowledge $\overline{\text{DMA/BREQ}}$ by setting BA and BS to a one. The requesting device will now have up to 15 bus cycles before the MPU retrieves the bus for self-refresh. Self-refresh requires one bus cycle with a leading and trailing dead cycle, see Figure 15.

Typically, the DMA controller will request to use the bus by asserting the $\overline{\text{DMA/BREQ}}$ pin low on the leading edge of E. When the MPU replies with BA = BS = 1, that cycle will be a dead cycle used to transfer control to the DMA controller.

False memory accesses should be prevented during any dead cycles. When BA is cleared (either as a result of $\overline{\text{DMA/BREQ}}$ = HIGH or MPU self-refresh), the DMA device should be taken off the bus.

Another dead cycle will elapse before the MPU is allowed a memory access to transfer control without contention.

MPU OPERATION

During normal operation, the MPU fetches an instruction from memory and then executes the requested function. This sequence begins at $\overline{\text{RESET}}$ and is repeated indefinitely unless altered by a special instruction or hardware occurrence. Software instructions that alter normal MPU operation are: SWI, SWI2, SWI3, CWAI, RTI and SYNC. An interrupt, $\overline{\text{HALT}}$ or $\overline{\text{DMA/BREQ}}$ can also alter the normal execution of instructions. Figure 16 illustrates the flow chart for the MC6809. The left-half of the flow chart represents normal operation; the right-half represents the flow when an interrupt or special instruction occurs.

FIGURE 12 — E/Q RELATIONSHIP

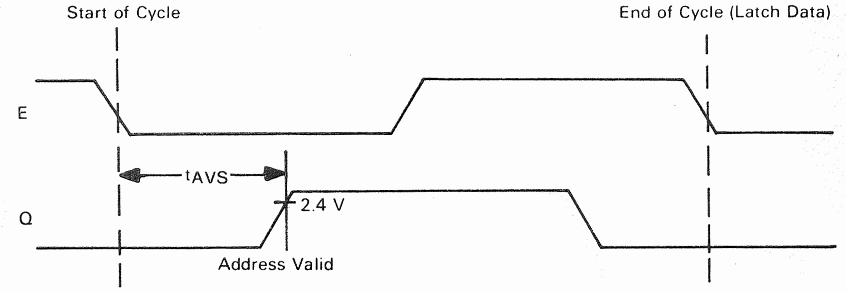


FIGURE 13 — MRDY TIMING

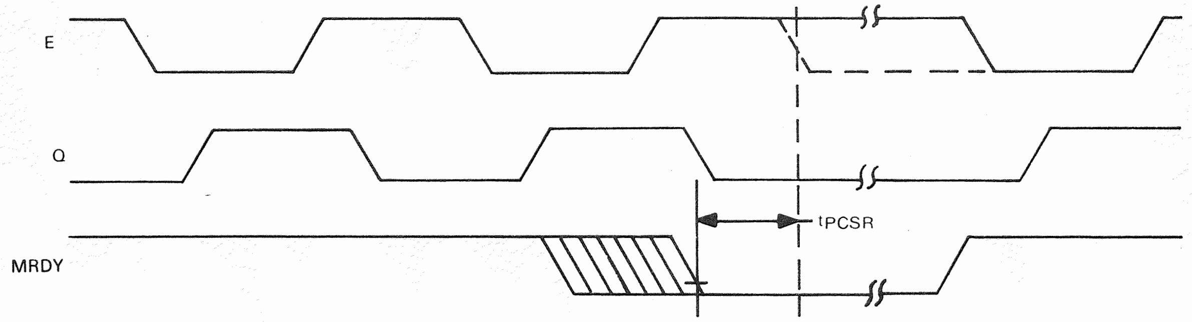
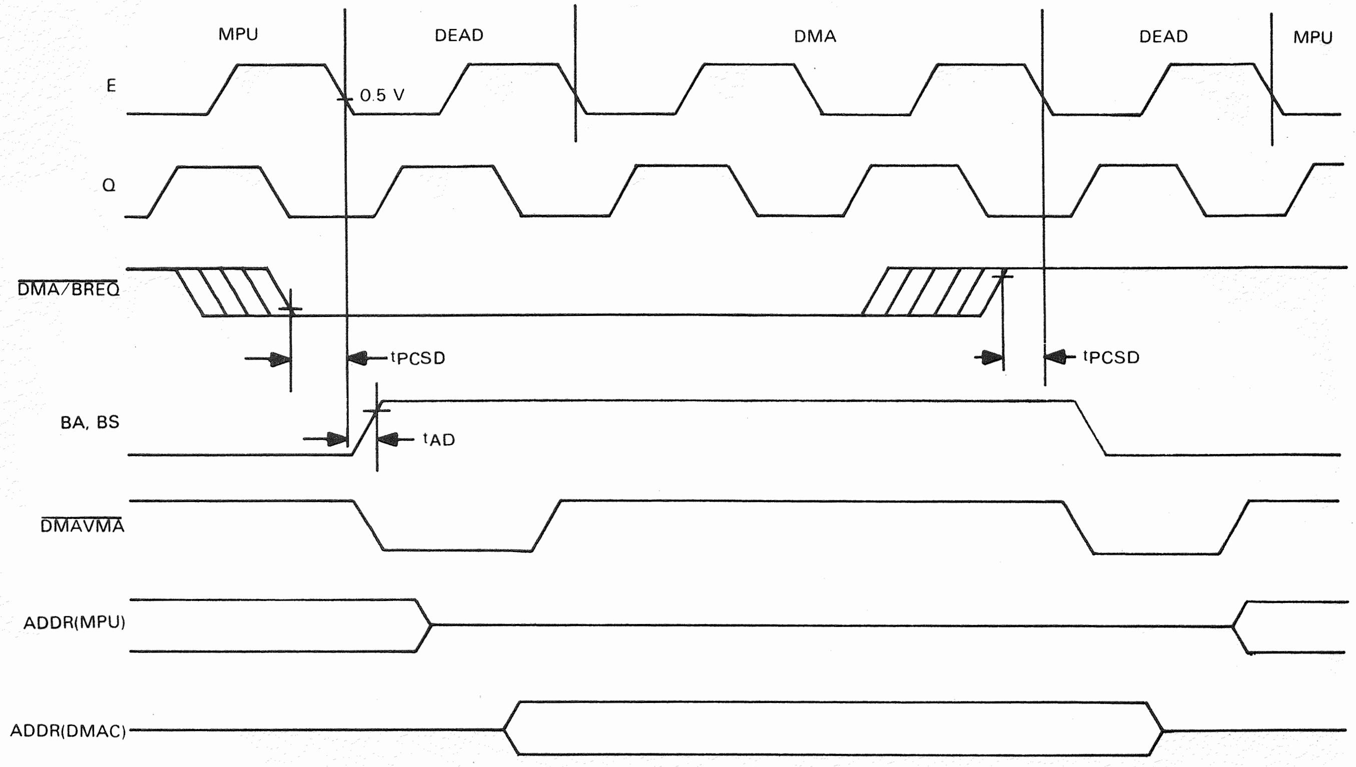


FIGURE 14 — TYPICAL DMA TIMING (‹ 14 CYCLES)



NOTE:
DMAVMA is a signal which
is developed externally, but
is a system requirement for DMA.

