THE

FACTS

Terry K. Laraway
SMALL GRAFX ETC.
41 N.W. Doncee Drive
Bremerton, WA 98310-9110

for the

TRS-80™ COLOR COMPUTER

TRS-80 IS A TRADEMARK OF THE TANDY CORPORATION

The

FACTS

for the

TRS-80 Color Computer

by SPECTRAL ASSOCIATES

3418 South 90th Street

Tacoma, WA 98409

Copyright (c) 1983

ALL RIGHTS RESERVED

The text of this publication, or any part thereof, may not be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, storage in an information retrieval system, or otherwise, without the prior written permission of the publisher.

Printed in the United States of America

First Printing	Nov 1981
Second Printing	Feb 1982
Third Printing	Aug 1982
Fourth Printing	Nov 1982
Fifth Printing	Jul 1983

Every effort has been made to insure the accuracy of the contents of this book and all facts contained herein are believed to be true. Spectral Associates does not guarantee the accuracy of the information contained in this book, nor does it assume any liability arising out of the application or use of any of the information herein.

L I S T O F F I G U R E S

FIGURE 1 BLOCK DIAGRAM

FIGURE 2 SOUND CIRCUITRY

FIGURE 3 MULTIPLEXER CONTROLS

FIGURE 4 SINGLE BIT SOUND

FIGURE 5 VDG MODE CONTROL

FIGURE 6 PIA INITIAL SETUP

FIGURE 7 1793 REGISTERS

FIGURE 8 1793 COMMANDS

FIGURE 9 1793 CONTROL REGISTER

TABLE OF CONTENTS

1. INTRODUCTION. 1
2. EVOLUTION OF THE CC 2
3. GENERAL DESCRIPTION 3
4. GENERAL COMPONENT DESCRIPTION 4
5. MICROPROCESSOR UNIT 12
6. SYNCHRONOUS ADDRESS MULTIPLEXER 13
7. VIDEO DISPLAY GENERATOR 17
8. PERIPHERAL INTERFACE ADAPTER. 20
9. INTERRUPTS. 23
10. BASIC 25
11. 32K MODEL COMPUTER. 26
12. 64K 28
13. DISK DRIVE. 29

COMMENTED BASIC ROUTINES. . . . . .APPENDIX A
COMMENTED DSKCON ROUTINE. . . . . .APPENDIX B
EXBAS & DISK TOKEN TABLES . . . . . .APPENDIX C
MPU DATA SHEET. . . . . .APPENDIX D
SAM DATA SHEET. . . . . .APPENDIX E
VDG DATA SHEET. . . . . .APPENDIX F
PIA DATA SHEET. . . . . .APPENDIX G
SOUND DRIVER ROUTINE. . . . . .APPENDIX H
SCHEMATICS. . . . . .APPENDIX I

SECTION 1
INTRODUCTION

The TRS 80 Color Computer is one of the best small color graphics computers available on the market today. One of the big drawbacks of this computer for the end user, however, is the extreme lack of documentation which has been available for its use. There is virtually nothing which explains in any great detail the inner workings of Color BASIC or the computer's hardware which will allow the user to build programs in order to take full advantage of the capabilities of the Color Computer.

THE FACTS is the first document to provide such information so that the user will be able to take advantage of all of the benefits of the Color Computer. THE FACTS is aimed at machine language users as opposed to the BASIC user. The internal workings of the Color Computer do not easily lend themselves to being manipulated with BASIC programs as BASIC is too slow and clumsy to fully utilize all of the features of the Color Computer. Any fast action, quick game will have to use machine language and will have to access the internal registers and functions of the Color Computer using machine language.

To this end, THE FACTS will attempt to explain and describe in detail how the user will avail himself of the internal features of the Color Computer. It will be broken up into two general sections: hardware and software; however, the primary emphasis will be upon the hardware capabilities and circuits which are built into the Color Computer and which have not been described in any detail. THE FACTS will provide full detailed explanations of all of the internal large scale integrated circuits which are the heart and soul of the Color Computer.

The reader should be aware that the Color Computer ROMs have gone through several revisions since the Color Computer was released in 1980. The current version number of the Color BASIC ROMs is 1.2, Extended BASIC is version 1.1 and Disk BASIC is version 1.1. The new revisions to Color Basic and Extended Basic corrected some minor bugs (including the famous PCLEAR bug) and left most of the code in the same relative position that it was located in the earlier ROMs Disk Basic 1.1, however, corrected many bugs and no effort was made to keep the code in the same address space as version 1.0. The primary reason for this, I am sure is because version 1.0 only used 75% of the ROM space available.

The comments and listings in THE FACTS book will deal with Color Basic 1.2, Extended Basic 1.1 and Disk Basic 1.1.

SECTION 2
EVOLUTION OF THE COLOR COMPUTER

The Color Computer has undergone several changes since it was first introduced in the Summer of 1980 when a 4K machine cost $400. As of this date (July 1983) a 16K machine costs $200. (a $300. drop) and I seriously doubt that the Color Computer in its present form (6809,74LS783, 6847) will be sold by Tandy Corp. after the Summer of 1984. There are two Printed Circuit Boards (PCBs) which have been used in the Color Computer. The first PCB spawned a total of 5 revisions; the A and B revisions never reached production (never sold) and only a few thousand of the C revision (the earliest) PCBs were sold. The D board was the first widely distributed board and could only handle 16K RAMs. The C & D boards could be expanded to 32K by piggybacking 16K RAMs but required major surgery to accept 64K RAMs. Piggybacking was cheaper but would not allow the SAM chip to locate the video display above the 16K boundary which was allowed by the 64K RAMs.

When Radio Shack became aware of the tremendous customer demand for 32K (which a disk badly needs), the decision to go to 32K by way of "half -good" 64K RAMs (instead of piggybacking) was made. Color Basic Verion 1.1 which allowed the computer (SAM chip) to set itself up for 64K RAMs was developed and the E board was born. 32K was here but now everyone wanted 64K. But a design flaw in the computer caused buss contention between the RAMs and ROMs if the 64K RAMs were switched into the all RAM mode.

Yep, you guessed it, now comes the F board. But wait, while all of this has been going on Commodore and Texas Instruments have been having a price war and Tandy had to compete so the Color's PCB was drastically redesigned and we start all over again with an A board and tack the letter A after the model number of the computer. The F board is a misnomer - it does not exist. It is understandable that people think it does since it came out after the E board but when Tandy put the A after the model number they also started the PCB version number at A. This is the same board used in all of the TDP computers and will allow the use of all good 64K RAMs.
BASIC was developed which has a DOS command that will allow you to boot a completely new operating system if the first two bytes of the bootstrap loader are the ASCII letters 'OS'. This will allow you to easily boot the OS-9 operating system by just typing DOS.

SECTION 3
GENERAL DESCRIPTION

The TRS 80 Color Computer is a Read Only Memory (ROM) based system. The operating system software, which is Microsoft Color Basic, extended BASIC and the disk operating system (DOS), is programmed into ROMs. The fact that this software is programmed into ROMs makes the Color Computer very portable and easy to use. You do not have to wait for the software to be loaded into the system from a cassette or disk. However, this leads to a certain number of problems, one of which is the fact that you cannot change BASIC. If there is any feature or improvement which you would like to add to BASIC, it cannot be done. This is because of the fact that Read Only Memories can only be read. It is not possible to write new information into the Read Only Memory. This presents no big problem for most users of the Color Computer because of the fact that they wish to take advantage of the features which BASIC presents to them - they do not want to change BASIC.

These Read Only Memories do take up a significant amount of space in the computer's memory and if you wish to write an extremely large program which makes full use of the total capacity of the memory capability of the computer, it is not possible to do so with these ROMS in place. It is the presence of these ROMS in the Color Computer which limit the amount of memory which Basic can access to 32K. The old style Printed Circuit Boards (PCBS)-series C, D and E - were not designed to accept 64K RAMs. The E series boards could be expanded to 32K with 64K RAMs which are only "half-good". These "fallouts" from 64K RAM production runs were relatively inexpensive at the time and filled a gap in the natural evolution of the Color Computer. A design flaw in the computer prevented the user from taking full advantage of the combination of the 6809, 74LS783 and all good 64K RAMs. This flaw was corrected in the new A series PCBs in the TDP computers and Color Computers.

It is important for the user of the TRS 80 Color Computer to realize a few basic limitations of the machine. One of the most powerful features of the Color Computer is the high resolution color graphics capabilities which it possesses. These capabilities are inherent to the 6847 Video Display Generator (VDG), and it is the controlling limitation on all graphics capabilities of the machine. The 6847 provides the signals which are modulated and fed to the antenna of your television set. Bandwidth limitations imposed by the circuits of your television set and the 6847 do not allow the Color Computer to normally display more than 32 characters across its display. If you wish to use 64K of memory, you can get 51 characters across the face of the screen and have reasonable readability. Some attempts have been made to put 64 columns on a high resolution (64K) screen, but the results are about 90% illegible. Its very difficult to try and get anything other than this without severe hardware modifications to the internal circuits of the Color Computer for home use and playing games. It is not designed nor intended to be used in a small business application which would require 64 and 80 characters per display line.

SECTION 4
GENERAL COMPONENT DESCRIPTION

In this section we shall describe the general components of the Color Computer and how they work together. Figure 1 is a block diagram of the Color Computer and shows all of the general components and how they relate to one another. In order to fully understand the operation of the Color Computer, it is necessary to understand in detail each one of these components and their strengths and weaknesses.

A. 6809E Central Processing Unit (CPU) - The microprocessor used in the Color Computer is the Motorola 6809E. The E suffix indicates that this particular chip does not have its own internal clock. There must be an external clock provided. This clock function is provided by the Synchronous Address Multiplexer (SAM) chip. The 6809 is undoubtedly the most powerful 8 bit microprocessor available on the market today. The power of the 6809 stems from the fact that it has a very powerful and highly organized addressing structure. This structure will be described in further detail later on in this text. Motorola's data sheet for the 6809E is included with this text. The user is referred to it if any detailed questions arise as to the operation of the 6809E CPU.

The CPU is the main controlling device for the entire computer. It governs the flow of information to and from the various internal components of the computer to the input/output devices, the keyboard, the cassette, the joysticks and the RS 232 channel. The other major devices in the Color Computer are the Video Display Generator (VDG), the Synchronous Address Multiplexer (SAM), and the Peripheral Interface Adapters (PIA's). These devices are all under the direct control of the CPU and serve to simplify the tasks of the CPU.

Section 4 will provide a detailed description of the CPU and its many operating features. In addition to the data sheet provided with this package, it should give the user a full and complete knowledge of the Color Computer CPU.

B. 74LS783 Synchronous Address Multiplexer (SAM) - The SAM chip provides the control necessary for all of the various addressing modes of the CPU. The addresses which the CPU would normally issue to its peripheral support chips are intercepted by the SAM chip and rerouted to provide different control functions. A detailed spec sheet on the SAM chip is also provided with this data package. A more detailed explanation of the SAM chip will be provided in section 5 of this manual. In general, the SAM chip provides the following functions:

It governs the flow of data from the random access memory (RAM) to the VDG chip. It does this in such a manner as to be transparent to the operation of the microprocessor. This means that when the VDG is getting the data which it requires from the RAMs, it is not affecting the flow of data between the RAMs and the CPU. This is the reason why the Color Computer does not have the annoying video display flicker which is very common to the TRS-80 Model III. The SAM chip also provides refresh signals to the dynamic RAMs of the system and it is capable of allowing either 4, 16, or 64K RAMs to be used in the system. The flow of all the data on the data bus and the peripheral devices which are activated for various memory accesses is also controlled by the SAM chip.

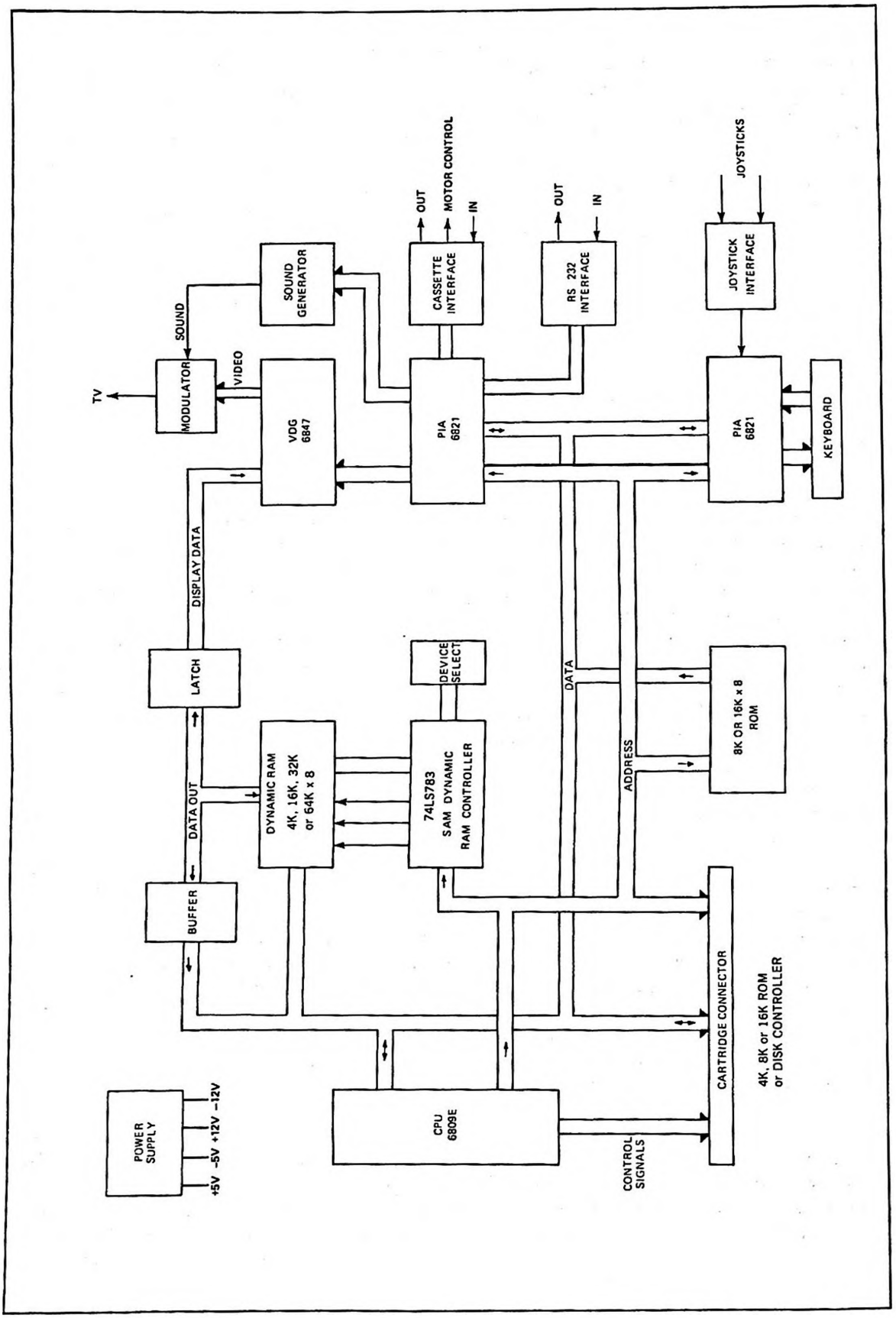


FIGURE 1. COLOR COMPUTER BLOCK DIAGRAM

C. The 6847 Video Display Generator (VDG) - The various display and graphics modes of the Color Computer are produced by the video display generator. The VDG gets its information from the RAM of the system under the control of the SAM chip; therefore, the VDG and the SAM must be programmed to the same graphic display mode or else the screen will generally show garbage results. This is not always the case as is explained in the appendix to the SAM data sheet which is included in this manual. It is the VDG which governs the upper case and lower case features of the Color Computer. For instance, if you do not like the fact that lower case letters are displayed in inverse video, you will have to take this discussion up with the VDG since it is the chip that forces the lower case characters to be in inverse video. A detailed description of the VDG will be provided in section 6 of this document and complete data sheet on the VDG is provided in the appendices. There are several lower case conversion kits on the market for the Color Computer. All of them use the external character generator ROM scheme detailed in Figure 23 of the 6847 VDG data sheet.

D. 6821 Peripheral Interface Adapter (PIA) - The PIA provides the interface from the CPU to the input/output devices with which the Color Computer communicates. These input/output devices are the keyboard, the cassette, the RS 232 channel and the joysticks.

E. Sound Generation - Sound is generated in the Color Computer by having the CPU generate all complex waveforms. This means that if there is a complex waveform which needs to be generated in order to produce a laser sound or an explosion or a high-pitched whistling sound, the CPU must spend all of its time generating this sound. The CPU will not, therefore, be able to move a lot of objects around on the screen such as space ships or missiles or explosions or any other features which have to be rapidly moved around on a high resolution screen. This is a severe limitation of the Color Computer and it prohibits you from having very good quality sound at the same time that high resolution fast action objects are being moved on the screen.

The actual analog waveform is produced by a Digital to Analog (D/A) converter. A D/A converter takes the digital byte from the computer and converts it into an analog voltage. This analog voltage is then fed to the TV's sound system through an analog multiplexer which is an integrated circuit used as a switch - it will choose one of four analog inputs and route that input to its output. The input which is selected is completely under the control of the computer. The sound output from this analog multiplexer is then given to the modulator device of the Color Computer which mixes it with the video signal that goes to your television set so that the television will recognize the sound input provided from the computer and amplify it on its own speaker. The D/A converter actually consists of a series of six resistors. This is the most inexpensive way to build a D/A converter and it has a certain number of drawbacks but works in the Color Computer.

Data is sent to the D/A converter through PIA1, Port A (FF20). Only the high order six bits of data are given to the D/A converter. This means that there are two additional bits sent to the PIA port which must be taken into consideration when providing data to the D/A. Bit 0 is the cassette input bit and since it is used as an input by the PIA, you do

not need to consider it when you are sending data to the D/A converter. Bit 1, however, is the RS 232 output bit and you must always keep this bit high when sending data to the PIA otherwise you will inadertently toggle the RS 232 output and if you happen to have an RS 232 device hooked up to the RS 232 output port, it will go crazy if you do not force the data on bit 1 to be high. The best way to handle this situation is to have one subroutine which sends data to the D/A converter and make sure that the last statement before the data is stored at FF20 is an ORA #2 statement. This will insure that bit 1 is always high when data is stored in the D/A converter from ACCA.

Although the D/A converter is used to make complex sound under computer control, it is not the only sound source which can be connected to the TV speaker. The analog multiplexer, which is an MC14529, is used to connect one of four sound sources to the speaker of the television set. The first sound source is the D/A converter. The second sound source is a signal identified as CSN which is the sound from the cassette input and is used to monitor the loading of cassette tapes into the computer. The third sound source is connected to the cartridge ROM pack port and is blank under normal conditions. This allows any type of device which is hooked up to the ROM pack port to send sound into the TV's sound amplifier under computer control. The fourth sound input is a no sound input. This input is hooked up to ground and whenever this sound source is selected, there will be no sound whatsoever. A graphic display of the various interconnections in the sound system of the Color Computer is shown in Figure 2.

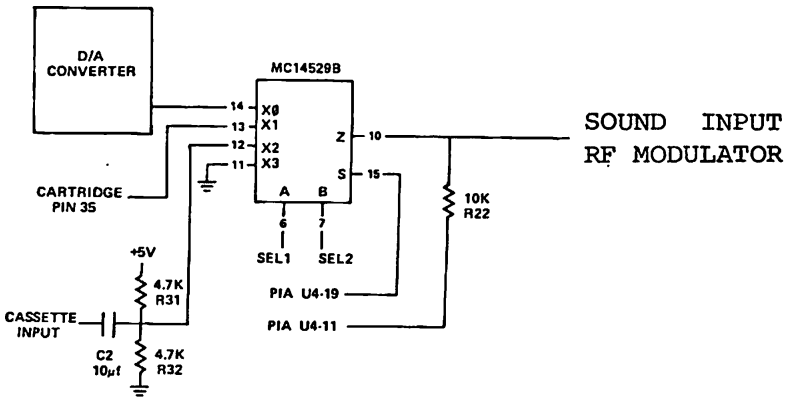


FIGURE 2. SOUND CIRCUITRY

The method of selecting which sound source you wish to have routed through to the speaker of the television set is determined by two control inputs to the analog multiplexer. These inputs are labeled SEL 1 and SEL 2 which is short for Select 1 and Select 2. Select 1 and 2 are the binary input controls for the MC14529 and they are controlled by the CA2 and CB2 outputs of PIAØ. Care must be exercised when programming the sound select inputs SEL 1 and SEL 2 because these select inputs are also used to select which joystick is being read by the computer. This is not normally a problem and you should not have to deal with this under normal circumstances; however, you should be aware that the MC14529 analog multiplexer is a dual multiplexer. That is, there are two four-channel to one-channel multiplexers in the same package and

they do not have independent select controls so that when you select one channel for the sound output, you are also selecting a channel for the joystick reading routine. The converse is also true, so that as you are stepping through reading one of the four joystick outputs and using the select inputs to choose which joystick you're reading, you will also be moving the sound select source from one to another. This can cause some very weird sound effects or foul up good sounds while the joysticks are being read. This problem can be circumvented by disabling the sound portion of the analog multiplexer. This is done through the CB2 output of PIA 1. If you store a zero in the output of CB2, sound from the analog multiplexer is disabled and no matter which sound source you select from the multiplexer, you will hear no sound on the TV. Figure 3 shows how the select pins control the inputs to the analog multiplexer.

If you program a 1 to this output, the sound will be enabled, and if there is a sound source on the respective input of the multiplexer, you will hear sound on the output of the TV speaker. There is a routine in BASIC which will perform this enabling and disabling function. If you wish to disable the sound, simply perform a JSR $A974 and the sound will be disabled. If you wish to enable the sound, perform a JSR $A976 and sound will be enabled.

		MULTIPLEXER	
SEL 1	SEL Ø	SOUND	JOYSTICK
0	0	D/A	RIGHT HORIZONTAL
0	1	CASSETTE	RIGHT VERTICAL
1	0	CARTRIDGE	LEFT HORIZONTAL
1	1	GROUND	LEFT VERTICAL

FIGURE 3 - MULTIPLEXER CONTROLS

If you wish to make complex sound, you will need to have the analog multiplexer selected with zeroes on both of the select inputs SEL 1 and SEL 2 in order to pass information from the D/A converter to the sound system of the TV set. It will also be necessary to enable the sound analog multiplexer as described above. Then you will have to write the data to the D/A converter through PIA1, Port A (FF20) in order to produce sound. A sample program is listed below which will take data stored in a table with a specified delay at each data point and store it into the D/A converter in order to produce sound. There is also a large looping routine which will allow you to reproduce the same sound over and over again.

If you put different values into the table for the amplitude value and the delay value, you will be able to program whatever type of sound you wish to hear.

F. Single Bit Sound - Single Bit Sound is used to send square waves to the sound system of the TV set independently of what the analog multiplexer is doing. The sound produced in this manner will be very simplistic but, in many cases will be perfectly satisfactory for the use at hand. You will not be able to produce the highly complex sound which you can produce with the D/A converter but there are many instances when you may wish to have a simple sound such as a beep or an alarm to let you know that something is happening and you will not want to use the D/A converter in order to do this. This is especially important if you are using the D/A converter to read joystick data since you cannot produce sound waveforms on the D/A converter while it is being used to read the joysticks.

In order to use the single bit sound, several things must first be set up. The first thing that's necessary to do is to disable the sound analog multiplexer as described in the sound generator section above. This must be done to prevent sound from the analog multiplexer causing interference with the single bit sound. Then it is necessary to program bit 1 of PIA 1 Port B (FF22) to be an output. You then write the sound data which you wish to hear to bit 1 of PIA 1 Port B (see Figure 4).

```
                    ** THIS PROGRAM WILL DEMONSTRATE HOW TO USE THE SINGLE BIT SOUND
                    ** FEATURE OF THE TRS-80 COLOR COMPUTER.
                    ********************************************************************
                    *THIS ROUTINE WILL ENABLE PIA1, PORT B, BIT 1 AS AN OUTPUT
                    *AND WILL DISABLE THE SOUND ANALOG MULTIPLEXER.
                    ********************************************************************
3100                        ORG     $3100
3100  8E    FF22    START   LDX     #$FF22  POINT X TO PIA1, PORT B
3103  A6 01                 LDA     1,X     GET THE CONTROL REGISTER
3105  84      FB            ANDA    #$FB    MASK OFF BIT 2
3107  A7 01                 STA     1,X     ENABLE DATA DIRECTION REGISTER
3109  E6 84                 LDB     ,X      GET DATA DIRECTION STATUS
310B  CA      02              ORB     #2      FORCE BIT  1 HIGH
310D  E7 84                 STB     ,X      STORE IT IN DATA DIRECTION REG - NOW BIT1 IS AN OUTPUT
310F  8A      04              ORA     #4      FORCE BIT 2 HIGH
3111  A7 01                 STA     1,X     ENABLE PERIPHERAL REGISTER
3113  BD    A974            JSR $A974       DISABLE THE SOUND ANALOG MULTIPLEXER - THIS IS
                    *                       NEEDED TO PREVENT THE ANA MUX SOUND ROM
                    *                       INTERFERING WITH THE SINGLE BIT SOUND
                    ********************************************************************
                    ** THIS ROUTINE WILL TAKE BIT 1 OF ACCA AND SEND IT TO SINGLE BIT SOUND
                    ********************************************************************
3116  84      02    SNDOUT  ANDA    #2      KEEP ONLY BIT 1
3118  34 02                 PSHS    A       SAVE IT ON THE STACK
311A  B6    FF22            LDA     $FF22   GET PIA1, PORT B
311D  84      FD            ANDA    #$FD    MASK OFF OLD SINGLE BIT DATA
311F  AA E0                 ORA     ,S+     INSERT NEW DATA AND FIX THE STACK
3121  B7    FF22            STA     $FF22   SAVE IT IN PIA1, PORT B
3124  39                    RTS             RETURN
```

G. Cassette Interface - Cassette data is stored onto the tape using a format called Frequency Shift Keying (FSK). This means that two sine waves of differing frequency are used to store zeroes and ones on the tape. A sine wave of 2400 Hertz is used to store a one, and a sine wave of 1200 Hertz is used to store a zero. These sine waves are generated by the computer with a lookup table and sent to the D/A converter (see Appendix A).

The table is scanned sequentially and the values are stored in the D/A converter. The output of the D/A converter is then sent to the auxiliary input of the tape recorder and stored on the magnetic cassette tape. When the data is to be read back into the computer, it is read back in through a circuit which detects when the input voltage has crossed zero and counts the time before it crosses zero again. A high frequency sine wave will cross zero more often than a low frequency sine wave. This fact is used to determine whether a zero or a one was stored on the tape. It is a very simple method but works well.

The cassette motor is controlled by the computer through CA2 of PIA1. The motor will be on if CA2 is high and off if CA2 is low.

CASSETTE TAPE FORMAT: The normal Color Computer tape contains the following information:

1) A leader of 128 bytes of $55
2) Namefile block. (header block)
3) About ½ second of blank tape.
4) 128 bytes of $55.
5) Data blocks.
6) End of file block.

The format for Data, Namefile or End of file blocks is:

1) One $55 byte.
2) One $3C sync byte.
3) Block Type byte: Namefile=00, Data=01, End of File=$FF.
4) Block length byte, 0 to 255.
5) Data, 0 to 255 bytes.
6) Checksum byte, the sum of all the data plus block type and block length.
7) One $55 byte.

The Namefile block is a normal block with a length of 15 bytes as described below:

1) Eight bytes for the program name.
2) File type byte; BASIC=00, Data=01, Machine language=02
3) ASCII flag byte; Binary=00, ASCII=$FF
4) File mode byte; Continuous=00, Non Continuous=$FF
5) Two bytes for the transfer address of a machine language program.
6) Two bytes for the start address of a machine language program.

The end of File block is a normal block with a length of zero.

H. RS 232 Interface - The Color Computer also has RS 232 input/output circuitry which is capable of transmitting data from the computer and loading data into the computer, and it has one status signal - the carrier detect signal. There is a built-in software routine in color BASIC which will drive this RS 232 I/O circuitry (see Appendix A). This routine has its limitations, however, and if you are listening to or talking to an RS 232 device with it, the Color Computer cannot do anything else. Normally this will not be a problem since when you are communicating over the RS 232 lines, there is probably nothing else that you will really want to be doing anyway.

I. Joystick Interface - Each joystick is composed of two potentiometers (pots), one for the horizontal direction, and one for the vertical direction. Moving the joystick in any combination of directions will vary the voltage on each of these joystick pots from zero to five volts. Therefore, there are four voltages which must be considered when trying to read the joysticks: the horizontal and vertical voltage for both the right and the left joystick.

The Color Computer uses the other half of the sound analog multiplexer in order to read these voltages into the computer and only one voltage at a time may be read into the computer. The voltage which is being read at any particular time is determined by the state of the SEL 1 and SEL 2 control outputs as described in the sound generator section above (see Figure 3). The analog multiplexer is set up to read one of the joystick voltages through it. The output of this multiplexer is connected to an electronic device called a comparator, which can compare two input voltages. If Voltage A is higher than Voltage B, the output of the comparator will be high. If Voltage A is lower than Voltage B, the output of the comparator will be low. The output from the analog multiplexer is hooked up to Comparator Input A. The other input of the comparator is connected to the D/A converter. A successive approximation method is then used to determine the voltage range which is present on input A of the comparator. A successive approximation conversion is done in the following manner: the D/A converter is programmed to its mid-range value which is $80 and the output will be approximately 2.5 volts. If the voltage being sensed from the analog multiplexer is greater than this value, then the D/A converter is increased by ½ of its present value - if it is less, the D/A converter is decreased by ½ of its current value. Another comparison is then made and the value of the D/A converter is decreased or increased by one-half of the distance between its current value and the value that was previously used. In this example, the first value used was 2.5 volts, the second value used was 3.75 volts so the D/A converter will then be programmed to be 2.5 + ½ (3.7 - 2.5) = 3.14. This procedure continues until six approximations have been made. Six is used because the D/A converter is only a six bit converter. If it were more accurate, say an eight or ten bit converter, you would continue for eight or ten approximations. Therefore, the input voltage loaded into the Color Computer can only be one of 64 different values. This fact is also caused because the D/A converter is only a six bit converter. Basic has a built-in joystick read routine located at $A9DE. The BASIC joystick conversion routine is explained in detail in Appendix A.

SECTION 5
6809E MICROPROCESSOR UNIT (MPU)

It is not the main purpose of THE FACTS to explain assembly language programming. Instead, it is to describe the specific application of the 6809E in the Color Computer. The suffix E is used to indicate that this particular version of the 6809 does not have its own internal clock source. The clock is the master timing signal source used in any computer system to keep all of the various components synchronized and whenever the 6809E is used, there must be an external clock which generates the system's timing pulses. In the case of the Color Computer, the Synchronous Address Multiplexer (SAM) chip is used to generate the system clock. In all of the discussions used in this document, we will use the term 6809 and 6809E interchangeably. This is generally done in the literature, but the reader should be advised that the Color Computer has a 6809E and not a 6809.

There is nothing unique or truly imaginative about the use of the 6809 in the Color Computer. One can easily determine this by looking at the first page of the SAM data sheet provided in this manual, or Figure 19 of the SAM data sheet, or Figure 1 of the 6847 data sheet. In each one of these figures, there is a rough block diagram outline of the computer system using the 6809E, the 6847 and the 74LS783 SAM chip. If you pay special attention to Figure 19 of the SAM sheet, you will see truly remarkable similarities between the schematic diagram of the Color Computer and this figure. The layouts in both diagrams are virtually identical. The 6809 has many features built into it which are not used by the Color Computer; however, most of these features are only useful when designing a multiprocessor system. The signals referred to as TSE, Busy, LIC, BS, BA, AVMA are not used in the Color Computer and there are no provisions to use these signals from the ROM pack connector. The only signal which goes to the ROM pack connector which may be useful for expansion applications is the Halt signal. When this particular signal is pulled low, the address bus, data bus, and R/W line are all placed in the tri-state condition. This will allow an external device to control the computer via the cartridge connector.

The 6809E data sheet provided in this manual has a complete description of the 6809E, its programming modes and a complete list of all of the instruction set. There are also many helpful explanations and programming hints provided in this data sheet. Adam Osborne Inc., publishes a good book on machine language programming for the 6809E which is aimed at the programmer who has written machine language before. If you wish to learn how to program in machine language and have never done so before, it will be necessary to get some reference books on general machine language programming. There are several of these books available and any one of them would be good for trying to learn the basics of computer terminology, architecture and programming style.

SECTION 6
74LS783 SYNCHRONOUS ADDRESS MULTIPLEXER (SAM)

The true heart and soul of the Color Computer is the SAM chip. It is a specially designed chip to mate the 6809E, 6847 Video Display Generator (VDG) and Dynamic Ram. An excellent example of a computer using these chips is shown in Figure 19 at the back of the SAM sheet provided in this manual. Once again one will notice the striking resemblance to the Color Computer circuitry as many of the circuits and interconnects described in Figure 19 are identical to those shown in the Color Computer schematic. The SAM is undoubtedly the least understood chip in the Color Computer. The SAM data sheet appended to this manual is the best description available of this particular chip.

The SAM is programmed via the 16 bit SAM control register. Page 14 of the SAM sheet has an excellent description of the controls of this register and how to program them. It is recommended that before continuing with this description, the reader look through the entire SAM sheet in order to get as familiar with the SAM as possible. Listed below will be the various registers which are controlled via the SAM control register and how to access them.

A. VDG Addressing Mode - These three bits are used to tell the SAM the current mode of the VDG. This is necessary because the SAM chip is used to control the flow of data from the dynamic RAMs to the VDG. If the VDG and the SAM are not programmed in the same manner, you will not necessarily get a valid display of data; however, it is not always the case that you will get a garbage display. This will be demonstrated very aptly in Appendix A of the SAM sheet by the presence of three additional semi-graphics programming modes which are caused by programming the VDG to one mode and programming the SAM chip to a different mode.

B. VDG Address Offset - In order to completely understand this particular control register, it is necessary to become acquainted with the fact that the display of the Color Computer is memory mapped. This is a special term which is used to indicate that the display is actually stored in the working memory of the computer. That is, if you wish to write a character onto the display, it is necessary only to store that character into the memory of the computer. The normal BASIC display starts at address $400. It is easy to see what happens by using the basic poke command to put a character into the video display. Try, for instance, the command POKE 1072,32. This will cause a blank to be displayed on the screen. The same thing can be done in machine language routines by just writing information into the memory area where the display is and the information will then show up on the TV screen.

Since the SAM is responsible for supplying all of the data to the VDG which, in turn, displays the data on the screen of the TV set, it makes sense that the SAM should be able to put the video display wherever you wish in memory. This particular function is performed by the VDG address offset. This register is a seven bit register and, therefore, there may be 1 of 128 different addresses in it. The memory display area of the Color Computer is divided into 128 one-half K pages of memory display. To choose which memory display you wish to see on

the screen, you program the seven bit VDG address offset register to the appropriate code. If you wish the display to start at the very bottom of the memory, 0000, then you clear all seven of the VDG address offset bits. If you wish it to be at the normal basic address, $400, you program bit one of the video display offset to be a one. This is done by writing any data into address FFC9. In this manner, the video display may be moved to any location in memory. This is not exactly true for the current configuration of the Color Computer. The Color Computer limits the video display to only being in dynamic RAM. This is because data is transferred from the system RAM to the VDG through a 74LS273, which is used as a temporary storage buffer. The strobe pulse which clocks data into the LS273 is the RASØ timing pulse from the SAM. This signal is only active when the system RAM is being used for the video display. As such, the video display will not function if set to any area of memory other than the internal dynamic RAM controlled by RASØ. A more severe limitation is the fact that extended BASIC will only allow the video display in the first 16K of RAM.

C. Page Switch - The page switch bit is used only when you have 64K RAMs in the system. When using 64K RAMs and you are in map type zero, this particular bit will be used to switch between the two 32K pages of RAM located in the bottom half of the 6809's addressing space.

D. MPU Rate - This is undoubtedly the most poorly understood feature of the Color Computer. There have been many articles published in the trade journals which speak of the ability of the Color Computer to run at double speed. This ability springs from the MPU rate control which is built into the SAM chip. The normal clock frequency of the Color Computer is .89 megahertz. If you use the MPU rate to boost the speed of the clock by doubling it, the clock frequency will be 1.78 megahertz. There are three severe limitations to having the Color Computer actually be able to run at 1.78 megahertz. If you look at the 6809 data sheet, you will notice that there are three versions of the 6809. They are the 6809E, the 68A09E and the 68B09E. The letter in the middle of the number of the microprocessor refers to the rate of speed at which it may be clocked. You will notice that it varies from one megahertz to two megahertz. The speed ability of the microprocessor is basically limited by the internal construction of the chip - it has to be constructed in a slightly more precise and accurate manner in order to run at a very high speed. The Color Computer comes with a 6809E in it - not a 68B09E. It often happens that 6809E will run at 1.78 megahertz because manufacturers generally are very conservative in spec-ing their devices. However, it does not guarantee that you will be able to run your Color Computer at double speed if you only have a 6809E in it. As it turns out, some will, some won't. In order to guarantee that you can run your Color Computer at double speed, you have to have a 68B09E inside the machine. You will also notice in the data sheet for the 6821 that there are three comparable 6821's which will run at three different speeds. The Color Computer also does not come with the 68B21 which is necessary if you're going to run the machine at double speed.

One further drawback will be noticed by looking at the Color Computer schematic. You will see the resistors R73 and R74 inserted in the line from the SAM chip to the CPU chip. These resistors are present in C and D version PCBs and are replaced by ferrite beads later and are

inserted in the clock lines in order to prevent static discharge damage. However, they also degrade the E and Q clock waveforms. The waveforms as they currently exist in the Color Computer are beyond the specs for the 6809 and when you insert a resistor in the path, they get even worse and this contributes to the inability of many Color Computers to operate at double speed.

A further consequence of attempting to use the Color Computer at double speed is that you change the timing loops which are built into many of the software routines. There are many routines in the Color Computer such as printer output routines, cassette routines, RS 232 routines, et cetera, which depend upon very specific timing delays between certain events. These timing delays are all based upon the processor running at normal speed. Whenever you double the speed of the processor, you are also going to cut in half the amount of time required for these timing loops which will invariably throw a monkey wrench into them, especially disk BASIC.

Very often people are tempted to increase the speed of the computer by changing the master crystal oscillator in the computer. Beware of doing this in the Color Computer because the master oscillator is divided by four by the SAM chip in order to produce the 3.58 megahertz color subcarrier frequency which is required by the VDG. If you change the master crystal oscillator, you will obviously change this 3.58 megahertz frequency and the VDG will no longer be able to function properly. If you attempt to upgrade your computer to achieve double speed or if you're lucky enough to have one which will work at double speed normally, you must read page 11 of the SAM spec sheet in order to determine the proper manner to change the MPU rate. You cannot just change the two MPU rate bits willy-nilly without paying very close attention to the MPU rate information given on page 11.

E. Memory Size - The SAM chip is designed to support 4K, 16K or 64K RAM chips. Which of these chips you use must be determined and then programmed into the SAM using the memory size bits. It is also possible to use two banks of 4K or two banks of 16K RAMs. See the RAM address section on page 4 and Figure 13 of the SAM spec sheet for more of the description on how to add a second bank of 16K RAMs to increase the memory of the Color Computer to 32K. This is the basis for the 32K piggyback expansion method. This was the technique used by Spectral Associates in its extremely popular RAMCHARGER board.

F. Map Type - A map type bit is used to switch between the two memory map types available in the SAM chip. Figures 14 and 15 in the SAM sheet give an excellent description of the two memory map types. Map type does not have any affect on the operation of the system unless you're using 64K RAMs. When using 64K RAMs, a map type will switch the 64K RAMs from either two banks of 32K in the lower memory of the computer or one bank of 64K which extends across the entire memory space of the computer.

The one problem with the 64K RAMs is that the BASIC, extended BASIC, and DOS ROM all occupy the same space as the upper half of the 64K RAMs when in map type one. The particular circuit layout used in the Color

Computer does not disable these ROMs from the bus of the computer when writing to them so that when you're in map type one and you attempt to write information into the 64K RAMs, there is a bus conflict because the ROMs will be enabled and the output of the CPU will also be enabled. Therefore, the data bus will have data provided to it by both the CPU and the ROMs and you will get a garbage result. Therefore, the only way to solve this problem is to disable the ROMs when you go to 64K RAM mode which requires a hardware modification if you do not have an "A" version PCB.

An additional benefit is gained by having 64K dynamic RAMs in the system in that it allows you to put the video display page anywhere in the 64K RAM memory of the computer which you wish. One thing needs to be noted. When you switch from map type zero to map type one, the second 32K RAM page which is located in the lower section of the memory is copied to the top half of the memory map in the map type one. There is no change in the data - it remains intact during the transition.

Dynamic Ram Refresh is also one of the built-in features of the SAM chip. Page 13 describes in detail how Dynamic Ram Refresh is accomplished and the manner in which it is done so that it does not interfere whatsoever with either the video display operation or the normal memory accesses performed by the CPU. This type of operation is referred to as transparent refresh.

One of the main benefits of the combination of the SAM and the VDG is the fact that new semi-graphics modes are created. Appendix A of the SAM sheet describes these modes in detail. This feature is only available because the SAM and the VDG are used together as explained earlier in this document.

SECTION 7
6847 VIDEO DISPLAY GENERATOR (VDG)

Once again a tremendous similarity between the Color Computer and Figure 1 of the VDG data sheet is obvious. The differences between Figure 1 and the Color Computer are caused because Figure 1 is a description of a TV game whereas the Color Computer is a full bore computer. As such, Figure 1 has no ROMs (BASIC, Extended BASIC, DOS) and a 6846 replaces the Color Computer's two 6821 peripheral interface adapters. The VDG is used to drive the video display of the Color Computer and data is transferred to it by the SAM chip. The mode control pins of the VDG are set by programming PIA1, port B (see Figure 5). Table 3 of the VDG data sheet describes the various alpha/graphics modes of the Color Computer. In order to develop high resolution fast action games or any type of display using these, it is necessary to thoroughly understand the various graphics modes available to the Color Computer.

As mentioned before, the display of the Color Computer is memory-mapped. The characters displayed on the screen are in consecutive memory locations and the amount of memory required to fill the entire display varies with the different graphics modes used. It varies from a minimum of 512 bytes for the alpha numeric displays to a full 6K bytes for the highest resolution graphics modes. When using the VDG, it is necessary to properly program its control pins to the correct graphics mode which you wish to use. This is done through PIA 1 and several examples of how to program the various modes are given below.

It is also necessary to properly program the SAM chip. Various programming requirements are detailed in Figure 5. When programming in the semi-graphic mode, each character position corresponds to one byte of memory. By looking under the column TV screen detail on page 19 of the 6847 data sheet, you will be able to see how each character position is divided into respective pixels (a pixel is a picture element). The VDG data bus describes how the memory byte controls the respective character position graphic display. For the semi-graphic four mode, the value stored in bits C2, C1, and C0 define the color for the entire graphic character block, and you may have one of eight colors for each character block. There are four picture elements: L0, L1, L2, and L3 in each character block. These picture elements may not have separate distinct colors. They may only have the color for the entire picture element as defined by C2, C1 and C0. In semi-graphic six mode, only C1 and C0 are available, therefore, you may only have four colors for each character block. Each pixel may be either on or off. This particular function is controlled by the state of the various bits of the data word labeled L0-L7. If the particular bit is zero, then that particular picture element is off and there is no color (black) for that element. When in semi-graphics mode six, there are two sets of four colors which are available. These colors are defined in Table 3 of the VDG data sheet under the column color for semi-graphic six mode. You may switch back and forth between these two sets of colors by changing the state of the CSS pin.

All of the alpha and semi-graphic modes require 512 bytes of memory which is arranged in 32 bytes per row and 16 rows. The most useful

graphics modes of the VDG are the high resolution graphics modes and there are two types of these modes: the color graphics mode and the resolution graphics mode. Each of these modes uses anywhere from one to six K of RAM. The higher amount of memory that is used, the higher the resolution available on the screen. Therefore, it is up to the programmer to decide which memory mode to use. To get the highest resolution (the best looking picture) requires that you use the higher resolution modes which absorb large amounts of memory.

The resolution graphics modes use one bit in each byte to control one picture element (pixel). The VDG data bus column of Table 3 of the VDG data sheet defines how each bit controls its corresponding picture element. In the graphics modes, the screen consists of consecutive memory bytes and each bit in a byte controls one picture element. The picture element is either on or off and may have only one color, green or buff. You do have a choice of two sets of foreground/background colors to use. These sets of color are described in resolution graphics one in the color column of Table 3. You have your choice between black and green or black and buff.

The color graphics mode requires two bits to define each pixel. Each byte is equal to four pixels and two consecutive bits (C1, C0) specify the color of a pixel. Once again, you have your choice of two sets of colors. These are described in the color graphics one section on the color column, Table 3. The four colors available are green, yellow, blue, and red or buff, cayenne, magenta and orange. The CSS pin is used to switch back and forth between these two sets of colors. The color graphics mode requires twice as much memory as the resolution graphics mode. This is because you have your choice of four colors instead of two colors for each memory location.

TV screen detail column of Table 3 gives the number of clock dots (absolute smallest possible picture element) used in each display mode. This table may be used to get an idea of the relative sizes of the various graphics display elements. Note the tremendous difference in detail which can be gotten by using the resolution graphics mode six compared to color graphics mode one.

Most of the better games today such as Lancer, Qubix, and Miss Gobbler use a high resolution (6K) mode which has a black background color and red, blue and buff as foreground colors. You will notice that such a mode is not listed on page 19 of the VDG Data Sheet, which is because this mode, which is called 'artifacting', is not a real mode of the VDG. It is a phenomena of the manner in which the graphic color information is displayed on the screen in North American television sets (NSTC). The black and buff colors appear if blocks of 0000 & 1111 bit patterns are stored on the screen. The blue and red colors appear if blocks of 1010 or 0101 bit patterns are put on the screen. Which pattern corresponds to blue or red cannot be controlled as it is determined by resetting the computer.

The blue and red colors will only appear if four or more bits contain a 1010 or 0101 pattern and these patterns cannot be closer than one bit

pair or the colors may flow together, which may be a quite acceptable result. You may not use the artifacting mode to get small detailed shapes with all four colors present, however, pleasant shading effects may be achieved by mixing the dot patterns such as D7D7.

The artifacting mode is enabled by setting the VDG and SAM to the Resolution Graphics Six mode and setting CSS on the VDG to 1. One last note about artifacting - it will only work on NSTC television sets. European television sets use a broadcast method called 'PAL' and artifacting will not work on PAL sets - only black and white (buff) will show up.

SECTION 8
6821 PERIPHERAL INTERFACE ADAPTER (PIA)

The Color Computer uses two peripheral interface adapters and has the capability of one expansion peripheral interface adapter on the ROM pack port. These devices provide a universal interface to the 6809 CPU and support all of the I/O functions in the Color Computer. The functional configuration of the PIA is programmed by the CPU during the reset routine. Each of the peripheral data lines may be programmed to act as an input or an output, and each of four control/interrupt lines may be programmed for one of several command modes. Figure 16 of the PIA data sheet shows a block diagram of the PIA. As shown in the block diagram, a PIA consists of two eight bit bi-directional data ports and four control/interrupt lines. The two eight bit data ports are controlled by two data direction registers. These direction control registers are set up by the reset routine and normally will not be changed. The four control/interrupt lines are controlled by the two control registers. Two of the four lines (CA1 and CB1) function only as interrupt inputs and the other two lines (CA2 and CB2) may be used as interrupt inputs or outputs; however, the Color Computer uses them only as outputs.

PIAØ is used mainly for the keyboard. Port B is programmed as an output and is used to strobe the keyboard columns. Port A is programmed as an input and bits Ø-6 are used to read the keyboard rows. Bit 0 and 1 are also used as fire button inputs for the joysticks. The fire buttons are in parallel with eight keys of the keyboard and this is the reason why garbage key strokes show up on the screen when you push the fire button in early versions of the Color Computer. The other pins of PIAØ serve various functions. Bit 7 of Port A is the input for the joystick interface and CA2 and CB2 are used as outputs. These are the two select lines which choose which joystick or sound input the analog multiplexer passes through it. The last two pins of PIAØ, CA1 and CB1, are used as interrupt inputs and they are both tied to the video clock outputs from the VDG. If enabled, CA1 provides an interrupt every 63.5 microseconds and CB1 provides an interrupt 60 times per second.

PIA1 is used for several different functions. Bits 2 through 7 of Port A are outputs used to drive the six bit D/A converter and bit 1 is an output which provides the RS 232 transmit signal. This signal is used to drive the printer and other RS 232 devices. Bit 0 is the input line for data from the cassette. Bits 3 through 7 of Port B are outputs used to control selection of the various alphanumeric and graphics modes of the VDG. Bit 2 is an input for the memory size jumper which is used by the 6809 to determine what memory size is available in the computer: 4K, 16K or 64K. Bit 1 is the single bit sound output and is not normally programmed as an output. If you wish to use the single bit sound, it is necessary to program it to be an output yourself because BASIC initially programs it as an input. This is done so that the single bit sound will not interfere with normal D/A generated sound. Bit 0 is the RS 232 receive input signal.

The control and interrupt pins of PIA1 also serve various functions. CA1 is the input for the signal CD (Carrier Detect, a status pin for the RS 232 interface), and CA2 is an output used to control

the cassette motor. CB 1 is the cartridge interrupt input. Whenever a cartridge is inserted into the computer, it will connect the 6809 FIRQ interrupt pin to the Q timing signal. This will cause an interrupt immediately and it will force BASIC to jump to the program in the cartridge. This is why many people put scotch tape over pin 7 or 8 in the ROM pack cartridge because it will defeat this cartridge interrupt feature. You may then jump to the cartridge yourself by typing EXEC 49152. Finally, CB2 is used as an output to enable the sound analog multiplexer. At this point, it should be noted that there is no enable hooked up to the joystick side of the analog multiplexer. It is always enabled and can never be disabled. Figure 6 details the various modes of the two PIAs after they are initialized by BASIC and they will generally not be changed by BASIC.

The PIAs occupy a defined area of the computer's memory. This is described on page 17 of the SAM data sheet. PIAØ occupies the addresses of FF00 to FF1F (I/O_0) and PIA1 occupies the addresses from FF20 to FF3F (I/O_1). There are only four register addresses in the PIA (the Data Direction Register and the Peripheral Register occupy the same address), yet 32 memory locations have been allocated for each PIA. You will find if you examine the additional memory locations which are allocated to the PIA output space by the SAM chip, that the four bottom memory locations (00-03) are merely repeated seven additional times throughout this memory space (00-1F). This is just a built-in feature of the way that the PIAs are addressed. The top three bits of the five bits required to address 32 memory locations are not used; only the bottom two bits are used, consequently, there is a duplication of the four PIA registers eight times in the 32 byte memory span allocated to each PIA.

MC6883 and VDG MODE REGISTER			PIA REGISTER BITS HEX ADDRESS (FF22)								DATA BITS		ALPHA/GRAPHIC MODE SELECT
V_2	V_1	$V_Ø$	7	6	5	4	3	2	1	Ø	7	6	
Ø	Ø	Ø	Ø	X	X	Ø	CSS	N	N	N	Ø	Ø	Alphanumerics
Ø	Ø	Ø	Ø	X	X	Ø	CSS	N	N	N	Ø	1	Alphanumerics Inverted
Ø	Ø	Ø	Ø	X	X	Ø	X	N	N	N	1	X	Semigraphics - 4
Ø	Ø	Ø	Ø	X	X	1	CSS	N	N	N	1	X	Semigraphics - 6
Ø	1	Ø	Ø	X	X	Ø	X	N	N	N	1	X	Semigraphics - 8
1	Ø	Ø	Ø	X	X	Ø	X	N	N	N	1	X	Semigraphics - 12
1	1	Ø	Ø	X	X	Ø	X	N	N	N	1	X	Semigraphics - 24
Ø	Ø	1	1	Ø	Ø	Ø	CSS	N	N	N	X	X	64 x 64 Color Graphics
Ø	Ø	1	1	Ø	Ø	1	CSS	N	N	N	X	X	128 x 64 Graphics
Ø	1	Ø	1	Ø	1	Ø	CSS	N	N	N	X	X	128 x 64 Color Graphics
Ø	1	1	1	Ø	1	1	CSS	N	N	N	X	X	128 x 96 Graphics
1	Ø	Ø	1	1	Ø	Ø	CSS	N	N	N	X	X	128 x 96 Color Graphics
1	Ø	1	1	1	Ø	1	CSS	N	N	N	X	X	128 x 192 Graphics
1	1	Ø	1	1	1	Ø	CSS	N	N	N	X	X	128 x 192 Color Graphics
1	1	Ø	1	1	1	1	CSS	N	N	N	X	X	256 x 192 Graphics

X = DON'T CARE
N = DO NOT CHANGE

FIGURE 5. VDG 2 & SAM GRAPHIC MODE SELECT

PIA 0 (U18) A BOARD

FF00
- BIT 0 = KEYBOARD ROW 1 and right joystick switch
- BIT 1 = KEYBOARD ROW 2 and left joystick switch
- BIT 2 = KEYBOARD ROW 3
- BIT 3 = KEYBOARD ROW 4
- BIT 4 = KEYBOARD ROW 5
- BIT 5 = KEYBOARD ROW 6
- BIT 6 = KEYBOARD ROW 7
- BIT 7 = JOYSTICK COMPARISON INPUT

FF01
- BIT 0 = Control of the Horizontal sync clock (63.5 microseconds) Interrupt Input — 0=IRQ to CPU Disabled; 1=IRQ to CPU Enabled
- BIT 1 = Control of the Horizontal sync clock (63.5 microseconds) Interrupt Input — 0=Flag set on the falling edge of HS; 1=Flag set on the rising edge of HS
- BIT 2 = Normally 1: 0=Changes FF00 to the data direction register
- BIT 3 = SEL 1: LSB of the two analog MUX select lines
- BIT 4 = 1 Always
- BIT 5 = 1 Always
- BIT 6 = Not Used
- BIT 7 = Horizontal sync interrupt flag

FF02
- BIT 0= KEYBOARD COLUMN 1
- BIT 1= KEYBOARD COLUMN 2
- BIT 2= KEYBOARD COLUMN 3
- BIT 3= KEYBOARD COLUMN 4
- BIT 4= KEYBOARD COLUMN 5
- BIT 5= KEYBOARD COLUMN 6
- BIT 6= KEYBOARD COLUMN 7
- BIT 7= KEYBOARD COLUMN 8 (32K/64K RAM Output Strobe)

FF03
- BIT 0 = Control of the field sync clock 16.667 Ms Interrupt Input — 0= IRQ to CPU Disabled; 1= IRQ to CPU Enabled
- BIT 1 = Control of the field sync clock 16.667 Ms Interrupt Input — 0= sets flag on falling edge FS; 1= sets flag on rising edge FS
- BIT 2 = NORMALLY 1: 0= changes FF02 to the data direction register
- BIT 3 = SEL 2: MSB of the two analog MUX select lines
- BIT 4 = 1 Always
- BIT 5 = 1 Always
- BIT 6 = Not Used
- BIT 7 = Field sync interrupt flag

PIA 1 (017) A BOARD

FF20
- BIT 0 = CASSETTE DATA INPUT
- BIT 1 = RS-232 DATA OUTPUT
- BIT 2 = 6 BIT D/A LSB
- BIT 3 = 6 BIT D/A
- BIT 4 = 6 BIT D/A
- BIT 5 = 6 BIT D/A
- BIT 6 = 6 BIT D/A
- BIT 7 = 6 BIT D/A MSB

FF21
- BIT 0 = Control of the CD RS-232 status Input — 0 = FIRQ to CPU Disabled; 1 = FIRQ to CPU Enabled
- BIT 1 = Control of the CD RS-232 status Input — 0 = set flag on falling edge CD; 1 = set flag on rising edge CD
- BIT 2 = Normally 1: 0 = changes FF20 to the data direction register
- BIT 3 = Cassette Motor Control: 0 = OFF 1 = ON
- BIT 4 = 1 Always
- BIT 5 = 1 Always
- BIT 6 = Not Used
- BIT 7 = CD Interrupt Flag

FF22
- BIT 0 = RS-232 DATA INPUT
- BIT 1 = SINGLE BIT SOUND OUTPUT
- BIT 2 = RAM SIZE INPUT LOW = 4K or 32K/64K HIGH = 16K
- BIT 3 = VDG CONTROL OUTPUT CSS
- BIT 4 = VDG CONTROL OUTPUT GM0 & $\overline{INT}$/EXT
- BIT 5 = VDG CONTROL OUTPUT GM1
- BIT 6 = VDG CONTROL OUTPUT GM2
- BIT 7 = VDG CONTROL OUTPUT $\overline{A}$/G

FF23
- BIT 0 = Control of the cartridge Interrupt Input — 0 = FIRQ to CPU Disabled; 1 = FIRQ to CPU Enabled
- BIT 1 = Control of the cartridge Interrupt Input — 0 = sets flag on falling edge CART; 1 = sets flag on rising edge CART
- BIT 2 = Normally 1: 0 = changes FF22 to the data direction register
- BIT 3 = Six BIT Sound Enable
- BIT 4 = 1 Always
- BIT 5 = 1 Always
- BIT 6 = Not Used
- BIT 7 = Cartridge Interrupt Flag

SECTION 9
INTERRUPTS

The Color Computer has 7 interrupts: Non-Maskable Interrupt (NMI), Fast Interrupt Request (FIRQ), Interrupt Request (IRQ), Software Interrupt (SWI), Software Interrupt 2 (SWI2), Software Interrupt 3 (SWI3), and RESET. Control of the operation of the computer will be transferred to these interrupts whenever they occur, assuming they are active.

The 6809 transfers control during an interrupt according to a jump table located at the top of memory (see Table 1 of the 6809 Data Sheet). The values stored in these jump addresses are set by the BASIC ROM and are listed below:

NMI	$1Ø9	SWI	$1Ø6
IRQ	$1ØC	SWI2	$1Ø3
FIRQ	$1ØF	SWI3	$1ØØ
RESET	$AØ27		

The addresses are called vectors since they direct the control of the computer elsewhere. BASIC sets all of the Color Computer's interrupt vectors, except RESET, to point at RAM. The RAM locations pointed to by the interrupt vectors contain JMP instructions which will once again vector control to another location. An important difference is that since these vectors are in RAM, they may be modified by BASIC or a user program. The initial values stored in these addresses by BASIC, EX BASIC, and DOS BASIC are given in the disassembled listing of BASIC.

The 6809 Data Sheet gives an excellent description of the operation of the various interrupts and how the computer processes them. The only remaining information required to use the interrupts is what physical action in the computer causes them to occur. The software interrupts, as their name implies, will occur whenever an SWI, SWI2 or SWI3 instruction is executed. The NMI interrupt is connected to the cartridge expansion port and may only be used by a device (such as a disk) connected to the cartridge port. The RESET interrupt is connected to the reset button at the rear of the computer. The IRQ and FIRQ interrupts are connected to the IRQ outputs of PIAØ and PIA1, respectively. Each PIA has two IRQ outputs which are connected together so that there are two possible IRQ and two FIRQ interrupts. Pages 7 and 10 explain the operation of the PIA interrupt structure in detail. Only the CA1 (CB1) inputs are used for interrupts. The CA2 (CB2) pin is only used as an output by the Color Computer and cannot cause an interrupt. The CA1 and CB1 inputs of PIAØ are connected to the horizontal and vertical sync signals from the VDG. This will cause an IRQ interrupt to the 6809 every 16.67 milliseconds or 63.5 microseconds. The CA1 and CB1 inputs of PIA1 are connected to the RS 232 Carrier Detect input and the cartridge connector respectively. This allows an FIRQ interrupt to be caused by the RS 232 CD input or from the cartridge port. The cartridge port FIRQ interrupt is connected to the computer's Q clock when you insert a cartridge into the cartridge connector. That is what causes the program in the cartridge to be executed immediately upon power on without going to BASIC first.

In order to use the FIRQ and IRQ interrupts you must perform the ollowing steps:

a. Enable the appropriate CA1 (CB1) interrupt request in the PIA control register.

b. Reset the PIA interrupt flag by reading data from the proper PIA peripheral register, i.e., to reset the IRQA1 flag, read data from the Port A Peripheral Register. Remember, CA2 or CB2 is never used as an input so the IRQA(B)2 flag will never need to be tested.

c. Enable the IRQ or FIRQ flag in the 6809 Condition Code Register.

Listed below is a sample program listing of a routine to enable both FIRQ interrupts and a servicing routine to determine which interrupt occured

```
                    ********************************************************************
                    *INTERRUPT ENABLING PROGRAM : BOTH FIRQ INTERRUPTS WILL BE ENABLED
                    *AND A SERVICING ROUTINE WHICH WILL DIFFERENTIATE BETWEEN THE TWO
                    *FIRQ INTERRUPTS - RS232 STATUS & CARTRIDGE.
                    ********************************************************************

                    *** ENABLE FIRQ INTERRUPT
0000  8E   FF20   ENABLE  LDX     #$FF20  POINT X TO PIA1
0003  A6 01             LDA     1,X     GET CONTROL REGISTER OF PORT A
0005  8A     01        ORA     #1      FORCE BIT 0 HIGH
0007  A7 01             STA     1,X     ENABLE IRQA MPU INTERRUPT ON CA1 (RS232 CD) TRANSITION
0009  A6 03             LDA     3,X     GET CONTROL REGISTER OF PORT B
000B  8A     01        ORA     #1      FORCE BIT 0 HIGH
000D  A7 03             STA     3,X     ENABLE IRQB MPU INTERRUPT ON CB1 (CARTRIDGE) TRANSITION
000F  8E   0017         LDX     #SERVIC GET ADDRESS OF      FIRQ INTERRUPT SERVICING ROUTINE
0012  BF   0110         STX     $110    STORE IN FIRQ VECTOR ADDRESS - THERE MUST ALSO
                    *                   BE A $7E BYTE (JMP OP CODE) IN $10F BUT YOU WILL NOT NEED
                    *                   TO PUT ONE THERE BECAUSE BASIC DOES THAT ON POWER UP
0015  1C     BF         ANDCC   #$BF    ENABLE 6809E FIRQ INTERRUPT FLAG

                    *** FIRQ INTERRUPT SERVICING ROUTINE
0017  B6   FF21   SERVIC  LDA     $FF21   GET CONTROL REGISTER OF PIA1, PORT A
001A  2B     05         BMI     RS232C  BRANCH IF THE IRQA1 FLAG WAS SET - IF IT WAS SET
                    *                   THE INTERRUPT WAS CAUSED BY RS232 CD

                    *                   IF IT WASN'T AN RS232CD INTERRUPT, THEN IT MUST
                    *                   HAVE BEEN A CARTRIDGE INTERRUPT.
001C  B6   FF20         LDA     $FF20   RESET THE IRQA2 FLAG
                    *       ********************************************
                    * PUT THE CARTRIDGE SERVICING ROUTINE HERE
                    *       ********************************************
001F  20     03         BRA     EXIT
0021  B6   FF22   RS232C  LDA     $FF22   RESET THE IRQB1 FLAG
                    *       ********************************************
                    * PUT THE RS232 CD SERVICING ROUTINE HERE
                    *       ********************************************
0024  3B          EXIT    RTI
```

SECTION 10
BASIC

There are many routines in BASIC which would be very useful to be able to use when making your programs. A good deal of time and memory space may be saved by using routines already existing in BASIC routines with complete comments.

When BASIC stores a command such as PRINT, EXEC, PCLEAR, etc. in program memory, it converts the command into a one byte abbreviation called a token. The allowable token numbers are from $80 to $FE for a total of 127 different commands. However, since Color BASIC has many more command than older Microsoft BASICs, a new type of command called a secondary function was needed. The token $FF is a special indicator token which means that the byte following the $FF is a secondary token. This is why there are two command look up tables - the regular reserved command table and the secondary function reserved word look up table. Both of these tables for BASIC, Extended BASIC, and DOS BASIC are given in Appendices A & C.

The listings of BASIC given in the FACTS are meant to provide a sample of some of the most useful routines in BASIC. Should the reader desire a more detailed listing of BASIC, Extended BASIC or Disk BASIC or a more complete description of the operation and theory of BASIC, he should investigate the BASIC unraveled trilogy to be published by Spectral Associates in September 1983. These books contain a complete explanation of Color Extended and Disk Basic and commented source listing of all three BASIC ROMs.

SECTION 11
32K MODEL COMPUTER

Several months after the initial release of the Color Computer, Tandy released a second version of the Color Computer which allows the use of 32K of RAM. There are enough differences between these two versions to warrant a more detailed description of the 32K model.

The 32K Color Computer uses "half good" 64K RAMs to get its 32K of memory. 64K RAMs are composed of two banks of 32K blocks of RAM in the same package, and sometimes one of the 32K blocks is defective while the other block is OK. These RAMs with one block of 32K RAM defective are called "half good" 64K RAMs and they function perfectly in the 32K Color Computer. Since there are two banks of 32K RAM (a high bank and a low bank), the computer must be told which bank is good. For that reason, there is a jumper in the 32K Color Computer labeled HIGH LOW which is used to tell the computer which bank of the 64K RAMs is good. A word of caution regarding "half good" 64K RAMs is in order at this point. Not all of the possible defects in a 64K RAM will cause it to malfunction at all times. Sometimes it will malfunction at high or low temperatures or intermittently malfunction when presented with certain data patterns. The "half good" 64K RAMs may appear to function properly as "all good" 64K RAMs only to intermittently malfunction when least expected. These types of malfunctions may appear to be software bugs and would be very difficult to track down. As such, one should not try to use "half good" 64K RAMs in RAM TYPE 1 mode or switch the HIGH/LOW jumper.

There are four other jumpers in the 32K model which do not appear in the older 4K/16K models. Three of these jumpers are labeled "32K-16K/4K" and are necessary because 64K RAMs require one power supply and 8 address lines while 16K/4K RAMs require 3 power supplies and 7 address lines. Great care must be exercised when exchanging 64K RAMs and 16K/4K RAMs in the Color Computer. If these jumpers are not properly set, the RAMs may be destroyed. The final jumper is used to program the type of RAM into the computer through the RAMSZ input of PIA1, port B.

Color BASIC also had to be modified in order to support the "half good" 64K RAMs. Although there were substantial changes to the first 750 bytes of the BASIC ROM, the functional alterations were slight and all of the major entry points were left unchanged. The functional changes to Color BASIC are listed below:

1. The keyboard input routine was modified to prevent the garbage keystroke input when the joystick buttons are depressed.

2. The power up initialization routine was modified to check for the existence of 64K RAMs and program the SAM chip to accept them.

3. The RS 232 output driver was altered to transmit an eight bit byte instead of a seven bit byte. This allows directly driving an eight bit graphic printer.

4. The CLOAD routine was modified to cause the upper left corner of the screen to blink when tape files are being skipped while searching for a valid file name.

5. The CLOSE routine was changed to cause an end of program block to be written to tape if the file was closed when the character buffer was empty.

SECTION 12
64K MODEL COMPUTER (A BOARD)

The latest version (A Series) of the color computer will allow the SAM chip to set the 64K RAMs to Map Type 1 (all RAM mode) and disable the System ROMs when the switch takes place. This will allow the user to load in his own operating system if he so chooses. It will also allow applications programs to use the entire 64K range of the computer.

The biggest problem facing you when using the computer in the all RAM mode is the interrupts which will not be relocated to $FFF0. The interrupt vectors will still be fetched from the BASIC ROM in the all RAM mode and they will point to $100-$111. This means that if your program uses interrupts the program itself may not use the RAM in the range $100-$111 or $BFF0-$BFFF. The interrupt vector space $BFE0-$BFFF may not be used as RAM because even the "A" board has not removed the buss contention from this block of RAM.

If you wish to run in the all RAM mode in BASIC, you must copy BASIC into RAM. Listed below is a short machine language routine which will do this:

```
        8E  80  00     LDX     #$8000   point X to start of extended BASIC
LOOP    B7  FF  DE     STA     $FFDE    set SAM to double 32K RAM mode
        EC  84         LDD     ,X       Get two bytes from BASIC
        B7  FF  DF     STA     $FFDF    set SAM to all RAM mode
        ED  81         STD     ,X++     save bytes to RAM
        8C  E0  00     CMPX    #$E000   compare to end of BASIC
        25  F3         BLO     LOOP     keep moving bytes
        7E             JMP     $A027    junp back to BASIC
```

Once BASIC is in RAM it may be modified either intentionally or accidentally. Also, if the SAM is reset out of the all RAM mode, the BASIC ROMs will be enabled again. You may reset the SAM to restore BASIC to RAM but if BASIC'S direct page variables have been changed it may not function as you expect.

The A board uses a 6822 in place of one of the 6821 PIA chips. The 6822 is functionally equivalent to the 6821, so that the data sheet for the 6821 will describe the function of the 6822.

SECTION 13
DISK BASIC

Adding a disk to the Color Computer will also give you an additional 8K of BASIC in ROM. Disk BASIC is entirely dedicated to providing the necessary routines to interface the disk to the computer and manage its file structure. There are no enhancements to the graphics routines of Extended BASIC nor are any new graphics routines added.

The heart of the Color Disk system is the Western Digital 1793 Floppy Disk Controller. The computer communicates with the 1793 through the four 1793 internal registers which are located at $FF48-$FF4B, and one write only latch located at $FF40. A description of these registers is given below:

ADDRESS	READ	WRITE
$FF48	STATUS	COMMAND
$FF49	TRACK	REGISTER
$FF4A	SECTOR	REGISTER
$FF4B	DATA	DATA

FIGURE 7 - 1793 REGISTERS

CODE	COMMAND
$03	RESTORE
$17	SEEK
$33	STEP
$53	STEP IN
$73	STEP OUT
$80	READ SECTOR
$A0	WRITE SECTOR
$C4	READ ADDRESS
$E4	READ TRACK
$F4	WRITE TRACK
$D0	FORCE INTERRUPT

FIGURE 8 - 1793 COMMANDS

BIT	FUNCTION
0	DRIVE SELECT 0
1	DRIVE SELECT 1
2	DRIVE SELECT 2
3	MOTOR ENABLE
4	PRECOMPENSATION ENABLE
5	SINGLE/DOUBLE DENSITY
6	DRIVE SELECT 3
7	NMI INTERRUPT ENABLE

FIGURE 9 - 1793 CONTROL REGISTER ($FF40)

The Color Disk User's Manual has a good description of several machine language routines in Disk BASIC and those routines will not be repeated in this book. The appendices contain a listing of Disk BASIC's tokens, jump tables and a commented listing of the DSKCON routine.

MEMORY MAP

HEX ADDRESS	
FFF2 FFFF	Interrupt Vectors
FFE0 FFF1	Reserved for future MPU enhancements
FFC0 FFDF	SAM Control Register
FF60 FFBF	Disk Boot ROM
FF40 FF5F	I/O_2
FF48 FF4B	1793 Internal Registers
FF40	Disk Controller Board Control Register
FF20 FF3F	I/O_1 PIA1
FF00 FF1F	I/O_0 PIA0
**********	BASIC ROMS
C000 DFFF	Disk BASIC
A000 BFFF	Color BASIC
8000 9FFF	Extended BASIC
**********	TOP OF RAM
	User program reserved area
	String space
	Stack

	FREE MEMORY

	Array variable storage area
	Simple variable storage area
	BASIC program
	Graphic pages reserved area - Extended only
	File control blocks - Disk only
	Direct access file reserved area - Disk only
0600 0989	Disk Variables
0400 05FF	Video display
0100 03FF	Color, Extended Variables
0000 00FF	Direct page
**********	BOTTOM OF RAM

DISPLAY CHARACTER SET

HEX VALUE		CHARACTER	HEX VALUE		CHARACTER
Non-Inverted	Inverted		Non-Inverted	Inverted	
00	40	@	18	58	X
01	41	A	19	59	Y
02	42	B	1A	5A	Z
03	43	C	1B	5B	[
04	44	D	1C	5C	\
05	45	E	1D	5D	]
06	46	F	1E	5E	↑
07	47	G	1F	5F	←
08	48	H	20	60	
09	49	I	21	61	!
0A	4A	J	22	62	"
0B	4B	K	23	63	#
0C	4C	L	24	64	$
0D	4D	M	25	65	%
0E	4E	N	26	66	&
0F	4F	O	27	67	'
10	50	P	28	68	(
11	51	Q	29	69	)
12	52	R	2A	6A	*
13	53	S	2B	6B	+
14	54	T	2C	6C	,
15	55	U	2D	6D	–
16	56	V	2E	6E	.
17	57	W	2F	6F	/

HEX VALUE		CHARACTER
Non-Inverted	Inverted	
30	70	0
31	71	1
32	72	2
33	73	3
34	74	4
35	75	5
36	76	6
37	77	7
38	78	8
39	79	9
3A	7A	:
3B	7B	;
3C	7C	<
3D	7D	=
3E	7E	>
3F	7F	?