FIGURE 15 — AUTO-REFRESH DMA TIMING (>14 CYCLES)

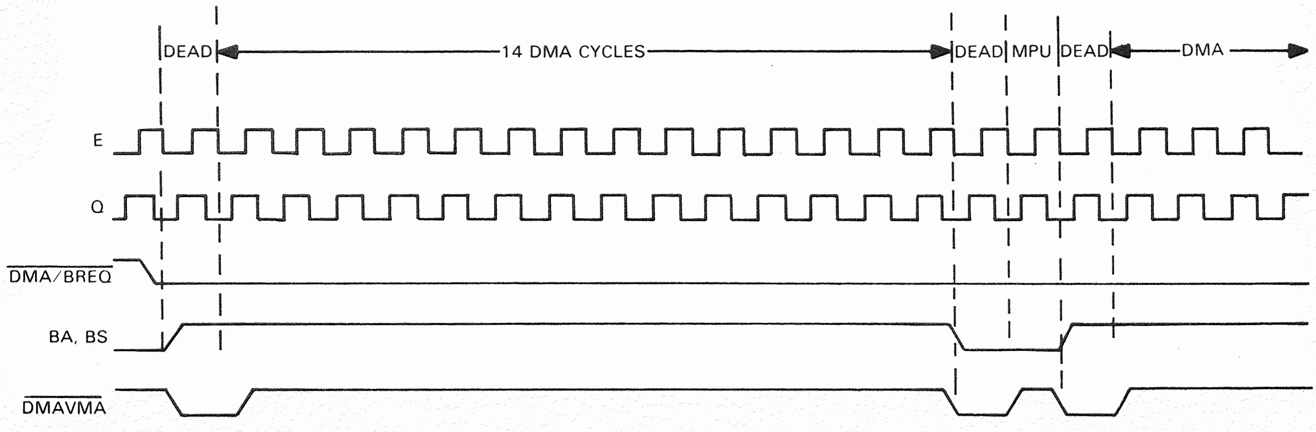
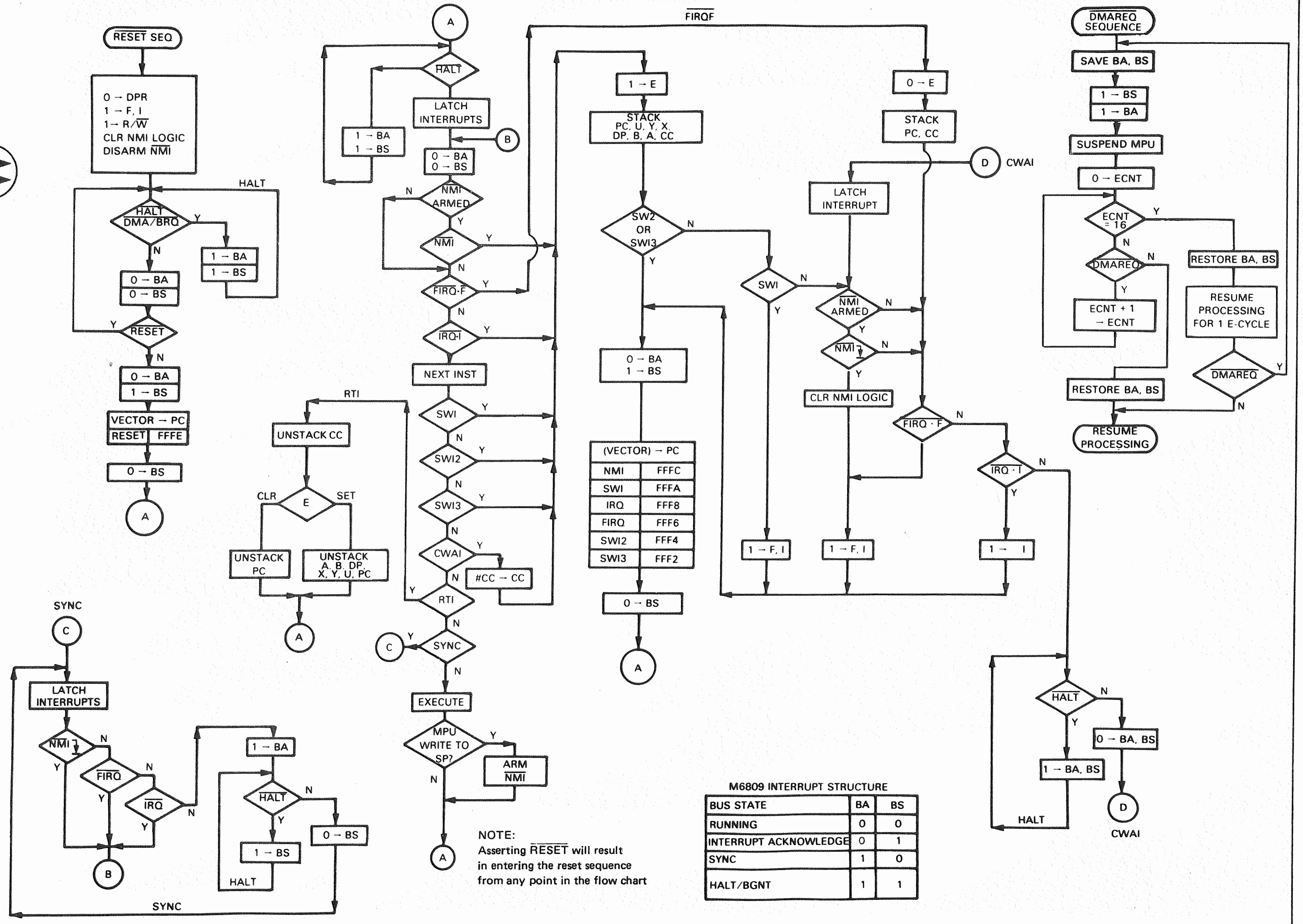


FIGURE 16 — MPU FLOWCHART



NOTE:
Asserting $\overline{RESET}$ will result in entering the reset sequence from any point in the flow chart

M6809 INTERRUPT STRUCTURE

BUS STATE	BA	BS
RUNNING	0	0
INTERRUPT ACKNOWLEDGE	0	1
SYNC	1	0
HALT/BGNT	1	1

ADDRESSING MODES

The basic instructions of any computer are greatly enhanced by the presence of powerful addressing modes. The MC6809 has the most complete set of addressing modes available on any microcomputer today. For example, the MC6809 has 59 basic instructions, however it recognizes 1464 different variations of instructions and addressing modes. The new addressing modes support modern programming techniques. The following addressing modes are available on the MC6809:

- Inherent (Includes Accumulator)
- Immediate
- Extended
  - Extended Indirect
- Direct
- Register
- Indexed
  - Zero-Offset
  - Constant Offset
  - Accumulator Offset
  - Auto Increment/Decrement
  - Indexed Indirect
- Relative
  - Short/Long Relative Branching
  - Program Counter Relative Addressing

INHERENT (INCLUDES ACCUMULATOR)

In this addressing mode, the opcode of the instruction contains all the address information necessary. Examples of Inherent Addressing are: ABX, DAA, SWI, ASRA, and CLRB.

IMMEDIATE ADDRESSING

In Immediate Addressing, the effective address of the data is the location immediately following the opcode; the data to be used in the instruction immediately follows the opcode of the instruction. The MC6809 uses both 8 and 16-bit immediate values depending on the size of argument specified by the opcode. Examples of instructions with Immediate Addressing are:

```
LDA #$20
LDX #$F000
LDY #CAT
```

Note: # signifies Immediate addressing, $ signifies hexadecimal value

EXTENDED ADDRESSING

In Extended Addressing the contents of the two bytes immediately following the opcode fully specify the 16-bit effective address used by the instruction. Note that the address generated by an extended instruction defines an absolute address and is not position independent. Examples of Extended Addressing include:

```
LDA CAT
STX MOUSE
LDD $2000
```

EXTENDED INDIRECT

As a special case of indexed addressing (discussed below), one level of indirection may be added to Extended Addressing. In Extended Indirect, the two bytes following the postbyte of an Indexed instruction contains the address of the address of the data.

```
LDA [CAT]
LDX [$FFFE]
STU [DOG]
```

DIRECT ADDRESSING

Direct addressing is similar to extended addressing except that only one byte of address follows the opcode. This byte specifies the lower 8 bits of the address to be used. The upper 8 bits of the address are supplied by the direct page register. Since only one byte of address is required in direct addressing, this mode requires less memory and executes faster than extended addressing. Of course, only 256 locations (one page) can be accessed without redefining the contents of the DP register. Since the DP register is set to $00 on Reset, direct addressing on the MC6809 is compatible with direct addressing on the M6800. Indirection is not allowed in direct addressing. Some examples of direct addressing are:

```
LDA     $30
SETDP   $10 (Assembler directive)
LDB     $1030
LDD     < CAT
```

Note: < is an assembler directive which forces direct addressing.