A P P E N D I X A

```
*** BASIC ASSEMBLY PROGRAM


000C  BASVER  EQU   12        BASIC 1.2

8000  EXBAS   EQU   $8000
C000  DOSBAS  EQU   $C000



      **** BASIC'S DIRECT PAGE AND VARIABLES
      *
      *
      ***
      ***

C000  ROMPAK  EQU   $C000
      *
0008  BS      EQU   8         BACKSPACE
000D  CR      EQU   $D        ENTER
001B  ESC     EQU   $1B       ESCAPE CODE
000A  LF      EQU   $A        LINE FEED
000C  FORMF   EQU   $C        FORM FEED
0020  SPACE   EQU   $20       SPACE (BLANK)
      *
003A  STKBUF  EQU   58        STACK BUFFER AREA
045E  DEBDEL  EQU   $45E      DEBOUNCE DELAY
00FA  LBUFMX  EQU   250       MAX NUMBER OF CHARS IN A BASIC LINE
00FA  MAXLIN  EQU   $FA       MAXIMUM MS BYTE OF LINE NUMBER
      *
2600  DOSBUF  EQU   $2600     RAM LOAD LOCATION FOR THE DOS COMMAND
0100  SECLEN  EQU   256       LENGTH OF SECTOR IN BYTES
1200  TRKLEN  EQU   18*SECLEN LENGTH OF TRACK IN BYTES
004A  FATLEN  EQU   74        FILE ALLOCATION TABLE LENGTH
0023  TRKMAX  EQU   35        MAX NUMBER OF TRACKS
0044  GRANMX  EQU   68        MAXIMUM NUMBER OF GRANULES
0119  FCBLEN  EQU   SECLEN+25 FILE CONTROL BLOCK LENGTH
0010  INPFIL  EQU   $10       INPUT FILE TYPE
0020  OUTFIL  EQU   $20       OUTPUT FILE TYPE
0040  RANFIL  EQU   $40       RANDOM/DIRECT FILE TYPE

      *
      * PSEUDO PSEUDO OPS
0021  SKP1    EQU   $21       OP CODE OF BRN - SKIP ONE BYTE
008C  SKP2    EQU   $8C       OP CODE OF CMPX # - SKIP TWO BYTES
0086  SKP1LD  EQU   $86       OP CODE OF LDA # - SKIP THE NEXT INSTRUCTION
      *                       AND LOAD THE VALUE OF THAT INSTRUCTION INTO ACCA - THIS
      *                       IS USUALLY USED TO LOAD ACCA WITH A NON ZERO VALUE
      *
      * REGISTER ADDRESSES
FF00  PIA0    EQU   $FF00     PERIPHERAL INPUT ADAPTER #0
FF20  PIA1    EQU   $FF20     PERIPHERAL INPUT ADAPTER #1
FF20  DA      EQU   PIA1+0    DIGITAL/ANALOG CONVERTER
FF40  DSKREG  EQU   $FF40     DISK CONTROL REGISTER
FF48  FDCREG  EQU   $FF48     1793 CONTROL REGISTER
FFC0  SAMREG  EQU   $FFC0     SAM CONTROL REGISTER
      *
```

```
                        *
                        *
                        *
0000                    ORG   0
                0000    SETDP 0
                        *
0000                    ENDFLG  RMB   1       STOP/END FLAG: POSITIVE=STOP, NEG=END
0001                    CHARAC  RMB   1       TERMINATOR FLAG 1
0002                    ENDCHR  RMB   1       TERMINATOR FLAG 2
0003                    TMPLOC  RMB   1       SCRATCH VARIABLE
0004                    IFCTR   RMB   1       IF COUNTER - HOW MANY IF STATEMENTS IN A LINE
0005                    DIMFLG  RMB   1       ARRAY FLAG 0=EVALUATE, 1=DIMENSIONING
0006                    VALTYP  RMB   1       *PV TYPE FLAG: 0=NUMERIC, $FF=STRING
0007                    GARBFL  RMB   1       *TV STRING SPACE HOUSEKEEPING FLAG
0008                    ARYDIS  RMB   1       DISABLE ARRAY SEARCH: 00=ALLOW SEARCH
0009                    INPFLG  RMB   1       *TV INPUT FLAG: READ=0, INPUT<>0
000A                    RELFLG  RMB   1       *TV RELATIONAL OPERATOR FLAG
000B                    TEMPPT  RMB   2       *PV TEMPORARY STRING STACK POINTER
000D                    LASTPT  RMB   2       *PV ADDR OF LAST USED STRING STACK ADDRESS
000F                    TEMPTR  RMB   2       TEMPORARY POINTER
0011                    TMPTR1  RMB   2       TEMPORARY DESCRIPTOR STORAGE (STACK SEARCH)
                        ** FLOATING POINT ACCUMULATOR #2 (MANTISSA ONLY)
0013                    FPA2    RMB   4       FLOATING POINT ACCUMULATOR #2 MANTISSA
0017                    BOTSTK  RMB   2       BOTTOM OF STACK AT LAST CHECK
0019                    TXTTAB  RMB   2       *PV BEGINNING OF BASIC PROGRAM
001B                    VARTAB  RMB   2       *PV START OF VARIABLES
001D                    ARYTAB  RMB   2       *PV START OF ARRAYS
001F                    ARYEND  RMB   2       *PV END OF ARRAYS (+1)
0021                    FRETOP  RMB   2       *PV START OF STRING STORAGE (TOP OF FREE RAM)
0023                    STRTAB  RMB   2       *PV START OF STRING VARIABLES
0025                    FRESPC  RMB   2       UTILITY STRING POINTER
0027                    MEMSIZ  RMB   2       *PV TOP OF STRING SPACE
0029                    OLDTXT  RMB   2       SAVED LINE NUMBER DURING A "STOP"
002B                    BINVAL  RMB   2       BINARY VALUE OF A CONVERTED LINE NUMBER
002D                    OLDPTR  RMB   2       SAVED INPUT PTR DURING A "STOP"
002F                    TINPTR  RMB   2       TEMPORARY INPUT POINTER STORAGE
0031                    DATTXT  RMB   2       *PV 'DATA' STATEMENT LINE NUMBER POINTER
0033                    DATPTR  RMB   2       *PV 'DATA' STATEMENT ADDRESS POINTER
0035                    DATTMP  RMB   2       DATA POINTER FOR 'INPUT' & 'READ'
0037                    VARNAM  RMB   2       *TV TEMP STORAGE FOR A VARIABLE NAME
0039                    VARPTR  RMB   2       *TV POINTER TO A VARIABLE DESCRIPTOR
003B                    VARDES  RMB   2       TEMP POINTER TO A VARIABLE DESCRIPTOR
003D                    RELPTR  RMB   2       POINTER TO RELATIONAL OPERATOR PROCESSING ROUTINE
003F                    TRELFL  RMB   1       TEMPORARY RELATIONAL OPERATOR FLAG BYTE
                        *
                        * FLOATING POINT ACCUMULATORS #3,4 & 5 ARE MOSTLY USED
                        * USED AS SCRATCH PAD REGISTERS.
                        ** FLOATING POINT ACCUMULATOR #3 :PACKED: ($40-$44)
0040                    V40     RMB   1
0041                    V41     RMB   1
0042                    V42     RMB   1
0043                    V43     RMB   1
0044                    V44     RMB   1
                        ** FLOATING POINT ACCUMULATOR #4 :PACKED: ($45-$49)
0045                    V45     RMB   1
0046                    V46     RMB   1
0047                    V47     RMB   1
0048                    V48     RMB   2
                        ** FLOATING POINT ACCUMULATOR #5 :PACKED: ($4A-$4E)
004A                    V4A     RMB   1
004B                    V4B     RMB   2
```

```
004D            V4D     RMB    2
                ** FLOATING POINT ACCUMULATOR #0
004F            FPOEXP  RMB    1       *PV FLOATING POINT ACCUMULATOR #0 EXPONENT
0050            FPA0    RMB    4       *PV FLOATING POINT ACCUMULATOR #0 MANTISSA
0054            FPOSGN  RMB    1       *PV FLOATING POINT ACCUMULATOR #0 SIGN
                *
0055            COEFCT  RMB    1       POLYNOMIAL COEFFICIENT COUNTER
0056            STRDES  RMB    5       TEMPORARY STRING DESCRIPTOR
005B            FPCARY  RMB    1       FLOATING POINT CARRY BYTE
                ** FLOATING POINT ACCUMULATOR #1
005C            FP1EXP  RMB    1       *PV FLOATING POINT ACCUMULATOR #1 EXPONENT
005D            FPA1    RMB    4       *PV FLOATING POINT ACCUMULATOR #1 MANTISSA
0061            FP1SGN  RMB    1       *PV FLOATING POINT ACCUMULATOR #1 SIGN
                *
0062            RESSGN  RMB    1       SIGN OF RESULT OF FLOATING POINT OPERATION
0063            FPSBYT  RMB    1       FLOATING POINT SUB BYTE (FIFTH BYTE)
0064            COEFPT  RMB    2       POLYNOMIAL COEFFICIENT POINTER
0066            LSTTXT  RMB    2       CURRENT LINE POINTER DURING LIST
0068            CURLIN  RMB    2       *PV CURRENT LINE # OF BASIC PROGRAM. $FFFF = DIRECT
006A            DEVCFW  RMB    1       *TV TAB FIELD WIDTH
006B            DEVLCF  RMB    1       *TV TAB ZONE
006C            DEVPOS  RMB    1       *TV PRINT POSITION
006D            DEVWID  RMB    1       *TV PRINT WIDTH
006E            PRTDEV  RMB    1       *TV PRINT DEVICE: 0=NOT CASSETTE, -1=CASSETTE
006F            DEVNUM  RMB    1       *PV DEVICE NUMBER: -3=DLOAD,-2=PRINTER,
                *                          -1=CASSETTE, 0=SCREEN, 1-16=DISK
0070            CINBFL  RMB    1       *PV CONSOLE IN BUFFER FLAG: 00=NOT EMPTY, $FF=EMPTY
0071            RSTFLG  RMB    1       *PV WARM START FLAG: $55=WARM, OTHER=COLD
0072            RSTVEC  RMB    2       *PV WARM START VECTOR - JUMP ADDRESS FOR WARM START
0074            TOPRAM  RMB    2       *PV TOP OF RAM
0076                    RMB    2       SPARE: UNUSED VARIABLES
0078            FILSTA  RMB    1       *PV FILE STATUS FLAG: 0=CLOSED,1=INPUT,2=OUTPUT
0079            CINCTR  RMB    1       *PV CONSOLE IN BUFFER CHAR COUNTER
007A            CINPTR  RMB    2       *PV CONSOLE IN BUFFER POINTER
007C            BLKTYP  RMB    1       *TV CASS BLOCK TYPE: 0=HEADER, 1=DATA, $FF=EOF
007D            BLKLEN  RMB    1       *TV CASSETTE BYTE COUNT
007E            CBUFAD  RMB    2       *TV CASSETTE LOAD BUFFER POINTER
0080            CCKSUM  RMB    1       *TV CASSETTE CHECKSUM BYTE
0081            CSRERR  RMB    1       *TV ERROR FLAG/CHARACTER COUNT
0082            CPULWD  RMB    1       *TV PULSE WIDTH COUNT
0083            CPERTM  RMB    1       *TV BIT COUNTER
0084            CBTPHA  RMB    1       *TV BIT PHASE FLAG
0085            CLSTSN  RMB    1       *TV LAST SINE TABLE ENTRY
0086            GRBLOK  RMB    1       *TV GRAPHIC BLOCK VALUE FOR SET, RESET AND POINT
0087            IKEYIM  RMB    1       *TV INKEY$ RAM IMAGE
0088            CURPOS  RMB    2       *PV CURSOR LOCATION
008A            ZERO    RMB    2       *PV DUMMY - THESE TWO BYTES ARE ALWAYS ZERO
008C            SNDTON  RMB    1       *TV TONE VALUE FOR SOUND COMMAND
008D            SNDDUR  RMB    2       *TV DURATION VALUE FOR SOUND COMMAND
                **********************************************************************
                ** THESE BYTES ARE MOVED DOWN FROM ROM
                ***                     INIT        DESCRIPTION
                *                       VALUE
008F            CMPMID  RMB    1        18    *PV 1200/2400 HERTZ PARTITION
0090            CMP0    RMB    1        24    *PV UPPER LIMIT OF 1200 HERTZ PERIOD
0091            CMP1    RMB    1        10    *PV UPPER LIMIT OF 2400 HERTZ PERIOD
0092            SYNCLN  RMB    2        128   *PV NUMBER OF $55'S TO CASSETTE LEADER
0094            BLKCNT  RMB    1        11    *PV CURSOR BLINK DELAY
0095            LPTBTD  RMB    2        97    *PV BAUD RATE CONSTANT (600)
0097            LPTLND  RMB    2        1     *PV PRINTER CARRIAGE RETURN DELAY
0099            LPTCFW  RMB    1        16    *PV TAB FIELD WIDTH
009A            LPTLCF  RMB    1        112   *PV LAST TAB ZONE
```

```
009B                    LPTWID  RMB     1       132   *PV PRINTER WIDTH
009C                    LPTPOS  RMB     1       0     *PV LINE PRINTER POSITION
009D                    EXECJP  RMB     2       LB4AA *PV JUMP ADDRESS FOR EXEC COMMAND

                        ** THIS ROUTINE PICKS UP THE NEXT INPUT CHARACTER FROM
                        ** BASIC. THE ADDRESS OF THE NEXT BASIC BYTE TO BE
                        ** INTERPRETED IS STORED AT CHARAD.

009F 0C  A7     GETNCH  INC     <CHARAD+1 *PV INCREMENT LS BYTE OF INPUT POINTER
00A1 26  02             BNE     GETCCH    *PV BRANCH IF NOT ZERO (NO CARRY)
00A3 0C  A6             INC     <CHARAD   *PV INCREMENT MS BYTE OF INPUT POINTER
00A5 B6         GETCCH  FCB     $B6       *PV OP CODE OF LDA EXTENDED
00A6            CHARAD  RMB     2         *PV THESE 2 BYTES CONTAIN ADDRESS OF THE CURRENT
                *                              CHARACTER WHICH THE BASIC INTERPRETER IS PROCESSING
00A8 7E  AA1A           JMP     BROMHK    JUMP BACK INTO THE BASIC ROM
                        ***************************************************************
00AB            VAB     RMB     1         = LOW ORDER FOUR BYTES OF THE PRODUCT
00AC            VAC     RMB     1         = OF A FLOATING POINT MULTIPLICATION
00AD            VAD     RMB     1         = THESE BYTES ARE USE AS RANDOM DATA
00AE            VAE     RMB     1         = BY THE RND STATEMENT
                *
                * EXTENDED BASIC VARIABLES
00AF            TRCFLG  RMB     1         *PV TRACE FLAG 0=OFF ELSE=ON
00B0            USRADR  RMB     2         *PV ADDRESS OF THE START OF USR VECTORS
00B2            FORCOL  RMB     1         *PV FOREGROUND COLOR
00B3            BAKCOL  RMB     1         *PV BACKGROUND COLOR
00B4            COLOR   RMB     1         *PV CURRENT COLOR BEING USED BY EX BASIC
00B5            ALLCOL  RMB     1         *PV ALL PIXELS IN THIS BYTE SET TO COLOR OF VB3
00B6            PMODE   RMB     1         *PV PMODE'S MODE ARGUMENT
00B7            ENDGRP  RMB     1         *PV END OF CURRENT GRAPHIC PAGE (MS BYTE)
00B8                    RMB     1         SPARE: UNUSED VARIABLE
00B9            HORBYT  RMB     1         *PV NUMBER OF BYTES/HORIZONTAL GRAPHIC LINE
00BA            BEGGRP  RMB     1         *PV START OF CURRENT GRAPHIC PAGE (MS BYTE)
00BB                    RMB     1         SPARE: UNUSED VARIABLE
00BC            GRPRAM  RMB     1         *PV START OF GRAPHIC RAM (MS BYTE)
00BD            HORBEG  RMB     2         *PV HORIZ COORD - START POINT
00BF            VERBEG  RMB     2         *PV VERT COORD  - START POINT
00C1            CSSVAL  RMB     1         *PV SCREEN'S COLOR SET ARGUMENT
00C2            SETFLG  RMB     1         *PV PRESET/PSET FLAG: 0=PRESET, 1=PSET
00C3            HOREND  RMB     2         *PV HORIZ COORD - ENDING POINT
00C5            VEREND  RMB     2         *PV VERT COORD - ENDING POINT
00C7            HORDEF  RMB     2         *PV HORIZ COORD - DEFAULT COORD
00C9            VERDEF  RMB     2         *PV VERT COORD - DEFAULT COORD
                *
                * EXTENDED BASIC SCRATCH PAD VARIABLES
00CB            VCB     RMB     2
00CD            VCD     RMB     2
00CF            VCF     RMB     2
00D1            VD1     RMB     2
00D3            VD3     RMB     1
00D4            VD4     RMB     1
00D5            VD5     RMB     1
00D6            VD6     RMB     1
00D7            VD7     RMB     1
00D8            VD8     RMB     1
00D9            VD9     RMB     1
00DA            VDA     RMB     1

00DB            CHGFLG  RMB     1         *TV FLAG TO INDICATE IF GRAPHIC DATA HAS BEEN CHANGED
00DC            TMPSTK  RMB     2         *TV STACK POINTER STORAGE DURING PAINT
00DE            OCTAVE  RMB     1         *PV OCTAVE VALUE (PLAY)
```

```
00DF            VOLHI   RMB   1       *PV VOLUME HIGH VALUE (PLAY)
00E0            VOLLOW  RMB   1       *PV VOLUME LOW VALUE (PLAY)
00E1            NOTELN  RMB   1       *PV NOTE LENGTH (PLAY)
00E2            TEMPO   RMB   1       *PV TEMPO VALUE (PLAY)
00E3            PLYTMR  RMB   2       *PV TIMER FOR THE "PLAY" COMMAND
00E5            DOTVAL  RMB   1       *TV DOTTED NOTE TIMER SCALE FACTOR
00E6            DLBAUD  RMB   1       *PV DLOAD BAUD RATE CONSTANT $B0=300. $2C=1200
00E7            TIMOUT  RMB   1       *PV DLOAD TIMEOUT CONSTANT
00E8            ANGLE   RMB   1       *PV ANGLE VALUE (DRAW)
00E9            SCALE   RMB   1       *PV SCALE VALUE (DRAW)

                * DSKCON VARIABLES
00EA            DCOPC   RMB   1       *PV DSKCON OPERATION CODE   0-3
00EB            DCDRV   RMB   1       *PV DSKCON DRIVE NUMBER     0-3
00EC            DCTRK   RMB   1       *PV DSKCON TRACK NUMBER     0-34
00ED            DSEC    RMB   1       *PV DSKCON SECTOR NUMBER    1-18
00EE            DCBPT   RMB   2       *PV DSKCON DATA POINTER
00F0            DCSTA   RMB   1       *PV DSKCON STATUS BYTE
                *
00F1            FCBTMP  RMB   2       TEMPORARY FCB POINTER
                *
00F3                    RMB   13      SPARE: UNUSED VARIABLES
                *

                *                     BASIC  EXBASIC  DOSBASIC

0100            SW3VEC  RMB   3       $0000  $0000   $3B3B    SWI3 VECTOR
0103            SW2VEC  RMB   3       $0000  $0000   $3B3B    SWI2 VECTOR
0106            SWIVEC  RMB   3       $C22D  $C22D   $5225    SWI VECTOR
0109            NMIVEC  RMB   3       $C22D  $C22D   $D7AE    NMI VECTOR
010C            IRQVEC  RMB   3       $A9B3  $894C   $D7BC    IRQ VECTOR
010F            FRQVEC  RMB   3       $A0F6  $A0F6   $A0F6    FIRQ VECTOR

0112            USRJMP  RMB   3       JUMP ADDRESS FOR BASIC'S USR FUNCTION
                TIMVAL
                * RMB   2       TIMER VALUE FOR EXBAS
                * RMB   1       UNUSED BY EXBAS OR DISK BASIC
0115            RVSEED  RMB   1       * FLOATING POINT RANDOM NUMBER SEED EXPONENT
0116                    RMB   4       * MANTISSA: INITIALLY SET TO $804FC75259
011A            CASFLG  RMB   1       UPPER CASE/LOWER CASE FLAG: $FF= UPPER,0=LOWER
011B            DEBVAL  RMB   2       KEYBOARD DEBOUNCE DELAY (SET TO $45E)
011D            EXPJMP  RMB   3       JUMP ADDRESS FOR EXPONENTIATION
                **                    INITIALLY SET TO ERROR FOR BASIC. $8489 FOR EX BASIC

                ***     COMMAND INTERPRETATION VECTOR TABLE

                ** FOUR SETS OF 10 BYTE TABLES:
                ** THE LAST USED TABLE MUST BE FOLLOWED BY A ZERO BYTE
                * THE JUMP TABLE VECTORS (3,4 AND 8,9) POINT TO THE JUMP TABLE FOR
                * THE FIRST TABLE. FOR ALL OTHER TABLES, THESE VECTORS POINT TO A
                * ROUTINE WHICH WILL POINT YOU TO THE CORRECT JUMP TABLE.
                *
                *    BYTE         DESCRIPTION
                *   0          NUMBER OF RESERVED WORDS
                *   1,2        TABLE OF RESERVED WORDS
                *   3,4        JUMP TABLE FOR COMMANDS
                *   5          NUMBER OF INTRINSIC FUNCTIONS
                *   6,7        TABLE OF INTRINSIC FUNCTIONS
                *   8,9        JUMP TABLE FOR INTRINSIC FUNCTIONS
                *   10         0 BYTE - END OF TABLE FLAG (LAST TABLE ONLY)
```

```
0120            CONVEC  RMB    10       BASIC'S TABLE
012A                    RMB    10       EX BASIC'S TABLE
0134                    RMB    10       DISC BASIC'S TABLE (UNUSED BY EX BASIC)

                ***** USR FUNCTION VECTOR ADDRESSES (EX BASIC ONLY)
013E                    RMB    2        USR 0 VECTOR
0140                    RMB    2        USR 1
0142                    RMB    2        USR 2
0144                    RMB    2        USR 3
0146                    RMB    2        USR 4
0148                    RMB    2        USR 5
014A                    RMB    2        USR 5
014C                    RMB    2        USR 7
014E                    RMB    2        USR 8
0150                    RMB    2        USR 9

                *** THE ABOVE 20 BYTE USR ADDR VECTOR TABLE IS MOVED TO
                *** $95F-$972 BY DISC BASIC. THE 20 BYTES FROM $13E-$151
                *** ARE REDEFINED AS FOLLOWS:

                * RMB 10 USER (SPARE) COMMAND INTERPRETATION TABLE SPACE
                * FCB 0  END OF COMM INTERP TABLE FLAG
                * RMB 9  UNUSED BY DISK BASIC
                *
                *       COMMAND INTERPRETATION TABLE VALUES
                *  BYTE  BASIC  EX BASIC  DISK BASIC
                *  0     $35                        BASIC TABLE
                *  1,2   $AA66
                *  3,4   $AB67
                *  5     $14
                *  6,7   $AB1A
                *  8,9   $AA29

                *  0             $19                EX BASIC TABLE
                *  1,2           $8183
                *  3,4           $813C    $CE2E
                *  5             $0E
                *  6,7           $821E
                *  8,9           $8168    $CE56

                *  0                      $13       DISK BASIC TABLE
                *  1,2                    $C17F
                *  3,4                    $C2C0
                *  5                      $06
                *  6,7                    $C201
                *  8,9                    $C236


0152            KEYBUF  RMB    8        KEYBOARD MEMORY BUFFER
015A            POTVAL  RMB    1        LEFT VERTICAL JOYSTICK DATA
015B                    RMB    1        LEFT HORIZONTAL JOYSTICK DATA
015C                    RMB    1        RIGHT VERTICAL JOYSTICK DATA
015D                    RMB    1        RIGHT HORIZONTAL JOYSTICK DATA

                * BASIC'S RAM VECTORS - INITIALIZED TO RTS BY COLOR BASIC
                * 25 SETS OF 3 BYTE INSTRUCTIONS WHICH ARE CALLED BY COLOR BASIC
                * EXTENDED AND DISK BASIC. THEIR PURPOSE IS TO ALLOW ENHANCEMENTS (SUCH
                * AS EX BASIC AND DOS BASIC) AS MORE ROMS ARE ADDED TO THE
                * SYSTEM BY EFFECTIVELY ALLOWING MORE CODE TO BE ADDED TO THE
                * ROUTINES IN EARLIER ROMS. THIS NEW CODE IS LOCATED IN THE NEW ROMS
```

```
                        * AND THE ADDRESS TO GET TO THE NEW CODE IS IN BYTES 1 & 2 OF THE
                        * RAM VECTOR. BYTE 0 WILL CONTAIN A $7E WHICH IS THE FIRST BYTE OF
                        * THE JMP INSTRUCTION.
                        * THE FIRST ADDRESS IN THIS TABLE IS THE ADDRESS IN BASIC WHICH
                        * CALLS THE RAM VECTOR. THE SECOND ADDRESS IS THE VALUE WHICH
                        * EX BASIC PUTS IN THE RAM VECTOR (IF ANY) AND THE THIRD ADDRESS
                        * IS THE VALUE WHICH DISK BASIC PUTS THERE (IF ANY)

015E                    RVEC0   RMB     3       $A5F6       $C426 OPEN COMMAND
0161                    RVEC1   RMB     3       $A5B9       $C838 CHECK DEVICE NUMBER
0164                    RVEC2   RMB     3       $A35F       $C843 SET PRINT PARAMETERS

0167                    RVEC3   RMB     3       $A282 $8273 $CB4A CONSOLE OUT
016A                    RVEC4   RMB     3       $A176 $8CF1 $C58F CONSOLE IN
016D                    RVEC5   RMB     3       $A3ED       $C818 INPUT FILE HANDLER
0170                    RVEC6   RMB     3       $A406       $C81B PRINT # HANDLER
0173                    RVEC7   RMB     3       $A426       $CA3B CLOSE ALL FILES
0176                    RVEC8   RMB     3       $A42D $8286 $CA4B CLOSE A FILE HANDLER
0179                    RVEC9   RMB     3       $B918 $8E90 $8E90 PRINT
017C                    RVEC10  RMB     3       $B061       $CC5B READ
017F                    RVEC11  RMB     3       $A549       $C859 BREAK CHECK
0182                    RVEC12  RMB     3       $A390       $C6B7 INPUTTING A BASIC LINE
0185                    RVEC13  RMB     3       $A4BF       $CA36 TERMINATE BASIC LINE INPUT
0188                    RVEC14  RMB     3       $A5CE       $C860 EOF COMMAND
018B                    RVEC15  RMB     3       $B223 $8846 $CDF6 EVALUATE AN EXPRESSION
018E                    RVEC16  RMB     3       $AC46       $C6B7 RESERVED FOR ON ERROR GOTO COMMAND
0191                    RVEC17  RMB     3       $AC49 $88F0 $C24D ERROR DRIVER
0194                    RVEC18  RMB     3       $AE75 $829C $C990 RUN
0197                    RVEC19  RMB     3       $BD22 $87E5       ASCII TO FLOATING POINT CONVERSION
019A                    RVEC20  RMB     3       $AD9E $82B9       BASIC'S COMMAND INTERPRETATION LOOP
019D                    RVEC21  RMB     3       $A8C4             RESET/SET/POINT COMMANDS
01A0                    RVEC22  RMB     3       $A910       $C29A CLS
                        *                       $8162              EXBAS' SECONDARY TOKEN HANDLER
                        *                       $8AFA              EXBAS' RENUM TOKEN CHECK
                        *                       $975C              EXBAS' GET/PUT
01A3                    RVEC23  RMB     3       $B821 $8304       CRUNCH BASIC LINE
01A6                    RVEC24  RMB     3       $B7C2             UNCRUNCH BASIC LINE

01A9                    STRSTK  RMB     8*5     STRING DESCRIPTOR STACK
01D1                    CFNBUF  RMB     9       CASSETTE FILE NAME BUFFER
01DA                    CASBUF  RMB     256     CASSETTE FILE DATA BUFFER
02DA                    LINHDR  RMB     2       LINE INPUT BUFFER HEADER
02DC                    LINBUF  RMB     LBUFMX+1 BASIC LINE INPUT BUFFER
03D7                    STRBUF  RMB     41      STRING BUFFER

0400                    VIDRAM  RMB     $200    VIDEO DISPLAY AREA

                        *START OF ADDITIONAL RAM VARIABLE STORAGE (DISK BASIC ONLY)
0600                    DBUF0   RMB     SECLEN  I/O BUFFER #0
0700                    DBUF1   RMB     SECLEN  I/O BUFFER #1
0800                    FATBL0  RMB     FATLEN  FILE ALLOCATION TABLE - DRIVE 0
084A                    FATBL1  RMB     FATLEN  FILE ALLOCATION TABLE - DRIVE 1
0894                    FATBL2  RMB     FATLEN  FILE ALLOCATION TABLE - DRIVE 2
08DE                    FATBL3  RMB     FATLEN  FILE ALLOCATION TABLE - DRIVE 3
0928                    FCBV1   RMB     16*2    FILE BUFFER VECTORS (16)
0948                    RNBFAD  RMB     2       START OF FREE RANDOM FILE BUFFER AREA
094A                    FCBADR  RMB     2       START OF FILE CONTROL BLOCKS
094C                    DNAMBF  RMB     8       DISK FILE NAME BUFFER
0954                    DEXTBF  RMB     3       DISK FILE EXTENSION NAME BUFFER
0957                    DFLTYP  RMB     1       DISK FILE TYPE: 0=BASIC, 1=DATA, 2=MACHINE
```

```
                        *                       LANGUAGE, 3=TEXT EDITOR SOURCE FILE
0958                    DASCFL  RMB     1       ASCII FLAG: 0=CRUNCHED, 1=ASCII
0959                    DRUNFL  RMB     1       RUN FLAG: (IF BIT 1=1 THEN RUN, IF BIT 0=1, THEN CLOSE
                        *                       ALL FILES BEFORE RUNNING
095A                    DEFDRV  RMB     1       DEFAULT DRIVE NUMBER
095B                    FCBACT  RMB     1       NUMBER OF FCBS ACTIVE
095C                    DRESFL  RMB     1       RESET FLAG: <>0 WILL CAUSE A 'NEW' & SHUT DOWN ALL FCBS
095D                    DLODFL  RMB     1       LOAD FLAG - CAUSE A 'NEW' FOLLOWING A LOAD ERROR
095E                    DMRGFL  RMB     1       MERGE FLAG: 0=NO MERGE, $FF=MERGE
095F                    DUSRVC  RMB     20      DISK BASIC USR COMMAND VECTORS
                        ***     DISK FILE WORK AREA FOR DIRECTORY SEARCH
                        *               EXISTING FILE
0973                    V973    RMB     1       SECTOR NUMBER
0974                    V974    RMB     2       RAM DIRECTORY IMAGE ADDRESS
0976                    V976    RMB     1       FIRST GRANULE NUMBER
                        *               UNUSED FILE
0977                    V977    RMB     1       SECTOR NUMBER
0978                    V978    RMB     2       RAM DIRECTORY IMAGE ADDRESS

097A                    WFATVL  RMB     2       WRITE FAT VALUE: # OF GRANULES WHICH MUST BE REMOVED
                        *                       FROM THE FAT TO TRIGGER A WRITE FAT TO DISK SEQUENCE
097C                    DFFLEN  RMB     2       DIRECT ACCESS FILE RECORD LENGTH
097E                    DR0TRK  RMB     1       CURRENT TRACK NUMBER, DRIVE 0
097F                    DR1TRK  RMB     1       CURRENT TRACK NUMBER, DRIVE 1
0980                    DR2TRK  RMB     1       CURRENT TRACK NUMBER, DRIVE 2
0981                    DR3TRK  RMB     1       CURRENT TRACK NUMBER, DRIVE 3
0982                    NMIFLG  RMB     1       NMI FLAG: 0=DON'T VECTOR <>0=VECTOR OUT
0983                    DNMIVC  RMB     2       NMI VECTOR - WHERE TO JUMP FOLLOWING AN NMI
                        *                       INTERRUPT IF THE NMI FLAG IS SET
0985                    RDYTMR  RMB     1       DRIVE NOT READY TIMER
0986                    DRGRAM  RMB     1       RAM IMAGE OF DSKREG ($FF40)
0987                    DVERFL  RMB     1       VERIFY FLAG: 0=OFF, $FF=ON
0988                    ATTCTR  RMB     1       READ/WRITE ATTEMPT COUNTER - NUMBER OF TIMES THE
                        *                       DISK WILL ATTEMPT TO RETRIEVE OR WRITE DATA
                        *                       BEFORE IT GIVES UP AND ISSUES AN ERROR.
                        *
0989                    DFLBUF  RMB     SECLEN  INITIALIZED TO SECLEN BY DISKBAS
                        ********************************
                        *RANDOM FILE RESERVED AREA
                        ********************************
                        *FILE CONTROL BLOCKS AND BUFFERS
                        ********************************
                        *GRAPHIC PAGE RESERVED AREA
                        ********************************
                        *BASIC PROGRAM
                        ********************************
                        *VARIABLE STORAGE AREA
                        ********************************
                        *ARRAY STORAGE AREA
                        ********************************
                        *
                        *   FREE MEMORY
                        *
                        ********************************
                        *STACK
                        ********************************
                        *STRING SPACE
                        ********************************
                        *USER PROGRAM RESERVED AREA
                        ********************************
                        *END OF RAM
```

```
                        *
                0032    BYEAR   EQU     '2
                        *
A000                    ORG     $A000       START OF COLOR BASIC

A000                    BASIC
                        *  INDIRECT JUMP TABLE
A000 A1CB               POLCAT  FDB     KEYIN       GET A KEYSTROKE
A002 A282               CHROUT  FDB     PUTCHR      OUTPUT A CHARACTER
A004 A77C               CSRDON  FDB     CASON       TURN ON CASSETTE MOTOR, START READING
A006 A70B               BLKIN   FDB     GETBLK      READ A BLOCK FROM CASSETTE
A008 A7F4               BLKOUT  FDB     SNDBLK      WRITE A BLOCK TO CASSETTE
A00A A9DE               JOYIN   FDB     GETJOY      READ JOY STICKS
A00C A7D8               WRTLDR  FDB     WRLDR       TURN ON MOTOR AND WRITE $55'S TO CASSETTE
                        *
A00E 10CE 03D7  LA00E   LDS     #LINBUF+LBUFMX+1 SET STACK TO TOP OF LINE INPUT BUFFER
A012 86   37            LDA     #$37        *
A014 B7   FF23          STA     PIA1+3      * ENABLE 63.5 MICROSECOND INTERRUPT
A017 96   71            LDA     RSTFLG      GET WARM START FLAG
A019 81   55            CMPA    #$55        IS IT A WARM START?
A01B 26   57            BNE     CLRRAM      NO - DO A COLD START
A01D 9E   72            LDX     RSTVEC      WARM START VECTOR
A01F A6   84            LDA     ,X          GET FIRST BYTE OF WARM START ADDR
A021 81   12            CMPA    #$12        IS IT NOP?
A023 26   4F            BNE     CLRRAM      NO - DO A COLD START
A025 6E   84            JMP     ,X          YES, GO THERE
                        *

A027 31   8C E4 RESVEC  LEAY    LA00E,PCR POINT Y TO WARM START CHECK CODE
A02A 8E   FF20  LA02A   LDX     #PIA1       POINT X TO PIA1
A02D 6F   1D            CLR     -3,X        CLEAR PIA0 CONTROL REGISTER A
A02F 6F   1F            CLR     -1,X        CLEAR PIA0 CONTROL REGISTER B
A031 6F   1C            CLR     -4,X        SET PIA0 SIDE A TO INPUT
A033 CC   FF34          LDD     #$FF34
A036 A7   1E            STA     -2,X        SET PIA0 SIDE B TO OUTPUT
A038 E7   1D            STB     -3,X        * ENABLE PIA0 PERIPHERAL REGISTERS, DISABLE PIA0
A03A E7   1F            STB     -1,X        * MPU INTERRUPTS, SET CA2, CA1 TO OUTPUTS
A03C 6F   01            CLR     1,X         CLEAR CONTROL REGISTER A ON PIA1
A03E 6F   03            CLR     3,X         CLEAR CONTROL REGISTER B ON PIA1
A040 4A                 DECA                A REG NOW HAS $FE
A041 A7   84            STA     ,X          BITS 1-7 ARE OUTPUTS, BIT 0 IS INPUT ON PIA1 SIDE A
A043 86   F8            LDA     #$F8
A045 A7   02            STA     2,X         BITS 0-2 ARE INPUTS, BITS 3-7 ARE OUTPUTS ON B SIDE
A047 E7   01            STB     1,X         * ENABLE PERIPHERAL REGISTERS, DISABLE PIA1 MPU
A049 E7   03            STB     3,X         * INTERRUPTS AND SET CA2, CB2 AS OUTPUTS
A04B 6F   02            CLR     2,X         SET 6847 MODE TO ALPHA-NUMERIC
A04D C6   02            LDB     #2          *
A04F E7   84            STB     ,X          * MAKE RS232 OUTPUT MARKING
A051 CE   FFC0          LDU     #SAMREG     SAM CONTROL REGISTER ADDR
A054 C6   10            LDB     #16         16 SAM CONTROL REGISTER BITS
A056 A7   C1    LA05E   STA     ,U++        ZERO OUT SAM CONTROL REGISTER BIT
A058 5A                 DECB                * DECREMENT COUNTER AND
A059 26   FB            BNE     LA05E       * BRANCH IF NOT DONE
A05B B7   FFC9          STA     SAMREG+9    SET DISPLAY PAGE AT $400
A05E 1F   9B            TFR     B,DP        SET DIRECT PAGE TO ZERO
A060 C6   04            LDB     #4          USE AS A MASK TO CHECK RAMSZ INPUT
A062 A7   1E            STA     -2,X        SET RAMSZ STROBE HIGH
A064 E5   02            BITB    2,X         CHECK RAMSZ INPUT
```

```
A066 27  0A          BEQ   LA072     BRANCH IF JUMPER SET FOR 4K RAMS
A068 6F  1E          CLR   -2,X      SET RAMSZ STROBE LOW
A06A E5  02          BITB  2,X       CHECK RAMSZ INPUT
A06C 27  02          BEQ   LA070     BRANCH IF JUMPER SET FOR 64K RAMS
A06E 33  5E          LEAU  -2,U      ADJUST POINTER TO SET SAM FOR 16K RAMS
A070 A7  5D    LA070 STA   -3,U      PROGRAM SAM FOR 16K OR 64K RAMS
A072 6E  A4    LA072 JMP   ,Y        GO DO A WARM OR COLD START
                     * COLD START ENTRY
A074 8E  0401  CLRRAM LDX  #VIDRAM+1 POINT X TO CLEAR 1ST 1K OF RAM
A077 6F  83    LA077 CLR   ,--X      MOVE POINTER DOWN TWO-CLEAR BYTE
A079 30  01          LEAX  1,X       ADVANCE POINTER ONE
A07B 26  FA          BNE   LA077     KEEP GOING IF NOT AT BOTTOM OF PAGE 0
A07D BD  A928        JSR   LA928     CLEAR SCREEN
A080 6F  80          CLR   ,X+       CLEAR 1ST BYTE OF BASIC PROGRAM
A082 9F  19          STX   TXTTAB    BEGINNING OF BASIC PROGRAM
A084 A6  02    LA084 LDA   2,X       LOOK FOR END OF MEMORY
A086 43              COMA            * COMPLIMENT IT AND PUT IT BACK
A087 A7  02          STA   2,X       * INTO SYSTEM MEMORY
A089 A1  02          CMPA  2,X       IS IT RAM?
A08B 26  06          BNE   LA093     BRANCH IF NOT (ROM, BAD RAM OR NO RAM)
A08D 30  01          LEAX  1,X       MOVE POINTER UP ONE
A08F 63  01          COM   1,X       RE-COMPLEMENT TO RESTORE BYTE
A091 20  F1          BRA   LA084     KEEP LOOKING FOR END OF RAM
A093 9F  74    LA093 STX   TOPRAM    SAVE ABSOLUTE TOP OF RAM
A095 9F  27          STX   MEMSIZ    SAVE TOP OF STRING SPACE
A097 9F  23          STX   STRTAB    SAVE START OF STRING VARIABLES
A099 30  89 FF38     LEAX  -200,X    CLEAR 200 - DEFAULT STRING SPACE TO 200 BYTES
A09D 9F  21          STX   FRETOP    SAVE START OF STRING SPACE
A09F 1F  14          TFR   X,S       PUT STACK THERE
A0A1 8E  A10D        LDX   #LA10D    POINT X TO ROM SOURCE DATA
A0A4 CE  008F        LDU   #CMPMID   POINT U TO RAM DESTINATION
A0A7 C6  1C          LDB   #28       MOVE 28 BYTES
A0A9 BD  A59A        JSR   LA59A     MOVE 28 BYTES FROM ROM TO RAM
A0AC CE  010C        LDU   #IRQVEC   POINT U TO NEXT RAM DESTINATION
A0AF C6  1E          LDB   #30       MOVE 30 MORE BYTES
A0B1 BD  A59A        JSR   LA59A     MOVE 30 BYTES FROM ROM TO RAM
A0B4 AE  14          LDX   -12,X     POINT X TO SYNTAX ERROR ADDRESS
A0B6 AF  43    LA0B6 STX   3,U       * SET EXBAS COMMAND INTERPRETATION
A0B8 AF  48          STX   8,U       * HANDLERS TO SYNTAX ERROR
A0BA 8E  015E        LDX   #RVEC0    POINT X TO START OF RAM VECTORS
A0BD CC  394B        LDD   #$394B    SET UP TO SAVE 75 RTS'
A0C0 A7  80    LA0C0 STA   ,X+       FILL THE RAM VECTORS WITH RTS'
A0C2 5A              DECB            * DECREMENT COUNTER AND
A0C3 26  FB          BNE   LA0C0     * BRANCH IF NOT DONE
A0C5 B7  02D9        STA   LINHDR-1  PUT RTS IN LINHDR-1


A0C8 BD  AD19        JSR   LAD19     GO DO A 'NEW'
A0CB 8E  4558        LDX   #$4558    ASCII 'EX' (FIRST TWO LETTERS OF 'EXTENDED')
A0CE BC  8000        CMPX  EXBAS     SEE IF EXTENDED ROM IS THERE
A0D1 1027 DF2D       LBEQ  EXBAS+2   IF IT IS, BRANCH TO IT
A0D5 1C  AF          ANDCC #$AF      ENABLE IRQ, FIRQ
A0D7 8E  A146        LDX   #LA147-1  POINT X TO COLOR BASIC COPYRIGHT MESSAGE
A0DA BD  B99C        JSR   LB99C     PRINT 'COLOR BASIC'
A0DD 8E  A0E8        LDX   #BAWMST   WARM START ADDRESS
A0E0 9F  72          STX   RSTVEC    SAVE IT
A0E2 86  55          LDA   #$55      WARM START FLAG
A0E4 97  71          STA   RSTFLG    SAVE IT
A0E6 20  0B          BRA   LA0F3     GO TO BASIC'S MAIN LOOP
A0E8 12        BAWMST NOP            NOP REQ'D FOR WARM START
```

```
A0E9 0F   6F            CLR    DEVNUM    SET DEVICE NUMBER TO SCREEN
A0EB BD   AD33          JSR    LAD33     DO PART OF A NEW
A0EE 1C   AF            ANDCC  #$AF      ENABLE IRQ,FIRQ
A0F0 BD   A928          JSR    LA928     CLEAR SCREEN
A0F3 7E   AC73    LA0F3 JMP    LAC73     GO TO MAIN LOOP OF BASIC
                  *
                  *    FIRQ SERVICE ROUTINE
A0F6 7D   FF23    BFRQSV TST   PIA1+3    CARTRIDGE INTERRUPT?
A0F9 2B   01            BMI    LA0FC     YES
A0FB 3B                 RTI
A0FC BD   A7D1    LA0FC JSR    LA7D1     DELAY FOR A WHILE
A0FF BD   A7D1          JSR    LA7D1     KEEP DELAYING
A102 31   8C 03  LA102  LEAY   <LA108,PCR Y = ROM-PAK START UP VECTOR
A105 7E   A02A          JMP    LA02A     GO DO INITIALIZATION
A108 0F   71      LA108 CLR    RSTFLG    CLEAR WARM START FLAG
A10A 7E   C000          JMP    ROMPAK    JUMP TO EXTERNAL ROM PACK
                  *
                  *  THESE BYTES ARE MOVED TO ADDRESSES $8F-$AA THE DIRECT PAGE
A10D 12           LA10D FCB    18        MID BAND PARTITION OF 1200/2400 HERTZ PERIOD
A10E 18                 FCB    24        UPPER LIMIT OF 1200 HERTZ PERIOD
A10F 0A                 FCB    10        UPPER LIMIT OF 2400 HERTZ PERIOD
A110 0080               FDB    128       NUMBER OF 55'S TO CASSETTE LEADER
A112 0B                 FCB    11        CURSOR BLINK DELAY
A113 0058               FDB    88        CONSTANT FOR 600 BAUD  VER 1.2 & UP
A115 0001               FDB    1         PRINTER CARRIAGE RETURN DELAY
A117 10                 FCB    16        TAB FIELD WIDTH
A118 70                 FCB    112       LAST TAB ZONE
A119 84                 FCB    132       PRINTER WIDTH
A11A 00                 FCB    0         LINE PRINTER POSITION
A11B B44A               FDB    LB44A     ARGUMENT OF EXEC COMMAND - SET TO 'FC' ERROR
                  * LINE INPUT ROUTINE
A11D 0C   A7            INC    CHARAD+1
A11F 26   02            BNE    *+4
A121 0C   A6            INC    CHARAD
A123 B6   0000          LDA    >0000
A126 7E   AA1A          JMP    BROMHK
                  *
                  * THESE BYTES ARE MOVED TO ADDRESSES $10C-$129
A129 7E   A9B3          JMP    BIRQSV    IRQ SERVICE
>A12C 7E  A0F6          JMP    BFRQSV    FIRQ SERVICE
A12F 7E   B44A          JMP    LB44A     USR ADDRESS FOR BK BASIC (INITIALIZED TO 'FC' ERROR)
A132 80                 FCB    $80       *RANDOM SEED
A133 4FC7               FDB    $4FC7     *RANDOM SEED OF MANTISSA
A135 5259               FDB    $5259     .811635157
A137 FF                 FCB    $FF       UPPER CASE/LOWER CASE FLAG (STARTS SET TO UPPER)
A138 045E               FDB    DEBDEL    KEYBOARD DEBOUNCE DELAY
A13A 7E   B277          JMP    LB277     DISPATCH FOR EXPONENTIATION (INITIALIZED TO SYNTAX ERROR)
                  *BASIC COMMAND INTERPRETATION TABLE ROM IMAGE
A13D 35                 FCB    $35       53 BASIC COMMANDS
A13E AA66               FDB    LAA66     POINTS TO RESERVED WORDS
A140 AB67               FDB    LAB67     POINTS TO JUMP TABLE FOR COMMANDS
A142 14                 FCB    $14       20 BASIC SECONDARY COMMANDS
A143 AB1A               FDB    LAB1A     POINTS TO INTRINSIC FUNCTION RESERVED WORDS
A145 AA29               FDB    LAA29     POINTS TO INTRINSIC FUNCTION JUMP TABLE
                  * COPYRIGHT MESSAGES
A147 43 4F 4C 4F  LA147 FCC    'COLOR BASIC 1.'
A155 32                 FCB    BASVER-10+'0
A156 0D                 FCB    CR
A157 28 43 29 20        FCC    '(C) 198'
A15E 32                 FCB    BYEAR
A15F 20 54 41 4E        FCC    ' TANDY'
```

```
A165 00                    FCB   0
A166 4D 49 43 52  LA166    FCC   'MICROSOFT'
A16F 0D 00                 FCB   CR,0

A171 8D  03       LA171    BSR   LA176     GET A CHARACTER FROM CONSOLE IN
A173 84  7F                ANDA  #$7F      MASK OFF BIT 7
A175 39                    RTS

                  * CONSOLE IN
A176 BD  016A     LA176    JSR   RVEC4     HOOK INTO RAM
A179 0F  70                CLR   CINBFL    RESET CONSOLE IN BUFFER FLAG = FULL
A17B 0D  6F       LA17B    TST   DEVNUM    CHECK DEVICE NUMBER
A17D 27  32                BEQ   LA1B1     GO DO CURSOR AND GET A KEY IF SCREEN MODE
A17F 0D  79                TST   CINCTR    TEST CHARACTER COUNTER
A181 26  03                BNE   LA186     NOT EMPTY - READ IN SOME CASSETTE DATA
A183 03  70                COM   CINBFL    SET TO $FF: CONSOLE IN BUFFER EMPTY
A185 39           LA185    RTS
                  *
A186 34  74       LA186    PSHS  B,X,Y,U   SAVE REGISTERS
A188 9E  7A                LDX   CINPTR    PICK UP BUFFER POINTER
A18A A6  80                LDA   ,X+       GET NEXT CHAR
A18C 34  02                PSHS  A         SAVE CHAR ON STACK
A18E 9F  7A                STX   CINPTR    SAVE NEW BUFFER POINTER
A190 0A  79       LA190    DEC   CINCTR    DECR CHAR COUNT
A192 26  03                BNE   LA197     RETURN IF BUFFER NOT EMPTY
A194 BD  A635              JSR   LA635     GO READ TAPE
A197 35  F6       LA197    PULS  A,B,X,Y,U,PC RESTORE REGISTERS
                  *
A199 0A  94       LA199    DEC   BLKCNT    CURSOR BLINK DELAY
A19B 26  0E                BNE   LA1AB     NOT TIME FOR NEW COLOR
A19D C6  0B                LDB   #11       *
A19F D7  94                STB   BLKCNT    *RESET DELAY COUNTER
A1A1 9E  BB                LDX   CURPOS    GET CURSOR POSITION
A1A3 A6  84                LDA   ,X        GET CURRENT CURSOR CHAR
A1A5 8B  10                ADDA  #$10      BUMP TO NEXT COLOR
A1A7 8A  8F                ORA   #$8F      MAKE SURE IT'S A SOLID GRAPHICS BLOCK
A1A9 A7  84                STA   ,X        STORE TO SCREEN
A1AB 8E  045E     LA1AB    LDX   #DEBDEL   CURSOR BLINK DELAY
A1AE 7E  A7D3     LA1AE    JMP   LA7D3     DELAY WHILE X DECREMENTS TO ZERO

                  *     BLINK CURSOR WHILE WAITING FOR A KEYSTROKE
A1B1 34  14       LA1B1    PSHS  B,X       SAVE REGISTERS
A1B3 8D  E4       LA1B3    BSR   LA199     GO DO CURSOR
A1B5 8D  14                BSR   KEYIN     GO CHECK KEYBOARD
A1B7 27  FA                BEQ   LA1B3     LOOP IF NO KEY DOWN
A1B9 C6  60                LDB   #$60      BLANK
A1BB E7  9F 0088           STB   [CURPOS]  BLANK CURRENT CURSOR CHAR ON SCREEN
A1BF 35  94       LA1BF    PULS  B,X,PC
                  *
                  * THIS ROUTINE GETS A KEYSTROKE FROM THE KEYBOARD IF A KEY
                  * IS DOWN.  IT RETURNS ZERO TRUE IF THERE WAS NO KEY DOWN.
                  *
A1C1 7F  FF02     LA1C1    CLR   PIA0+2    CLEAR COLUMN STROBE
A1C4 B6  FF00              LDA   PIA0      READ KEY ROWS
A1C7 43                    COMA           COMPLEMENT ROW DATA
A1C8 48                    ASLA           SHIFT OFF JOYSTICK DATA
A1C9 27  79                BEQ   LA243     RETURN IF NO KEYS OR BUTTONS DOWN
A1CB 34  54       KEYIN    PSHS  U,X,B     SAVE REGISTERS
A1CD CE  FF00              LDU   #PIA0     POINT U TO PIA0
A1D0 8E  0152              LDX   #KEYBUF   POINT X TO KEYBOARD MEMORY BUFFER
A1D3 4F                    CLRA           * CLEAR CARRY FLAG, SET COLUMN COUNTER (ACCA)
```

```
A1D4 4A              DECA            ; TO $FF
A1D5 34  12          PSHS   X,A      SAVE COLUMN CTR & 2 BLANK (X REG) ON STACK
A1D7 A7  42          STA    2,U      INITIALIZE COLUMN STROBE TO $FF
A1D9 69  42    LA1D5 ROL    2,U      ; ROTATE COLUMN STROBE DATA LEFT 1 BIT, CARRY
A1DB 24  43          BCC    LA220    ; INTO BIT 0 - BRANCH IF 8 SHIFTS DONE
A1DD 6C  E4          INC    ,S       INCREMENT COLUMN COUNTER
A1DF 8D  59          BSR    LA239    READ KEYBOARD ROW DATA
A1E1 A7  61          STA    1,S      TEMP STORE KEY DATA
A1E3 A8  84          EORA   ,X       SET ANY BIT WHERE A KEY HAS MOVED
A1E5 A4  84          ANDA   ,X       ACCA=0 IF NO NEW KEY DOWN, <70 IF KEY WAS RELEASED
A1E7 E6  61          LDB    1,S      GET NEW KEY DATA
A1E9 E7  80          STB    ,X+      STORE IT IN KEY MEMORY
A1EB 4D              TSTA            WAS A NEW KEY DOWN?
A1EC 27  EB          BEQ    LA1D5    NO-CHECK ANOTHER COLUMN
A1EE E6  42          LDB    2,U      ; GET COLUMN STROBE DATA AND
A1F0 E7  62          STB    2,S      ; TEMP STORE IT ON THE STACK

                     ;; THIS ROUTINE CONVERTS THE KEY DEPRESSION INTO A NUMBER
                     ;; FROM 0-50 IN ACCB CORRESPONDING TO THE KEY THAT WAS DOWN
A1F2 C6  F8          LDB    #$F8     TO MAKE SURE ACCB=0 AFTER FIRST ADDB #8
A1F4 CB  08    LA1F1 ADDB   #8       ADD 8 FOR EACH ROW OF KEYBOARD
A1F6 44              LSRA            ACCA HAS THE ROW # OF THIS KEY - ADD 8 FOR EACH ROW
A1F7 24  FB          BCC    LA1F1    GO ON UNTIL A ZERO APPEARS IN THE CARRY FLAG
A1F9 EB  E4          ADDB   ,S       ADD IN THE COLUMN #
                     ;; NOW CONVERT THE VALUE IN ACCB INTO ASCII
A1FB 27  48          BEQ    LA244    THE 'AT SIGN' KEY WAS DOWN
A1FD C1  1A          CMPB   #26      WAS IT A LETTER?
A1FF 22  46          BHI    LA246    NO
A201 CA  40          ORB    #$40     YES, CONVERT TO UPPER CASE ASCII
A203 8D  29          BSR    LA22E    CHECK FOR THE SHIFT KEY
A205 BA  011A        ORA    CASFLG   ; 'OR' IN THE CASE FLAG & BRANCH IF IN UPPER
A208 26  02          BNE    LA20B    ; CASE MODE OR SHIFT KEY DOWN
A20A CA  20          ORB    #$20     CONVERT TO LOWER CASE
A20C E7  E4    LA20B STB    ,S       TEMP STORE ASCII VALUE
A20E BE  011B        LDX    DEBVAL   GET KEYBOARD DEBOUNCE
A211 8D  9B          BSR    LA1AE
A213 C6  FF          LDB    #$FF     ; SET COLUMN STROBE TO ALL ONES (NO
A215 8D  21          BSR    LA237    ; STROBE) AND READ KEYBOARD
A217 4C              INCA            = INCR ROW DATA, ACCA NOW 0 IF NO JOYSTICK
A218 26  06          BNE    LA220    = BUTTON DOWN. BRANCH IF JOYSTICK BUTTON DOWN
A21A E6  62    LA21A LDB    2,S      GET COLUMN STROBE DATA
A21C 8D  1A          BSR    LA237    READ A KEY
A21E A1  61          CMPA   1,S      IS IT THE SAME KEY AS BEFORE DEBOUNCE?
A220 35  12    LA220 PULS   X,A      REMOVE TEMP SLOTS FROM THE STACK AND RECOVER
                     ;                THE ASCII VALUE OF THE KEY
A222 26  07          BNE    LA22B    NOT THE SAME KEY OR JOYSTICK BUTTON
A224 81  12          CMPA   #$12     IS SHIFT ZERO DOWN?
A226 26  04          BNE    LA22C    NO
A228 73  011A        COM    CASFLG   YES, TOGGLE UPPER CASE/LOWER CASE FLAG
A22B 4F        LA22B CLRA            SET ZERO FLAG TO INDICATE NO NEW KEY DOWN
A22C 35  D4    LA22C PULS   X,U,B,PC RESTORE REGISTERS

                     ;;; TEST FOR THE SHIFT KEY
A22E 96  7F    LA22E LDA    #$7F     COLUMN STROBE
A230 A7  42          STA    2,U      STORE TO PIA
A232 A6  C4          LDA    ,U       READ KEY DATA
A234 43              COMA            ;
A235 84  40          ANDA   #$40     ; SET BIT 6 IF SHIFT KEY DOWN
A237 39              RTS             RETURN

                     ;;; READ THE KEYBOARD
```

```
A238 E7  42      LA237  STB   2,U      SAVE NEW COLUMN STROBE VALUE
A23A A6  C4      LA239  LDA   ,U       READ PIA0, PORT A TO SEE IF KEY IS DOWN
                 *                     A BIT WILL BE ZERO IF ONE IS
A23C 8A  80             ORA   #$80     MASK OFF THE JOYSTICK COMPARATOR INPUT
A23E 6D  42             TST   2,U      ARE WE STROBING COLUMN 7?
A240 2B  02             BMI   LA243    NO
A242 8A  C0             ORA   #$C0     YES, FORCE ROW 6 TO BE HIGH - THIS WILL CAUSE
                 *                     THE SHIFT KEY TO BE IGNORED
A244 39          LA243  RTS            RETURN

A245 C6  33      LA244  LDB   #51       CODE FOR 'AT SIGN'
A247 8E  A238    LA246  LDX   #CONTAB-$36 POINT X TO CONTROL CODE TABLE
A24A C1  21             CMPB  #33       KEY NUMBER <33?
A24C 25  16             BLO   LA263     YES (ARROW KEYS, SPACE BAR, ZERO)
A24E 8E  A21A           LDX   #CONTAB-$54 POINT X TO MIDDLE OF CONTROL TABLE
A251 C1  30             CMPB  #48       KEY NUMBER >48?
A253 24  0F             BHS   LA263     YES (ENTER,CLEAR,BREAK,AT SIGN)
A255 BD  D7             BSR   LA22E     CHECK SHIFT KEY (ACCA WILL CONTAIN STATUS)
A257 C1  2B             CMPB  #43       IS KEY A NUMBER, COLON OR SEMICOLON?
A259 23  02             BLS   LA25C     YES
A25B 88  40             EORA  #$40      TOGGLE BIT 6 OF ACCA WHICH CONTAINS THE SHIFT DATA
                 *                     ONLY FOR SLASH,HYPHEN,PERIOD,COMMA
A25D 4D          LA25C  TSTA            SHIFT KEY DOWN?
A25E 26  AC             BNE   LA20B     YES
A260 CB  10             ADDB  #$10      NO. ADD IN ASCII OFFSET CORRECTION
A262 20  A8             BRA   LA20B     GO CHECK FOR DEBOUNCE
A264 5B          LA263  ASLB           MULT ACCB BY 2 - THERE ARE 2 ENTRIES IN CONTROL
                 *                     TABLE FOR EACH KEY - ONE SHIFTED, ONE NOT
A265 BD  C7             BSR   LA22E     CHECK SHIFT KEY
A267 27  01             BEQ   LA269     NOT DOWN
A269 5C                 INCB           ADD ONE TO GET THE SHIFTED VALUE
A26A E6  85      LA269  LDB   B,X       GET ASCII CODE FROM CONTROL TABLE
A26C 20  9E             BRA   LA20B     GO CHECK DEBOUNCE



                 *
                 *** CONTROL TABLE    UNSHIFTED, SHIFTED VALUES
A26E 5E 5F       CONTAB FCB   $5E,$5F   UP ARROW
A270 0A 5B             FCB   $A,$5B    DOWN ARROW
A272 08 15             FCB   $8,$15    RIGHT ARROW
A274 09 5D             FCB   $9,$5D    LEFT ARROW
A276 20 20             FCB   $20,$20   SPACE BAR
A278 30 12             FCB   $30,$12   ZERO

A27A 0D 0D             FCB   $D,$D     ENTER
A27C 0C 5C             FCB   $C,$5C    CLEAR
A27E 03 03             FCB   $3,$3     BREAK
A280 40 13             FCB   $40,$13   AT SIGN

                 * CONSOLE OUT
A282 BD  0167    PUTCHR JSR   RVEC3     HOOK INTO RAM
A285 34  04             PSHS  B         SAVE ACCB
A287 D6  6F             LDB   DEVNUM    GET DEVICE NUMBER
A289 5C                 INCB           SET FLAGS
A28A 35  04             PULS  B         RESTORE ACCB
A28C 2B  31             BMI   LA2BF     SEND TO LINE PRINTER
A28E 26  7A             BNE   LA30A     SEND TO SCREEN
                 * SEND TO CASSETTE
A290 34  16             PSHS  A,B,X     RESTORE REGISTERS
A292 D6  78             LDB   FILSTA    GET FILE STATUS
```

```
A294 5A                 DECB            INPUT FILE?
A295 27  0F             BEQ     LA2A6   YES
A297 D6  79             LDB     CINCTR  TEMP CHAR CTR
A299 5C                 INCB            IS THE BUFFER FULL
A29A 26  02             BNE     LA29E   NO
A29C 8D  0A             BSR     LA2A8   YES. WRITE DATA BLOCK TO TAPE
A29E 9E  7A     LA29E   LDX     CINPTR  GET BUFFER POINTER
A2A0 A7  80             STA     ,X+     PUT CHAR IN CASSETTE BUFFER
A2A2 9F  7A             STX     CINPTR  STORE NEW BUFFER POINTER
A2A4 0C  79             INC     CINCTR  INCR BYTE COUNT
A2A6 35  96     LA2A6   PULS    A,B,X,PC

                *WRITE A BLOCK OF DATA TO TAPE
A2A8 C6  01     LA2A8   LDB     #1      DATA BLOCK TYPE - NOT A HEADER BLOCK
A2AA D7  7C     LA2AA   STB     BLKTYP  BLOCK #
A2AC 8E  01DA           LDX     #CASBUF CASSETTE BUFFER
A2AF 9F  7E             STX     CBUFAD  STARTING ADDRESS
A2B1 D6  79             LDB     CINCTR  GET NUMBER OF BYTES
A2B3 D7  7D             STB     BLKLEN  BYTE COUNT
A2B5 34  62             PSHS    A,Y,U   SAVE REGISTERS
A2B7 BD  A7E5           JSR     LA7E5   WRITE A BLOCK ON TAPE
A2BA 35  62             PULS    A,Y,U   RESTORE REGISTERS
A2BC 7E  A650           JMP     LA650   RESET BUFFER POINTERS
                *
                *  SOFTWARE UART TO LINE PRINTER
A2BF 34  17     LA2BF   PSHS    CC,A,B,X SAVE REGISTERS AND INTERRUPT STATUS
A2C1 1A  50             ORCC    #$50    DISABLE IRQ,FIRQ
                *
A2C3 F6  FF22   LA2C3   LDB     PIA1+2  GET RS 232 STATUS
A2C6 54                 LSRB            SHIFT READY BIT TO CARRY
A2C7 25  FA             BCS     LA2C3   LOOP IF NOT READY
A2C9 8D  30             BSR     LA2FB   SET OUTPUT TO MARKING
A2CB 5F                 CLRB            *
A2CC 8D  2F             BSR     LA2FD   * TRANSMIT ONE START BIT
A2CE C6  08             LDB     #8      SEND 8 BITS
A2D0 34  04     LA2CA   PSHS    B       SAVE BIT COUNTER
A2D2 5F                 CLRB            CLEAR DA IMAGE - ZEROES TO DA WHEN SENDING RS 232 DATA
A2D3 44                 LSRA            ROTATE NEXT BIT OF OUPUT CHARACTER TO CARRY FLAG
A2D4 59                 ROLB            * ROTATE CARRY FLAG INTO BIT 1
A2D5 58                 ASLB            * ALL OTHER BITS SET TO ZERO
A2D6 8D  25             BSR     LA2FD   TRANSMIT DATA BYTE
A2D8 35  04             PULS    B       GET BIT COUNTER
A2DA 5A                 DECB            SENT ALL 8 BITS?
A2DB 26  F3             BNE     LA2CA   NO
A2DD 8D  1C             BSR     LA2FB   SEND STOP BIT (ACCB=0)
A2DF 35  03             PULS    CC,A    RESTORE OUTPUT CHARACTER AND INTERRUPT STATUS
A2E1 81  0D             CMPA    #CR     IS IT CARRIAGE RETURN?
A2E3 27  08             BEQ     LA2E7   YES
A2E5 0C  9C             INC     LPTPOS  INCREMENT CHARACTER COUNTER
A2E7 D6  9C             LDB     LPTPOS  CHECK FOR END OF LINE PRINTER LINE
A2E9 D1  9B             CMPB    LPTWID  AT END OF LINE PRINTER LINE?
A2EB 25  06             BLO     LA2ED   NO
A2ED 0F  9C     LA2E7   CLR     LPTPOS  RESET CHARACTER COUNTER
A2EF 8D  14             BSR     LA305   *
A2F1 8D  12             BSR     LA305   *DELAY FOR CARRIAGE RETURN
A2F3 F6  FF22   LA2ED   LDB     PIA1+2  WAIT FOR HANDSHAKE
A2F6 54                 LSRB            CHECK FOR RS232 STATUS
A2F7 25  FA             BCS     LA2ED   NOT READY YET
A2F9 35  94             PULS    B,X,PC  RESTORE REGISTERS
```

```
                        *
                        *
A2FB C6   02      LA2FB LDB   #2        SET RS232 OUTPUT HIGH (MARKING)
A2FD F7   FF20    LA2FD STB   DA        STORE TO THE D/A CONVERTER REGISTER
A300 8D   00            BSR   LA302     GO WAIT A WHILE
A302 9E   95      LA302 LDX   LPTBTD    GET BAUD RATE
A304 8C                 FCB   SKP2      SKIP NEXT TWO BYTES
A305 9E   97      LA305 LDX   LPTLND    PRINTER CARRIAGE RETURN DELAY
A307 7E   A7D3          JMP   LA7D3     DELAY ON DECREMENTING X
                        *
                        * PUT A CHARACTER ON THE SCREEN
A30A 34   16      LA30A PSHS  A,B,X     SAVE REGISTERS
A30C 9E   88            LDX   CURPOS    POINT X TO CURRENT CHARACTER POSITION
A30E 81   08            CMPA  #BS       IS IT BACKSPACE?
A310 26   0B            BNE   LA31D     NO
A312 8C   0400          CMPX  #VIDRAM   AT TOP OF SCREEN?
A315 27   46            BEQ   LA35D     YES - DO NOT ALLOW BACKSPACE
A317 86   60            LDA   #$60      BLANK
A319 A7   82            STA   ,-X       PUT IN PREVIOUS POSITION
A31B 20   27            BRA   LA344     SAVE NEW CURPOS
A31D 81   0D      LA31D CMPA  #CR       ENTER KEY?
A31F 26   0E            BNE   LA32F     BRANCH IF NOT
A321 9E   88            LDX   CURPOS    GET CURRENT CHAR POSITION
A323 86   60      LA323 LDA   #$60      BLANK
A325 A7   80            STA   ,X+       PUT IT ON SCREEN
A327 1F   10            TFR   X,D       *
A329 C5   1F            BITB  #$1F      *TEST FOR BEGINNING OF NEW LINE
A32B 26   F6            BNE   LA323     PUT OUT BLANKS TILL NEW LINE
A32D 20   15            BRA   LA344     CHECK FOR SCROLLING
A32F 81   20      LA32F CMPA  #SPACE    *
A331 25   2A            BLO   LA35D     * BRANCH IF CONTROL CHARACTER
A333 4D                 TSTA            SET FLAGS
A334 2B   0C            BMI   LA342     IT IS GRAPHIC CHARACTER
A336 81   40            CMPA  #$40      *
A338 25   06            BLO   LA340     * BRANCH IF NUMBER OR SPECIAL CHARACTER
A33A 81   60            CMPA  #$60      UPPER/LOWER CASE?
A33C 25   04            BLO   LA342     BRANCH IF UPPER CASE ALPHA
A33E 84   DF            ANDA  #$DF      CLEAR BIT 5, FORCE ASCII LOWER CASE TO BE UPPER CASE
A340 88   40      LA340 EORA  #$40      INVERT BIT 6, CHANGE UPPER CASE TO LOWER & VICE VERSA
A342 A7   80      LA342 STA   ,X+       STORE CHARACTER TO SCREEN
A344 9F   88      LA344 STX   CURPOS    SAVE CURRENT CHAR POSITION
A346 8C   05FF          CMPX  #VIDRAM+$1FF END OF SCREEN BUFFER?
A349 23   12            BLS   LA35D     RETURN IF NO NEED TO SCROLL
A34B 8E   0400          LDX   #VIDRAM   TOP OF SCREEN

                        *** SCROLL THE SCREEN
A34E EC   88 20   LA34E LDD   32,X      GET TWO BYTES
A351 ED   81            STD   ,X++      MOVE THEM UP ONE ROW
A353 8C   05E0          CMPX  #VIDRAM+$1E0 AT THE LAST LINE?
A356 25   F6            BLO   LA34E     NO
A358 C6   60            LDB   #$60      BLANK
A35A 8D   A92D          JSR   LA92D     BLANK LAST LINE
A35D 35   96      LA35D PULS  A,B,X,PC  RESTORE REGISTERS
                        *
                        *  SET UP TAB FIELD WIDTH, TAB ZONE, CURRENT POSITION
                        *  AND LINE WIDTH ACCORDING TO THE DEVICE SELECTED
A35F BD   0164    LA35F JSR   RVEC2     HOOK INTO RAM
A362 34   16            PSHS  A,B,X     SAVE REGISTERS
A364 0F   6E            CLR   PRTDEV    RESET PRINT DEVICE NUMBER
A366 96   6F            LDA   DEVNUM    GET DEVICE NUMBER
A368 27   09            BEQ   LA373     BRANCH IF SCREEN
```

```
A36A 4C                 INCA            CHECK FOR CASSETTE
A36B 27  17             BEQ     LA384   BRANCH IF CASSETTE
                    ** END UP HERE IF PRINTER
A36D 9E  99             LDX     LPTCFW  TAB FIELD WIDTH AND TAB ZONE
A36F DC  9B             LDD     LPTWID  PRINTER WIDTH AND POSITION
A371 20  09             BRA     LA37C   SET PRINT PARAMETERS
                    *       SCREEN DISPLAY VALUES
A373 D6  89     LA373   LDB     CURPOS+1 GET CURSOR LOC LS BYTE
A375 C4  1F             ANDB    #$1F    KEEP ONLY COLUMN POSITION
A377 8E  1010           LDX     #$1010  TAB FIELD WIDTH AND LAST TAB ZONE
A37A 86  20             LDA     #32     DISPLAY SCREEN LINE WIDTH
A37C 9F  6A     LA37C   STX     DEVCFW  SAVE TAB FIELD WIDTH AND ZONE
A37E D7  6C             STB     DEVPOS  SAVE PRINT POSITION
A380 97  6D             STA     DEVWID  SAVE PRINT WIDTH
A382 35  96             PULS    A,B,X,PC RESTORE REGISTERS
A384 03  6E     LA384   COM     PRTDEV  SET TO $FF FOR CASSETTE
A386 8E  0100           LDX     #$0100  * TAB FIELD WIDTH = 1; ALL OTHER
A389 4F                 CLRA            * PARAMETERS = 0
A38A 5F                 CLRB            *
A38B 20  EF             BRA     LA37C   SET PRINT PARAMETERS
                    *
                    *   THIS IS THE ROUTINE THAT GETS AN INPUT LINE FOR BASIC
                    *   EXIT WITH BREAK KEY: CARRY = 1
                    *   EXIT WITH ENTER KEY: CARRY = 0
A38D BD  A928   LA38D   JSR     LA928   CLEAR SCREEN
A390 9D  0182   LA390   JSR     RVEC12  HOOK INTO RAM
A393 0F  87             CLR     IKEYIM  RESET BREAK CHECK KEY TEMP KEY STORAGE
A395 8E  02DD           LDX     #LINBUF+1 INPUT LINE BUFFER
A398 C6  01             LDB     #1      ACCB = CHAR COUNTER: SET TO 1 TO ALLOW A
                    *                   BACKSPACE AS FIRST CHARACTER
A39A BD  A171   LA39A   JSR     LA171   GO GET A CHARACTER FROM CONSOLE IN
A39D 0D  70             TST     CINBFL  GET CONSOLE IN BUFFER FLAG
A39F 26  2B             BNE     LA3CC   BRANCH IF NO MORE CHARACTERS IN INPUT FILE
A3A1 0D  6F             TST     DEVNUM  CHECK DEVICE NUMBER
A3A3 26  23             BNE     LA3C8   BRANCH IF NOT SCREEN
A3A5 81  0C             CMPA    #FORMF  FORM FEED
A3A7 27  E4             BEQ     LA38D   YES - CLEAR SCREEN
A3A9 81  08             CMPA    #BS     BACKSPACE
A3AB 26  07             BNE     LA3B4   NO
A3AD 5A                 DECB            YES - DECREMENT CHAR COUNTER
A3AE 27  E0             BEQ     LA390   BRANCH IF BACK AT START OF LINE AGAIN
A3B0 30  1F             LEAX    -1,X    DECREMENT BUFFER POINTER
A3B2 20  34             BRA     LA3E8   ECHO CHAR TO SCREEN
A3B4 81  15     LA3B4   CMPA    #$15    SHIFT RIGHT ARROW?
A3B6 26  0A             BNE     LA3C2   NO
                    **YES, RESET BUFFER TO BEGINNING AND ERASE CURRENT LINE
A3B8 5A         LA3B8   DECB            DEC CHAR CTR
A3B9 27  D5             BEQ     LA390   GO BACK TO START IF CHAR CTR = 0
A3BB 86  08             LDA     #BS     BACKSPACE?
A3BD BD  A282           JSR     PUTCHR  SEND TO CONSOLE OUT (SCREEN)
A3C0 20  F6             BRA     LA3B8   KEEP GOING
A3C2 81  03     LA3C2   CMPA    #3      BREAK KEY?
A3C4 1A  01             ORCC    #1      SET CARRY FLAG
A3C6 27  05             BEQ     LA3CD   BRANCH IF BREAK KEY DOWN
A3C8 81  0D     LA3C8   CMPA    #CR     ENTER KEY?
A3CA 26  0D             BNE     LA3D9   NO
A3CC 4F         LA3CC   CLRA            CLEAR CARRY FLAG IF ENTER KEY - END LINE ENTRY
A3CD 34  01     LA3CD   PSHS    CC      SAVE CARRY FLAG
A3CF BD  B958           JSR     LB958   SEND CR TO SCREEN
```

```
A3D2 6F  84          CLR   ,X        MAKE LAST BYTE IN INPUT BUFFER = 0
A3D4 8E  02DC        LDX   #LINBUF   RESET INPUT BUFFER POINTER
A3D7 35  81          PULS  CC,PC     RESTORE CARRY FLAG

                     *    INSERT A CHARACTER INTO THE BASIC LINE INPUT BUFFER
A3D9 81  20     LA3D9  CMPA  #$20     IS IT CONTROL CHAR?
A3DB 25  BD          BLO   LA39A     BRANCH IF CONTROL CHARACTER
A3DD 81  7B          CMPA  #'z+1     *
A3DF 24  B9          BHS   LA39A     *IGNORE IF > LOWER CASE Z
A3E1 C1  FA          CMPB  #LBUFMX   HAVE 250 OR MORE CHARACTERS BEEN ENTERED?
A3E3 24  B5          BHS   LA39A     YES, IGNORE ANY MORE
A3E5 A7  80          STA   ,X+       PUT IT IN INPUT BUFFER
A3E7 5C              INCB            INCREMENT CHARACTER COUNTER
A3E8 BD  A282   LA3E8  JSR   PUTCHR   ECHO IT TO SCREEN
A3EB 20  AD          BRA   LA39A     GO GET SOME MORE

                     *    INPUT FILE HANDLER
A3ED BD  016D   LA3ED  JSR   RVEC5    HOOK INTO RAM
A3F0 96  6F          LDA   DEVNUM    DEVICE NUMBER
A3F2 27  21          BEQ   LA415     BRANCH IF SCREEN
A3F4 4C              INCA            *
A3F5 26  0C          BNE   LA403     *BRANCH IF NOT CASSETTE (BAD FILE MODE)
A3F7 96  78          LDA   FILSTA    GET FILE STATUS
A3F9 26  05          BNE   LA400     FILE IS OPEN
A3FB C6  2C     LA3FB  LDB   #22*2    'FILE NOT OPEN' ERROR
A3FD 7E  AC46        JMP   LAC46     JUMP TO ERROR SERVICING ROUTINE
A400 4A         LA400  DECA          *
A401 27  12          BEQ   LA415     *FILE IS IN INPUT MODE, RETURN
A403 7E  A616   LA403  JMP   LA616    'BAD FILE MODE' ERROR
                     *
                     *    PRINT # HANDLER
A406 BD  0170   LA406  JSR   RVEC6    HOOK INTO RAM
A409 96  6F          LDA   DEVNUM    GET DEVICE NUMBER
A40B 4C              INCA            *
A40C 26  07          BNE   LA415     *BRANCH IF NOT TAPE
A40E 96  78          LDA   FILSTA    GET FILE STATUS
A410 27  E9          BEQ   LA3FB     'FILE NOT OPEN' ERROR
A412 4A              DECA            *
A413 27  EE          BEQ   LA403     *'BAD FILE MODE' - FILE IN INPUT MODE
A415 39         LA415  RTS
                     *
                     *            CLOSE
A416 27  0E     CLOSE  BEQ   LA426     BRANCH IF NO NAME SPECIFIED
A418 BD  A5A5        JSR   LA5A5     CHECK DEVICE NUMBER
A41B 8D  10     LA41B  BSR   LA42D     GO CLOSE A FILE
A41D 9D  A5          JSR   GETCCH    GET CURRENT BASIC CHARACTER
A41F 27  2A          BEQ   LA44B     RETURN IF NO MORE FILES
A421 BD  A5A2        JSR   LA5A2     CHECK SYNTAX AND DEVICE NUMBER
A424 20  F5          BRA   LA41B     KEEP CLOSING FILES

                     *    CLOSE ALL FILES HANDLER
A426 BD  0173   LA426  JSR   RVEC7    HOOK INTO RAM
A429 86  FF          LDA   #-1       CASSETTE DEVICE NUMBER
A42B 97  6F          STA   DEVNUM    SET DEVICE NUMBER
                     *    CLOSE FILE HANDLER
A42D BD  0176   LA42D  JSR   RVEC8    HOOK INTO RAM
A430 96  6F          LDA   DEVNUM    GET DEVICE NUMBER
A432 0F  6F          CLR   DEVNUM    SET TO SCREEN
A434 4C              INCA            *
A435 26  14          BNE   LA44B     *BRANCH IF WAS NOT CASSETTE
A437 96  78          LDA   FILSTA    GET FILE STATUS
```

```
A439 81 02          CMPA  #2       IS IT OUTPUT MODE
A43B 26 0C          BNE   LA449    NO
A43D 96 79          LDA   CINCTR   GET CHARACTER BUFFER CTR
A43F 27 03          BEQ   LA444    WRITE END OF PROG BLOCK IF BUFFER EMPTY
A441 BD A2A8        JSR   LA2A8    WRITE A BLOCK TO TAPE
A444 C6 FF    LA444 LDB   #$FF     END OF FILE TYPE BLOCK #
A446 BD A2AA        JSR   LA2AA    WRITE END OF FILE TYPE BLOCK
A449 0F 78    LA449 CLR   FILSTA   CASSETTE FILE STATUS = CLOSED
A44B 39       LA44B RTS
                    *
                    *        CSAVE
A44C BD A578  CSAVE JSR   LA578    GO SCAN OFF NAME
A44F 9D A5          JSR   GETCCH   GET CURRENT CHARACTER IN THE BASIC LINE
A451 27 16          BEQ   LA469    BRANCH IF NONE
A453 BD B26D        JSR   LB26D    SYNTAX ERROR IF NOT ','
A456 C6 41          LDB   #'A      IS THIS AN ASCII SAVE?
A458 BD B26F        JSR   LB26F    SYNTAX ERROR IF NOT 'A'
A45B 26 EE          BNE   LA44B    RETURN IF NOT END OF LINE
A45D 4F             CLRA           FILE TYPE = 0
A45E BD A65C        JSR   LA65C    WRITE OUT HEADER BLOCK
A461 86 FF          LDA   #-1      CASSETTE CODE
A463 97 6F          STA   DEVNUM   SET DEVICE NUMBER TO CASSETTE
A465 4F             CLRA           CLEAR CARRY - FORCE LIST TO BEGIN AT PROGRAM START
A466 7E B764        JMP   LIST     GO DO A 'LIST' TO CASSETTE
                    *
                    * NON-ASCII CSAVE
A469 4F       LA469 CLRA           FILE TYPE = 0
A46A 9E 8A          LDX   ZERO     ZERO OUT ASCII FLAG AND FILE MODE
A46C BD A65F        JSR   LA65F    WRITE HEADER BLOCK
A46F 0F 78          CLR   FILSTA   CLOSE FILES
A471 0C 7C          INC   BLKTYP   INCREMENT BLOCK #
A473 BD A7D8        JSR   WRLDR    WRITE 55'S TO CASSETTE
A476 9E 19          LDX   TXTTAB   ADDRESS OF PROGRAM START
A478 9F 7E    LA478 STX   CBUFAD   STORE CURRENT BLOCK START ADDR
A47A 86 FF          LDA   #255     255 BYTE BLOCKS
A47C 97 7D          STA   BLKLEN   BLOCK SIZE
A47E DC 1B          LDD   VARTAB   END OF PROGRAM
A480 93 7E          SUBD  CBUFAD   CURRENT BLOCK STARTING ADDR
A482 27 0D          BEQ   LA491    BRANCH IF IT CAME OUT EXACT
A484 1083 00FF      CMPD  #255     MORE THAN 255 BYTES LEFT?
A488 24 02          BHS   LA48C    YES
A48A D7 7D          STB   BLKLEN   USE ACTUAL BLOCK SIZE IF LESS THAN 255
A48C BD A7F4  LA48C JSR   SNDBLK   WRITE BLOCK TO CASSETTE
A48F 20 E7          BRA   LA478    DO ANOTHER BLOCK
A491 00 7C    LA491 NEG   BLKTYP   MAKE BLOCK NUMBER NEGATIVE (EOF BLOCK)
A493 0F 7D          CLR   BLKLEN   ZERO BLOCK SIZE
A495 7E A7E7        JMP   LA7E7    WRITE A BLOCK, TURN OFF MOTOR
                    *
                    *   CLOAD
A498 0F 78    CLOAD CLR   FILSTA   CLOSE FILES
A49A 81 4D          CMPA  #'M      IS IT CLOADM?
A49C 27 60          BEQ   LA4FE    BRANCH IF SO
A49E 32 62          LEAS  2,S      GET RID OF THE RETURN
A4A0 BD A5C5        JSR   LA5C5    GO GET FILE NAME
A4A3 BD A648        JSR   LA648    SEARCH FOR FILE
A4A6 7D 01E4        TST   CASBUF+10 GET FILE MODE (NON-ZERO=DATA OR ASCII)
A4A9 27 1D          BEQ   LA4C8    ZERO = CRUNCHED BASIC OR MACHINE LANG
A4AB B6 01E3        LDA   CASBUF+9 GET ASCII FLAG
A4AE 27 1D          BEQ   LA4CD    'BAD FILE MODE' 0 = CRUNCHED OR MACH LANG
A4B0 BD AD19        JSR   LAD19    DO A 'NEW'
A4B3 86 FF          LDA   #-1      TAPE DEVICE NUMBER
A4B5 97 6F          STA   DEVNUM   SET DEVICE NUMBER TO TAPE
```

```
A4B7 0C   78          INC   FILSTA   FILE TYPE = INPUT
A4B9 BD   A635        JSR   LA635    GO LOAD ASCII RECORD
A4BC 7E   AC7C        JMP   LAC7C    GO LOAD AND CRUNCH INPUT
                *
                *  COME HERE FROM BASIC'S DIRECT LOOP IF CONSOLE
                *  IN BUFFER EMPTY
A4BF BD   0185  LA4BF JSR   RVEC13   HOOK INTO RAM
A4C2 BD   A42D        JSR   LA42D    CLOSE FILE
A4C5 7E   AC73        JMP   LAC73    GO TO BASIC'S DIRECT LOOP

                * CLOAD A CRUNCHED BASIC FILE
A4C8 B6   01E2  LA4C8 LDA   CASBUF+8 FILE TYPE
A4CB 27   03          BEQ   LA4D0    ZERO IS CSAVE TYPE
A4CD 7E   A616  LA4CD JMP   LA616    'BAD FILE MODE' IF NOT BASIC FILE
A4D0 BD   AD19  LA4D0 JSR   LAD19    DO A 'NEW'
A4D3 BD   A77C        JSR   CASON    TURN ON TAPE, START READING
A4D6 9E   19          LDX   TXTTAB   GET START OF PROGRAM ADDRESS
A4D8 9F   7E    LA4D8 STX   CBUFAD   STORE IT IN LOAD BUFFER
A4DA DC   7E          LDD   CBUFAD   GET START ADDRESS TO D REG
A4DC 4C               INCA           ADD 256 TO LOAD ADDRESS
A4DD BD   AC37        JSR   LAC37    SEE IF ROOM BELOW STACK FOR ONE BLOCK
A4E0 BD   A70B        JSR   GETBLK   READ A BLOCK
A4E3 26   13          BNE   LA4F8    GOT AN ERROR DURING READ
A4E5 96   7C          LDA   BLKTYP   BLOCK #
A4E7 27   0F          BEQ   LA4F8    I/O ERROR IF HEADER BLOCK TYPE
A4E9 2A   ED          BPL   LA4D8    READ MORE IF BLOCK NUMBER POSITIVE
A4EB 9F   1B          STX   VARTAB   SET END OF PROGRAM ADDRESS
A4ED 8D   4C          BSR   LA53B    TURN OFF TAPE DECK
A4EF 8E   ABEC        LDX   #LABED-1 POINT TO 'OK' MESSAGE
A4F2 BD   B99C        JSR   LB99C    PRINT 'OK' TO CONSOLE OUT
A4F5 7E   ACE9        JMP   LACE9    RESET INPUT POINTER, CLEAR VARIABLES AND
                *                     RETURN TO MAIN LOOP OF BASIC
                *
A4F8 BD   AD19  LA4F8 JSR   LAD19    DO A 'NEW'
A4FB 7E   A619  LA4FB JMP   LA619    'I/O ERROR'
                *
                *   CLOADM
A4FE 9D   9F    LA4FE JSR   GETNCH   GET NEXT CHARACTER IN BASIC LINE
A500 8D   76          BSR   LA578    GO SCAN OFF NAME
A502 BD   A648        JSR   LA648    SEARCH FOR FILE
A505 9E   BA    LA505 LDX   ZERO     STORE ZERO TO X REG, DEFAULT OFFSET VALUE
A507 9D   A5          JSR   GETCCH   CHECK FOR AN OFFSET
A509 27   06          BEQ   LA511    BRANCH IF NO OFFSET
A50B BD   B26D        JSR   LB26D    SYNTAX CHECK FOR COMMA
A50E BD   B73D        JSR   LB73D    GET OFFSET IN X (CALLS SUPERINT)
A511 B6   01E2  LA511 LDA   CASBUF+8 CHECK FILE MODE
A514 81   02          CMPA  #2       IS IT MACHINE LANGUAGE?
A516 26   B5          BNE   LA4CD    'BAD FILE MODE' ERROR IF NOT
A518 FC   01E5        LDD   CASBUF+11 GET XFER ADDR FROM TAPE
A51B 33   8B          LEAU  D,X      ADD OFFSET
A51D DF   9D          STU   EXECJP   STORE TRANSFER ADDR IN EXEC ARGUMENT
A51F 7D   01E4        TST   CASBUF+10 CHECK FILE MODE
A522 26   A9          BNE   LA4CD    'BAD FILE MODE' ERROR
A524 FC   01E7        LDD   CASBUF+13 GET LOAD ADDR FROM TAPE
A527 30   8B          LEAX  D,X      ADD OFFSET
A529 9F   7E          STX   CBUFAD   STORE IN BUFFER START ADDRESS POINTER
A52B BD   A77C        JSR   CASON    START UP TAPE
A52E BD   A70B  LA52E JSR   GETBLK   READ A BLOCK
A531 26   C8          BNE   LA4FB    BRANCH IF I/O ERROR
A533 9F   7E          STX   CBUFAD   STORE NEW START ADDR (ONE BLOCK HIGHER)
A535 0D   7C          TST   BLKTYP   CHECK BLOCK #
```

```
A537 27   C2          BEQ    LA4FB     BRANCH IF I/O ERROR (HEADER BLOCK)
A539 2A   F3          BPL    LA52E     GO READ SOME MORE
A53B 7E   A7E9  LA53B JMP    LA7E9     GO TURN OFF TAPE DECK
                *
                *             EXEC
A53E 27   05    EXEC  BEQ    LA545     BRANCH IF NO ARGUMENT
A540 BD   B73D        JSR    LB73D     EVALUATE ARGUMENT - ARGUMENT RETURNED IN X
A543 9F   9D          STX    EXECJP    STORE X TO EXEC JUMP ADDRESS
A545 6E   9F 009D LA545 JMP    [EXECJP]  GO DO IT

                *    BREAK CHECK
A549 BD   017F  LA549 JSR    RVEC11    HOOK INTO RAM
A54C 96   6F          LDA    DEVNUM    GET DEVICE NUMBER
A54E 4C               INCA             CHECK FOR TAPE
A54F 27   50          BEQ    LA5A1     RETURN IF TAPE
A551 7E   ADEB        JMP    LADEB     GO DO BREAK KEY CHECK
                *
                * THIS ROUTINE EVALUATES AN ARGUMENT
                * AND MAKES SURE IT IS WITHIN LIMITS OF VIDEO DISPLAY RAM
A554 BD   B3E4  LA554 JSR    LB3E4     EVALUATE EXPRESSION AND RETURN VALUE IN ACCD
A557 83   01FF        SUBD   #511      ONLY 512 VIDEO DISPLAY LOCATIONS
A55A 1022 0EEC        LBHI   LB44A     BRANCH IF > 511 TO 'ILLEGAL FUNCTION CALL'
A55E C3   0400        ADDD   #VIDRAM   ADD BACK IN OFFSET + START OF VIDEO RAM
A561 DD   88          STD    CURPOS    PUT THE CURSOR THERE
A563 39               RTS
                *
                *             INKEY$
A564 96   87    INKEY LDA    IKEYIM    WAS A KEY DOWN IN THE BREAK CHECK?
A566 26   03          BNE    LA56B     YES
A568 BD   A1CB        JSR    KEYIN     GO GET A KEY
A56B 0F   87    LA56B CLR    IKEYIM    CLEAR INKEY RAM IMAGE
A56D 97   53          STA    FPA0+3    STORE THE KEY IN FPA0
A56F 1026 111C        LBNE   LB68F     CONVERT FPA0+3 TO A STRING
A573 97   56          STA    STRDES    SET LENGTH OF STRING = 0
A575 7E   B69B        JMP    LB69B     PUT A NULL STRING ONTO THE STRING STACK

                *    STRIP A FILENAME OFF OF THE BASIC INPUT LINE
A578 8E   01D1  LA57B LDX    #CFNBUF   POINT TO FILE NAME BUFFER
A57B 6F   80          CLR    ,X+       CLEAR THE FIRST BYTE - IT WILL CONTAIN THE COUNT
                *                      OF THE NUMBER OF CHARACTERS IN THE NAME
A57D 86   20          LDA    #SPACE    SPACE
A57F A7   80    LA57F STA    ,X+       BLANK FILL 8 CHARS
A581 8C   01DA        CMPX   #CASBUF   DONE?
A584 26   F9          BNE    LA57F     NO
A586 9D   A5          JSR    GETCCH    GET CURRENT INPUT CHAR
A588 27   17          BEQ    LA5A1     RETURN IF NO NAME
A58A BD   B156        JSR    LB156     GET THE FILE NAME  - EVALUATE EXPRESSION
A58D BD   B654        JSR    LB654     POINT X TO START OF NAME (TOP STRING
                *                      ON STRING STACK)
A590 CE   01D1        LDU    #CFNBUF   CASSETTE FILE NAME BUFFER
A593 E7   C0          STB    ,U+       STORE THE NUMBER OF BYTES IN THE NAME
A595 27   0A          BEQ    LA5A1     NULL NAME (BLANK NAME)
A597 8C               FCB    SKP2      SKIP THE NEXT TWO BYTES
A598 C6   08    LA598 LDB    #8        MOVE 8 BYTES
                *
                *    MOVE B BYTES FROM (X) TO (U)
A59A A6   80    LA59A LDA    ,X+       GET BYTE FROM X
A59C A7   C0          STA    ,U+       STORE IT AT U
A59E 5A               DECB             MOVED ALL BYTES?
A59F 26   F9          BNE    LA59A     NO
A5A1 39         LA5A1 RTS
```

```
                        * GET DEVICE NUMBER FROM BASIC LINE - CHECK VALIDITY
A5A2 BD   B26D    LA5A2 JSR   LB26D    CHECK FOR COMMA, SYNTAX ERROR IF NONE
A5A5 81   23      LA5A5 CMPA  #'#      IS NEXT CHARACTER A NUMBER?
A5A7 26   02            BNE   LA5AB    NO
A5A9 9D   9F            JSR   GETNCH   GET NEXT BASIC INPUT CHARACTER
A5AB BD   B141    LA5AB JSR   LB141    EVALUATE EXPRESSION
A5AE BD   B3ED    LA5AE JSR   INTCNV   CONVERT FPA0 TO INTEGER, RETURN VALUE IN ACCD
A5B1 59                 ROLB           MSB OF ACCB TO CARRY
A5B2 89   00            ADCA  #0       ADD MSB OF ACCB TO ACCA
A5B4 26   69            BNE   LA61F    'DEVICE # ERROR' IF ACCA<FF80 OR >007F
A5B6 56                 RORB           RESTORE ACCB
A5B7 D7   6F            STB   DEVNUM   STORE B IN DEVICE NUMBER
A5B9 BD   0161          JSR   RVEC1    HOOK INTO RAM
A5BC 27   06            BEQ   LA5C4    BRANCH IF DEVICE NUMBER SET TO SCREEN
A5BE 2A   5F            BPL   LA61F    'DEVICE #' ERROR IF POSITIVE DEVICE #
A5C0 C1   FE            CMPB  #-2      LOWEST LEGAL DEVICE #
A5C2 2D   5B            BLT   LA61F    'DEVICE # ERROR'
A5C4 39           LA5C4 RTS

                        ** THIS ROUTINE WILL SCAN OFF THE FILE NAME FROM A BASIC LINE
                        ** AND RETURN A SYNTAX ERROR IF THERE ARE ANY CHARACTERS
                        ** FOLLOWING THE END OF THE NAME
A5C5 8D   B1      LA5C5 BSR   LA578    SCAN OFF NAME
A5C7 9D   A5            JSR   GETCCH   GET CURRENT INPUT CHAR FROM BASIC LINE
A5C9 27   F9      LA5C9 BEQ   LA5C4    RETURN IF END OF LINE
A5CB 7E   B277          JMP   LB277    SYNTAX ERROR IF ANY MORE CHARACTERS
                        *
                        *               EOF
A5CE BD   0188    EOF   JSR   RVEC14   HOOK INTO RAM
A5D1 96   6F            LDA   DEVNUM   GET DEVICE NUMBER
A5D3 34   02            PSHS  A        SAVE IT
A5D5 8D   D7            BSR   LA5AE    CHECK DEVICE #
A5D7 BD   A3ED          JSR   LA3ED    CHECK FOR PROPER FILE AND MODE
A5DA 5F                 CLRB           NOT EOF FLAG = 0
A5DB 96   6F            LDA   DEVNUM   TEST DEVICE NUMBER
A5DD 27   05            BEQ   LA5E4    BRANCH IF NOT SET TO DISPLAY
A5DF 0D   79            TST   CINCTR   ANY CHARACTERS LEFT TO SEND?
A5E1 26   01            BNE   LA5E4    YES
A5E3 53                 COMB           NO - EOF: SET FLAG = -1 ($FF)
A5E4 35   02      LA5E4 PULS  A        GET DEVICE NUMBER BACK AGAIN
A5E6 97   6F            STA   DEVNUM   RESTORE IT
A5E8 1D           LA5E8 SEX            CONVERT ACCB TO 2 DIGIT SIGNED INTEGER
A5E9 7E   B4F4          JMP   GIVABF   CONVERT ACCD TO FLOATING POINT
                        *
                        *   SKIPF
A5EC 8D   D7      SKIPF BSR   LA5C5    SCAN OFF THE BASIC FILE NAME
A5EE 8D   58            BSR   LA648    LOOK FOR THAT FILE ON TAPE
A5F0 BD   A6D1          JSR   LA6D1    READ THE FILE
A5F3 26   24            BNE   LA619    'I/O ERROR'
A5F5 39                 RTS
                        *
                        *               OPEN
                        *
A5F6 BD   015E    OPEN  JSR   RVEC0    HOOK INTO RAM
A5F9 BD   B156          JSR   LB156    GET FILE STATUS (INPUT,OUTPUT)