REGISTER ADDRESSING

Some opcodes are followed by a byte that defines a register or set of registers to be used by the instruction, this is called a POSTBYTE. Some examples of register addressing are:

```
TFR     X,Y        Transfers X into Y
EXG     A,B        Exchanges A with B
PSHS    A,B,X,Y    Push onto S Y,X,B, then A
PULU    X,Y,D      Pull from U D,X, then Y
```

INDEXED ADDRESSING

In all indexed addressing one of the pointer registers (X, Y, U, S, and sometimes PC) is used in a calculation of the effective address of the operand to be used by the instruction. Five basic types of indexing are available and are discussed below. The postbyte of an indexed instruction specifies the basic type and variation of the addressing mode as well as the pointer register to be used. Figure 17 lists the legal formats for the postbyte. Table 2 gives the assembler form and the number of cycles and bytes added to the basic values for indexed addressing for each variation.

FIGURE 17 — INDEXED ADDRESSING POSTBYTE REGISTER BIT ASSIGNMENTS

Post-Byte Register Bit								Indexed Addressing Mode
7	6	5	4	3	2	1	0	
0	R	R	X	X	X	X	X	EA = ,R ± 4 Bit Offset
1	R	R	0	0	0	0	0	,R+
1	R	R	I	0	0	0	1	,R++
1	R	R	0	0	0	1	0	,-R
1	R	R	I	0	0	1	1	,--R
1	R	R	I	0	1	0	0	EA = ,R ± 0 Offset
1	R	R	I	0	1	0	1	EA = ,R ± ACCB Offset
1	R	R	I	0	1	1	0	EA = ,R ± ACCA Offset
1	R	R	I	1	0	0	0	EA = ,R ± 7 Bit Offset
1	R	R	I	1	0	0	1	EA = ,R ± 15 Bit Offset
1	R	R	I	1	0	1	1	EA = ,R ± D Offset
1	X	X	I	1	1	0	0	EA = ,PC ± 7 Bit Offset
1	X	X	I	1	1	0	1	EA = ,PC ± 15 Bit Offset
1	R	R	1	1	1	1	1	EA = ,Address

Addressing Mode Field

Indirect Field
Sign bit when B7 = 0

Register Field
00:R = X
01:R = Y
10:R = U
11:R = S
X = Don't Care

Zero-Offset Indexed — In this mode, the selected pointer register contains the effective address of the data to be used by the instruction. This is the fastest indexing mode.

Examples are:

```
LDD 0,X
LDA 0,S
```

Constant Offset Indexed — In this mode a two's-complement offset and the contents of one of the pointer registers are added to form the effective address of the operand. The pointer register's initial content is unchanged by the addition.

Three sizes of offsets are available:

± 4-bit (-16 to +15)
± 7-bit (-128 to +127)
± 15-bit (-32768 to +32767)

The two's complement 5-bit offset is included in the postbyte and therefore is most efficient in use of bytes and cycles. The two's complement 8-bit offset is contained in a single byte following the postbyte. The two's complement 16-bit offset is in the two bytes following the postbyte. In most cases the programmer need not be concerned with the size of this offset since the assembler will select the optional size automatically.

Examples of constant-offset indexing are:

```
LDA 23,X
LDX -2,S
LDY 300,X
LDU CAT,Y
```

TABLE 2 — INDEXED ADDRESSING MODES

		Non Indirect				Indirect			
Type	Forms	Assembler Form	Postbyte OP Code	× ~	+ #	Assembler Form	Postbyte OP Code	+ ~	+ #
Constant Offset From R (Signed Offsets)	No Offset	,R	1RR00100	0	0	[,R]	1RR10100	3	0
	5 Bit Offset	n, R	0RRnnnnn	1	0	defaults to 8-bit			
	8 Bit Offset	n, R	1RR01000	1	1	[n, R]	1RR11000	4	1
	16 Bit Offset	n, R	1RR01001	4	2	[n, R]	1RR11001	7	2
Accumulator Offset From R (Signed Offsets)	A — Register Offset	A, R	1RR00110	1	0	[A, R]	1RR10110	4	0
	B — Register Offset	B, R	1RR00101	1	0	[B, R]	1RR10101	4	0
	D — Register Offset	D, R	1RR01011	4	0	[D, R]	1RR11011	7	0
Auto Increment/Decrement R	Increment By 1	,R+	1RR00000	2	0	not allowed			
	Increment By 2	,R++	1RR00001	3	0	[,R++]	1RR10001	6	0
	Decrement By 1	,-R	1RR00010	2	0	not allowed			
	Decrement By 2	,--R	1RR00011	3	0	[,--R]	1RR10011	6	0
Constant Offset From PC	8 Bit Offset	n, PCR	1XX01100	1	1	[n, PCR]	1XX11100	4	1
	16 Bit Offset	n, PCR	1XX01101	5	2	[n, PCR]	1XX11101	8	2
Extended Indirect	16 Bit Address	—	—	—	—	[n]	10011111	5	2

R = X, Y, U or S X = 00 Y = 01
X = Don't Care U = 10 S = 11

$\underset{\sim}{+}$ and $\underset{\#}{+}$ Indicate the number of additional cycles and bytes for the particular variation.

Accumulator-Offset Indexed — This mode is similar to constant offset indexed except that the two's-complement value in one of the accumulators (A, B or D) and the content of one of the pointer registers are added to form the effective address of the operand. The contents of both the accumulator and the pointer register are unchanged by the addition. The postbyte specifies which accumulator to use as an offset and no additional bytes are required. The advantage of an accumulator offset is that the value of the offset can be calculated by a program at run-time.

Some examples are:

```
LDA B,Y
LDX D,Y
LEAX B,X
```

Auto Increment/Decrement Indexed — In the auto increment addressing mode, the pointer register contains the address of the operand. Then, after the pointer register is used it is incremented by one or two. This addressing mode is useful in stepping through tables, moving data, or for the creation of software stacks. In auto decrement, the pointer register is decremented prior to use as the address of the data. The use of auto decrement is similiar to that of auto increment but the tables, etc. are scanned from the high to low addresses. The size of the increment/decrement can be either one or two to allow for tables of either 8 or 16-bit data to be accessed and is selectable by the programmer. The pre-decrement, post-increment nature of these modes allow them to be used to create additional software stacks that behave identically to the U and S stacks.