A5FC BD   B6A4          JSR   LB6A4    GET FIRST BYTE OF STATUS STRING TO ACCB
A5FF 34   04            PSHS  B        SAVE IT ON STACK
A601 8D   9F            BSR   LA5A2    CHECK FOR SYNTAX AND GET DEVICE #
A603 BD   B26D          JSR   LB26D    SYNTAX CHECK FOR COMMA, SYNTAX ERROR IF NOT
A606 8D   BD            BSR   LA5C5    GET FILE NAME
A608 96   6F            LDA   DEVNUM   GET DEVICE NUMBER
```

```
A60A 0F   6F          CLR    DEVNUM    SET DEVICE NUMBER TO SCREEN
A60C 35   04          PULS   P         GET STATUS AGAIN
A60E C1   49          CMPB   #'I       IS IT INPUT MODE?
A610 27   12          BEQ    LA624     YES
A612 C1   4F          CMPB   #'O       IS IT OUTPUT MODE?
A614 27   42          BEQ    LA658     YES
                      *
                      *  IF IT ISN'T INPUT OR OUTPUT, BAD FILE MODE
A616 C6   2A    LA616 LDB    #21*2     ERROR # 21 'BAD FILE MODE'
A618 8C               FCB    SKP2      SKIP TWO BYTES
A619 C6   28    LA619 LDB    #20*2     ERROR # 20 'I/O ERROR
A61B 8C               FCB    SKP2      SKIP TWO BYTES
A61C C6   24    LA61C LDB    #18*2     ERROR # 18  'FILE ALREADY OPEN'
A61E 8C               FCB    SKP2      SKIP TWO BYTES
A61F C6   26    LA61F LDB    #19*2     ERROR # 19 'DEVICE NUMBER ERROR'
A621 7E   AC46        JMP    LAC46     JUMP TO ERROR HANDLER
                      *
A624 4C         LA624 INCA              DEVICE NUMBER SET TO TAPE?
A625 2B   EF          BMI    LA616     'BAD FILE MODE' IF DEVNUM = NEG BUT NOT CASSETTE
A627 26   2E          BNE    LA657     RETURN IF DEVNUM WAS SET TO SCREEN OR DISK
                      *** SET TO TAPE
A629 8D   1D          BSR    LA648     GET HEADER BLOCK
A62B B6   01E3        LDA    CASBUF+9  GET ASCII FLAG
A62E B4   01E4        ANDA   CASBUF+10 'AND' IT WITH FILE MODE
A631 27   E3          BEQ    LA616     'BAD FILE MODE' - CRUNCHED FILE OR MACH LANG
A633 0C   78          INC    FILSTA    OPEN FILE FOR INPUT
A635 BD   A701  LA635 JSR    LA701     START TAPE. READ A BLOCK
A638 26   DF          BNE    LA619     'I/O ERROR'
A63A 0D   7C          TST    BLKTYP    CHECK BLOCK NUMBER
A63C 27   DB          BEQ    LA619     'I/O ERROR' IF HEADER BLOCK
A63E 2B   17          BMI    LA657     BRANCH IF THIS IS THE LAST BLOCK
A640 96   7D          LDA    BLKLEN    CHAR COUNT
A642 27   F1          BEQ    LA635     READ ANOTHER BLOCK IF NULL BLOCK
A644 97   79    LA644 STA    CINCTR    STORE IN TEMP CHARACTER COUNTER
A646 20   0A          BRA    LA652     RESET BUFFER POINTER

                      *    SEARCH FOR FILE NAME IN CNMBUF
A648 0D   78    LA648 TST    FILSTA    IS THE FILE OPEN?
A64A 26   D0          BNE    LA61C     YES- 'FILE ALREADY OPEN'
A64C 8D   33          BSR    LA681     SEARCH FOR CORRECT FILE NAME
A64E 26   C9          BNE    LA619     'I/O ERROR'
A650 0F   79    LA650 CLR    CINCTR    CLEAR CHARACTER COUNTER
A652 8E   01DA  LA652 LDX    #CASBUF   CASSETTE INPUT BUFFER ADDRESS
A655 9F   7A          STX    CINPTR    RESET IT
A657 39         LA657 RTS
                      *
                      *** WRITE OUT THE HEADER BLOCK
                      *
                      ** CASBUF        FILE NAME
                      ** CASBUF+8       FILE TYPE
                      ** CASBUF+9       ASCII FLAG
                      ** CASBUF+10      FILE MODE
                      ** CASBUF+11,12   TRANSFER ADDRESS
                      ** CASBUF+13,14   START ADDRESS

                      * ENTER HERE FOR DATA FILES W/DEVICE NUMBER IN ACCA
A658 4C         LA658 INCA              CHECK FOR CASSETTE DEVICE NUMBER
A659 26   FC         BNE    LA657     RETURN IF DEVICE NUMBER WASN'T TAPE
A65B 4C               INCA             MAKE FILE TYPE = 1
                      * ENTER HERE FOR ASCII FILES
A65C 8E   FFFF  LA65C LDX    #$FFFF    SET ASCII FLAG AND MODE = $FF
```

```
A65F 0D  78      LA65F  TST   FILSTA   IS FILE OPEN?
A661 26  B9             BNE   LA61C    YES- 'FILE ALREADY OPEN'
A663 CE  01DA           LDU   #CASBUF  CASSETTE INPUT BUFFER
A666 DF  7E             STU   CBUFAD   STORE IN STARTING ADDRESS
A668 A7  48             STA   8,U      FILE TYPE IN CASBUF+8
A66A AF  49             STX   9,U      ASCII FLAG & MODE IN CASBUF+9, CASBUF+10
                 *        CASBUF    +8     +9     +10
                 *                  TYPE  ASCII  MODE
                 * BASIC CRUNCHED    00     00     00
                 * BASIC ASCII       00     FF     FF
                 * DATA              01     FF     FF
                 * MACHINE LANGUAGE  02     00     00
                 * MACHINE BLK LOAD  02     00     FF

A66C 8E  01D2           LDX   #CFNBUF+1 POINT X TO FILE NAME BUFFER
A66F BD  A598           JSR   LA598     MOVE 8 BYTES FROM (X) TO (U)
A672 0F  7C             CLR   BLKTYP    ZERO BLOCK #
A674 86  0F             LDA   #15       15 BYTES IN THE HEADER BLOCK
A676 97  7D             STA   BLKLEN    CHAR COUNT
A678 BD  A7E5           JSR   LA7E5     GO WRITE ONE BLOCK
A67B 86  02             LDA   #2        OUTPUT FILE
A67D 97  78             STA   FILSTA    STORE IN FILE MODE
A67F 20  CF             BRA   LA650     RESET POINTERS

                 * SEARCH FOR CORRECT CASSETTE FILE NAME
A681 8E  01DA  LA681    LDX   #CASBUF   CASSETTE BUFFER
A684 9F  7E             STX   CBUFAD    LOAD ADDRESS POINTER
A686 96  68    LA686    LDA   CURLIN    GET CURRENT LINE NUMBER MSB (CURLIN)
A688 4C                 INCA            IN DIRECT MODE IF ACCA = $FF
A689 26  0B             BNE   LA696     BRANCH IF NOT DIRECT MODE
A68B BD  A928           JSR   LA928     CLEAR SCREEN
A68E 9E  88             LDX   CURPOS    CURRENT SCREEN CHAR POSITION
A690 C6  53             LDB   #'S       'S' MEANS SEARCHING
A692 E7  81             STB   ,X++      PUT AN 'S' ON THE SCREEN
A694 9F  88             STX   CURPOS    STORE NEW CURSOR LOCATION
A696 8D  69    LA696    BSR   LA701     READ ONE BLOCK FROM TAPE
A698 DA  7C             ORB   BLKTYP    OR ERROR FLAG WITH BLOCK NUMBER
A69A 26  34             BNE   LA6D0     BRANCH IF NOT BLOCK ZERO OR ERROR
A69C 8E  01DA           LDX   #CASBUF   POINT TO CASSETTE BUFFER
A69F CE  01D2           LDU   #CFNBUF+1 POINT TO DESIRED NAME
A6A2 C6  08             LDB   #8        EIGHT CHARACTERS MAX IN NAME
A6A4 6F  E2             CLR   ,-S       ZERO A BYTE ON THE STACK
A6A6 A6  80    LA6A6    LDA   ,X+       GET CHAR FROM CASSETTE BLOCK
A6A8 109E 6B            LDY   CURLIN    GET CURLIN
A6AB 31  21             LEAY  1,Y       DIRECT MODE?
A6AD 26  05             BNE   LA6B4     FALL THROUGH IF DIRECT MODE
A6AF 0F  6F             CLR   DEVNUM    SET DEVICE NUMBER TO SCREEN
A6B1 BD  A282           JSR   PUTCHR    OUTPUT A CHAR
A6B4 A0  C0    LA6B4    SUBA  ,U+       SUBTRACT A CHAR FROM DESIRED NAME
                 *                      NON-ZERO RESULT IF NO MATCH
A6B6 AA  E4             ORA   ,S        OR WITH TOP OF STACK. RESULT WILL BE NON-ZERO IF MISMATCH
A6B8 A7  E4             STA   ,S        SAVE IT
A6BA 5A                 DECB            DONE ALL 8 CHARACTERS?
A6BB 26  E9             BNE   LA6A6     NO
A6BD A6  E0             LDA   ,S+       SEE IF ALL CHARS WERE OK
A6BF 27  0A             BEQ   LA6CB     BRANCH IF GOOD COMPARE
A6C1 6D  57             TST   -9,U      CHECK THE NUMBER OF CHARACTERS IN THE CLOAD STATEMENT
A6C3 27  06             BEQ   LA6CB     IF NO NAME SPECIFIED, ANY FILE IS OK
                 ** DIDN'T FIND THE RIGHT FILE IF HERE
```

```
A6C5 8D  0A          BSR   LA6D1     LOOK FOR FILE
A6C7 26  07          BNE   LA6D0     RETURN IF ERROR
A6C9 20  BB          BRA   LA686     GO LOOK SOME MORE
A6CB 86  46  LA6CB   LDA   #'F       *
A6CD 8D  29          BSR   LA6F8     *PUT 'F' ON THE SCREEN IF DIRECT MODE
A6CF 4F              CLRA            SET ZERO FLAG TO INDICATE NO ERRORS
A6D0 39      LA6D0   RTS
A6D1 7D  01E4 LA6D1   TST   CASBUF+10 CHECK FILE MODE
A6D4 26  09          BNE   LA6DF     BRANCH IF ASCII OR DATA
A6D6 BD  A77C        JSR   CASON     TURN ON TAPE DECK
A6D9 8D  30  LA6D9   BSR   GETBLK    LOAD A BLOCK FROM TAPE
A6DB 8D  08          BSR   LA6E5     CHECK FOR ERROR OR LAST BLOCK
A6DD 20  FA          BRA   LA6D9     KEEP GOING
A6DF 8D  20  LA6DF   BSR   LA701     READ ONE BLOCK FROM TAPE
A6E1 8D  02          BSR   LA6E5     CHECK FOR ERROR OR LAST BLOCK
A6E3 20  FA          BRA   LA6DF     KEEP READING BLOCKS
A6E5 26  06  LA6E5   BNE   LA6ED     GOT AN ERROR ON READING IN BLOCK
A6E7 96  7C          LDA   BLKTYP    GET BLOCK #
A6E9 40              NEGA            CHECK FOR LAST BLOCK
A6EA 2B  14          BMI   LA700     THIS WILL CAUSE THE UPPER LEFT CORNER
                     *               OF THE SCREEN TO BLINK WHEN SKIPPING OVER
                     *               CASSETTE FILES WHILE SEARCHING FOR THE
                     *               CORRECT FILE NAME.
A6EC 4A              DECA            IF BLOCK NUMBER WAS $FF, ACCA IS NOW ZERO - THIS WILL
                     *               CAUSE CLOAD TO IGNORE ERRORS IN THE
                     *               BLOCKS WHICH IT IS SKIPPING WHILE
                     *               LOOKING FOR THE CORRECT FILE NAME.
A6ED 97  81  LA6ED   STA   CSRERR    STORE ACCA TO ERROR FLAG
A6EF 32  62          LEAS  2,S       REMOVE RETURN FROM STACK
A6F1 20  12          BRA   LA705     TURN OFF MOTOR
A6F3 B6  0400 LA6F3  LDA   VIDRAM    GET FIRST CHAR ON SCREEN
A6F6 88  40          EORA  #$40      REVERSE THE VIDEO
A6F8 D6  68  LA6F8   LDB   CURLIN    GET CURLIN MSB
A6FA 5C              INCB            CHECK FOR DIRECT MODE
A6FB 26  03          BNE   LA700     BRANCH IF NOT DIRECT MODE
A6FD B7  0400        STA   VIDRAM    PUT IT ON SCREEN
A700 39      LA700   RTS

                     ****** READ A BLOCK FROM CASSETTE
A701 8D  79  LA701   BSR   CASON     START TAPE, AND LOOK FOR A BUNCH OF 55 OR AA BYTES
A703 8D  06          BSR   GETBLK    READ A BLOCK
A705 BD  A7E9 LA705  JSR   LA7E9     TURN OFF MOTOR
A708 D6  81          LDB   CSRERR    GET ERROR STATUS
A70A 39              RTS
A70B 1A  50  GETBLK  ORCC  #$50      DISABLE IRQ,FIRQ
A70D 8D  E4          BSR   LA6F3     REVERSE VIDEO UPPER LEFT CHAR IF DIRECT MODE
A70F 9E  7E          LDX   CBUFAD    GET LOAD ADDRESS
A711 4F              CLRA            RESET ACCA
A712 8D  41  LA712   BSR   LA755     READ A BIT FROM TAPE, RETURN IT IN CARRY FLAG
A714 46              RORA            PUT BIT IN MSB OF ACCA
A715 81  3C          CMPA  #$3C      GET SYNC'ED ON $3C
A717 26  F9          BNE   LA712     NOT SYNC'ED YET
A719 8D  2E          BSR   LA749     GET BLOCK NUMBER
A71B 97  7C          STA   BLKTYP    SAVE IT
A71D 8D  2A          BSR   LA749     GET CHAR COUNT
A71F 97  7D          STA   BLKLEN    SAVE IT
A721 9B  7C          ADDA  BLKTYP    ACCUMULATE CHECKSUM
A723 97  80          STA   CCKSUM    SAVE IT
A725 96  7D          LDA   BLKLEN    GET BACK CHAR COUNT
A727 97  81          STA   CSRERR    TEMP SAVE
```

```
A729 27  10          BEQ   LA73B     NULL SET OF CHARACTERS
A72B 8D  1E   LA72B   BSR   LA749     GET BYTE FROM TAPE
A72D A7  84          STA   ,X        FILL MEMORY WITH TAPE DATA
A72F A1  80          CMPA  ,X+       SEE IF WE READ BACK SAME THING
A731 26  11          BNE   LA744     BRANCH IF NOT PUTTING IT IN RAM
A733 9B  80          ADDA  CCKSUM    ACCUMULATE CHECKSUM
A735 97  80          STA   CCKSUM    TEMP STORE CHECKSUM
A737 0A  81          DEC   CSRERR    DECR TEMP CHAR COUNT
A739 26  F0          BNE   LA72B     GET ANOTHER CHARACTER
A73B 8D  0C   LA73B   BSR   LA749     GET CHECKSUM FROM TAPE
A73D 90  80          SUBA  CCKSUM    COMPARE TO CALCULATED CHECKSUM
A73F 27  05          BEQ   LA746     BRANCH IF OK
A741 86  01          LDA   #1        CHECKSUM ERROR FLAG
A743 8C              FCB   SKP2      SKIP TWO BYTES
A744 86  02   LA744   LDA   #2        NON-RAM ERROR FLAG
A746 97  81   LA746   STA   CSRERR   1 IF CHECKSUM ERROR. 2 IF LOADING INTO NON-RAM
A748 39              RTS

                     ***** GET A BYTE FROM TAPE
A749 86  08   LA749   LDA   #8        8 BITS/BYTE
A74B 97  82          STA   CPULWD    TEMP COUNTER
A74D 8D  06   LA74D   BSR   LA755     READ A BIT FROM TAPE
A74F 46              RORA            PUT IT INTO ACCA
A750 0A  82          DEC   CPULWD    GOT ALL 8 BITS
A752 26  F9          BNE   LA74D     NO
A754 39              RTS

                     **** READ A BIT FROM THE TAPE
A755 8D  06   LA755   BSR   LA75D     GET THE TIME BETWEEM TRANSITIONS
A757 D6  83          LDB   CPERTM    GET PERIOD TIMER
A759 5A              DECB
A75A D1  BF          CMPB  CMPMID    CMPMID CONTAINS 18 INITIALLY, AND IS USED TO DETERMINE
                     *               WHETHER THE BIT READ IS A ONE OR ZERO
                     *               IF THE PERIOD TIMER IS < 18, THE BIT
                     *               IS CONSIDERED TO BE A ONE, IF > 18, IT IS ZERO
A75C 39              RTS

                     *** MAIN TIMING LOOP
A75D 0F  83   LA75D   CLR   CPERTM    RESET PERIOD TIMER
A75F 0D  84          TST   CBITPHA   CHECK TO SEE IF SYNC'ED ON THE HI-LO TRANSITION OR LO-HI
A761 26  10          BNE   LA773     BRANCH ON HI-LO TRANSITION
                     **** LO - HI TRANSITION
A763 8D  07   LA763   BSR   LA76C     READ CASSETTE INPUT BIT
A765 25  FC          BCS   LA763     LOOP UNTIL IT IS LO
A767 8D  03   LA767   BSR   LA76C     READ CASSETTE INPUT DATA
A769 24  FC          BCC   LA767     WAIT UNTIL IT GOES HI
A76B 39              RTS

                     **** READ THE CASSETTE INPUT BIT OF THE PIA
A76C 0C  83   LA76C   INC   CPERTM    INCREMENT PERIOD TIMER
A76E F6  FF20        LDB   PIA1      GET CASSETTE INPUT BIT
A771 56              RORB            PUT CASSETTE BIT INTO THE CARRY FLAG
A772 39              RTS

                     *** WAIT FOR HI - LO TRANSITION
A773 8D  F7   LA773   BSR   LA76C     READ CASSETTE INPUT DATA
A775 24  FC          BCC   LA773     LOOP UNTIL IT IS HI
A777 8D  F3   LA777   BSR   LA76C     READ CASSETTE INPUT
A779 25  FC          BCS   LA777     LOOP UNTIL IT IS LO
A77B 39              RTS
```

```
                        ;#### LOOK FOR THE SYNC BYTES - RETURN WITH ACCA = 0 IF SYNC'ED
                        ;#### ON HI - LO TRANSITION, ACCA = $A0 IF SYNC'ED ON THE
                        ;#### LO - HI TRANSITION OF THE INPUT SIGNAL FROM THE CASSETTE.
A77C 1A  50     CASON   ORCC    #$50    DISABLE IRQ,FIRQ
A77E 8D  4A             BSR     LA7CA   TURN ON TAPE DECK MOTOR
A780 0F  82             CLR     CPULWD  RESET UP TO SPEED COUNTER
A782 8D  DF     LA782   BSR     LA763   WAIT FOR LO-HI TRANSITION
A784 8D  27     LA784   BSR     LA7AD   WAIT FOR HI-LO TRANSITION
A786 22  0F             BHI     LA797   CASSETTE SPEED IN RANGE FOR 1200 HZ
A788 8D  1D     LA788   BSR     LA7A7   WAIT FOR LO-HI TRANSITION
A78A 25  0F             BCS     LA79B   CASSETTE SPEED IN RANGE FOR 2400 HZ
A78C 0A  82             DEC     CPULWD  DECREMENT UP TO SPEED COUNTER IF SYNC'ED ON LO-HI
A78E 96  82             LDA     CPULWD  GET IT
A790 81  A0             CMPA    #-96    HAVE THERE BEEN 96 CONSECUTIVE 1-0-1-0 PATTERNS
A792 26  EE     LA792   BNE     LA782   NO
A794 97  84             STA     CBTPHA  SAVE WHICH TRANSITION (HI-LO OR LO-HI)
A796 39                 RTS
A797 8D  0E     LA797   BSR     LA7A7   WAIT FOR LO-HI TRANSITION
A799 22  E9             BHI     LA784   BRANCH IF TWO CONSECUTIVE 1200 HZ PULSES
A79B 8D  10     LA79B   BSR     LA7AD   WAIT FOR HI-LO TRANSITION
A79D 25  E9             BCS     LA788   BRANCH IF TWO CONSECUTIVE 2400 HZ PULSES
A79F 0C  82             INC     CPULWD  INCREMENT UP TO SPEED COUNTER IF SYNC'ED ON HI-LO
A7A1 96  82             LDA     CPULWD  GET IT
A7A3 80  60             SUBA    #96     GOT ENOUGH SYNC PULSES? - ACCA WILL BE ZERO IF
                        ;                       THERE HAVE BEEN 96 CONSECUTIVE
                        ;                       0-1-0-1 PATTERNS
A7A5 20  EB             BRA     LA792
A7A7 0F  83     LA7A7   CLR     CPERTM  RESET PERIOD TIMER
A7A9 8D  BC             BSR     LA767   WAIT UNTIL CASSETTE INPUT GOES HI
A7AB 20  04             BRA     LA7B1
A7AD 0F  83     LA7AD   CLR     CPERTM  RESET PERIOD TIMER
A7AF 8D  C6             BSR     LA777   WAIT UNTIL CASSETTE GOES LO
A7B1 D6  83     LA7B1   LDB     CPERTM  GET PERIOD TIMER
A7B3 D1  90             CMPB    CMP0    UPPER LIMIT OF 1200 HZ PERIOD
A7B5 22  03             BHI     LA7BA   BRANCH IF CASSETTE SPEED IS TOO SLOW OR DROPOUT
A7B7 D1  91             CMPB    CMP1    UPPER LIMIT OF 2400 HZ PERIOD
A7B9 39                 RTS
A7BA 0F  82     LA7BA   CLR     CPULWD  RESET UP TO SPEED COUNTER
A7BC 39                 RTS
                        ;
                        ;                MOTOR
A7BD 1F  89     MOTOR   TFR     A,B     SAVE CURRENT TOKEN IN ACCB
A7BF 9D  9F             JSR     GETNCH  GET NEXT INPUT CHARACTER FROM BASIC
A7C1 C1  AA             CMPB    #$AA    'OFF' TOKEN
A7C3 27  24             BEQ     LA7E9   YES
A7C5 C1  88             CMPB    #$88    'ON' TOKEN
A7C7 BD  A5C9           JSR     LA5C9   SYNTAX ERROR IF IT WASN'T 'ON' OR 'OFF'
A7CA B6  FF21   LA7CA   LDA     PIA1+1  READ CRA OF U4
A7CD 8A  08             ORA     #8      TURN ON BIT 3 WHICH ENABLES MOTOR RELAY
A7CF 8D  1F             BSR     LA7F0   PUT IT BACK
A7D1 9E  8A     LA7D1   LDX     ZERO    GET READY TO WAIT A WHILE
                        ;
                        ;  DELAY WHILE DECREMENTING X TO ZERO
A7D3 30  1F     LA7D3   LEAX    -1,X    DECREMENT X
A7D5 26  FC             BNE     LA7D3   BRANCH IF NOT ZERO
A7D7 39                 RTS

                        ;#### SEND SYNCLN 55'S TO TAPE
A7D8 1A  50     WRLDR   ORCC    #$50    DISABLE INTERRUPTS
A7DA 8D  EE             BSR     LA7CA   TURN ON TAPE DECK MOTOR
A7DC 9E  92             LDX     SYNCLN  GET COUNT OF 55'S TO SEND
```

```
A7DE 8D  48      LA7DE  BSR   LA828    SEND $55 TO TAPE
A7E0 30  1F             LEAX  -1,X     ARE ALL 55'S SENT?
A7E2 26  FA             BNE   LA7DE    NO
A7E4 39                 RTS

                ***** WRITE SYNC BYTES AND A BLOCK TO TAPE
A7E5 8D  F1     LA7E5  BSR   WRLDR     WRITE SYNC BYTES TO TAPE
A7E7 8D  0B     LA7E7  BSR   SNDBLK    GO WRITE A BLOCK
                *
                *    TURN OFF TAPE DECK MOTOR
A7E9 1C  AF     LA7E9  ANDCC #$AF      ENABLE IRQ,FIRQ
A7EB B6  FF21          LDA   PIA1+1    READ CRA OF U4
A7EE 84  F7            ANDA  #$F7      TURN OFF BIT 3
A7F0 B7  FF21  LA7F0   STA   PIA1+1    PUT IT BACK
A7F3 39                RTS
                *
                *     WRITE A BLOCK TO CASSETTE
                *       BUFFER SIZE IN BLKLEN
                *       STARTING ADDR IN CBUFAD
                *       BLOCK NUMBER IN BLKTYP
A7F4 1A  50     SNDBLK ORCC  #$50      DISABLE IRQ,FIRQ
A7F6 D6  7D            LDB   BLKLEN    GET CHAR COUNT
A7F8 D7  81            STB   CSRERR    TEMP CHAR COUNT
A7FA 96  7D            LDA   BLKLEN    GET CHAR COUNT (INCLUDED IN CHECKSUM)
A7FC 27  07            BEQ   LA805     BRANCH IF NO CHARACTERS - NULL
A7FE 9E  7E            LDX   CBUFAD    GET STARTING ADDRESS
A800 AB  80     LA800  ADDA  ,X+       CHECKSUM THE BUFFER
A802 5A                DECB            DONE ALL CHARACTERS?
A803 26  FB            BNE   LA800     NO
A805 9B  7C     LA805  ADDA  BLKTYP     ADD IN THE BLOCK NUMBER
A807 97  80            STA   CCKSUM    SAVE THE CHECKSUM
A809 9E  7E            LDX   CBUFAD    GET STARTING ADDRESS
A80B 8D  1B            BSR   LA828     SEND $55 TO TAPE
A80D 86  3C            LDA   #$3C      SYNC CHAR
A80F 8D  19            BSR   LA82A     SEND TO TAPE
A811 96  7C            LDA   BLKTYP    GET BLOCK NUMBER
A813 8D  15            BSR   LA82A     SEND BLOCK NUMBER TO TAPE
A815 96  7D            LDA   BLKLEN    GET CHARACTER COUNT
A817 8D  11            BSR   LA82A     SEND CHAR COUNT TO TAPE
A819 4D                TSTA            SET FLAGS
A81A 27  08            BEQ   LA824     BRANCH IF CHAR COUNT IS ZERO
A81C A6  80     LA81C  LDA   ,X+       GET BUFFER CHARACTER
A81E 8D  0A            BSR   LA82A     SEND BUFFER TO TAPE
A820 0A  81            DEC   CSRERR    DECR TEMP CHAR COUNT
A822 26  FB            BNE   LA81C     NOT DONE YET
A824 96  80     LA824  LDA   CCKSUM    GET CHECKSUM
A826 8D  02            BSR   LA82A     SEND CHECKSUM TO TAPE
A828 86  55     LA828  LDA   #$55      SEND A $55 TO TAPE
                *
                *   THIS ROUTINE SENDS THE A REG TO TAPE
A82A 34  02     LA82A  PSHS  A         SAVE OUTPUT CHARACTER
A82C C6  01            LDB   #1        ACCB CONTAINS A MASK USED TO DETERMINE WHETHER A
                *                      BIT IN THE OUTPUT CHARACTER IS HI OR LO
A82E 96  85     LA82E  LDA   CLSTSN    GET THE ENDING VALUE OF THE LAST SINE CYCLE
A830 B7  FF20          STA   DA        STORE IN THE D/A CONVERTER
A833 108E A85C         LDY   #LA85C    SINE LOOK-UP TABLE FOR GENERATING FSK
A837 E5  E4            BITB  ,S        IS THE CURRENT BIT A ONE OR A ZERO ?
A839 26  0D            BNE   LA848     IF A 1, DO HIGH FREQ
                ***** LOW FREQUENCY LOOK UP
A83B A6  A0     LA83B  LDA   ,Y+       USE EVERY BYTE IN TABLE IF LOW FREQUENCY
A83D 108C A880         CMPY  #LA85C+36 END OF SINE TABLE?
```

```
A841 27   12          BEQ   LA855     YES
A843 B7   FF20        STA   DA        SEND NEXT VALUE TO D/A CONVERTER
A846 20   F3          BRA   LA83B     GET NEXT VALUE
                  *** HIGH FREQUENCY LOOK UP
A848 A6   A1      LA848 LDA   ,Y++     USE EVERY OTHER BYTE IF HIGH FREQUENCY
A84A 108C A880        CMPY  #LA85C+36 END OF SINE TABLE?
A84E 27   05          BEQ   LA855     YES
A850 B7   FF20        STA   DA        SEND NEXT VALUE TO D/A CONVERTER
A853 20   F3          BRA   LA848     GET NEXT VALUE
A855 97   85      LA855 STA   CLSTSN   SAVE THE LAST VALUE SENT TO THE D/A CONVERTER
A857 58               ASLB            SHIFT MASK BIT LEFT
A858 24   D4          BCC   LA82E     DONE WHEN MASK BIT IS SHIFTED INTO CARRY FLAG
A85A 35   82          PULS  A,PC      RESTORE OUTPUT CHARACTER AND RETURN
                  *
                  *  THIS IS A LOOK-UP TABLE OF SINE VALUES FOR THE TAPE DECK FSK
                  *     (BIT 1 IS USED TO KEEP THE SERIAL OUTPUT MARKING)
A85C 82 92 AA BA  LA85C FCB   $82,$92,$AA,$BA,$CA,$DA,$EA,$F2
A864 FA FA FA F2      FCB   $FA,$FA,$FA,$F2,$EA,$DA,$CA,$BA
A86C AA 92 7A 6A      FCB   $AA,$92,$7A,$6A,$52,$42,$32,$22
A874 12 0A 02 02      FCB   $12,$0A,$02,$02,$02,$0A,$12,$22
A87C 32 42 52 6A      FCB   $32,$42,$52,$6A
                  *
                  *              SET
A880 8D   3F      SET   BSR   LA8C1     GET ABSOLUTE SCREEN POSITION OF GRAPHICS BLOCK
A882 34   10          PSHS  X         SAVE CHARACTER LOCATION
A884 BD   B738        JSR   LB738     SYNTAX CHECK FOR COMMA - RETURN EXPR VALUE IN ACCB
A887 35   10          PULS  X         REGET CHARACTER LOCATION
A889 C1   08          CMPB  #8        NINE ALLOWABLE COLORS
A88B 22   48          BHI   LA8D5     ILLEGAL COLOR - 'ILLEGAL FUNCTION CALL'
A88D 5A               DECB            CHANGE COLOR NUMBERS FROM 0-8 TO (-1 TO 7)
A88E 2B   05          BMI   LA895     BRANCH IF SET (X,Y,0)
A890 86   10          LDA   #$10      $10 OFFSET BETWEEN DIFFERENT COLORS
A892 3D               MUL             MULT BY COLOR FOR TOTAL OFFSET
A893 20   08          BRA   LA89D     GO SAVE THE COLOR
A895 E6   84      LA895 LDB   ,X        GET CURRENT CHAR FROM SCREEN
A897 2A   03          BPL   LA89C     BRANCH IF NOT GRAPHIC
A899 C4   70          ANDB  #$70      SAVE ONLY THE COLOR INFO
A89B 21               FCB   SKP1      SKIP THE NEXT BYTE
A89C 5F           LA89C CLRB            RESET ASCII BLOCK TO ZERO COLOR
A89D 34   04      LA89D PSHS  B         SAVE COLOR INFO
A89F 8D   6C          BSR   LA90D     SYNTAX CHECK FOR ")"
A8A1 A6   84          LDA   ,X        GET CURRENT CHARACTER FROM SCREEN
A8A3 2B   01          BMI   LA8A6     BRANCH IF GRAPHIC
A8A5 4F               CLRA            RESET ASCII CHARACTER TO ALL PIXELS OFF
A8A6 84   0F      LA8A6 ANDA  #$0F      SAVE ONLY PIXEL ON/OFF INFO
A8A8 9A   86          ORA   GRBLOK    OR WITH WHICH PIXEL TO TURN ON
A8AA AA   E0          ORA   ,S+       OR IN THE COLOR
A8AC 8A   80      LA8AC ORA   #$80      FORCE GRAPHIC
A8AE A7   84          STA   ,X        DISPLAY IT ON THE SCREEN
A8B0 39               RTS
                  *
                  *              RESET
A8B1 8D   0E      RESET BSR   LA8C1     GET ABSOLUTE SCREEN ADDRESS OF THIS CHARACTER
A8B3 8D   58          BSR   LA90D     SYNTAX CHECK FOR ")"
A8B5 4F               CLRA            * ACCA=ZERO GRAPHIC BLOCK - FOR USE IN CASE YOU'RE
                  *                   * TRYING TO RESET A NON GRAPHIC BLOCK
A8B6 E6   84          LDB   ,X        GET CURRENT CHAR FROM SCREEN
A8B8 2A   F2          BPL   LA8AC     BRANCH IF NON-GRAPHIC
A8BA 03   86          COM   GRBLOK    INVERT PIXEL ON/OFF MASK
A8BC D4   86          ANDB  GRBLOK    AND IT WITH CURRENT ON/OFF DATA
A8BE E7   84          STB   ,X        DISPLAY IT
A8C0 39               RTS
```

```
                        *** THIS ROUTINE WILL CHECK SYNTAX AND CHECK FOR LEGAL VALUES
                        *** OF SET,RESET & POINT HORIZONTAL AND VERTICAL PARAMETERS
                        *** AND RETURN THEIR ABSOLUTE SCREEN ADDRESS IN THE X REGISTER
                        *** WHICH OF THE FOUR PIXELS OF THE GRAPHIC BLOCK SELECTED
                        *** IS RETURNED IN GRBLOK.
A8C1 BD  B26A  LA8C1    JSR   LB26A     SYNTAX CHECK FOR "("
A8C4 BD  019D  LA8C4    JSR   RVEC21    HOOK INTO RAM
A8C7 BD  B70B           JSR   LB70B     EVALUATE EXPRESSION
A8CA C1  3F             CMPB  #63       ONLY 64 HORIZONTAL GRAPHIC BLOCKS
A8CC 22  07             BHI   LA8D5     'ILLEGAL FUNCTION CALL'
A8CE 34  04             PSHS  B         SAVE HOR COORD
A8D0 BD  B738           JSR   LB738     SYNTAX CHECK FOR COMMA AND EVALUATE EXPR
A8D3 C1  1F             CMPB  #31       ONLY 32 VERTICAL BLOCKS
A8D5 22  71   LA8D5      BHI   LA948     'ILLEGAL FUNCTION CALL'
A8D7 34  04             PSHS  B         SAVE VERT COORD
A8D9 54                 LSRB            DIVIDE BY TWO BECAUSE THERE ARE 2 GRAPHIC PIXELS/HOR
                        *               CHARACTER POSITION (BYTE)
A8DA 86  20             LDA   #32       32 BYTES/ROW
A8DC 3D                 MUL             GET ROW OFFSET OF CHAR POSITION
A8DD 8E  0400           LDX   #VIDRAM   SCREEN BUFFER ADDRESS
A8E0 30  8B             LEAX  D,X       ADD ROW OFFSET TO SCREEN BUFFER ADDRESS
A8E2 E6  61             LDB   1,S       GET HOR COORD
A8E4 54                 LSRB            2 VERTICAL PIXELS/CHARACTER POSITION
A8E5 3A                 ABX             ADD VERTICAL OFFSET TO CHARACTER ADDRESS
A8E6 35  06             PULS  A,B       GET VER COORD TO ACCA, HOR COORD TO ACCB
A8E8 84  01             ANDA  #1        KEEP ONLY LSB OF VER COORD
A8EA 56                 RORB            LSB OF HOR COORD TO CARRY FLAG
A8EB 49                 ROLA            LSB OF HOR TO BIT 0 OF ACCA
A8EC C6  10             LDB   #$10      MAKE A BIT MASK - TURN ON BIT 4
A8EE 54   LA8EE         LSRB            SHIFT IT RIGHT ONCE
A8EF 4A                 DECA            SHIFTED IT ENOUGH?
A8F0 2A  FC             BPL   LA8EE     NO
A8F2 D7  86             STB   GRBLOK    ACCB=8 FOR UPPER LEFT PIXEL, =4 FOR UPPER RIGHT
                        *               PIXEL =2 FOR LOWER LEFT, =1 FOR LOWER RIGHT
A8F4 39                 RTS
                        *
                        *                POINT
A8F5 8D  CD   POINT     BSR   LA8C4     EVALUATE EXPRESSION
A8F7 C6  FF             LDB   #$FF      INITIAL VALUE OF ON/OFF FLAG = OFF (FALSE)
A8F9 A6  84             LDA   ,X        GET CURRENT GRAPHIC CHARACTER
A8FB 2A  0D             BPL   LA90A     BRANCH IF NON-GRAPHIC (ALWAYS FALSE)
A8FD 94  86             ANDA  GRBLOK    'AND' CURR CHAR WITH THE PIXEL IN QUESTION
A8FF 27  08             BEQ   LA909     BRANCH IF THE ELEMENT IS OFF
A901 E6  84             LDB   ,X        GET CURRENT CHARACTER
A903 54                 LSRB            * SHIFT RIGHT
A904 54                 LSRB            * SHIFT RIGHT
A905 54                 LSRB            * SHIFT RIGHT
A906 54                 LSRB            * SHIFT RIGHT - NOW THE HIGH NIBBLE IS IN THE LOW NIBBLE
A907 C4  07             ANDB  #7        KEEP ONLY THE COLOR INFO
A909 5C   LA909         INCB            ACCB=0 FOR NO COLOR, =1TO8 OTHERWISE
A90A BD  A5E8  LA90A    JSR   LA5E8     CONVERT ACCB TO FLOATING POINT
A90D 7E  B267  LA90D    JMP   LB267     SYNTAX CHECK FOR ")"
                        *
                        *                CLS
A910 BD  01A0  CLS      JSR   RVEC22    HOOK INTO RAM
A913 27  13             BEQ   LA928     BRANCH IF NO ARGUMENT
A915 BD  B70B           JSR   LB70B     CALCULATE ARGUMENT. RETURN VALUE IN ACCB
A918 C1  08             CMPB  #8        VALID ARGUMENT?
A91A 22  1B             BHI   LA937     IF ARGUMENT >8. GO PRINT 'MICROSOFT'
A91C 5D                 TSTB            SET FLAGS
```

```
A91D 27  06          BEQ   LA925     COLOR 0
A91F 5A              DECB           ACCB NOW CONTAINS 0-7
A920 86  10          LDA   #$10     EACH GRAPHIC BLOCK SEPARATED BY $10 FROM ONE ANOTHER
A922 3D              MUL            ACCB CONTAINS ONE OF 8 OFFSETS
A923 CA  0F          ORB   #$0F     BITS 0-3 SET FOR SOLID COLOR GRAPHIC BLOCK
A925 CA  80    LA925 ORB   #$80     BIT 7 SET FOR GRAPHICS
A927 8C              FCB   SKP2     SKIP TWO BYTES
                     *
                     *       CLEAR SCREEN
A928 C6  60    LA928 LDB   #$60     BLANK
A92A 8E  0400        LDX   #VIDRAM  GET ADDR OF START OF SCREEN BUFFER
A92D 9F  88    LA92D STX   CURPOS   SAVE IT IN CURPOS
A92F E7  80    LA92F STB   .X+      FILL SCREEN WITH CONTENTS OF ACCB
A931 8C  05FF        CMPX  #VIDRAM+$1FF END OF SCREEN?
A934 23  F9          BLS   LA92F    NO
A936 39              RTS
A937 BD  EF    LA937 BSR   LA928    CLEAR SCREEN
A939 8E  A165        LDX   #LA166-1 *
A93C 7E  B99C        JMP   LB99C    * PRINT 'MICROSOFT'

A93F BD  B26D  LA93F JSR   LB26D    SYNTAX CHECK FOR A COMMA
A942 BD  B70B  LA942 JSR   LB70B    EVALUATE EXPRESSION. RETURN VALUE IN ACCB
A945 5D              TSTB           SET FLAGS
A946 26  3C          BNE   LA984    RETURN IF NON ZERO
A948 7E  B44A  LA948 JMP   LB44A    'ILLEGAL FUNCTION CALL' IF ZERO

                     *         SOUND
A94B BD  F5    SOUND BSR   LA942    EVALUATE EXPRESSION (FREQUENCY)
A94D D7  8C          STB   SNDTON   SAVE IT
A94F 8D  EE          BSR   LA93F    EVALUATE EXPRESSION (SOUND LENGTH)
A951 86  04          LDA   #4       CONSTANT FACTOR
A953 3D              MUL            EXPAND LENGTH EXPRESSION
A954 DD  8D          STD   SNDDUR   SAVE LENGTH OF SOUND
A956 B6  FF03        LDA   PIA0+3   GET CONTROL REGISTER OF PIA0, PORT B
A959 8A  01          ORA   #1       *
A95B B7  FF03        STA   PIA0+3   * ENABLE 60 HZ INTERRUPT (PIA0 IRQ)
A95E 0F  08          CLR   ARYDIS   CLEAR THE ARRAY DISABLE FLAG
A960 8D  40          BSR   LA9A2    CONNECT D/A SOUND INPUT TO OUTPUT OF SOUND MUX
A962 8D  12          BSR   LA976    TURN ON AUDIO - ENABLE SOUND MUX
A964 8D  1F    LA964 BSR   LA985    STORE 2.5 VOLTS TO D/A AND WAIT
A966 86  FE          LDA   #$FE     DATA TO MAKE D/A OUT = 5 VOLTS
A968 8D  1D          BSR   LA987    STORE IT TO D/A AND WAIT
A96A 8D  19          BSR   LA985    STORE 2.5 VOLTS TO D/A AND WAIT
A96C 86  02          LDA   #2       DATA TO MAKE D/A OUT = 0 VOLTS
A96E 8D  17          BSR   LA987    STORE IT TO D/A AND WAIT
A970 9E  8D          LDX   SNDDUR   * IS SNDDUR = 0? - THE IRQ INTERRUPT SERVICING
                     *              * ROUTINE WILL DECREMENT SNDDUR
A972 26  F0          BNE   LA964    NOT DONE YET

                     *** THESE ROUTINES WILL ENABLE/DISABLE THE ANALOG MUX
A974 4F        LA974 CLRA           BIT 3 OF ACCA = 0, DISABLE ANALOG MUX
A975 8C              FCB   SKP2     SKIP TWO BYTES
A976 86  08    LA976 LDA   #8       BIT 3 OF ACCA = 1, ENABLE ANALOG MUX
A978 A7  E2          STA   ,-S      SAVE ACCA ON STACK
A97A B6  FF23        LDA   PIA1+3   GET CONTROL REGISTER OF PIA1, PORT B
A97D 84  F7          ANDA  #$F7     RESET BIT 3
A97F AA  E0          ORA   ,S+      OR IN BIT 3 OF ACCA (SAVED ON STACK)
A981 B7  FF23        STA   PIA1+3   SET/RESET CB2 OF U4
A984 39        LA984 RTS
A985 86  7E    LA985 LDA   #$7E     DATA VALUE TO MAKE D/A OUTPUT = 2.5 VOLTS
A987 B7  FF20  LA987 STA   DA       STORE IT IN D/A
```

```
A98A 96  8C          LDA   SNDTON   GET FREQUENCY
A98C 4C        LA98C INCA           INCREMENT IT
A98D 26  FD          BNE   LA98C    LOOP UNTIL DONE
A98F 39              RTS
                     *
                     *   AUDIO
A990 1F  89    AUDIO TFR   A,B      SAVE ON/OFF TOKEN IN ACCB
A992 9D  9F          JSR   GETNCH   MOVE BASIC POINT TO NEXT CHARACTER
A994 C1  AA          CMPB  #$AA     'OFF' TOKEN?
A996 27  DC          BEQ   LA974    YES - TURN OFF ANALOG MUX
A998 C0  88          SUBB  #$88     'ON' TOKEN
A99A BD  A5C9        JSR   LA5C9    SYNTAX ERROR IF NOT 'OFF' OR 'ON'
A99D 5C              INCB           NOW ACCB = 1
A99E 8D  02          BSR   LA9A2    ROUTE CASSETTE TO SOUND MULTIPLEXER
A9A0 20  D4          BRA   LA976    ENABLE SOUND MULTIPLEXER

                     *** THIS ROUTINE WILL TRANSFER BIT 0 OF ACCB TO SEL 1 OF
                     *** THE ANALOG MULTIPLEXER AND BIT 1 OF ACCB TO SEL 2.
A9A2 CE  FF01  LA9A2 LDU   #PIA0+1  POINT U TO PIA0 CONTROL REG
A9A5 8D  00          BSR   LA9A7    PROGRAM 1ST CONTROL REGISTER
A9A7 A6  C4    LA9A7 LDA   ,U       GET PIA CONTROL REGISTER
A9A9 84  F7          ANDA  #$F7     RESET CA2 (CB2) OUTPUT BIT
A9AB 57              ASRB           SHIFT ACCB BIT 0 TO CARRY FLAG
A9AC 24  02          BCC   LA9B0    BRANCH IF CARRY = ZERO
A9AE 8A  08          ORA   #8       FORCE BIT 3=1; SET CA2(CB2)
A9B0 A7  C1    LA9B0 STA   ,U++     PUT IT BACK IN THE PIA CONTROL REGISTER
A9B2 39              RTS
                     *
                     *  IRQ SERVICE
A9B3 B6  FF03  BIRQSV LDA  PIA0+3   CHECK FOR 60HZ INTERRUPT
A9B6 2A  0D          BPL   LA9C5    RETURN IF 63.5 MICROSECOND INTERRUPT
A9B8 B6  FF02        LDA   PIA0+2   RESET PIA0, PORT B INTERRUPT FLAG
A9BB BE  008D        LDX   >SNDDUR  GET INTERRUPT TIMER (SOUND COMMAND)
A9BE 27  05          BEQ   LA9C5    RETURN IF TIMER = 0
A9C0 30  1F          LEAX  -1,X     DECREMENT TIMER IF NOT = 0
A9C2 BF  008D  LA9C2 STX   >SNDDUR  SAVE NEW TIMER VALUE
A9C5 3B        LA9C5 RTI            RETURN FROM INTERRUPT
                     *
                     *    JOYSTK
A9C6 BD  B70E  JOYSTK JSR  LB70E     EVALUATE JOYSTICK ARGUMENT
A9C9 C1  03          CMPB  #3       TWO JOYSTICKS MAXIMUM (HOR & VER FOR EACH)
A9CB 1022 0A7B       LBHI  LB44A    'ILLEGAL FUNCTION CALL' IF >3
A9CF 5D              TSTB           SET FLAGS
A9D0 26  02          BNE   LA9D4    GET NEW DATA ONLY IF JOYSTK(0)
A9D2 8D  0A          BSR   GETJOY   GET NEW DATA FOR ALL JOYSTICKS
A9D4 BE  015A  LA9D4 LDX   #POTVAL  POINT X TO JOYSTICK DATA BUFFER
A9D7 D6  53          LDB   FPA0+3   WHICH JOYSTICK DID YOU WANT?
A9D9 E6  85          LDB   B,X      PUT ITS DATA INTO ACCB
A9DB 7E  B4F3        JMP   LB4F3    CONVERT ACCB INTO FLOATING POINT NUMBER
                     *
                     * JOYSTK DATA AT:
                     *     $15A      $15B      $15C      $15D
                     *     LEFT      LEFT      RIGHT     RIGHT
                     *     VERT      HORIZ     VERT      HORIZ

                     *** THIS IS A 6 BIT SOFTWARE A/D CONVERSION ROUTINE
A9DE 8D  94    GETJOY BSR  LA974     TURN OFF AUDIO
A9E0 8E  015E        LDX   #POTVAL+4 POINT X TO JOYSTICK DATA BUFFER
A9E3 C6  03          LDB   #3        GET FOUR SETS OF DATA (4 JOYSTICKS)
A9E5 86  0A    LA9E5 LDA   #10       10 TRIES TO GET STABLE READING
A9E7 ED  E3          STD   ,--S      STORE JOYSTICK NUMBER AND TRY NUMBER ON THE STACK
```

```
A9E9 8D   B7          BSR   LA9A2    SET THE SELECT INPUTS ON ANALOG MULTIPLEXER
A9EB CC   4080  LA9EB LDD   #$4080   ACCA IS A SHIFT COUNTER OF HOW MANY BITS TO CONVERT
                *                    AND WILL BE $40 (6 BITS) FOR THE COLOR
                *                    COMPUTER. ACCB CONTAINS A VALUE EQUAL TO 1/2
                *                    THE CURRENT TRIAL DIFFERENCE. INITIALLY =$80 (2.5 VOLTS).
A9EE A7   E2    LA9EE STA   .-S      TEMP STORE SHIFT COUNTER ON STACK
A9F0 CA   02          ORB   #2       KEEP RS 232 SERIAL OUT MARKING
A9F2 F7   FF20        STB   DA       STORE IN D/A CONVERTER
A9F5 C8   02          EORB  #2       PUT RS232 BIT BACK TO ZERO
A9F7 B6   FF00        LDA   PIA0     HIGH BIT IS FROM COMPARATOR
A9FA 2B   03          BMI   LA9FF    BRANCH IF COMPARATOR OUTPUT IS HIGH
A9FC E0   E4          SUBB  ,S       SUBTRACT 1/2 THE CURRENT TRIAL DIFFERENCE
A9FE 8C               FCB   SKP2     SKIP NEXT TWO BYTES
A9FF EB   E4    LA9FF ADDB  ,S       ADD 1/2 OF THE CURRENT TRIAL DIFFERENCE
AA01 A6   E0          LDA   ,S+      PULL SHIFT COUNTER OFF THE STACK
AA03 44               LSRA           SHIFT IT RIGHT ONCE
AA04 81   01          CMPA  #1       HAVE ALL THE SHIFTS BEEN DONE?
AA06 26   E6          BNE   LA9EE    NO
AA08 54               LSRB           YES - THE DATA IS IN THE TOP 6 BYTES OF ACCB
AA09 54               LSRB           PUT IT INTO THE BOTTOM SIX
AA0A E1   1F          CMPB  -1,X     IS THIS VALUE EQUAL TO THE LAST TRY?
AA0C 27   04          BEQ   LAA12    YES - GO SAVE THE VALUE
AA0E 6A   E4          DEC   ,S       NO-DECREMENT TRIES COUNTER
AA10 26   D9          BNE   LA9EB    BRANCH IF YOU HAVEN'T TRIED 10 TIMES
                *                    IF YOU FALL THROUGH HERE YOU HAVE TRIED TO GET THE SAME READING
                *                    10 TIMES AND NEVER GOTTEN A MATCH. AS A RESULT YOU JUST FALL
                *                    THROUGH AND USE THE LAST VALUE READ IN.
AA12 E7   82    LAA12 STB   .-X      SAVE THE DIGITIZED VALUE
AA14 EC   E1          LDD   ,S++     GET THE NUMBER OF THE JOYSTICK JUST DONE
AA16 5A               DECB           DECR JOYSTK NUMBER
AA17 2A   CC          BPL   LA9E5    BRANCH IF THE LAST ONE DONE WASN'T #0
AA19 39               RTS
                *
                *    SET CARRY IF NUMERIC - RETURN WITH
                *    ZERO FLAG SET IF ACCA = 0 OR 3A(:) - END
                *    OF BASIC LINE OR SUB LINE
AA1A 81   3A    BROMHK CMPA  #'9+1     IS THIS CHARACTER >=(ASCII 9)+1?
AA1C 24   0A          BHS   LAA28    BRANCH IF > 9; Z SET IF = COLON
AA1E 81   20          CMPA  #SPACE   SPACE?
AA20 26   02          BNE   LAA24    NO - SET CARRY IF NUMERIC
AA22 0E   9F          JMP   GETNCH   IF SPACE, GET NEXT CHAR (IGNORE SPACES)
AA24 80   30    LAA24 SUBA  #'0      SET CARRY IF
AA26 80   D0          SUBA  #-'0     CHARACTER > ASCII 0
AA28 39         LAA28 RTS
                *
```

APPENDIX B

DSKCON ROUTINE

```
D766 34 76      DSKCON  PSHS    U,Y,X,D     SAVE REGISTERS
D768 86 05      LD66E   LDA     #5          * GET RETRY COUNT AND
D76A 34 02              PSHS    A           * SAVE IT ON THE STACK
D76C 7F 0985    LD672   CLR     RDYTMR      RESET DRIVE NOT READY TIMER
D76F D6 EB              LDB     DCDRV       GET DRIVE NUMBER
D771 8E D8A2            LDX     #LD7AA      POINT X TO MOTOR ENABLE MASKS
D774 B6 0986            LDA     DRGRAM      GET DSKREG IMAGE
D777 84 A8              ANDA    #$A8        KEEP MOTOR STATUS, DOUBLE DENSITY, NMI ENABLE
D779 AA 85              ORA     B,X         'OR' IN DRIVE ENABLE DATA
D77B 8A 20              ORA     #$20        'OR' IN DOUBLE DENSITY
D77D D6 EC              LDB     DCTRK       GET TRACK NUMBER
D77F C1 16              CMPB    #22         PRECOMPENSATION STARTS AT TRACK 22
D781 25 00              BLO     LD68B       BRANCH IF LESS THAN 22
D783 8A 10              ORA     #$10        TURN ON WRITE PRECOMPENSATION IF >= 22
D785 1F 89      LD68B   TFR     A,B         SAVE PARTIAL IMAGE IN ACCB
D787 8A 08              ORA     #8          'OR' IN MOTOR ON BIT
D789 B7 0986            STA     DRGRAM      SAVE IMAGE IN RAM
D78C B7 FF40            STA     DSKREG      PROGRAM THE 1793 CONTROL REGISTER
D78F C5 08              BITB    #8          = WAS MOTOR ALREADY ON?
D791 26 04              BNE     LD69F       = DON'T WAIT FOR IT TO COME UP
                        *                   TO SPEED IF ALREADY ON
D793 BD A7D1            JSR     LA7D1       * WAIT A WHILE
D796 BD A7D1            JSR     LA7D1       * WAIT SOME MORE FOR MOTOR TO COME UP TO SPEED
D799 9D 3B      LD69F   BSR     LD6DE       WAIT UNTIL NOT BUSY OR TIME OUT
D79B 26 08              BNE     LD6AD       BRANCH IF TIMED OUT (DOOR OPEN, NO DISK, NO POWER, ETC.)
D79D 0F F0              CLR     DCSTA       CLEAR STATUS REGISTER
D79F 8E D89A            LDX     #LD7A2      POINT TO COMMAND JUMP VECTORS
D7A2 D6 EA              LDB     DCOPC       GET COMMAND
D7A4 58                 ASLB                2 BYTES PER COMMAND JUMP ADDRESS
D7A5 AD 95              JSR     [B,X]       GO DO IT
D7A7 35 02      LD6AD   PULS    A           GET RETRY COUNT
D7A9 D6 F0              LDB     DCSTA       GET STATUS
D7AB 27 09              BEQ     LD6BE       BRANCH IF NO ERRORS
D7AD 4A                 DECA                DECREMENT RETRIES COUNTER
D7AE 27 06              BEQ     LD6BE       BRANCH IF NO RETRIES LEFT
D7B0 34 02              PSHS    A           SAVE RETRY COUNT ON STACK
D7B2 8D 09              BSR     LD6C5       RESTORE HEAD TO TRACK 0
D7B4 26 EF              BNE     LD6AD       BRANCH IF SEEK ERROR
D7B6 20 B2              BRA     LD672       GO TRY COMMAND AGAIN IF NO ERROR
D7B8 86 78      LD6BE   LDA     #120        120*1/60 = 2 SECONDS
D7BA B7 0985            STA     RDYTMR      WAIT 2 SECONDS BEFORE TURNING OFF MOTOR
D7BD 35 F6              PULS    PC,U,Y,X,D  RESTORE REGISTER - EXIT DSKCON
                        *       RESTORE HEAD TO TRACK 0
D7B8 8E 097E    LD6C5   LDX     #DROTRK     POINT TO TRACK TABLE
D7BB D6 EB              LDB     DCDRV       GET DRIVE NUMBER
D7BD 6F 85              CLR     B,X         ZERO TRACK NUMBER
D7BF 86 03              LDA     #3          * RESTORE HEAD TO TRACK 0, UNLOAD THE HEAD
D7C1 B7 FF48            STA     FDCREG      * AT START, 30 MS STEPPING RATE
D7C4 1E 88              EXG     A,A         =
D7C6 1E 88              EXG     A,A         = WAIT FOR 1793 TO RESPOND TO COMMAND
D7C8 8D 07              BSR     LD6DE       WAIT TILL DRIVE NOT BUSY
D7CA 8D 24              BSR     LD6FD       WAIT SOME MORE
D7CC 84 10              ANDA    #$10        1793 STATUS : KEEP ONLY SEEK ERROR
D7CE 97 F0              STA     DCSTA       SAVE IN DSKCON STATUS
```

```
D7D7 39          LD6DD   RTS
                 *     WAIT FOR THE 1793 TO BECOME UNBUSY.  IF IT DOES
                 *     NOT BECOME UNBUSY, FORCE AN INTERRUPT AND ISSUE
                 *     A 'DRIVE NOT READY' 1793 ERROR.
D7D8 9E   8A     LD6DE   LDX     ZERO      GET ZERO TO X REGISTER - LONG WAIT
D7DA 30   1F     LD6E0   LEAX    -1,X      DECREMENT LONG WAIT COUNTER
D7DC 27   06             BEQ     LD6EC     IF NOT READY BY NOW, FORCE INTERRUPT
D7DE B6   FF48           LDA     FDCREG    * GET 1793 STATUS AND TEST
D7E1 85   01             BITA    #1        * BUSY STATUS BIT
D7E3 26   F3             BNE     LD6E0     BRANCH IF BUSY
D7E5 39                  RTS
D7E6 86   D0     LD6EC   LDA     #$D0      FORCE INTERRUPT - TERMINATE
D7E8 B7   FF48           STA     FDCREG    = SEND TO 1793 - BREAK OUT OF ANY
                 *                         COMMAND IN PROCESS
D7EB 1E   88             EXG     A,A       * WAIT BEFORE READING 1793
D7ED 1E   88             EXG     A,A       *
D7EF B6   FF48           LDA     FDCREG    GET STATUS FROM 1793
D7F2 86   80             LDA     #$80      RETURN DRIVE NOT READY STATUS IF THE
                 *                          DRIVE DOES NOT BECOME UNBUSY
D7F4 97   F0             STA     DCSTA     SAVE DSKCON STATUS BYTE
D7F6 39                  RTS
                 *     MEDIUM DELAY
D7F7 8E   222E   LD6FD   LDX     #8750     DELAY FOR A WHILE
D7FA 30   1F     LD700   LEAX    -1,X      * DECREMENT DELAY COUNTER AND
D7FC 26   FA             BNE     LD700     * BRANCH IF NOT DONE
D7FE 39                  RTS
                 *     READ ONE SECTOR
D7FF 86   80     LD705   LDA     #$80      $80 IS READ FLAG (1793 READ SECTOR)
D801 8C                  FCB     SKP2      SKIP TWO BYTES
                 *     WRITE ONE SECTOR
D802 86   A0     LD708   LDA     #$A0      $A0 IS WRITE FLAG (1793 WRITE SECTOR)
D804 34   02             PSHS    A         SAVE READ/WRITE FLAG ON STACK
D806 8E   097E           LDX     #DROTRK   POINT X TO TRACK # TABLE IN RAM
D809 D6   EB             LDB     DCDRV     GET DRIVE NUMBER
D80B 3A                  ABX               POINT X TO CORRECT DRIVE'S TRACK BYTE
D80C E6   84             LDB     ,X        GET TRACK NUMBER
D80E F7   FF49           STB     FDCREG+1  SEND TO 1793 TRACK REGISTER
D811 D1   EC             CMPB    DCTRK     CHECK CURRENT TRACK
D813 27   1C             BEQ     LD739     BRANCH IF ON CORRECT TRACK
D815 96   EC             LDA     DCTRK     GET CURRENT TRACK DESIRED
D817 B7   FF4B           STA     FDCREG+3  SEND TO 1793 DATA REGISTER
D81A A7   84             STA     ,X        SAVE IN RAM TRACK IMAGE
D81C 86   17             LDA     #$17      * SEEK COMMAND FOR 1793: DO NOT LOAD THE
D81E B7   FF48           STA     FDCREG    * HEAD AT START, VERIFY DESTINATION TRACK,
D821 1E   88             EXG     A,A       * 30 MS STEPPING RATE - WAIT FOR
D823 1E   88             EXG     A,A       * VALID STATUS FROM 1793
D825 8D   AF             BSR     LD6DE     WAIT TILL NOT BUSY
D827 26   06             BNE     LD737     BRANCH IF TIMED OUT
D829 8D   CA             BSR     LD6FD     WAIT SOME MORE
D82B 84   18             ANDA    #$18      CHECK FOR SEEK ERROR OR CRC ERROR IN ID FIELD
D82D 27   02             BEQ     LD739     BRANCH IF NO ERRORS - HEAD ON CORRECT TRACK
D82F 97   F0             STA     DCSTA     SAVE IN DSKCON STATUS
D831 35   82   LD737   PULS    PC,A
                 *       HEAD POSITIONED ON CORRECT TRACK
```

```
D833 96   ED      LD739   LDA    DSEC      GET SECTOR NUMBER
D835 B7   FF4A            STA    FDCREG+2  SEND TO 1793 SECTOR REGISTER
D838 8E   D890            LDX    #LD798    * POINT X TO ROUTINE TO BE VECTORED
D83B BF   0983            STX    DNMIVC    * TO BY NMI UPON COMPLETION OF
                  *                        DISK I/O AND SAVE VECTOR
D83E 9E   EE              LDX    DCBPT     POINT X TO I/O BUFFER
D840 B6   FF48            LDA    FDCREG    RESET STATUS REGISTER
D843 86   0986            LDA    DRGRAM    GET DSKREG IMAGE
D846 8A   80              ORA    #$80      SET FLAG TO ENABLE 1793 TO HALT 6809
D848 35   04              PULS   B         GET READ/WRITE COMMAND FROM STACK
D84A 109E 8A              LDY    ZERO      ZERO OUT Y - TIMEOUT INITIAL VALUE
D84D CE   FF48            LDU    #FDCREG   U POINTS TO 1793 INTERFACE REGISTERS
D850 73   0982            COM    NMIFLG    NMI FLAG = $FF: ENABLE NMI VECTOR
D853 1A   50              ORCC   #$50      DISABLE FIRQ,IRQ
D855 F7   FF48            STB    FDCREG    SEND READ/WRITE COMMAND TO 1793: SINGLE
                  *                        RECORD, COMPARE
D858 1E   88              EXG    A,A       * FOR SIDE 0, NO 15 MS DELAY, DISABLE SIDE SELECT
D85A 1E   88              EXG    A,A       * COMPARE, WRITE DATA ADDRESS MARK (FB)
                  *                       *WAIT FOR STATUS
D85C C1   80              CMPB   #$80      WAS THIS A READ?
D85E 27   1A              BEQ    LD782     IF SO, GO LOOK FOR DATA
                  *    WAIT FOR THE 1793 TO ACKNOWLEDGE READY TO WRITE DATA
D860 C6   02              LDB    #2        DRQ MASK BIT
D862 E5   C4      LD768   BITB   ,U        IS 1793 READY FOR A BYTE? (DRQ SET)
D864 26   0A              BNE    LD778     BRANCH IF SO
D866 31   3F              LEAY   -1,Y      DECREMENT WAIT TIMER
D868 26   F6              BNE    LD768     KEEP WAITING FOR THE 1793 DRQ
D86A 7F   0982    LD770   CLR    NMIFLG    RESET NMI FLAG
D86D 1C   AF              ANDCC  #$AF      ENABLE FIRQ,IRQ
D86F 7E   D7E4            JMP    LD6EC     FORCE INTERRUPT, SET DRIVE NOT READY ERROR
                  *
                  * ********** WRITE A SECTOR **********
D872 E6   80      LD778   LDB    ,X+       GET A BYTE FROM RAM
D874 F7   FF4B            STB    FDCREG+3  SEND IT TO 1793
D877 B7   FF40            STA    DSKREG    RE-ENABLE 1793 TO HALT 6809
D87A 20   F4              BRA    LD778     SEND MORE DATA
                  *   WAIT FOR THE 1793 TO ACKNOWLEDGE READY TO READ DATA
D87C C6   02      LD782   LDB    #2        DRQ MASK BIT
D87E E5   C4      LD784   BITB   ,U        DOES THE 1793 HAVE A BYTE? (DRQ SET)
D880 26   04              BNE    LD78E     YES, GO READ A SECTOR
D882 31   3F              LEAY   -1,Y      DECREMENT WAIT TIMER
D884 26   F6              BNE    LD784     KEEP WAITING FOR 1793 DRQ
D886 20   E0              BRA    LD770     GENERATE DRIVE NOT READY ERROR
                  *
                  * ********** READ A SECTOR **********
D888 F6   FF4B    LD78E   LDB    FDCREG+3  GET DATA BYTE FROM 1793
D88B E7   80              STB    ,X+       PUT IT IN RAM
D88D B7   FF40            STA    DSKREG    RE-ENABLE 1793 TO HALT 6809
D890 20   F4              BRA    LD78E     KEEP GETTING DATA
                  *   BRANCH HERE ON COMPLETION OF SECTOR READ/WRITE
D892 1C   AF      LD798   ANDCC  #$AF      ENABLE IRQ, FIRQ
D894 B6   FF48            LDA    FDCREG    * GET STATUS & KEEP WRITE PROTECT RECORD
D897 84   7C              ANDA   #$7C      * TYPE/WRITE FAULT, RECORD NOT FOUND, CRC
                  *              * ERROR OR LOST DATA
```

```
D899 97   F0            STA    DCSTA     SAVE IN DSKCON STATUS
D89B 39                 RTS
                        *
                        * DSKCON OPERATION CODE JUMP VECTORS
D89C D7BD       LD7A2   FDB    LD6C5     RESTORE HEAD TO TRACK ZERO
D89E D7D5               FDB    LD6DD     NO OP - RETURN
D8A0 D7FD               FDB    LD705     READ SECTOR
D8A2 D800               FDB    LD708     WRITE SECTOR
                        *
                        *     DSKREG MASKS FOR DISK DRIVE MOTOR ENABLE
D8A4 01         LD7AA   FCB    1         DRIVE SEL 0
D8A5 02                 FCB    2         DRIVE SEL 1
D8A6 04                 FCB    4         DRIVE SEL 2
D8A7 40                 FCB    $40       DRIVE SEL 3
                        *
                        *    ******* NMI SERVICE  ******
                        *
D8A8 B6   0982  DNMISV  LDA    NMIFLG    GET NMI FLAG
D8AB 27   06            BEQ    LD7BB     RETURN IF NOT ACTIVE
D8AD BE   0983          LDX    DNMIVC    GET NEW RETURN VECTOR
D8B0 AF   6A            STX    10,S      STORE AT STACKED PC SLOT ON STACK
D8B2 7F   0982          CLR    NMIFLG    RESET NMI FLAG
D8B5 3B         LD7BB   RTI
                        *
                        ********** IRQ SERVICE  ********
D8B6 B6   FF03  DIRQSV  LDA    PIA0+3    63.5 MICRO SECOND OR 60 HZ INTERRUPT
D8B9 2A   F8            BPL    LD7BB     RETURN IF 63.5 MICROSECOND
D8BB B6   FF02          LDA    PIA0+2    RESET 60 HZ PIA FLAG
D8BE B6   0985          LDA    RDYTMR    GET TIMER
D8C1 27   0F            BEQ    LD7DA     BRANCH IF NOT ACTIVE
D8C3 4A                 DECA              DECREMENT THE TIMER
D8C4 B7   0985          STA    RDYTMR    SAVE IT
D8C7 26   09            BNE    LD7DA     BRANCH IF NOT TIME TO TURN OFF DISK MOTORS
D8C9 B6   0986          LDA    DRGRAM    = GET DSKREG IMAGE
D8CC 84   B0            ANDA   #$B0      = TURN ALL MOTORS OFF
D8CE B7   0986          STA    DRGRAM    = PUT IT BACK IN RAM IMAGE
D8D1 B7   FF40          STA    DSKREG    SEND TO CONTROL REGISTER (MOTORS OFF)
D8D4 7E   8955  LD7DA   JMP    L8955     JUMP TO EXTENDED BASIC'S IRQ HANDLER

DF00                    ORG    DOSBAS+$1F00
                        *
                        ** THIS IS THE CODE FOR THE DOS COMMAND
                        *
DF00 113F       DOSCOM  SWI3              DO A SOFTWARE INTERRUPT (#3)
DF02 0F   03            CLR    TMPLOC    RESET SECTOR COUNTER
DF04 CC   2600          LDD    #DOSBUF   RAM LOAD ADDRESS FOR SECTOR DATA
DF07 34   06            PSHS   B,A       SAVE RAM LOAD ADDRESS
DF09 BE   C006  LDF09   LDX    DSKVAR    POINT X TO DSKCON VARIABLES
DF0C 0C   03            INC    TMPLOC    INCREMENT SECTOR COUNTER
DF0E 96   03            LDA    TMPLOC    GET THE SECTOR COUNTER
DF10 81   12            CMPA   #18       LOADED IN 18 SECTORS? (ONE TRACK)
DF12 22   22            BHI    LDF36     YES - EXIT
DF14 A7   03            STA    3,X       NO - SAVE SECTOR NUMBER IN DSEC
DF16 CC   0200          LDD    #$0200    GET FDC OP CODE (READ) AND DRIVE # (0)
DF19 A7   84            STA    ,X        SAVE THEM IN DSKCON VARIABLES
DF1B 86   22            LDA    #34       GET TRACK NUMBER (34)
```

```
DF1D A7   02          STA   2,X       SAVE IT IN DSKCON VARIABLES TOO
DF1F 35   06          PULS  B,A       GET RAM LOAD ADDRESS
DF21 ED   04          STD   4,X       AND SAVE IT
DF23 8B   01          ADDA  #1        ADD 256 (ONE SECTOR) TO RAM LOAD ADDRESS
DF25 34   06          PSHS  B,A       SAVE NEW RAM LOAD ADDRESS
DF27 AD   9F C004     JSR   [DCNVEC]  GO READ A SECTOR
DF2B 6D   06          TST   6,X       CHECK FOR ERRORS
DF2D 27   DA          BEQ   LDF09     KEEP READING IF NONE
DF2F 35   06          PULS  B,A       PULL LOAD ADDRESS OFF OF THE STACK
DF31 C6   28          LDB   #2*20     'IO' ERROR
DF33 7E   AC46        JMP   LAC46     JUMP TO ERROR SERVICING ROUTINE
DF36 35   06    LDF36 PULS  B,A       PULL RAM LOAD ADDRESS OFF OF THE STACK
DF38 FC   2600        LDD   DOSBUF    GET FIRST TWO BYTES OF RAM DATA
DF3B 1083 4F53        CMPD  #'O*256+'S LOOK FOR 'OS' (OS9) AT START OF BUFFER
DF3F 1027 46BF        LBEQ  DOSBUF+2  IF 'OS' THEN BRANCH TO DATA LOADED IN RAM
DF43 7F   2600        CLR   DOSBUF    * OTHERWISE CLEAR THE FIRST TWO
DF46 7F   2601        CLR   DOSBUF+1  * BYTES OF RAM DATA
DF49 7E   A0E8        JMP   BAWMST    JUMP TO BASIC'S WARM START
                *
DF4C CC   3B3B  DOSINI LDD  #$3B3B    TWO RTI INSTRUCTIONS
DF4F FD   0100        STD   SW3VEC    *
DF52 FD   0102        STD   SW3VEC+2  * LOAD THE SWI2 AND SWI3 JUMP
DF55 FD   0104        STD   SW2VEC+1  * VECTORS WITH RTIS
DF58 39               RTS
```

APPENDIX C : TOKEN TABLES

```
                        *   THIS IS THE RESERVED WORD TABLE
                        *
AA66 46 4F D2     LAA66 FCC   'FO',$80+'R 80
AA69 47 CF              FCC   'G',$80+'O 81
AA6B 52 45 CD           FCC   'RE',$80+'M 82
AA6E A7                 FCB   ''+$80    83
AA6F 45 4C 53 C5        FCC   'ELS',$80+'E 84
AA73 49 C6              FCC   'I',$80+'F 85
AA75 44 41 54 C1        FCC   'DAT',$80+'A 86
AA79 50 52 49 4E        FCC   'PRIN',$80+'T 87
AA7D D4
AA7E 4F CE              FCC   'O',$80+'N 88
AA80 49 4E 50 55        FCC   'INPU',$80+'T 89
AA84 D4
AA85 45 4E C4           FCC   'EN',$80+'D 8A
AA88 4E 45 58 D4        FCC   'NEX',$80+'T 8B
AA8C 44 49 CD           FCC   'DI',$80+'M 8C
AA8F 52 45 41 C4        FCC   'REA',$80+'D 8D
AA93 52 55 CE           FCC   'RU',$80+'N 8E
AA96 52 45 53 54        FCC   'RESTOR',$80+'E 8F
AA9A 4F 52 C5
AA9D 52 45 54 55        FCC   'RETUR',$80+'N 90
AAA1 52 CE
AAA3 53 54 4F D0        FCC   'STO',$80+'P 91
AAA7 50 4F 4B C5        FCC   'POK',$80+'E 92
AAAB 43 4F 4E D4        FCC   'CON',$80+'T 93
AAAF 4C 49 53 D4        FCC   'LIS',$80+'T 94
AAB3 43 4C 45 41        FCC   'CLEA',$80+'R 95
AAB7 D2
AABB 4E 45 D7           FCC   'NE',$80+'W 96
AABB 43 4C 4F 41        FCC   'CLOA',$80+'D 97
AABF C4
AAC0 43 53 41 56        FCC   'CSAV',$80+'E 98
AAC4 C5
AAC5 4F 50 45 CE        FCC   'OPE',$80+'N 99
AAC9 43 4C 4F 53        FCC   'CLOS',$80+'E 9A
AACD C5
AACE 4C 4C 49 53        FCC   'LLIS',$80+'T 9B
AAD2 D4
AAD3 53 45 D4           FCC   'SE',$80+'T 9C
AAD6 52 45 53 45        FCC   'RESE',$80+'T 9D
AADA D4
AADB 43 4C D3           FCC   'CL',$80+'S 9E
AADE 4D 4F 54 4F        FCC   'MOTO',$80+'R 9F
AAE2 D2
AAE3 53 4F 55 4E        FCC   'SOUN',$80+'D A0
AAE7 C4
AAE8 41 55 44 49        FCC   'AUDI',$80+'O A1
AAEC CF
AAED 45 58 45 C3        FCC   'EXE',$80+'C A2
AAF1 53 4B 49 50        FCC   'SKIP',$80+'F A3
AAF5 C6
AAF6 54 41 42 A8        FCC   'TAB',$80+'( A4
AAFA 54 CF              FCC   'T',$80+'O A5
AAFC 53 55 C2           FCC   'SU',$80+'B A6
AAFF 54 48 45 CE        FCC   'THE',$80+'N A7
AB03 4E 4F D4           FCC   'NO',$80+'T A8
AB06 53 54 45 D0        FCC   'STE',$80+'P A9
AB0A 4F 46 C6           FCC   'OF',$80+'F AA
```

```
                        *   DISPATCH TABLE FOR COMMANDS
                        *
AB67 AD47         LAB67 FDB   LAD47     FOR 80
AB69 AE86               FDB   LAE86     GO 81
AB6B AEE3               FDB   LAEE3     REM 82
AB6D AEE3               FDB   LAEE3     REM 83
AB6F AEE3               FDB   LAEE3     ELSE 84
AB71 AF14               FDB   LAF14     IF 85
AB73 AEE0               FDB   LAEE0     DATA 86
AB75 B8F7               FDB   LB8F7     PRINT 87
AB77 AF42               FDB   LAF42     ON 88
AB79 AFF5               FDB   LAFF5     INPUT 89
AB7B AE02               FDB   LAE02     END 8A
AB7D B0F8               FDB   LB0F8     NEXT 8B
AB7F B34E               FDB   LB34E     DIM 8C
AB81 B046               FDB   LB046     READ 8D
AB83 AE75               FDB   LAE75     RUN 8E
AB85 ADE4               FDB   LADE4     RESTORE 8F
AB87 AEC0               FDB   LAEC0     RETURN 90
AB89 AE09               FDB   LAE09     STOP 91
AB8B B757               FDB   LB757     POKE 92
AB8D AE30               FDB   LAE30     CONT 93
AB8F B764               FDB   LB764     LIST 94
AB91 AE41               FDB   LAE41     CLEAR 95
AB93 AD17               FDB   LAD17     NEW 96
AB95 A498               FDB   LA498     CLOAD 97
AB97 A44C               FDB   LA44C     CSAVE 98
AB99 A5F6               FDB   LA5F6     OPEN 99
AB9B A416               FDB   LA416     CLOSE 9A
AB9D B75E               FDB   LB75E     LLIST 9B
AB9F A880               FDB   LA880     SET 9C
ABA1 A8B1               FDB   LA8B1     RESET 9D
ABA3 A910               FDB   LA910     CLS 9E
ABA5 A7BD               FDB   LA7BD     MOTOR 9F
ABA7 A94B               FDB   LA94B     SOUND A0
ABA9 A990               FDB   LA990     AUDIO A1
ABAB A53E               FDB   LA53E     EXEC A2
ABAD A5EC               FDB   LA5EC     SKIPF A3
*
* THIS TABLE CONTAINS PRECEDENCES AND DISPATCH ADDRESSES
* FOR ARITHMETIC AND LOGICAL OPERATORS
*
AA51 79           LAA51 FCB   $79
AA52 B9C5               FDB   LB9C5     +
AA54 79                 FCB   $79
AA55 B9BC               FDB   LB9BC     -
AA57 7B                 FCB   $7B
AA58 BACC               FDB   LBACC     *
AA5A 7B                 FCB   $7B
AA5B BB91               FDB   LBB91     /
AA5D 7F                 FCB   $7F
AA5E 011D               FDB   $11D      EXPONENTIATION
AA60 50                 FCB   $50
AA61 B2D5               FDB   LB2D5     AND
AA63 46                 FCB   $46
AA64 B2D4               FDB   LB2D4     OR
```

```
AB0D AB                     FCB   '++$80    AB
AB0E AD                     FCB   '-+$80    AC
AB0F AA                     FCB   '*+$80    AD
AB10 AF                     FCB   '/+$80    AE
AB11 DE                     FCB   '^+$80    AF
AB12 41 4E C4               FCC   'AN',$80+'D B0
AB15 4F D2                  FCC   'O',$80+'R B1
AB17 BE                     FCB   '>+$80    B2
AB18 BD                     FCB   '=+$80    B3
AB19 BC                     FCB   '<+$80    B4
```

```
*    RESERVED WORD TABLE FOR EXTENDED BASIC

8183 44 45 CC      L8183    FCC   'DE',$80+'L B5
8186 45 44 49 D4            FCC   'EDI',$80+'T B6
818A 54 52 4F CE            FCC   'TRO',$80+'N B7
818E 54 52 4F 46            FCC   'TROF',$80+'F B8
8192 C6
8193 44 45 C6               FCC   'DE',$80+'F B9
8196 4C 45 D4               FCC   'LE',$80+'T BA
8199 4C 49 4E C5            FCC   'LIN',$80+'E BB
819D 50 43 4C D3            FCC   'PCL',$80+'S BC
81A1 50 53 45 D4            FCC   'PSE',$80+'T BD
81A5 50 52 45 53            FCC   'PRESE',$80+'T BE
81A9 45 D4
81AB 53 43 52 45            FCC   'SCREE',$80+'N BF
81AF 45 CE
81B1 50 43 4C 45            FCC   'PCLEA',$80+'R C0
81B5 41 D2
81B7 43 4F 4C 4F            FCC   'COLO',$80+'R C1
81BB D2
81BC 43 49 52 43            FCC   'CIRCL',$80+'E C2
81C0 4C C5
81C2 50 41 49 4E            FCC   'PAIN',$80+'T C3
81C6 D4
81C7 47 45 D4               FCC   'GE',$80+'T C4
81CA 50 55 D4               FCC   'PU',$80+'T C5
81CD 44 52 41 D7            FCC   'DRA',$80+'W C6
81D1 50 43 4F 50            FCC   'PCOP',$80+'Y C7
81D5 D9
81D6 50 4D 4F 44            FCC   'PMOD',$80+'E C8
81DA C5
81DB 50 4C 41 D9            FCC   'PLA',$80+'Y C9
81DF 44 4C 4F 41            FCC   'DLOA',$80+'D CA
81E3 C4
81E4 52 45 4E 55            FCC   'RENU',$80+'M CB
81E8 CD
81E9 46 CE                  FCC   'F',$80+'N CC
81EB 55 53 49 4E            FCC   'USIN',$80+'G CD
81EF C7
```

```
*    DISPATCH TABLE FOR EXTENDED BASIC COMMANDS

81F0 8970          L81F0    FDB   L8970     DEL B5
81F2 8533                   FDB   L8533     EDIT B6
81F4 86A7                   FDB   L86A7     TRON B7
81F6 86A8                   FDB   L86A8     TROFF B8
81F8 8871                   FDB   L8871     DEF B9
81FA AF89                   FDB   LAF89     LET BA
81FC 93BB                   FDB   L93BB     LINE BB
81FE 9532                   FDB   L9532     PCLS BC
8200 9361                   FDB   L9361     PSET BD
8202 9365                   FDB   L9365     PRESET BE
8204 9670                   FDB   L9670     SCREEN BF
8206 968B                   FDB   L968B     PCLEAR C0
8208 9546                   FDB   L9546     COLOR C1
820A 9E9D                   FDB   L9E9D     CIRCLE C2
820C 98EC                   FDB   L98EC     PAINT C3
820E 9755                   FDB   L9755     GET C4
8210 9758                   FDB   L9758     PUT C5
8212 9CB6                   FDB   L9CB6     DRAW C6
8214 9723                   FDB   L9723     PCOPY C7
8216 9621                   FDB   L9621     PMODE C8
8218 9A22                   FDB   L9A22     PLAY C9
821A 8C18                   FDB   L8C18     DLOAD CA
821C 8A09                   FDB   L8A09     RENUM CB
```

```
* COMMAND TABLE FOR DISK BASIC

VER 1.1

C192 44 49 D2      LC17F   FCC     'DI',$80+'R CE
C195 44 52 49 56           FCC     'DRIV',$80+'E CF
C19A 46 49 45 4C           FCC     'FIEL',$80+'D D0
C19F 46 49 4C 45           FCC     'FILE',$80+'S D1
C1A4 4B 49 4C CC           FCC     'KIL',$80+'L D2
C1A8 4C 4F 41 C4           FCC     'LOA',$80+'D D2
C1AC 4C 53 45 D4           FCC     'LSE',$80+'T D4
C1B0 4D 45 52 47           FCC     'MERG',$80+'E D5
C1B5 52 45 4E 41           FCC     'RENAM',$80+'E D6
C1BB 52 53 45 D4           FCC     'RSE',$80+'T D7
C1BF 53 41 56 CE           FCC     'SAV',$80+'E D8
C1C3 57 52 49 54           FCC     'WRIT',$80+'E D9
C1C8 56 45 52 49           FCC     'VERIF',$80+'Y DA
C1CE 55 4E 4C 4F           FCC     'UNLOA',$80+'D DB
C1D4 44 53 4B 49           FCC     'DSKIN',$80+'I DC
C1DA 42 41 43 4B           FCC     'BACKU',$80+'P DD
C1E0 43 4F 50 D9           FCC     'COP',$80+'Y DE
C1E4 44 53 4B 49           FCC     'DSKI',$80+'$ DF
C1E9 44 53 4B 4F           FCC     'DSKO',$80+'$ E0
                *
C1EE 44 4F D3              FCC     'DO',$80+'S E1
```

```
* DISPATCH TABLE FOR DISK BASIC COMMANDS

VER 1.1

C1F1 CCA9          LC1DB   FDB     DIR         DIR
C1F3 CEC5                  FDB     DRIVE       DRIVE
C1F5 D0BC                  FDB     FIELD       FIELD
C1F7 D15C                  FDB     FILES       FILES
C1F9 C6EF                  FDB     KILL        KILL
C1FB CA48                  FDB     LOAD        LOAD
C1FD D102                  FDB     LSET        LSET
C1FF CA39                  FDB     MERGE       MERGE
C201 D01B                  FDB     RENAME      RENAME
C203 D101                  FDB     RSET        RSET
C205 C9E0                  FDB     SAVE        SAVE
C207 D066                  FDB     WRITE       WRITE
C209 D74E                  FDB     VERIFY      VERIFY
C20B D233                  FDB     UNLOAD      UNLOAD
C20D D599                  FDB     DSKINI      DSKINI
C20F D262                  FDB     BACKUP      BACKUP
C211 D3B9                  FDB     COPY        COPY
C213 D4ED                  FDB     DSKI        DSKI$
C215 D5E2                  FDB     DSKO        DSKO$
                *
C217 D6EC                  FDB     DOS         DOS
```

```
*
*   TOKENS FOR THE INTRINSIC FUNCTIONS ARE PRECEEDED BY $FF
*

AB1A 53 47 CE      LAB1A   FCC     'SG',$80+'N 80
AB1D 49 4E D4              FCC     'IN',$80+'T 81
AB20 41 42 D3              FCC     'AB',$80+'S 82
AB23 55 53 D2              FCC     'US',$80+'R 83
AB26 52 4E C4              FCC     'RN',$80+'D 84
AB29 53 49 CE              FCC     'SI',$80+'N 85
AB2C 50 45 45 CB           FCC     'PEE',$80+'K 86
AB30 4C 45 CE              FCC     'LE',$80+'N 87
AB33 53 54 52 A4           FCC     'STR',$80+'$ 88
AB37 56 41 CC              FCC     'VA',$80+'L 89
AB3A 41 53 C3              FCC     'AS',$80+'C 8A
AB3D 43 48 52 A4           FCC     'CHR',$80+'$ 8B
AB41 45 4F C6              FCC     'EO',$80+'F 8C
AB44 4A 4F 59 53           FCC     'JOYST',$80+'K 8D
AB48 54 CB
AB4A 4C 45 46 54           FCC     'LEFT',$80+'$ 8E
AB4E A4
AB4F 52 49 47 48           FCC     'RIGHT',$80+'$ 8F
AB53 54 A4
```

```
*
*   DISPATCH TABLE FOR INTRINSIC FUNCTIONS

AA29 BC7A          LAA29   FDB     LBC7A       SGN 80
AA2B BCEE                  FDB     LBCEE       INT 81
AA2D BC93                  FDB     LBC93       ABS 82
AA2F 0112                  FDB     $112        USR 83
AA31 BF1F                  FDB     LBF1F       RND 84
AA33 BF78                  FDB     LBF78       SIN 85
AA35 B750                  FDB     LB750       PEEK 86
AA37 B6B1                  FDB     LB681       LEN 87
AA39 B4FD                  FDB     LB4FD       STR$ 88
AA3B B716                  FDB     LB716       VAL 89
AA3D B6A0                  FDB     LB6A0       ASC 8A
AA3F B6BC                  FDB     LB68C       CHR$ 8B
AA41 A5CE                  FDB     LA5CE       EOF 8C
AA43 A9C6                  FDB     LA9C6       JOYSTK 8D
AA45 B6AB                  FDB     LB6AB       LEFT$ 8E
AA47 B6CB                  FDB     LB6C8       RIGHT$ 8F
AA49 B6CF                  FDB     LB6CF       MID$ 90
AA4B A8F5                  FDB     LA8F5       POINT 91
AA4D A564                  FDB     LA564       INKEY$ 92
AA4F B4EE                  FDB     LB4EE       MEM 93
```

```
AB55 4D 49 44 A4          FCC   'MID',$80+'$ 90
AB59 50 4F 49 4E          FCC   'POIN',$80+'T 91
AB5D D4
AB5E 49 4E 4B 45          FCC   'INKEY',$80+'$ 92
AB62 59 A4
AB64 4D 45 CD             FCC   'ME',$80+'M 93
```

```
* INTRINSIC FUNCTION FOR EXTENDED BASIC
* TOKENS ARE PRECEEDED WITH AN $FF BYTE

821E 41 54 CE      L821E  FCC   'AT',$80+'N 94
8221 43 4F D3             FCC   'CO',$80+'S 95
8224 54 41 CE             FCC   'TA',$80+'N 96
8227 45 58 D0             FCC   'EX',$80+'P 97
822A 46 49 D8             FCC   'FI',$80+'X 98
822D 4C 4F C7             FCC   'LO',$80+'G 99
8230 50 4F D3             FCC   'PO',$80+'S 9A
8233 53 51 D2             FCC   'SQ',$80+'R 9B
8236 48 45 58 A4          FCC   'HEX',$80+'$ 9C
823A 56 41 52 50          FCC   'VARPT',$80+'R 9D
823E 54 D2
8240 49 4E 53 54          FCC   'INST',$80+'R 9E
8244 D2
8245 54 49 4D 45          FCC   'TIME',$80+'R 9F
8249 D2
824A 50 50 4F 49          FCC   'PPOIN',$80+'T A0
824E 4E D4
8250 53 54 52 49          FCC   'STRING',$80+'$ A1
8254 4E 47 A4
```

```
* INTRINSIC FUNCTIONS FOR DISK BASIC
  TOKENS ARE PRECEEDED BY $FF    VER 1.1

C219 43 4E CE      LC201  FCC   'CN',$80+'N A2
C21C 46 52 45 C5          FCC   'FRE',$80+'E A3
C220 4C 4F C3             FCC   'LO',$80+'C A4
C223 4C 4F C6             FCC   'LO',$80+'F A5
C226 4D 4B 4E A4          FCC   'MKN',$80+'$ A6
C22A 41 D3                FCC   'A',$80+'S A7
```

```
* DISPATCH TABLE FOR EXTENDED BASIC INTRINSIC FUNCTIONS

8257 83B0          L8257  FDB   L83B0     ATN 94
8259 8378                 FDB   L8378     COS 95
825B 8381                 FDB   L8381     TAN 96
825D 84F2                 FDB   L84F2     EXP 97
825F 8524                 FDB   L8524     FIX 98
8261 8446                 FDB   L8446     LOG 99
8263 86AC                 FDB   L86AC     POS 9A
8265 8480                 FDB   L8480     SQR 9B
8267 8BDD                 FDB   L8BDD     HEX$ 9C
8269 86BE                 FDB   L86BE     VARPTR 9D
826B 877E                 FDB   L877E     INSTR 9E
826D 8968                 FDB   L8968     TIMER 9F
826F 9339                 FDB   L9339     PPOINT A0
8271 874E                 FDB   L874E     STRING$ A1
```

```
* DISPATCH TABLE FOR DISK BASIC INTRINSIC FUNCTIONS
                VER 1.1

C22C CDF4          LC214  FDB   LCD1A     CVN A2
C22E CE9C                 FDB   LCDC0     FREE A3
C230 CE10                 FDB   LCD36     LOC A4
C232 CE37                 FDB   LCD5B     LOF A5
C234 CE02                 FDB   LCD28     MKN$ A6
C236 B277                 FDB   LB277     AS A7
                                          (SET TO SYNTAX ERROR)
```

MOTOROLA
SEMICONDUCTORS
3501 ED BLUESTEIN BLVD., AUSTIN, TEXAS 78721

MC6809E

8-BIT MICROPROCESSING UNIT

The MC6809E 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 M6800 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 MC6809E has the most complete set of addressing modes available on any 8-bit microprocessor today.