Some examples of the auto increment/decrement addressing modes are:

```
LDA ,X+
STD ,Y++
LDB ,-Y
LDX ,--S
```

INDEXED INDIRECT

All of the indexing modes with the exception of auto increment/decrement by one, or a ± 4-bit offset may have an additional level of indirection specified. In Indirect addressing, the effective address is contained at the location specified by the content of the Index register plus any offset. In the example below, the A accumulator is loaded indirectly using an effective address calculated from the Index register and an offset.

```
                         Before Execution
                        A = XX (don't care)
                            X = $F000
$0100   LDA     [10, X]   EA is now $F010

$F010   $F1               F150 is now the
$F011   $50               new EA

$F150   $AA
                          After Execution
                    A = $AA Actual Data Loaded
```

All modes of indexed indirect are included except those which are meaningless (e.g. auto increment/decrement by 1 indirect). Some examples of indexed indirect are:

```
LDA   [,X]
LDD   [10,S]
LDA   [B,Y]
LDD   [,X++]
```

RELATIVE ADDRESSING

The byte(s) following the branch opcode is (are) treated as a signed offset which is added to the program counter. If the branch condition is true then the calculated address (PC + signed offset) is loaded into the program counter. Program execution continues at the new location as indicated by the PC; Short (1 byte offset) and long (2 bytes offset) relative addressing modes are available. All of memory can be reached in long relative addressing as an effective address is interpreted modulo 2^{16}. Some examples of relative addressing are:

```
          BEQ    CAT       (short)
          BGT    DOG       (short)
CAT       LBEQ   RAT       (long)
DOG       LBGT   RABBIT    (long)
          •
          •
          •
RAT       NOP
RABBIT    NOP
```

PROGRAM COUNTER RELATIVE

The PC can be used as the pointer register with 8 or 16-bit signed offsets. As in relative addressing the offset is added to the current PC to create the effective address. The effective address is then used as the address of the operand or data. Program Counter Relative Addressing is used for writing position independent programs. Tables related to a particular routine will maintain the same relationship after the routine is moved, if referenced relative to the Program Counter. Examples are:

```
LDA    CAT, PCR
LEAX   TABLE, PCR
```

Since program counter relative is a type of indexing, an additional level of indirection is available.

```
LDA   [CAT, PCR]
LDU   [DOG, PCR]
```

MC6809 INSTRUCTION SET

The instruction set of the MC6809 is similar to that of the MC6800 and is upward compatible at the source code level. The number of opcodes has been reduced from 72 to 59, but because of the expanded architecture and additional addressing modes, the number of available opcodes (with different addressing modes) has risen from 197 to 1464.

Some of the new instructions and addressing modes are described in detail below:

PSHU/PSHS

The push instructions have the capability of pushing onto either the hardware stack (S) or user stack (U) any or all of the MPU registers with a single instruction.

PULU/PULS

The pull instructions have the same capability of the push instruction, in reverse order. The byte immediate following the push or pull opcode determines which register or registers are to be pushed or pulled. The actual PUSH/PULL sequence is fixed; each bit defines a unique register to push or pull as shown in Figure 16.

TFR/EXG

Within the MC6809, any register may be transferred to or exchanged with another of like-size, i.e. 8-bit to 8-bit or 16-bit to 16-bit. Bits 4-7 of postbyte define the source register, while bits 0-3 represent the destination register. These are denoted as follows:

0000 — D	0101 — PC
0001 — X	1000 — A
0010 — Y	1001 — B
0011 — U	1010 — CC
0100 — S	1011 — DP

Note: All other combinations are undefined and INVALID.

Load Effective Address

The LEA works by calculating the effective address used in an indexed instruction and stores that address value, rather than the data at that address, in a pointer register. This makes all the features of the internal addressing hardware available to the programmer. Some of the implications of this instruction are illustrated in the following table of examples:

The LEA instruction also allows the user to access data in a position independent manner. For example:

```
        LEAX    MSG1, PCR
        LBSR    PDATA (Print message routine)


     MSG1 FCC   'MESSAGE'
```

FIGURE 16 — PUSH/PULL POSTBYTE

```
           ← Pull Order   Push Order →
PC    U    Y    X    DP    B    A    CC     PSHS/PULS
   FFFF ...... ← increasing memory address → ...... 0000
PC    S    Y    X    DP    B    A    CC     PSHU/PULU
```

TABLE 3 — LEA EXAMPLES

Instruction		Operation		Comment
LEAX	10, X	X + 10	→ X	Adds 5-bit constant 10 to X
LEAX	500, X	X + 500	→ X	Adds 16-bit constant 500 to X
LEAY	A, Y	Y + A	→ Y	Adds 8-bit accumulator to Y
LEAY	D, Y	Y + D	→ Y	Adds 16-bit D accumulator to Y
LEAU	-10, U	U - 10	→ U	Subtracts 10 from U
LEAS	-10, S	S - 10	→ S	Used to reserve area on stack
LEAS	10, S	S + 10	→ S	Used to 'clean up' stack
LEAX	5, S	S + 5	→ X	Transfers as well as adds

This sample program prints "message". By writing MSG1,PCR, the assembler computes the distance between the present address and MSG1. This result is placed as a constant into the LEAX instruction which will be indexed from the PC value at the time of execution. No matter where the code is located, when it is executed, the computed offset from the PC will put the absolute address of MSG1 into the X pointer register. This code is totally position independent.

MUL

Multiplies the unsigned binary numbers in the A and B accumulator and places the unsigned result into the 16-bit D accumulator. This unsigned multiply also allows multiple-precision multiplications.

Long And Short Relative Branches

The MC6809 has the capability of program counter relative branching throughout the entire memory map. In this mode, if the branch is to be taken, the 8 or 16-bit signed offset is added to the value of the program counter to be used as the effective address. This allows the program to branch anywhere in the 64K memory map. Position independent code can be easily generated through the use of relative branching. Both short (8-bit) and long (16-bit) branches are available.

SYNC

After encountering a Sync instruction, the MPU enters a Sync state, stops processing instructions and waits for an interrupt. If the pending interrupt is non-maskable (NMI) or maskable (FIRQ, IRQ) with its mask bit (F or I) clear, the processor will clear the Sync state and perform the normal interrupt stacking and service routine. Since FIRQ and IRQ are not edge-triggered, a low level with a minimum duration of three cycles is required to assure that the interrupt will be taken. If the pending interrupt is maskable (FIRQ, IRQ) with its mask bit (F or I) set, the processor will clear the Sync state and continue processing in sequence. Figure 18 depicts Sync timing.

Software Interrupts

A Software Interrupt is an instruction which will cause an interrupt, and its associated vector fetch. These Software Interrupts are useful in operating system calls, software debugging, trace operations, memory mapping, and software development systems. Three levels of SWI are available on this MC6809, and are prioritized in the following order: SWI, SWI2, SWI3.

16-Bit Operations

The MC6809 has the capability of processing 16-bit data. These instructions include loads, stores, compares, adds, subtracts, transfers, exchanges, pushes and pulls.

CYCLE-BY-CYCLE OPERATION

The address bus cycle-by-cycle performance chart illustrates the memory-access sequence corresponding to each possible instruction and addressing mode in the MC6809. Each instruction begins with an opcode fetch. While that opcode is being internally decoded, the next program byte is always fetched. (Most instructions will use the next byte, so this technique cnsiderably speeds throughput). Next, the operation of each opcode will follow the flow chart. $\overline{VMA}$ is an indication of FFFF$_{16}$ on the address bus, $R/\overline{W}$ = 1 and BS = 0. The following examples illustrate the use of the chart; see Figure 19.

LBSR (Branch taken)

Cycle #	
1	opcode Fetch
2	opcode +
3	opcode +
4	$\overline{VMA}$
5	$\overline{VMA}$
6	ADDR
7	$\overline{VMA}$
8	STACK (write)
9	STACK (write)

DEC (Extended)

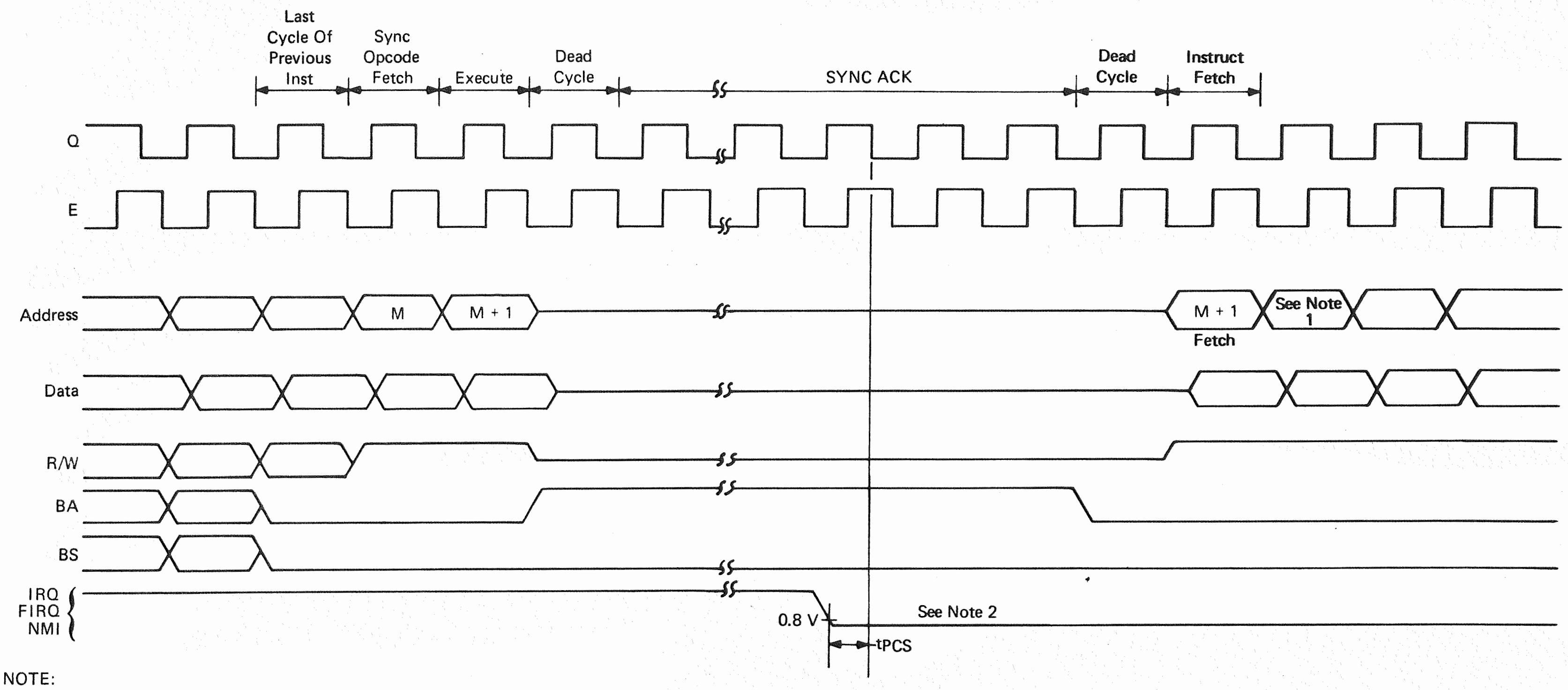
1	opcode Fetch
2	opcode +
3	opcode +
4	$\overline{VMA}$
5	ADDR (read)
6	$\overline{VMA}$
7	ADDR (write)

MC6809 INSTRUCTION SET TABLES

The instructions of the MC6809 have been broken down into five different categories. They are as follows:

8-Bit operation (Table 4)
16-Bit operation (Table 5)
Index register/stack pointer instructions (Table 6)
Relative branches (long and short) (Table 7)
Miscellaneous instructions (Table 8)
Hexadecimal Value instructions (Table 9)

FIGURE 18 — SYNC TIMING



NOTE:

1. If the mask bit is set when the interrupt is requested processing will continue with instruction execution fetched from previous step. However, if an NMI or an unmasked FIRQ or IRQ caused interrupt, the address placed on bus from previous cycle (M + 1) remains on bus and processing continues with this cycle as (m + 1) or (n + 1) of interrupt timing.
2. If mask bits are clear IRQ & FIRQ must be held low for three cycles to guarantee interrupt to be taken, although only one cycle is necessary to bring the processor out of SYNC.

FIGURE 19 — ADDRESS BUS CYCLE-BY-CYCLE PERFORMANCE

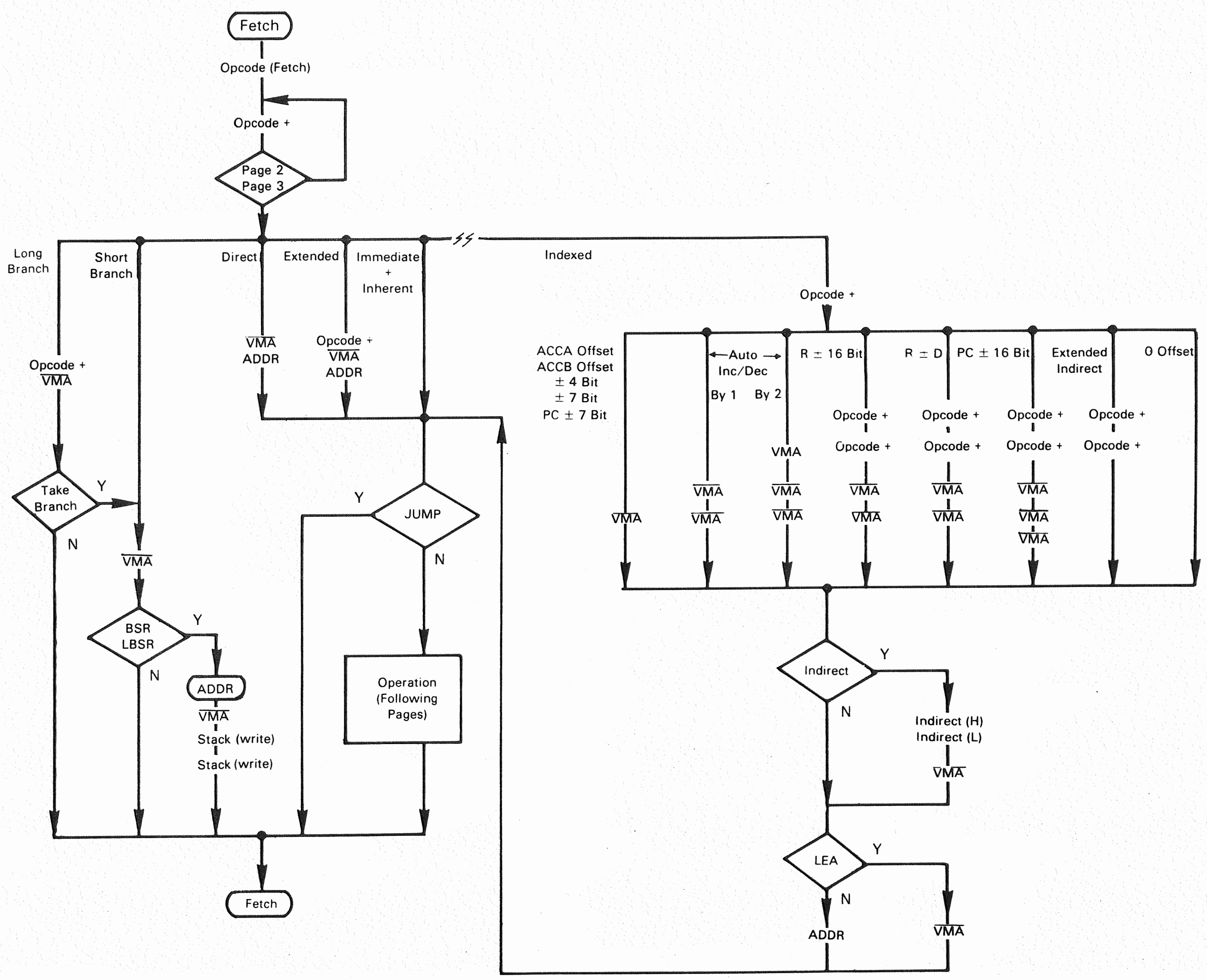


FIGURE 19 — ADDRESS BUS CYCLE-BY-CYCLE PERFORMANCE (CONT.)