The MC6809E has hardware and software features which make it an ideal processor for higher level language execution or standard controller applications. External clock inputs are provided to allow synchronization with peripherals, systems, or other MPUs.

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

HARDWARE FEATURES

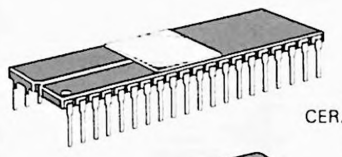
- External Clock Inputs, E and Q, Allow Synchronization
- TSC Input Controls Internal Bus Buffers
- LIC Indicates Opcode Fetch
- AVMA Allows Efficient Use of Common Resources in a Multiprocessor System
- BUSY is a Status Line for Multiprocessing
- Fast Interrupt Request Input Stacks Only Condition Code Register and Program Counter
- Interrupt Acknowledge Output Allows Vectoring By Devices
- Sync Acknowledge Output Allows for Synchronization to External Event
- Single Bus-Cycle $\overline{\text{RESET}}$
- Single 5-Volt Supply Operation
- $\overline{\text{NMI}}$ Inhibited After $\overline{\text{RESET}}$ Until After First Load of Stack Pointer
- Early Address Valid Allows Use With Slower Memories
- Early Write Data for Dynamic Memories

SOFTWARE FEATURES

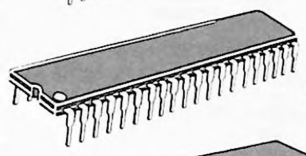
- 10 Addressing Modes
  - M6800 Upward Compatible Addressing Modes
  - Direct Addressing Anywhere in Memory Map
  - Long Relative Branches
  - Program Counter Relative
  - True Indirect Addressing
  - Expanded Indexed Addressing
    0-, 5-, 8-, or 16-Bit Constant Offsets
    8- or 16-Bit Accumulator Offsets
    Auto-Increment/Decrement by 1 or 2
- Improved Stack Manipulation
- 1464 Instruction with Unique Addressing Modes
- 8 × 8 Unsigned Multiply
- 16-Bit Arithmetic
- Transfer/Exchange All Registers
- Push/Pull Any Registers or Any Set of Registers
- Load Effective Address

HMOS
(HIGH-DENSITY N-CHANNEL, SILICON-GATE)

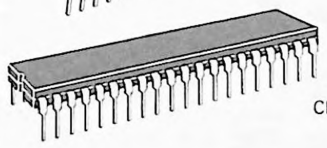
8-BIT
MICROPROCESSING
UNIT



L SUFFIX
CERAMIC PACKAGE
CASE 715

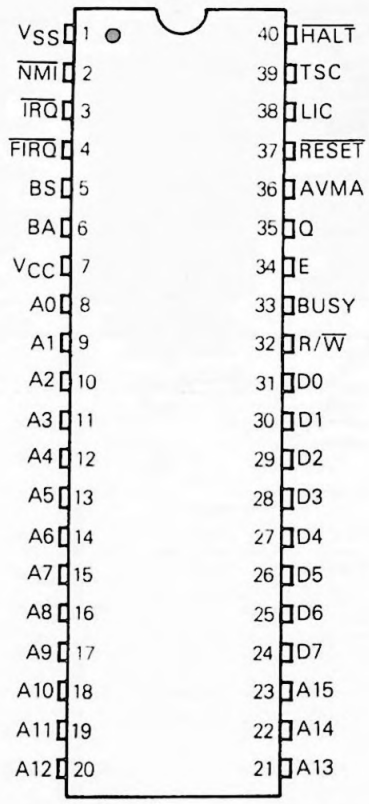


P SUFFIX
PLASTIC PACKAGE
CASE 711



S SUFFIX
CERDIP PACKAGE
CASE 734

PIN ASSIGNMENTS



©MOTOROLA INC., 1982

MAXIMUM RATINGS

Rating	Symbol	Value	Unit
Supply Voltage	V_{CC}	−0.3 to +7.0	V
Input Voltage	V_{in}	−0.3 to +7.0	V
Operating Temperature Range MC6809E, MC68A09E, MC68B09E MC6809EC, MC68A09EC, MC68B09EC	T_A	T_L to T_H 0 to +70 −40 to +85	°C
Storage Temperature Range	T_{stg}	−55 to +150	°C

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.

Reliability of operation is enhanced if unused inputs are tied to an appropriate logic voltage level (e.g., either V_{SS} or V_{CC}).

THERMAL CHARACTERISTICS

Characteristic	Symbol	Value	Unit
Thermal Resistance Ceramic Cerdip Plastic	θ_{JA}	 50 60 100	°C/W

POWER CONSIDERATIONS

The average chip-junction temperature, T_J, in °C can be obtained from:

$$T_J = T_A + (P_D \cdot \theta_{JA}) \quad (1)$$

Where:

$T_A \equiv$ Ambient Temperature, °C

$\theta_{JA} \equiv$ Package Thermal Resistance, Junction-to-Ambient, °C/W

$P_D \equiv P_{INT} + P_{PORT}$

$P_{INT} \equiv I_{CC} \times V_{CC}$, Watts — Chip Internal Power

$P_{PORT} \equiv$ Port Power Dissipation, Watts — User Determined

For most applications $P_{PORT} \ll P_{INT}$ and can be neglected. P_{PORT} may become significant if the device is configured to drive Darlington bases or sink LED loads.

An approximate relationship between P_D and T_J (if P_{PORT} is neglected) is:

$$P_D = K \div (T_J + 273°C) \quad (2)$$

Solving equations 1 and 2 for K gives:

$$K = P_D \cdot (T_A + 273°C) + \theta_{JA} \cdot P_D^2 \quad (3)$$

Where K is a constant pertaining to the particular part. K can be determined from equation 3 by measuring P_D (at equilibrium) for a known T_A. Using this value of K the values of P_D and T_J can be obtained by solving equations (1) and (2) iteratively for any value of T_A.

DC ELECTRICAL CHARACTERISTICS (V_{CC} = 5.0 V ±5%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H unless otherwise noted)

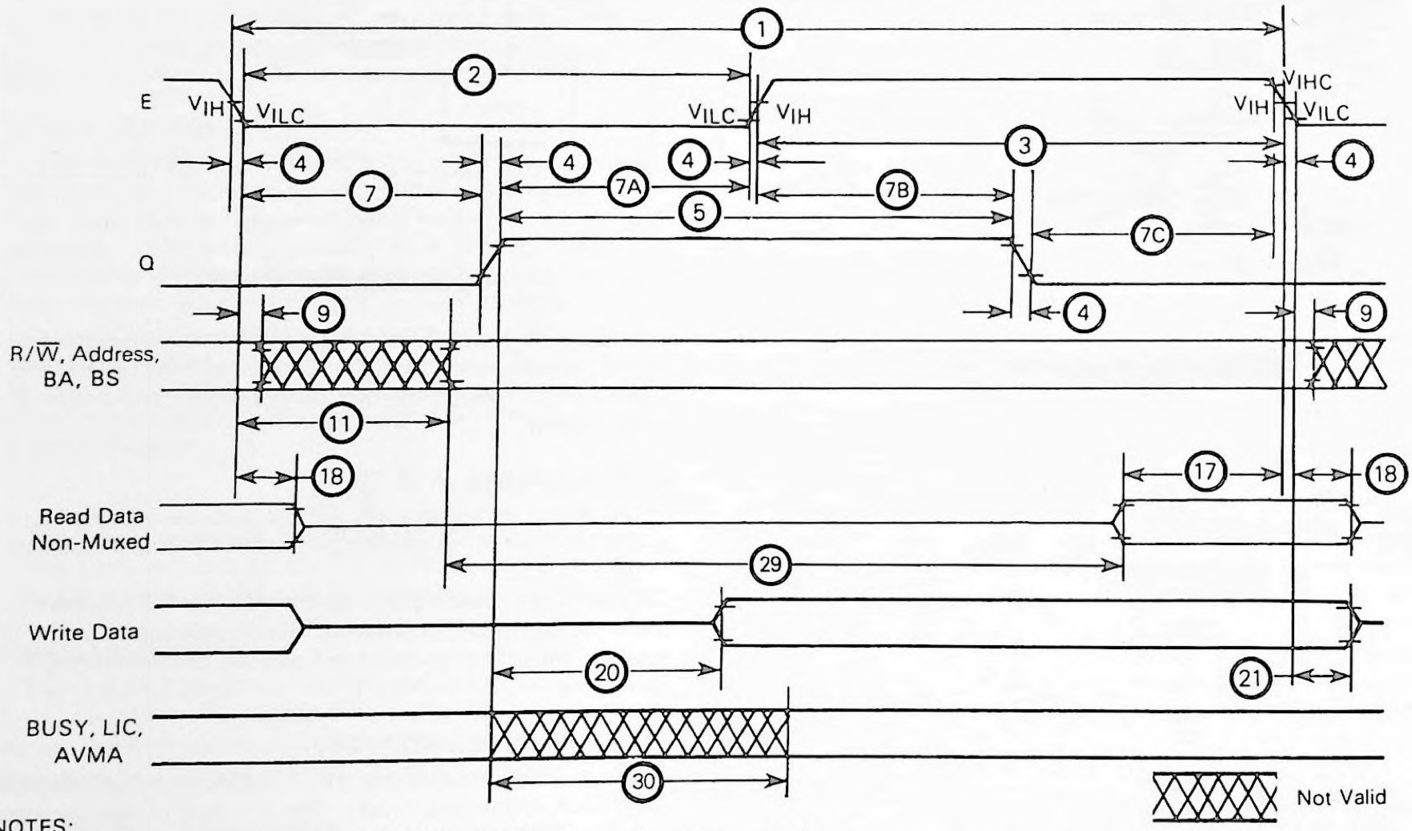
Characteristic		Symbol	Min	Typ	Max	Unit
Input High Voltage	Logic, Q, $\overline{RESET}$	V_{IH}	V_{SS} + 2.0	–	V_{CC}	V
		V_{IHR}	V_{SS} + 4.0	–	V_{CC}	
	E	V_{IHC}	V_{CC} − 0.75	–	V_{CC} + 0.3	
Input Low Voltage	Logic, Q, $\overline{RESET}$	V_{IL}	V_{SS} − 0.3	–	V_{SS} + 0.8	V
	E	V_{ILC}	V_{SS} − 0.3	–	V_{SS} + 0.4	
Input Leakage Current (V_{in} = 0 to 5.25 V, V_{CC} = max)	Logic, Q, $\overline{RESET}$	I_{in}	–	–	2.5	μA
	E		–	–	100	
dc Output High Voltage		V_{OH}				V
(I_{Load} = −205 μA, V_{CC} = min)	D0-D7		V_{SS} + 2.4	–	–	
(I_{Load} = −145 μA, V_{CC} = min)	A0-A15, R/$\overline{W}$		V_{SS} + 2.4	–	–	
(I_{Load} = −100 μA, V_{CC} = min)	BA, BS, LIC, AVMA, BUSY		V_{SS} + 2.4	–	–	
dc Output Low Voltage (I_{Load} = 2.0 mA, V_{CC} = min)		V_{OL}	–	–	V_{SS} + 0.5	V
Internal Power Dissipation (Measured at $T_A = T_L$ in Steady State Operation)		P_{INT}	–	–	1.0	W
Capacitance* (V_{in} = 0, T_A = 25°C, f = 1.0 MHz)		C_{in}				pF
	D0-D7, Logic Inputs, Q, $\overline{RESET}$		–	10	15	
	E		–	30	50	
	A0-A15, R/$\overline{W}$, BA, BS, LIC, AVMA, BUSY	C_{out}	–	10	15	pF
Frequency of Operation (E and Q Inputs)	MC6809E	f	0.1	–	1.0	MHz
	MC68A09E		0.1	–	1.5	
	MC68B09E		0.1	–	2.0	
Hi-Z (Off State) Input Current (V_{in} = 0.4 to 2.4 V, V_{CC} = max)	D0-D7	I_{TSI}	–	2.0	10	μA
	A0-A15, R/$\overline{W}$		–	–	100	

* Capacitances are periodically tested rather than 100% tested.

BUS TIMING CHARACTERISTICS (See Notes 1, 2, 3, and 4)

Ident. Number	Characteristics	Symbol	MC6809E		MC68A09E		MC68B09E		Unit
			Min	Max	Min	Max	Min	Max	
1	Cycle Time	t_{cyc}	1.0	10	0.667	10	0.5	10	μs
2	Pulse Width, E Low	PW_{EL}	450	9500	295	9500	210	9500	ns
3	Pulse Width, E High	PW_{EH}	450	9500	280	9500	220	9500	ns
4	Clock Rise and Fall Time	t_r, t_f	–	25	–	25	–	20	ns
5	Pulse Width, Q High	PW_{QH}	450	9500	280	9500	220	9500	ns
7	Delay Time, E to Q Rise	t_{EQ1}	200	–	130	–	100	–	ns
7A	Delay Time, Q High to E Rise	t_{EQ2}	200	–	130	–	100	–	ns
7B	Delay Time, E High to Q Fall	t_{EQ3}	200	–	130	–	100	–	ns
7C	Delay Time, Q High to E Fall	t_{EQ4}	200	–	130	–	100	–	ns
9	Address Hold Time	t_{AH}	20	–	20	–	20	–	ns
11	Address Delay Time from E Low (BA, BS, R/$\overline{W}$)	t_{AD}	–	200	–	140	–	110	ns
17	Read Data Setup Time	t_{DSR}	80	–	60	–	40	–	ns
18	Read Data Hold Time	t_{DHR}	10	–	10	–	10	–	ns
20	Data Delay Time from Q	t_{DDQ}	–	200	–	140	–	110	ns
21	Write Data Hold Time	t_{DHW}	30	–	30	–	30	–	ns
29	Usable Access Time	t_{ACC}	695	–	440	–	330	–	ns
30	Control Delay Time	t_{CD}	–	300	–	250	–	200	ns
	Interrupts, $\overline{HALT}$, $\overline{RESET}$, and TSC Setup Time (Figures 6, 7, 8, 9, 12, and 13)	t_{PCS}	200	–	140	–	110	–	ns
	TSC Drive to Valid Logic Level (Figure 13)	t_{TSV}	–	210	–	150	–	120	ns
	TSC Release MOS Buffers to High Impedance (Figure 13)	t_{TSR}	–	200	–	140	–	110	ns
	TSC Hi-Z Delay Time (Figure 13)	t_{TSD}	–	120	–	85	–	80	ns
	Processor Control Rise and Fall Time (Figure 7)	t_{PCr}, t_{PCf}	–	100	–	100	–	100	ns

FIGURE 1 — READ/WRITE DATA TO MEMORY OR PERIPHERALS TIMING DIAGRAM



NOTES:
1. Voltage levels shown are $V_L \leq 0.4$ V, $V_H \geq 2.4$ V, unless otherwise specified.
2. Measurement points shown are 0.8 V and 2.0 V, unless otherwise specified.
3. Hold time (⑨) for BA and BS is not specified.
4. Usable access time is computed by: 1 – 4 – 11 max – 17.

FIGURE 2 — EXPANDED BLOCK DIAGRAM

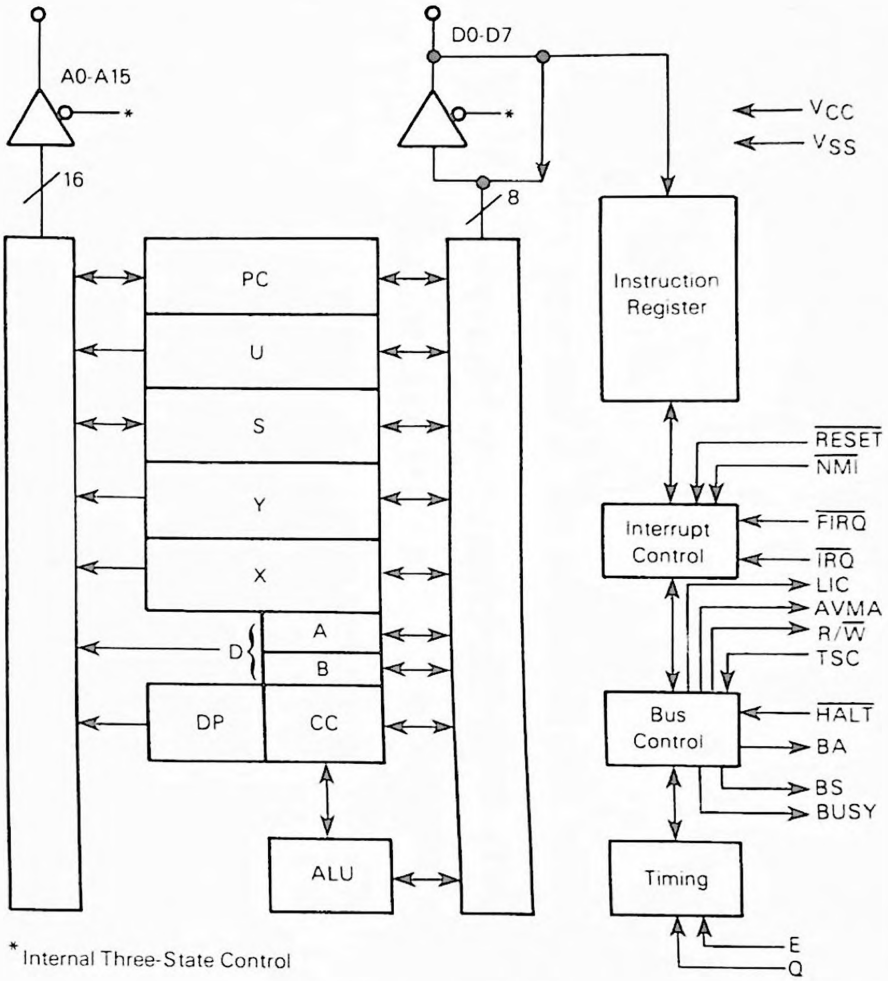
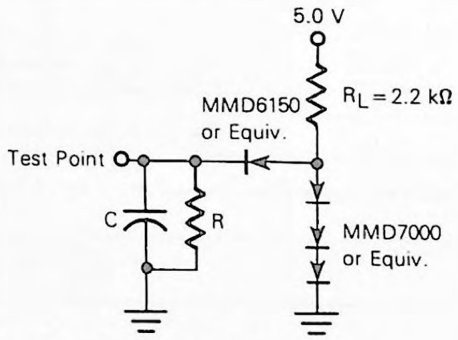


FIGURE 3 — BUS TIMING TEST LOAD



C = 30 pF for BA, BS, LIC, AVMA, BUSY
130 pF for D0-D7
90 pF for A0-A15, R/$\overline{W}$

R = 11.7 kΩ for D0-D7
16.5 kΩ for A0-A15, R/$\overline{W}$
24 kΩ for BA, BS, LIC, AVMA, BUSY

PROGRAMMING MODEL

As shown in Figure 4, the MC6809E 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)

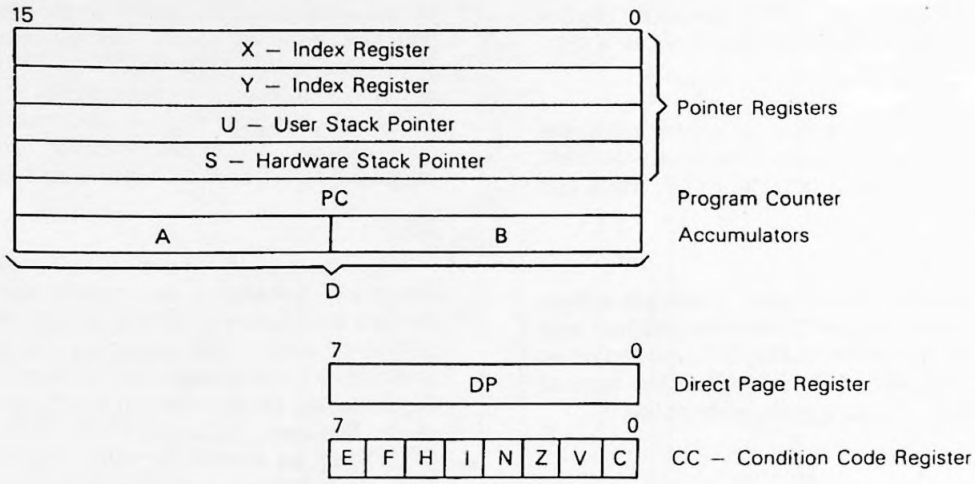
The A and B registers are general purpose accumulators which are used for arithmetic calculations and manipulation of data.

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 MC6809E 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 ensure M6800 compatibility, all bits of this register are cleared during processor reset.

FIGURE 4 — 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 POINTER (U, S)

The hardware stack pointer (S) is used automatically by the processor during subroutine calls and interrupts. The user stack pointer (U) is controlled exclusively by the programmer. This allows arguments to be passed to and from subroutines with ease. The U register is frequently used as a stack marker. 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 MC6809E to be used efficiently as a stack processor, greatly enhancing its ability to support higher level languages and modular programming.

NOTE

The stack pointers of the MC6809E point to the top of the stack in contrast to the MC6800 stack pointer, which pointed to the next free location on stack.

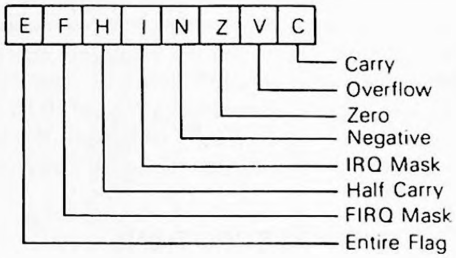
PROGRAM COUNTER

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 4.

FIGURE 5 — 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) and 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 twos 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 preceding operation. Thus, a negative twos complement result will leave N set to a one.

BIT 4 (I)

Bit 4 is the $\overline{IRQ}$ mask bit. The processor will not recognize interrupts from the $\overline{IRQ}$ line if this bit is set to a one. $\overline{NMI}$, $\overline{FIRQ}$, $\overline{IRQ}$, $\overline{RESET}$, 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 $\overline{FIRQ}$ mask bit. The processor will not recognize interrupts from the $\overline{FIRQ}$ line if this bit is a one. $\overline{NMI}$, $\overline{FIRQ}$, SWI, and $\overline{RESET}$ all set F to a one. $\overline{IRQ}$, 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.

PIN DESCRIPTIONS

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 $+5.0\ V \pm 5\%$.

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 $FFFF_{16}$, $R/\overline{W} = 1$, and $BS = 0$; this is a "dummy access" or $\overline{VMA}$ cycle. All address bus drivers are made high-impedance when output bus available (BA) is high or when TSC is asserted. Each pin will drive one Schottky TTL load or four LSTTL loads and 90 pF.

DATA BUS (D0-D7)

These eight pins provide communication with the system bidirectional data bus. Each pin will drive one Schottky TTL load or four LSTTL loads and 130 pF.

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

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. $R/\overline{W}$ is made high impedance when BA is high or when TSC is asserted.

$\overline{RESET}$

A low level on this Schmitt-trigger input for greater than one bus cycle will reset the MPU, as shown in Figure 6. The reset vectors are fetched from locations $FFFE_{16}$ and $FFFF_{16}$ (Table 1) when interrupt acknowledge is true, ($\overline{BA} \bullet BS = 1$). During initial power on, the reset line should be held low until the clock input signals are fully operational.

Because the MC6809E $\overline{RESET}$ 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 ensures that all peripherals are out of the reset state before the processor.

$\overline{HALT}$

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 state. While halted, the MPU will not respond to external real-time requests ($\overline{FIRQ}$, $\overline{IRQ}$) although $\overline{NMI}$ or $\overline{RESET}$ will be latched for later response. During the halt state, Q and E should continue to run normally. A halted state ($\overline{BA} \bullet BS = 1$) can be achieved by pulling $\overline{HALT}$ low while $\overline{RESET}$ is still low. See Figure 7.

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. When BA goes low, a dead cycle will elapse before the MPU acquires the bus. BA will not be asserted when TSC is active, thus allowing dead cycle consistency.

The bus status output signal, when decoded with BA, represents the MPU state (valid with leading edge of Q).

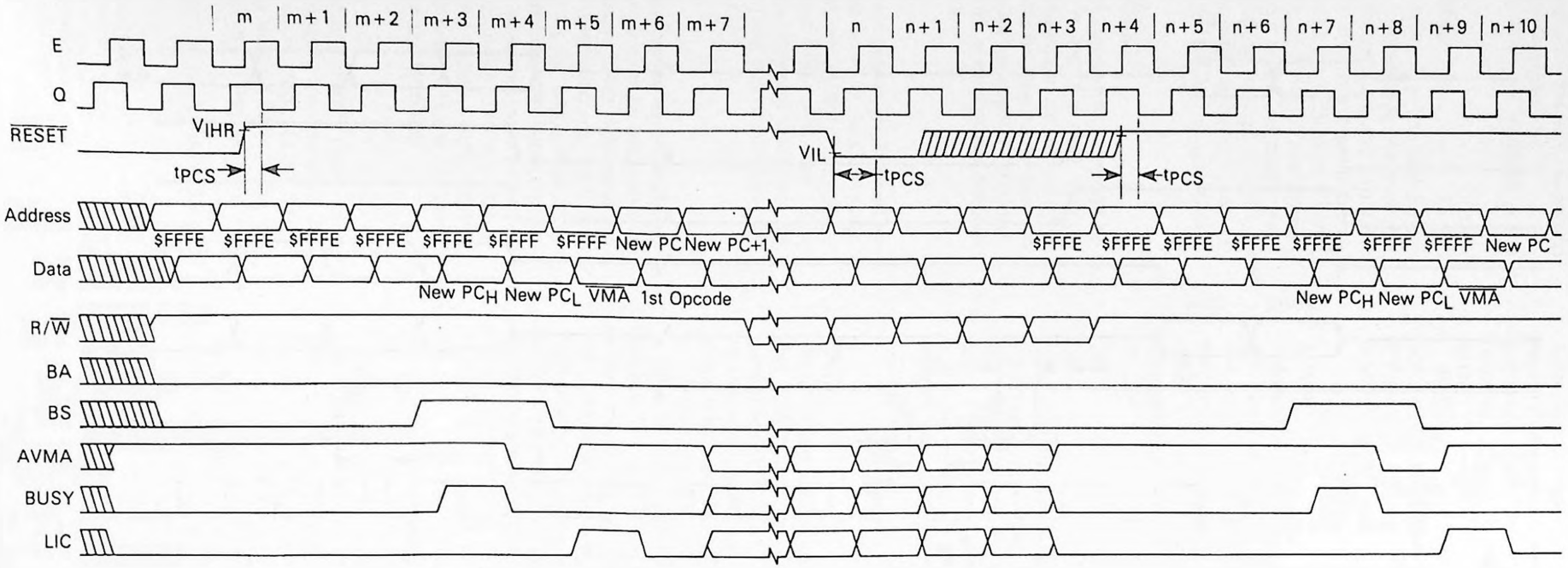
MPU State		MPU State Definition
BA	BS	
0	0	Normal (Running)
0	1	Interrupt or Reset Acknowledge
1	0	Sync Acknowledge
1	1	Halt Acknowledge

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

TABLE 1 — MEMORY MAP FOR INTERRUPT VECTORS

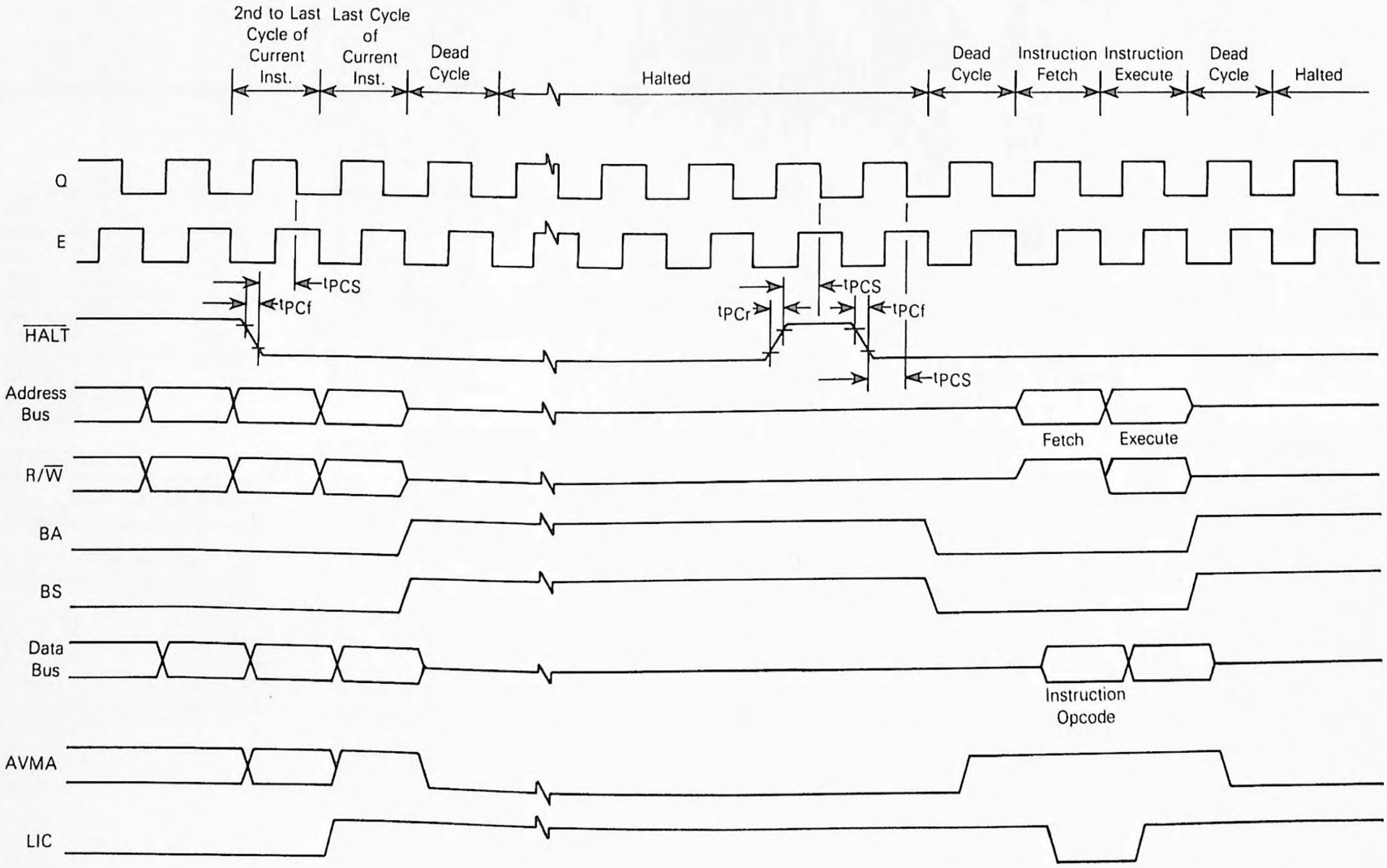
Memory Map For Vector Locations		Interrupt Vector Description
MS	LS	
FFFE	FFFF	$\overline{RESET}$
FFFC	FFFD	$\overline{NMI}$
FFFA	FFFB	SWI
FFF8	FFF9	$\overline{IRQ}$
FFF6	FFF7	$\overline{FIRQ}$
FFF4	FFF5	SWI2
FFF2	FFF3	SWI3
FFF0	FFF1	Reserved

FIGURE 6 — $\overline{\text{RESET}}$ TIMING



NOTE: Timing measurements are referenced to and from a low voltage of 0.8 volts and a high voltage of 2.0 volts, unless otherwise noted.

FIGURE 7 -- $\overline{\text{HALT}}$ AND SINGLE INSTRUCTION EXECUTION TIMING FOR SYSTEM DEBUG



NOTE: Timing measurements are referenced to and from a low voltage of 0.8 volts and a high voltage of 2.0 volts, unless otherwise noted.

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

Halt Acknowledge is indicated when the MC6809E is in a halt condition.

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

A negative transition 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 $\overline{FIRQ}$, $\overline{IRQ}$, or software interrupts. During recognition of an $\overline{NMI}$, the entire machine state is saved on the hardware stack. After reset, an $\overline{NMI}$ will not be recognized until the first program load of the hardware stack pointer (S). The pulse width of $\overline{NMI}$ low must be at least one E cycle. If the $\overline{NMI}$ input does not meet the minimum set up with respect to Q, the interrupt will not be recognized until the next cycle. See Figure 8.

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

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 ($\overline{IRQ}$) 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 9.

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

A low level input on this pin will initiate an interrupt request sequence provided the mask bit (I) in the CC is clear. Since $\overline{IRQ}$ stacks the entire machine state, it provides a slower response to interrupts than $\overline{FIRQ}$. $\overline{IRQ}$ also has a lower priority than $\overline{FIRQ}$. Again, the interrupt service routine should clear the source of the interrupt before doing an RTI. See Figure 8.

CLOCK INPUTS E, Q

E and Q are the clock signals required by the MC6809E. Q must lead E; that is, a transition on Q must be followed by a similar transition on E after a minimum delay. Addresses will be valid from the MPU, t_{AD} after the falling edge of E, and data will be latched from the bus by the falling edge of E. While the Q input is fully TTL compatible, the E input directly drives internal MOS circuitry and, thus, requires a high level above normal TTL levels. This approach minimizes clock skew inherent with an internal buffer. Refer to **BUS TIMING CHARACTERISTICS** for E and Q and to Figure 10 which shows a simple clock generator for the MC6809E.

BUSY

BUSY will be high for the read and modify cycles of a read-modify-write instruction and during the access of the first byte of a double-byte operation (e.g., LDX, STD, ADDD). BUSY is also high during the first byte of any indirect or other vector fetch (e.g., jump extended, SWI indirect, etc.).

In a multiprocessor system, BUSY indicates the need to defer the rearbitration of the next bus cycle to insure the integrity of the above operations. This difference provides the indivisible memory access required for a "test-and-set" primitive, using any one of several read-modify-write instructions.

BUSY does not become active during PSH or PUL operations. A typical read-modify-write instruction (ASL) is shown in Figure 11. Timing information is given in Figure 12. BUSY is valid t_{CD} after the rising edge of Q.

AVMA

AVMA is the advanced VMA signal and indicates that the MPU will use the bus in the following bus cycle. The predictive nature of the AVMA signal allows efficient shared-bus multiprocessor systems. AVMA is low when the MPU is in either a $\overline{HALT}$ or SYNC state. AVMA is valid t_{CD} after the rising edge of Q.

LIC

LIC (last instruction cycle) is high during the last cycle of every instruction, and its transition from high to low will indicate that the first byte of an opcode will be latched at the end of the present bus cycle. LIC will be high when the MPU is halted at the end of an instruction (i.e., not in CWAI or $\overline{RESET}$), in sync state, or while stacking during interrupts. LIC is valid t_{CD} after the rising edge of Q.

TSC

TSC (three-state control) will cause MOS address, data, and R/$\overline{W}$ buffers to assume a high-impedance state. The control signals (BA, BS, BUSY, AVMA, and LIC) will not go to the high-impedance state. TSC is intended to allow a single bus to be shared with other bus masters (processors or DMA controllers).

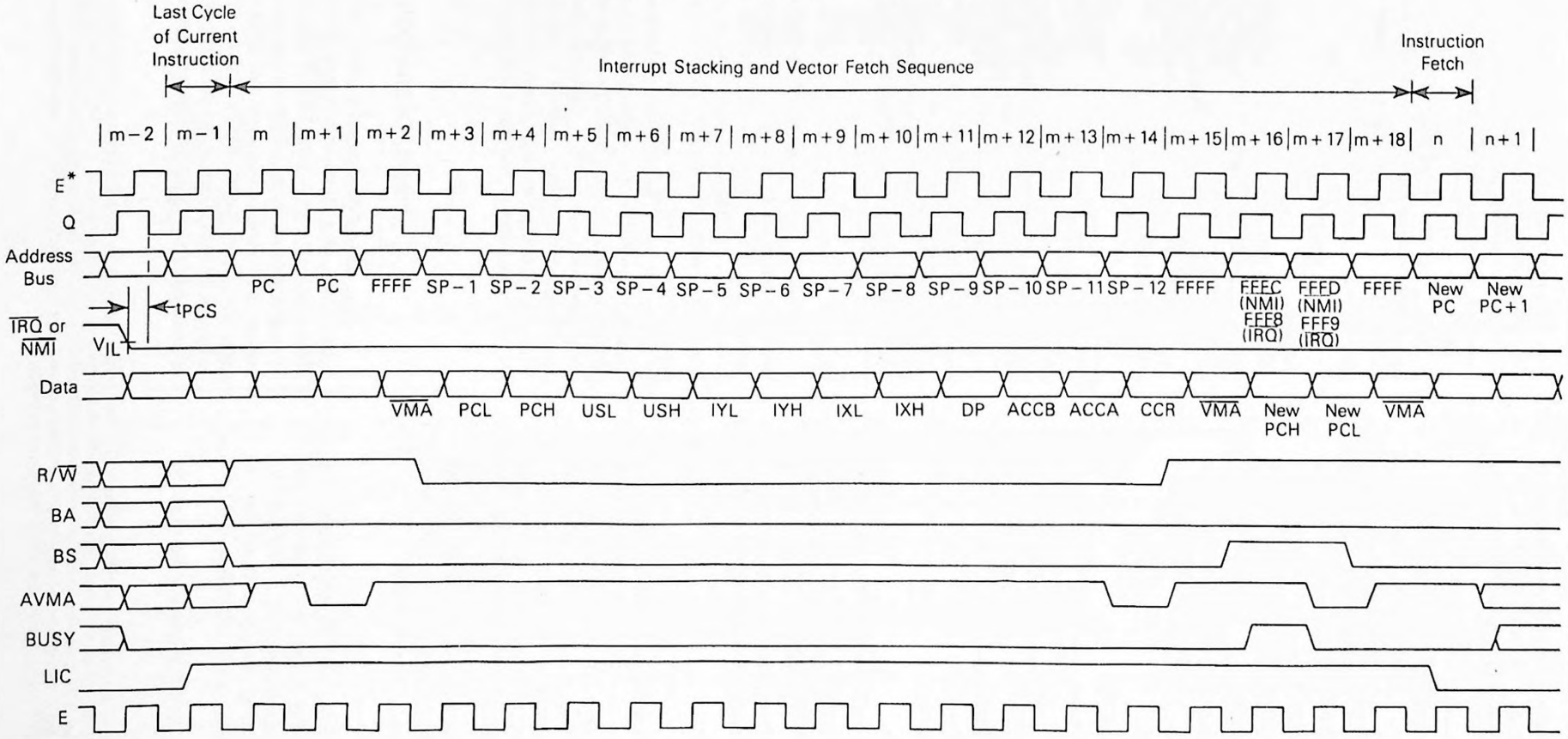
While E is low, TSC controls the address buffers and R/$\overline{W}$ directly. The data bus buffers during a write operation are in a high-impedance state until Q rises at which time, if TSC is true, they will remain in a high-impedance state. If TSC is held beyond the rising edge of E, then it will be internally latched, keeping the bus drivers in a high-impedance state for the remainder of the bus cycle. See Figure 13.