Inherent Page

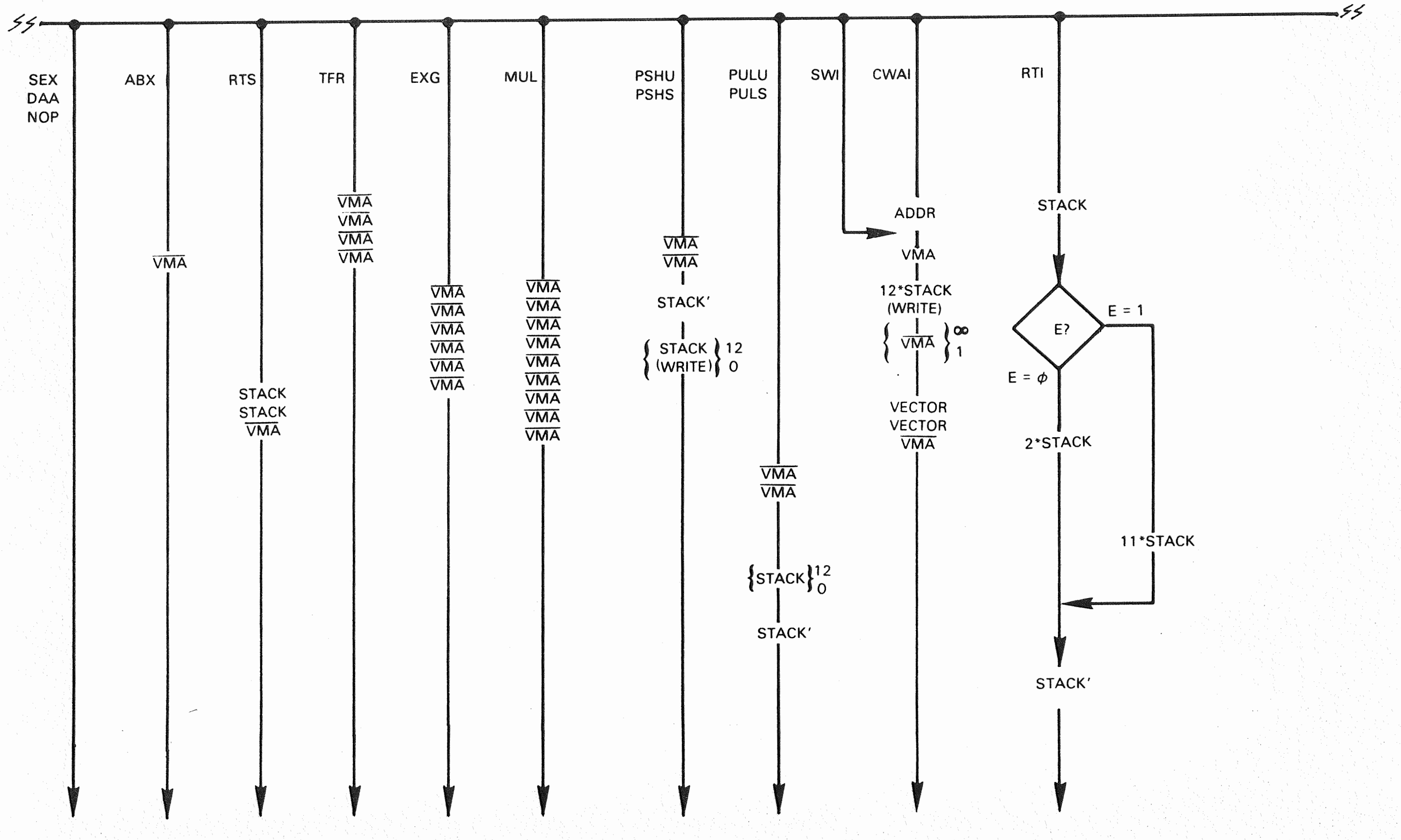


FIGURE 19 — ADDRESS BUS CYCLE-BY-CYCLE PERFORMANCE (CONT.)

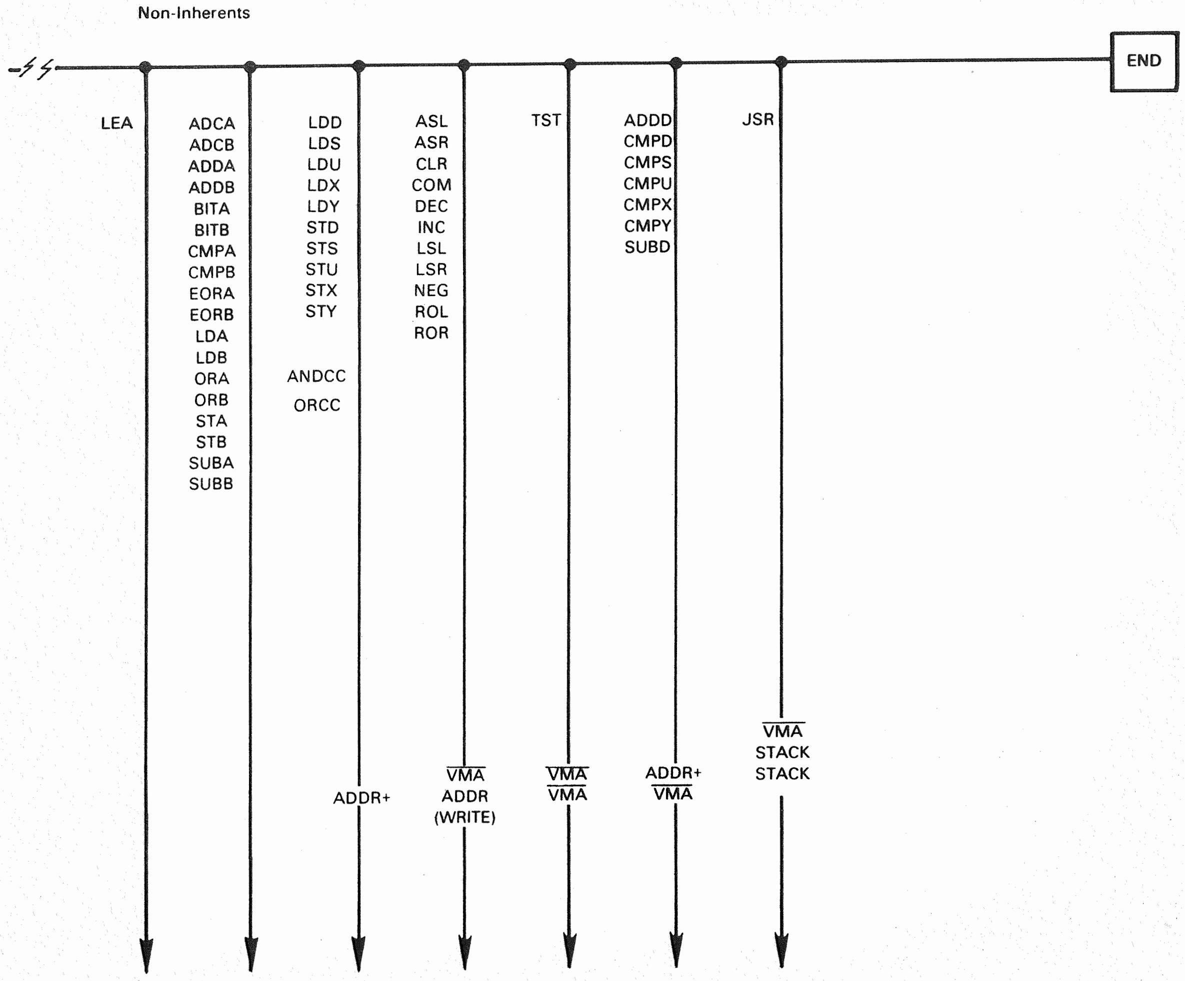


TABLE 4 — 8-BIT ACCUMULATOR AND MEMORY INSTRUCTIONS

Mnemonic(s)	Operation
ADCA, ADCB	Add memory to accumulator with carry
ADDA, ADDB	Add memory to accumulator
ANDA, ANDB	And memory with accumulator
ASL, ASLA, ASLB	Arithmetic shift of accumulator or memory left
ASR, ASRA, ASRB	Arithmetic shift of accumulator or memory right
BITA, BITB	Bit test memory with accumulator
CLR, CLRA, CLRB	Clear accumulator or memory location
CMPA, CMPB	Compare memory from accumulator
COM, COMA, COMB	Complement accumulator or memory location
DAA	Decimal adjust A-accumulator
DEC, DECA, DECB	Decrement accumulator or memory location
EORA, EORB	Exclusive or memory with accumulator
EXG R1, R2	Exchange R1 with R2 (R1, R2 = A, B, CC, DP)
INC, INCA, INCB	Increment accumulator or memory location
LDA, LDB	Load accumulator from memory
LSL, LSLA, LSLB	Logical shift left accumulator or memory location
LSR, LSRA, LSRB	Logical shift right accumulator or memory location
MUL	Unsigned multiply (A x B → D)
NEG, NEGA, NEGB	Negate accumulator or memory
ORA, ORB	Or memory with accumulator
ROL, ROLA, ROLB	Rotate accumulator or memory left
ROR, RORA, RORB	Rotate accumulator or memory right
SBCA, SBCB	Subtract memory from accumulator with borrow
STA, STB	Store accumulator to memory
SUBA, SUBB	Subtract memory from accumulator
TST, TSTA, TSTB	Test accumulator or memory location
TFR, R1, R2	Transfer R1 to R2 (R1, R2 = A, B, CC, DP)

NOTE: A, B, CC, or DP may be pushed to (pulled from) either stack with PSHS, PSHU, (PULS, PULU) instructions.

TABLE 5 — 16-BIT ACCUMULATOR AND MEMORY INSTRUCTIONS

Mnemonic(s)	Operation
ADDD	Add memory to D accumulator
CMPD	Compare memory from D accumulator
EXG D, R	Exchange D with X, Y, S, U or PC
LDD	Load D accumulator from memory
SEX	Sign Extend B accumulator into A accumulator
STD	Store D accumulator to memory
SUBD	Subtract memory from D accumulator
TFR D, R	Transfer D to X, Y, S, U or PC
TFR R, D	Transfer X, Y, S, U or PC to D

TABLE 6 — INDEX REGISTER/STACK POINTER INSTRUCTIONS

Mnemonic(s)	Operation
CMPS, CMPU	Compare memory from stack pointer
CMPX, CMPY	Compare memory from index register
EXG R1, R2	Exchange D, X, Y, S, U, or PC with D, X, Y, S, U or PC
LEAS, LEAU	Load effective address into stack pointer
LEAX, LEAY	Load effective address into index register
LDS, LDU	Load stack pointer from memory
LDX, LDY	Load index register from memory
PSHS	Push any register(s) onto hardware stack (except S)
PSHU	Push any register(s) onto user stack (except U)
PULS	Pull any register(s) from hardware stack (except S)
PULU	Pull any register(s) from hardware stack (except U)
STS, STU	Store stack pointer to memory
STX, STY	Store index register to memory
TFR R1, R2	Transfer D, X, Y, S, U or PC to D, X, Y, S, U or PC
ABX	Add B accumulator to X (unsigned)

TABLE 7 — BRANCH INSTRUCTIONS

Mnemonic(s)	Operation
BCC, LBCC	Branch if carry clear
BCS, LBCS	Branch if carry set
BEQ, LBEQ	Branch if equal
BGE, LBGE	Branch if greater than or equal (signed)
BGT, LBGT	Branch if greater (signed)
BHI, LBHI	Branch if higher (unsigned)
BHS, LBHS	Branch if higher or same (unsigned)
BLE, LBLE	Branch if less than or equal (signed)
BLO, LBLO	Branch if lower (unsigned)
BLS, LBLS	Branch if lower or same (unsigned)
BLT, LBLT	Branch if less than (signed)
BMI, LBMI	Branch if minus
BNE, LBNE	Branch if not equal
BPL, LBPL	Branch if plus
BRA, LBRA	Branch always
BRN, LBRN	Branch never
BSR, LBSR	Branch to subroutine
BVC, LBVC	Branch if overflow clear
BVS, LBVS	Branch if overflow set

TABLE 8 — MISCELLANEOUS INSTRUCTIONS

Mnemonic(s)	Operation
ANDCC	AND condition code register
CWAI	AND condition code register, then wait for interrupt
NOP	No operation
ORCC	OR condition code register
JMP	Jump
JSR	Jump to subroutine
RTI	Return from interrupt
RTS	Return from subroutine
SWI, SWI2, SWI3	Software interrupt (absolute indirect)
SYNC	Synchronize with interrupt line

TABLE 9 — HEXADECIMAL VALUES OF MACHINE CODES

OP Mnem	Mode	~	#	OP Mnem	Mode	~	#	OP Mnem	Mode	~	#
00 NEG	Direct	6	2	30 LEAX	Indexed	4+	2+	60 NEG	Indexed	6+	2+
01 *				31 LEAY		4+	2+	61 *			
02 *				32 LEAS		4+	2+	62 *			
03 COM		6	2	33 LEAU	Indexed	4+	2+	63 COM		6+	2+
04 LSR		6	2	34 PSHS	Inherent	5+	2	64 LSR		6+	2+
05 *				35 PULS		5+	2	65 *			
06 ROR		6	2	36 PSHU		5+	2	66 ROR		6+	2+
07 ASR		6	2	37 PULU		5+	2	67 ASR		6+	2+
08 ASL/LSL		6	2	38 *				68 ASL/LSL		6+	2+
09 ROL		6	2	39 RTS		5	1	69 ROL		6+	2+
0A DEC		6	2	3A ABX		3	1	6A DEC		6+	2+
0B *				3B RTI		6/15	1	6B *			
0C INC		6	2	3C CWAI		20	2	6C INC		6+	2+
0D TST		6	2	3D MUL		11	1	6D TST		6+	2+
0E JMP		3	2	3E *				6E JMP		3+	2+
0F CLR	Direct	6	2	3F SWI	Inherent	19	1	6F CLR	Indexed	6+	2+
10 Page 2	—	—	—	40 NEGA	Inherent	2	1	70 NEG	Extended	7	3
11 Page 3	—	—	—	41 *				71 *			
12 NOP	Inherent	2	1	42 *				72 *			
13 SYNC	Inherent	2	1	43 COMA		2	1	73 COM		7	3
14 *				44 LSRA		2	1	74 LSR		7	3
15 *				45 *				75 *			
16 LBRA	Relative	5	3	46 RORA		2	1	76 ROR		7	3
17 LBSR	Relative	9	3	47 ASRA		2	1	77 ASR		7	3
18 *				48 ASLA/LSLA		2	1	78 ASL/LSL		7	3
19 DAA	Inherent	2	1	49 ROLA		2	1	79 ROL		7	3
1A ORCC	Immed	3	2	4A DECA		2	1	7A DEC		7	3
1B *				4B *				7B *			
1C ANDCC	Immed	3	2	4C INCA		2	1	7C INC		7	3
1D SEX	Inherent	2	1	4D TSTA		2	1	7D TST		7	3
1E EXG		8	2	4E *				7E JMP		4	3
1F TFR	Inherent	6	2	4F CLRA	Inherent	2	1	7F CLR	Extended	7	3
20 BRA	Relative	3	2	50 NEGB	Inherent	2	1	80 SUBA	Immed	2	2
21 BRN		3	2	51 *				81 CMPA		2	2
22 BHI		3	2	52 *				82 SBCA		2	2
23 BLS		3	2	53 COMB		2	1	83 SUBD		4	3
24 BHS/BCC		3	2	54 LSRB		2	1	84 ANDA		2	2
25 BLO/BCS		3	2	55 *				85 BITA		2	2
26 BNE		3	2	56 *				86 LDA		2	2
27 BEQ		3	2	56 RORB		2	1	87 *			
28 BVC		3	2	57 ASRA		2	1	88 EORA		2	2
29 BVS		3	2	58 ASLB/LSLB		2	1	89 ADCA		2	2
2A BPL		3	2	59 ROLB		2	1	8A ORA		2	2
2B BMI		3	2	5A DECB		2	1	8B ADDA		2	2
2C BGE		3	2	5B *				8C CMPX	Immed	4	3
2D BLT		3	2	5C INCB		2	1	8D BSR	Relative	7	2
2E BGT		3	2	5D TSTB		2	1	8E LDX	Immed	3	3
2F BLE	Relative	3	2	5E *				8F *			
				5F CLRB	Inherent	2	1				

Legend:

~ Number of MPU cycles (less possible push/pull or indexed-mode cycles)

# Number of program bytes

* Denotes unused opcode

TABLE 9 – HEXADECIMAL VALUES OF MECHANICAL CODES (CONTINUED)

OP Mnem	Mode	~	#
90 SUBA	Direct	4	2
91 CMPA		4	2
92 SBCA		4	2
93 SUBD		6	2
94 ANDA		4	2
95 BITA		4	2
96 LDA		4	2
97 STA		4	2
98 EORA		4	2
99 ADCA		4	2
9A ORA		4	2
9B ADDA		4	2
9C CMPX		6	2
9D JSR		7	2
9E LDX		5	2
9F STX	Direct	5	2
A0 SUBA	Indexed	4+	2+
A1 CMPA		4+	2+
A2 SBCA		4+	2+
A3 SUBD		6+	2+
A4 ANDA		4+	2+
A5 BITA		4+	2+
A6 LDA		4+	2+
A7 STA		4+	2+
A8 EORA		4+	2+
A9 ADCA		4+	2+
AA ORA		4+	2+
AB ADDA		4+	2+
AC CMPX		6+	2+
AD JSR		7+	2+
AE LDX		5+	2+
AF STX	Indexed	5+	2+
B0 SUBA	Extended	5	3
B1 CMPA		5	3
B2 SBCA		5	3
B3 SUBD		7	3
B4 ANDA		5	3
B5 BITA		5	3
B6 LDA		5	3
B7 STA		5	3
B8 EORA		5	3
B9 ADCA		5	3
BA ORA		5	3
BB ADDA		5	3
BC CMPX		7	3
BD JSR		8	3
BE LDX		6	3
BF STX	Extended	6	3
C0 SUBB	Immed	2	2
C1 CMPB		2	2
C2 SBCB		2	2
C3 ADDD		4	3
C4 ANDB		2	2
C5 BITB	Immed	2	2

OP Mnem	Mode	~	#
C6 LDB	Immed	2	2
C7 *			
C8 EORB		2	2
C9 ADCB		2	2
CA ORB		2	2
CB ADDB		2	2
CC LDD		3	3
CD *			
CE LDU	Immed	3	3
CF *			
D0 SUBB	Direct	4	2
D1 CMPB		4	2
D2 SBCB		4	2
D3 ADDD		6	2
D4 ANDB		4	2
D5 BITB		4	2
D6 LDB		4	2
D7 STB		4	2
D8 EORB		4	2
D9 ADCB		4	2
DA ORB		4	2
DB ADDB		4	2
DC LDD		5	2
DD STD		5	2
DE LDU		5	2
DF STU	Direct	5	2
E0 SUBB	Indexed	4+	2+
E1 CMPB		4+	2+
E2 SBCB		4+	2+
E3 ADDD		6+	2+
E4 ANDB		4+	2+
E5 BITB		4+	2+
E6 LDB		4+	2+
E7 STB		4+	2+
E8 EORB		4+	2+
E9 ADCB		4+	2+
EA ORB		4+	2+
EB ADDB		4+	2+
EC LDD		5+	2+
ED STD		5+	2+
EE LDU		5+	2+
EF STU	Indexed	5+	2+
F0 SUBB	Extended	5	3
F1 CMPB		5	3
F2 SBCB		5	3
F3 ADDD		7	3
F4 ANDB		5	3
F5 BITB		5	3
F6 LDB		5	3
F7 STB		5	3
F8 EORB		5	3
F9 ADCB		5	3
FA ORB		5	3
FB ADDB	Extended	5	3

OP Mnem	Mode	~	#
FC LDD	Extended	6	3
FD STD		6	3
FE LDU		6	3
FF STU	Extended	6	3

OP	Mnem	Mode	~	#
1021	LBRN	Relative	5	4
1022	LBHI		5(6)	4
1023	LBLS		5(6)	4
1024	LBHS/LBCC		5(6)	4
1025	LBCS/LBLO		5(6)	4
1026	LBNE		5(6)	4
1027	LBEQ		5(6)	4
1028	LBVC		5(6)	4
1029	LBVS		5(6)	4
102A	LBPL		5(6)	4
102B	LBMI		5(6)	4
102C	LBGE		5(6)	4
102D	LBLT	Relative	5(6)	4
102E	LBGT	Relative	5(6)	4
102F	LBLE	Relative	5(6)	4
103F	SWI/2	Inherent	20	2
1083	CMPD	Immed	5	4
108C	CMPY		5	4
108E	LDY	Immed	4	4
1093	CMPD	Direct	7	3
109C	CMPY		7	3
109E	LDY		6	3
109F	STY	Direct	6	3
10A3	CMPD	Indexed	7+	3+
10AC	CMPY		7+	3+
10AE	LDY		6+	3+
10AF	STY	Indexed	6+	3+
10B3	CMPD	Extended	8	4
10BC	CMPY		8	4
10BE	LDY		7	4
10BF	STY	Extended	7	4
10CE	LDS	Immed	4	4
10DE	LDS	Direct	6	3
10DF	STS	Direct	6	3
10EE	LDS	Indexed	6+	3+
10EF	STS	Indexed	6+	3+
10FE	LDS	Extended	7	4
10FF	STS	Extended	7	4
113F	SWI/3	Inherent	20	2
1183	CMPU	Immed	5	4
118C	CMPS	Immed	5	4
1193	CMPU	Direct	7	3
119C	CMPS	Direct	7	3
11A3	CMPU	Indexed	7+	3+
11AC	CMPS	Indexed	7+	3+
11B3	CMPU	Extended	8	4
11BC	CMPS	Extended	8	4

NOTE: All unused opcodes are both undefined and illegal.