MPU OPERATION

During normal operation, the MPU fetches an instruction from memory and then executes the requested function. This sequence begins after $\overline{RESET}$ 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 or $\overline{HALT}$ input can also alter the normal execution of instructions. Figure 14 is the flowchart for the MC6809E.

* $\overline{NMI}$, $\overline{FIRQ}$, and $\overline{IRQ}$ requests are sampled on the falling edge of Q. One cycle is required for synchronization before these interrupts are recognized. The pending interrupt(s) will not be serviced until completion of the current instruction unless a SYNC or CWAI condition is present. If $\overline{IRQ}$ and $\overline{FIRQ}$ do not remain low until completion of the current instruction, they may not be recognized. However, $\overline{NMI}$ is latched and need only remain low for one cycle. No interrupts are recognized or latched between the falling edge of $\overline{RESET}$ and the rising edge of BS indicating $\overline{RESET}$ acknowledge. See $\overline{RESET}$ sequence in the MPU flowchart in Figure 14.

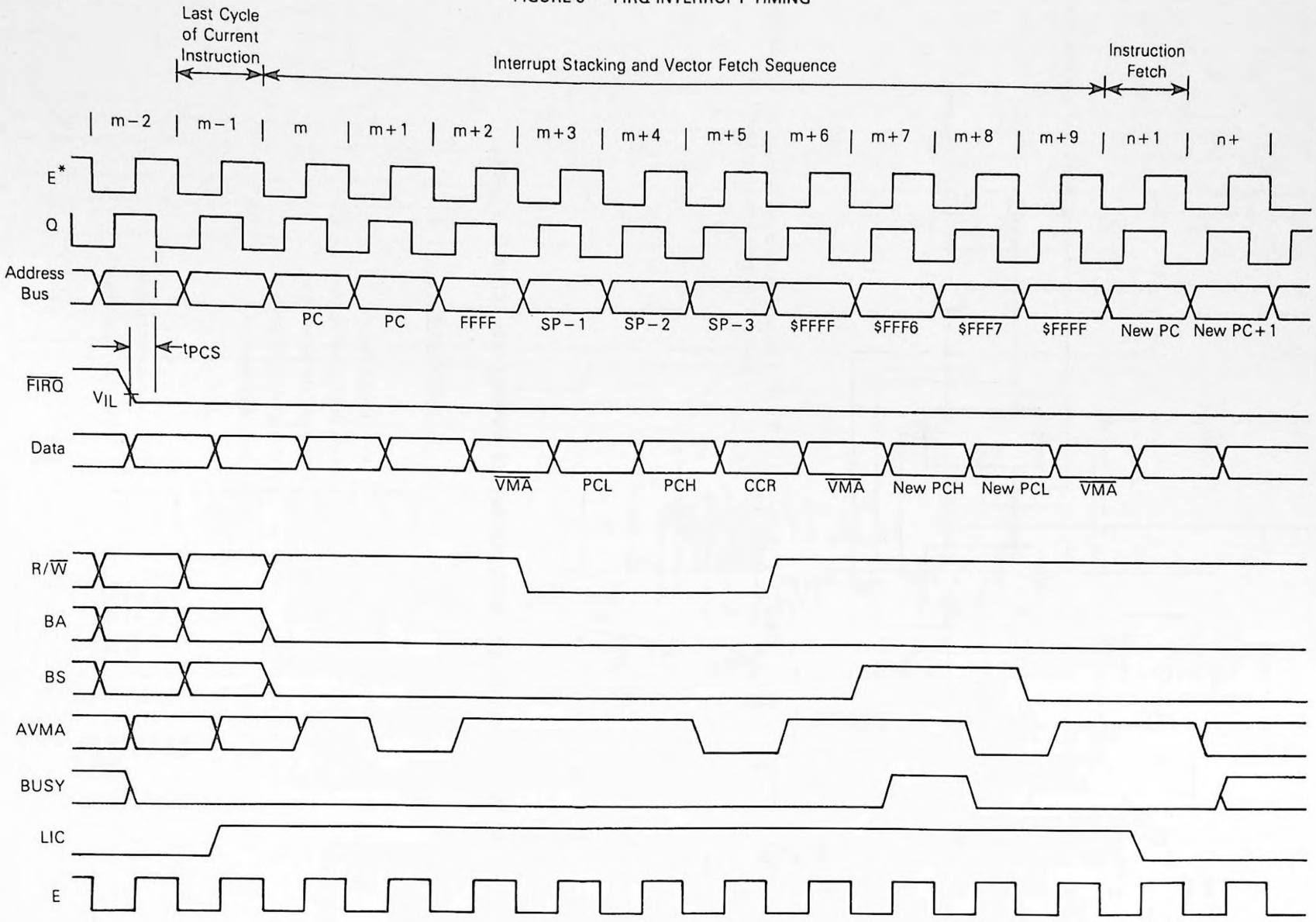
FIGURE 8 – $\overline{IRQ}$ AND $\overline{NMI}$ INTERRUPT TIMING



*E clock shown for reference only.

NOTE: Timing measurements are referenced to and from a low voltage of 0.8 volts and a high voltage of 2.0 volts, unless otherwise noted.

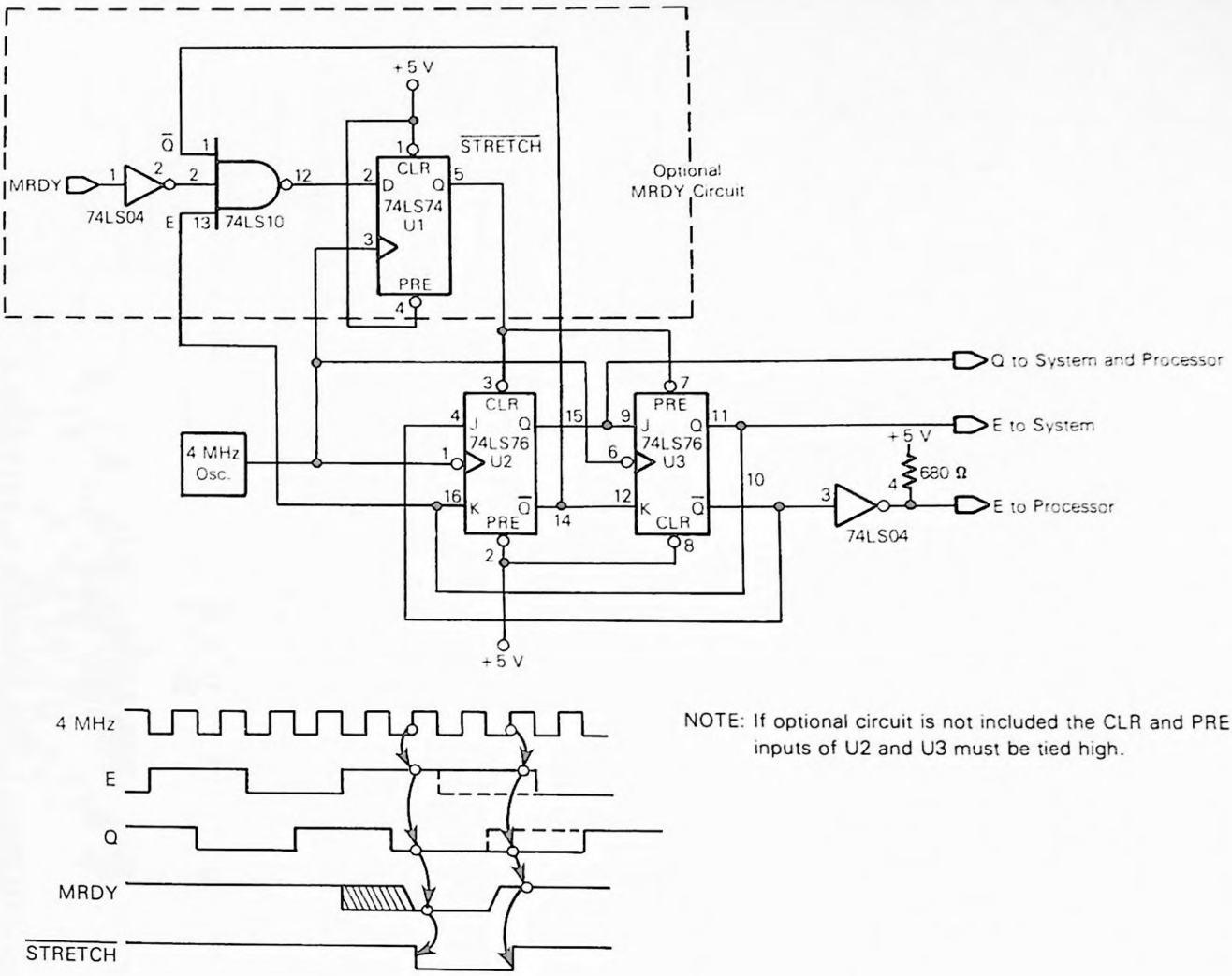
FIGURE 9 — $\overline{\text{FIRQ}}$ INTERRUPT TIMING



*E clock shown for reference only.

NOTE: Timing measurements are referenced to and from a low voltage of 0.8 volts and a high voltage of 2.0 volts, unless otherwise noted.

FIGURE 10 – CLOCK GENERATOR



NOTE: If optional circuit is not included the CLR and PRE inputs of U2 and U3 must be tied high.

FIGURE 11 – READ-MODIFY-WRITE INSTRUCTION EXAMPLE (ASL EXTENDED INDIRECT)

Memory Location	Memory Contents	Contents Description
PC → $0200	$68	ASL Indexed Opcode
$0201	$9F	Extended Indirect Postbyte
$0202	$63	Indirect Address Hi-Byte
$0203	$00	Indirect Address Lo-Byte
$0204		Next Main Instruction
$6300	$E3	Effective Address Hi-Byte
$6301	$D6	Effective Address Lo-Byte
$E3D6	$5C	Target Data

FIGURE 12 — BUSY TIMING

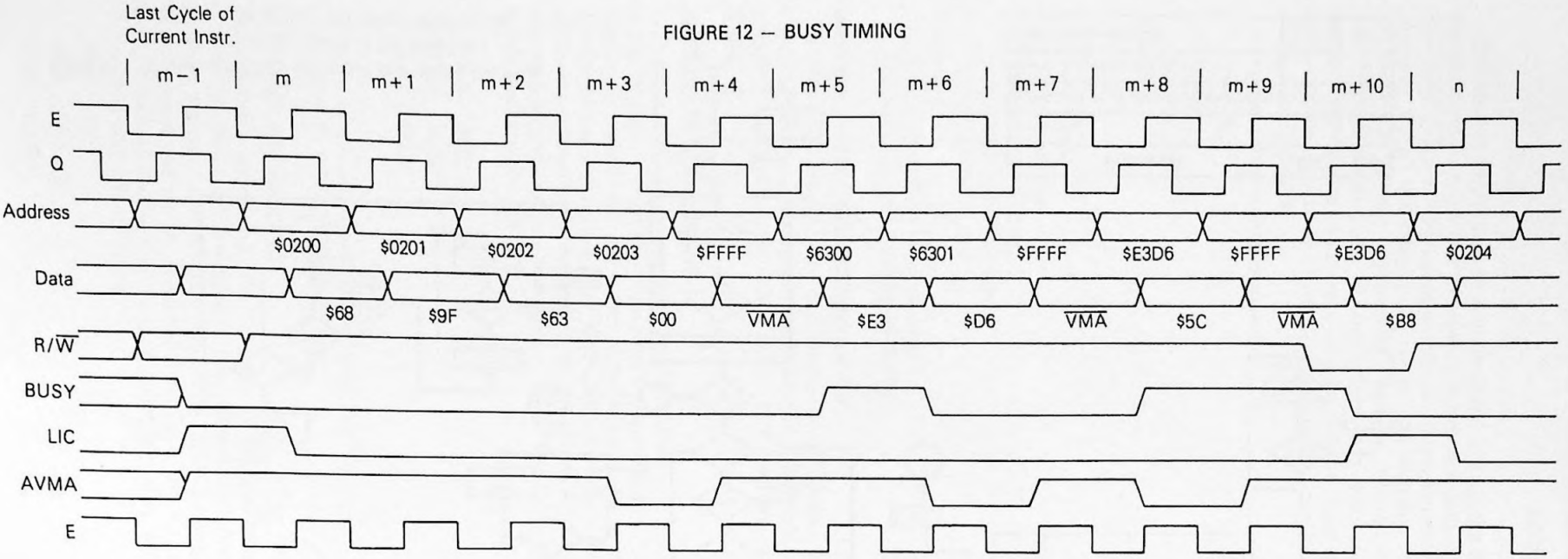
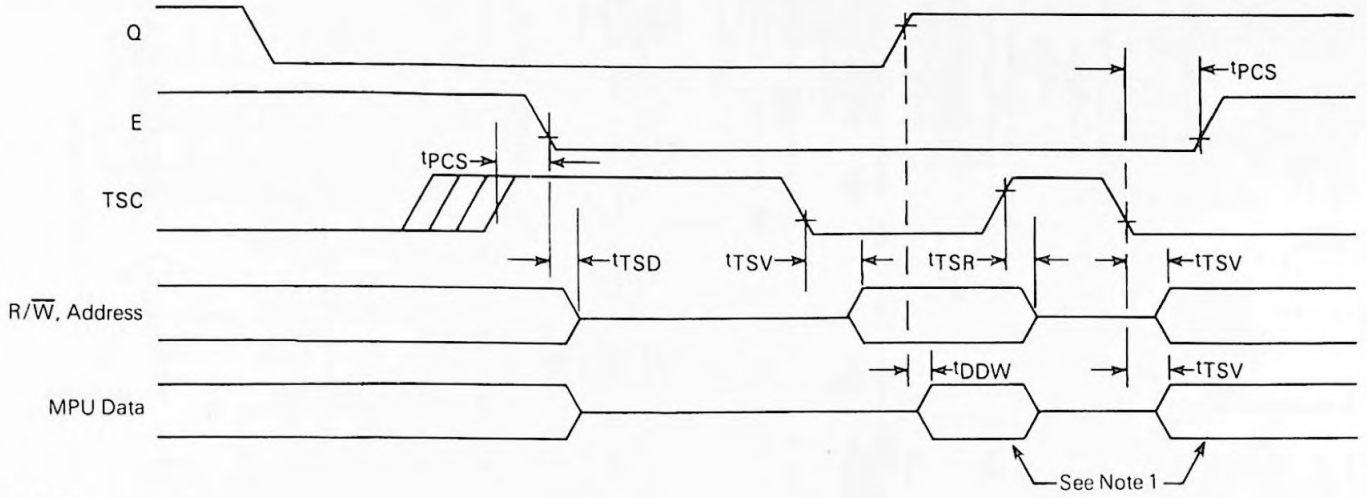


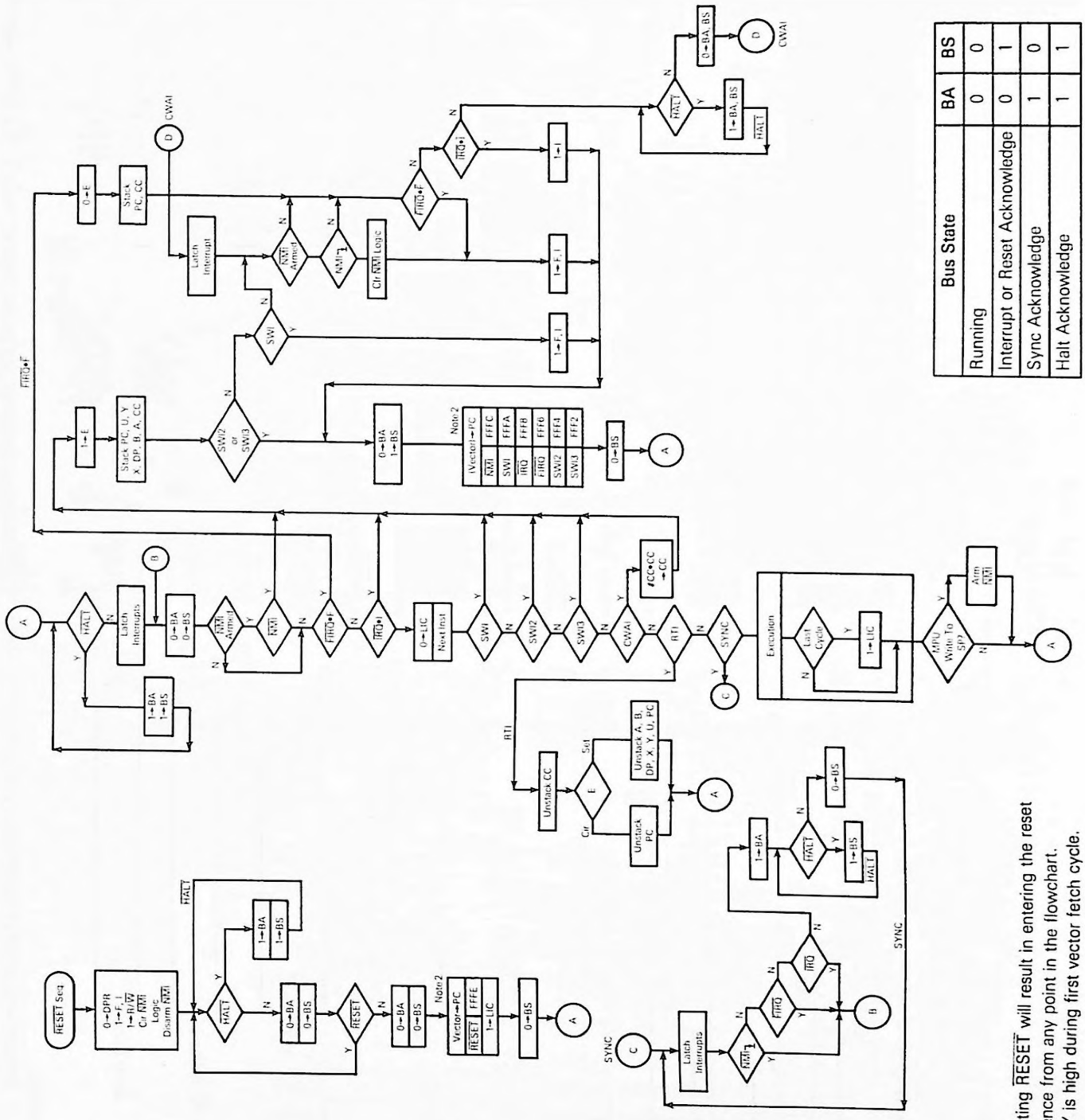
FIGURE 13 — TSC TIMING



NOTES:

1. Data will be asserted by the MPU only during the interval while R/$\overline{W}$ is low and (E or Q) is high. A composite bus cycle is shown to give most cases of timing.
2. Timing measurements are referenced to and from a low voltage of 0.8 volts and a high voltage of 2.0 volts, unless otherwise noted.

FIGURE 14 — FLOWCHART FOR MC6809E INSTRUCTIONS



Bus State	BA	BS
Running	0	0
Interrupt or Reset Acknowledge	0	1
Sync Acknowledge	1	0
Halt Acknowledge	1	1

NOTES: 1. Asserting $\overline{\text{RESET}}$ will result in entering the reset sequence from any point in the flowchart.
2. BUSY is high during first vector fetch cycle.

ADDRESSING MODES

The basic instructions of any computer are greatly enhanced by the presence of powerful addressing modes. The MC6809E has the most complete set of addressing modes available on any microcomputer today. For example, the MC6809E has 59 basic instructions; however, it recognizes 1464 different variations of instructions and addressing modes. The addressing modes support modern programming techniques. The following addressing modes are available on the MC6809E:

- 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 (i.e., the data to be used in the instruction immediately following the opcode of the instruction). The MC6809E 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 to the MC6809 assembler.

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 contain 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 eight bits of the address to be used. The upper eight 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 MC6809E is upward compatible with direct addressing on the M6800. Indirection is not allowed in direct addressing. Some examples of direct addressing are:

```
LDA   where DP = $00
LDB   where DP = $10
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 Y, X, B and A onto S stack
PULU	X, Y, D	Pull D, X, and Y from U stack

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 15 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 15 — INDEXED ADDRESSING POSTBYTE REGISTER BIT ASSIGNMENTS

Post-Byte Register Bit								Indexed Addressing Mode
7	6	5	4	3	2	1	0	
0	R	R	d	d	d	d	d	EA = ,R + 5 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 +8 Bit Offset
1	R	R	i	1	0	0	1	EA = ,R +16 Bit Offset
1	R	R	i	1	0	1	1	EA = ,R +D Offset
1	x	x	i	1	1	0	0	EA = ,PC +8 Bit Offset
1	x	x	i	1	1	0	1	EA = ,PC +16 Bit Offset
1	R	R	i	1	1	1	1	EA = [,Address]

Addressing Mode Field

Indirect Field
(Sign Bit when b7 = 0)

Register Field: RR
00 = X
01 = Y
10 = U
11 = S

x = Don't Care
d = Offset Bit
i = 0 = Not Indirect, 1 = Indirect

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   ,S
```

CONSTANT OFFSET INDEXED — In this mode, twos 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 offset are available:

5-bit (−16 to +15)
8-bit (−128 to +127)
16-bit (−32768 to +32767)

The twos complement 5-bit offset is included in the postbyte and, therefore, is most efficient in use of bytes and cycles. The twos complement 8-bit offset is contained in a single byte following the postbyte. The twos 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 optimal size automatically.

Examples of constant-offset indexing are:

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

TABLE 2 — INDEXED ADDRESSING MODE

Type	Forms	Non Indirect				Indirect			
		Assembler Form	Postbyte Opcode	+ ~	+ #	Assembler Form	Postbyte Opcode	+ ~	+ #
Constant Offset From R (2s Complement 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 (2s Complement 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 (2s Complement Offsets)	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 = Don't Care

RR:
00 = X
01 = Y
10 = U
11 = S

$^{+}_{\sim}$ and $^{+}_{\#}$ indicate the number of additional cycles and bytes respectively for the particular indexing variation.

ACCUMULATOR-OFFSET INDEXED – This mode is similar to constant offset indexed except that the twos complement value in one of the accumulators (A, B, or D) and the contents 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 creating 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 similar 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 allows 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
```

Care should be taken in performing operations on 16-bit pointer registers (X, Y, U, S) where the same register is used to calculate the effective address.

Consider the following instruction:

STX 0,X++ (X initialized to 0)

The desired result is to store a zero in locations $0000 and $0001, then increment X to point to $0002. In reality, the following occurs:

0 → temp	calculate the EA; temp is a holding register
X + 2 → X	perform auto increment
X → (temp)	do store operation

INDEXED INDIRECT

All of the indexing modes, with the exception of auto increment/decrement by one or a ±5-bit offset, may have an additional level of indirection specified. In indirect addressing, the effective address is contained at the location specified by the contents 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)
X = $F000

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 may be 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 (one byte offset) and long (two bytes offset) relative addressing modes are available. All of memory can be reached in long relative addressing as an effective address 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]
```

INSTRUCTION SET

The instruction set of the MC6809E 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.

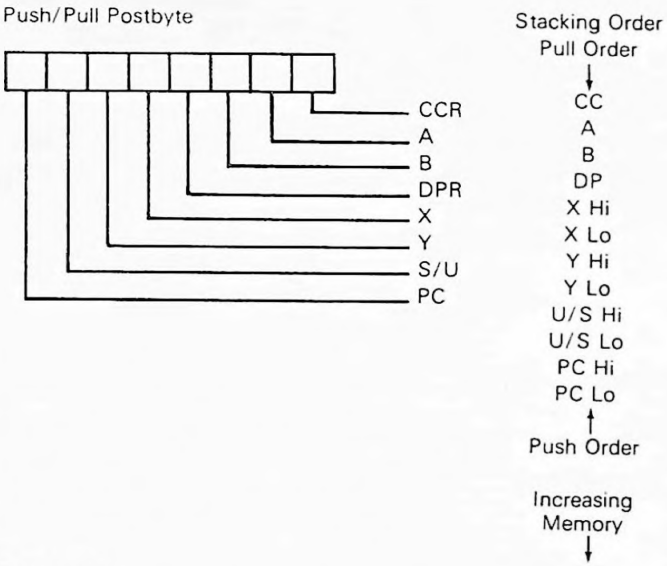
Some of the new instructions are described in detail below.

PSHU/PSHS

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

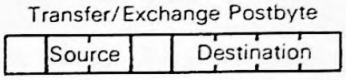
PULU/PULS

The pull instructions have the same capability of the push instruction, in reverse order. The byte immediately 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 below.



TFR/EXG

Within the MC6809E, 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:



Register Field

0000 = D (A:B)	1000 = A
0001 = X	1001 = B
0010 = Y	1010 = CCR
0011 = U	1011 = DPR
0100 = S	
0101 = PC	

NOTE

All other combinations are undefined and INVALID.

LEAX/LEAY/LEAU/LEAS

The LEA (load effective address) 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 Table 3.

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

```
        LEAX    MSG1, PCR
        LBSR    PDATA (Print message routine)
        .
        .
MSG1    FCC     'MESSAGE'
```

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.

The LEA instructions are very powerful and use an internal holding register (temp). Care must be exercised when using the LEA instructions with the auto increment and auto decrement addressing modes due to the sequence of internal operations. The LEA internal sequence is outlined as follows:

LEAa ,b+ (any of the 16-bit pointer registers X, Y, U, or S may be substituted for a and b.)

1. b → temp (calculate the EA)
2. b + 1 → b (modify b, postincrement)
3. temp → a (load a)

LEAa ,−b

1. b − 1 → temp (calculate EA with predecrement)
2. b − 1 → b (modify b, predecrement)
3. temp → a (load a)

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 A Accumulator to Y
LEAY D, Y	Y + D → Y	Adds 16-Bit D Accumulator to Y
LEAU −10, U	U − 10 → U	Substracts 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

Auto increment-by-two and auto decrement-by-two instructions work similarly. Note that LEAX ,X+ does not change X; however LEAX, −X does decrement X.LEAX 1,X should be used to increment X by one.

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 MC6809E 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 ($\overline{NMI}$) or maskable ($\overline{FIRQ}$, $\overline{IRQ}$) 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 $\overline{FIRQ}$ and $\overline{IRQ}$ are not edge-triggered, a low level with a minimum duration of three bus cycles is required to assure that the interrupt will be taken. If the pending interrupt is maskable ($\overline{FIRQ}$, $\overline{IRQ}$) with its mask bit (F or I) set, the processor will clear the sync state and continue processing by executing the next in-line instruction. Figure 16 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 MC6809E and are prioritized in the following order: SWI, SWI2, SWI3.

16-BIT OPERATION

The MC6809E 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 (Figure 16) illustrates the memory-access sequence corresponding to each possible instruction and addressing mode in the MC6809E. 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 considerably speeds throughput.) Next, the operation of each opcode will follow the flowchart. $\overline{VMA}$ 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.

Example 1: LBSR (Branch Taken)
Before Execution SP = F000

```
                    •
                    •
                    •
$8000               LBSR    CAT
                    •
                    •
                    •
$A000   CAT         •
```

CYCLE-BY-CYCLE FLOW

Cycle #	Address	Data	R/$\overline{W}$	Description
1	8000	17	1	Opcode Fetch
2	8001	20	1	Offset High Byte
3	8002	00	1	Offset Low Byte
4	FFFF	*	1	$\overline{VMA}$ Cycle
5	FFFF	*	1	$\overline{VMA}$ Cycle
6	A000	*	1	Computed Branch Address
7	FFFF	*	1	$\overline{VMA}$ Cycle
8	EFFF	80	0	Stack High Order Byte of Return Address
9	EFFE	03	0	Stack Low Order Byte of Return Address

Example 2: DEC (Extended)

```
$8000   DEC     $A000
$A000   FCB     $80
```

CYCLE-BY-CYCLE FLOW

Cycle #	Address	Data	R/$\overline{W}$	Description
1	8000	7A	1	Opcode Fetch
2	8001	A0	1	Operand Address, High Byte
3	8002	00	1	Operand Address, Low Byte
4	FFFF	*	1	$\overline{VMA}$ Cycle
5	A000	80	1	Read the Data
6	FFFF	*	1	$\overline{VMA}$ Cycle
7	FFFF	7F	0	Store the Decremented Data

* The data bus has the data at that particular address.

INSTRUCTION SET TABLES

The instructions of the MC6809E have been broken down into five different categories. They are as follows:

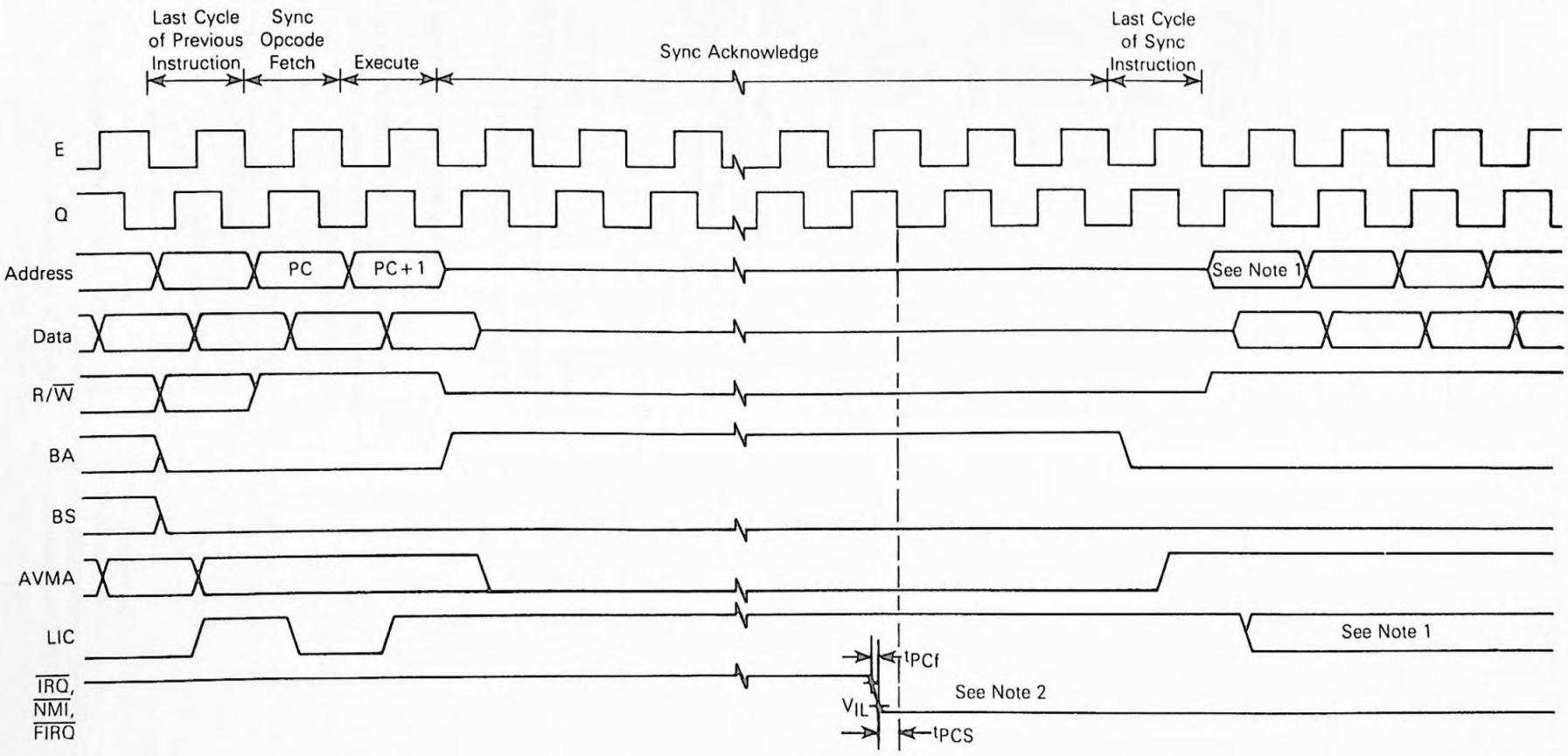
8-bit operation (Table 4)
16-bit operation (Table 5)
Index register/stack pointer instructions (Table 6)
Relative branches (long or short) (Table 7)
Miscellaneous instructions (Table 8)

Hexadecimal values for the instructions are given in Table 9.

PROGRAMMING AID

Figure 18 contains a compilation of data that will assist you in programming the MC6809E.

FIGURE 16 — SYNC TIMING

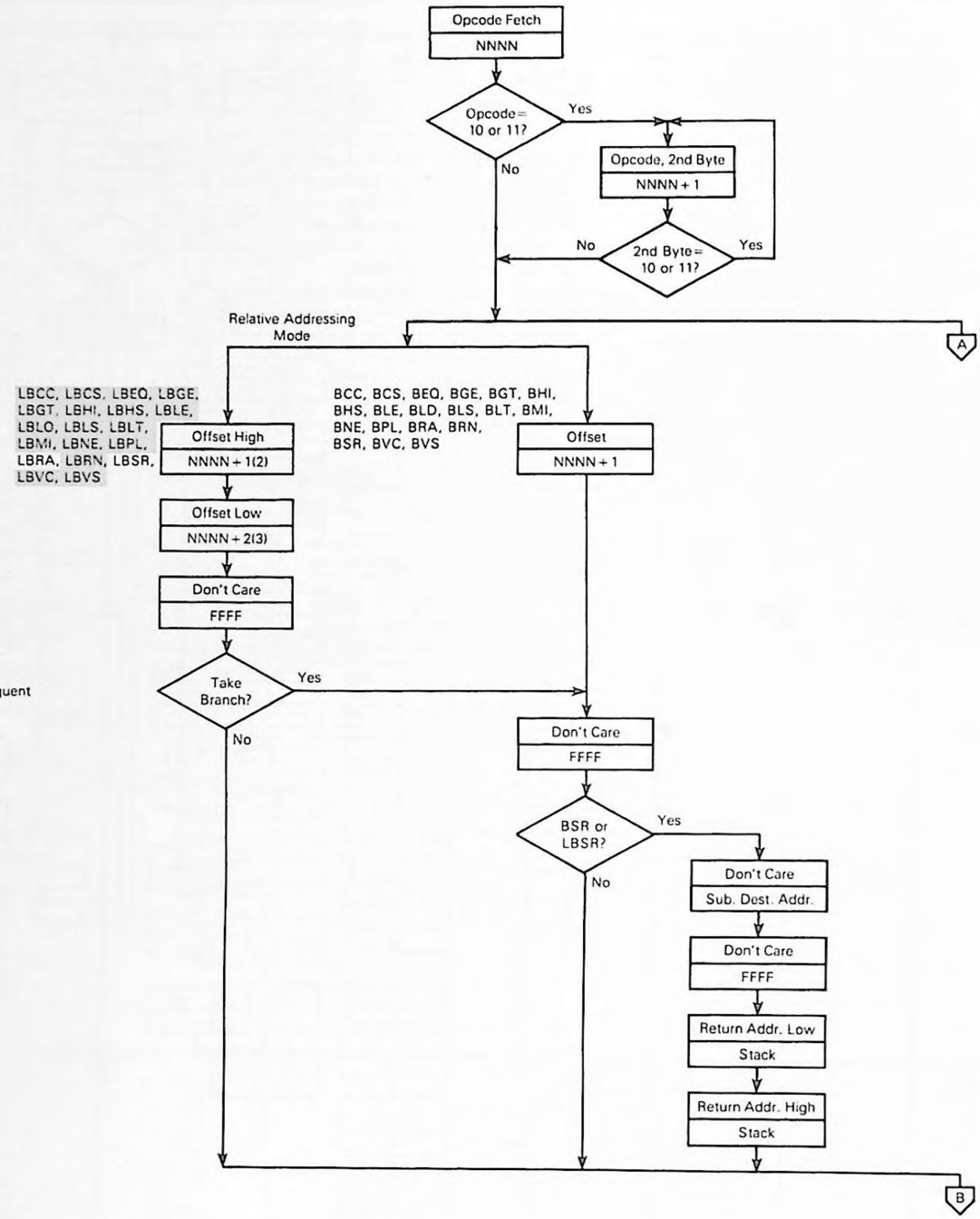


NOTES: 1. If the associated mask bit is set when the interrupt is requested, LIC will go low and this cycle will be an instruction fetch from address location PC+1. However, if the interrupt is accepted ($\overline{NMI}$ or an unmasked $\overline{FIRQ}$ or $\overline{IRQ}$) LIC will remain high and interrupt processing will start with this cycle as m on Figures 8 and 9 (Interrupt Timing).

2. If mask bits are clear, $\overline{IRQ}$ and $\overline{FIRQ}$ must be held low for three cycles to guarantee that interrupt will be taken, although only one cycle is necessary to bring the processor out of SYNC.

3. Timing measurements are referenced to and from a low voltage of 0.8 volts and a high voltage of 2.0 volts, unless otherwise noted.

FIGURE 17 — CYCLE-BY-CYCLE PERFORMANCE (Sheet 1 of 5)



NOTES:

1. Each state shows:

Data Bus	Offset High
Address Bus	NNNN + 1(2)

2. Address NNNN is location of opcode.
3. If opcode is a two byte opcode subsequent addresses are in parenthesis (—).
4. Two-byte opcodes are highlighted.

FIGURE 17 — CYCLE-BY-CYCLE PERFORMANCE (Sheet 2 of 5)

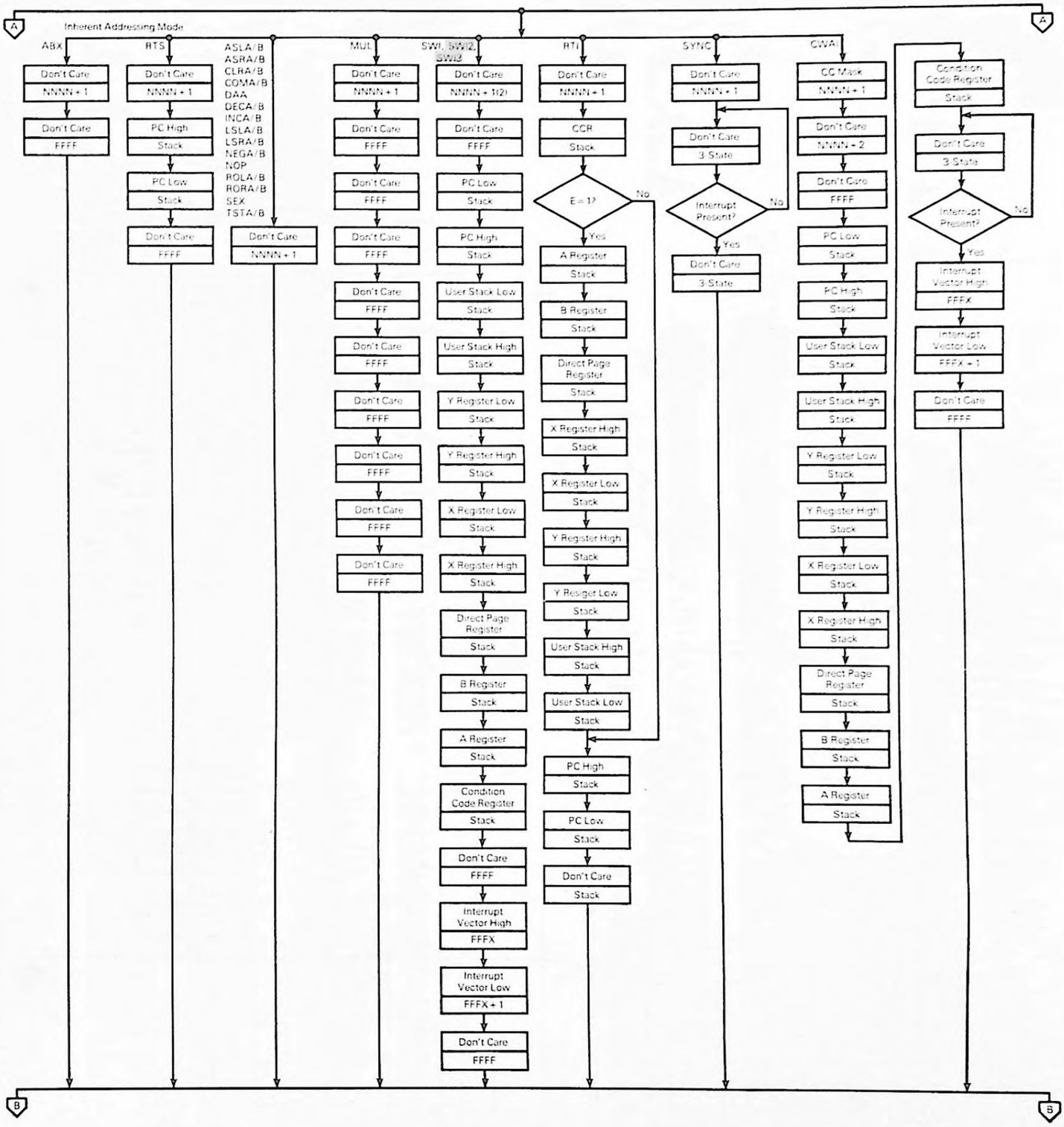


FIGURE 17 — CYCLE-BY-CYLE PERFORMANCE (Sheet 3 of 5)

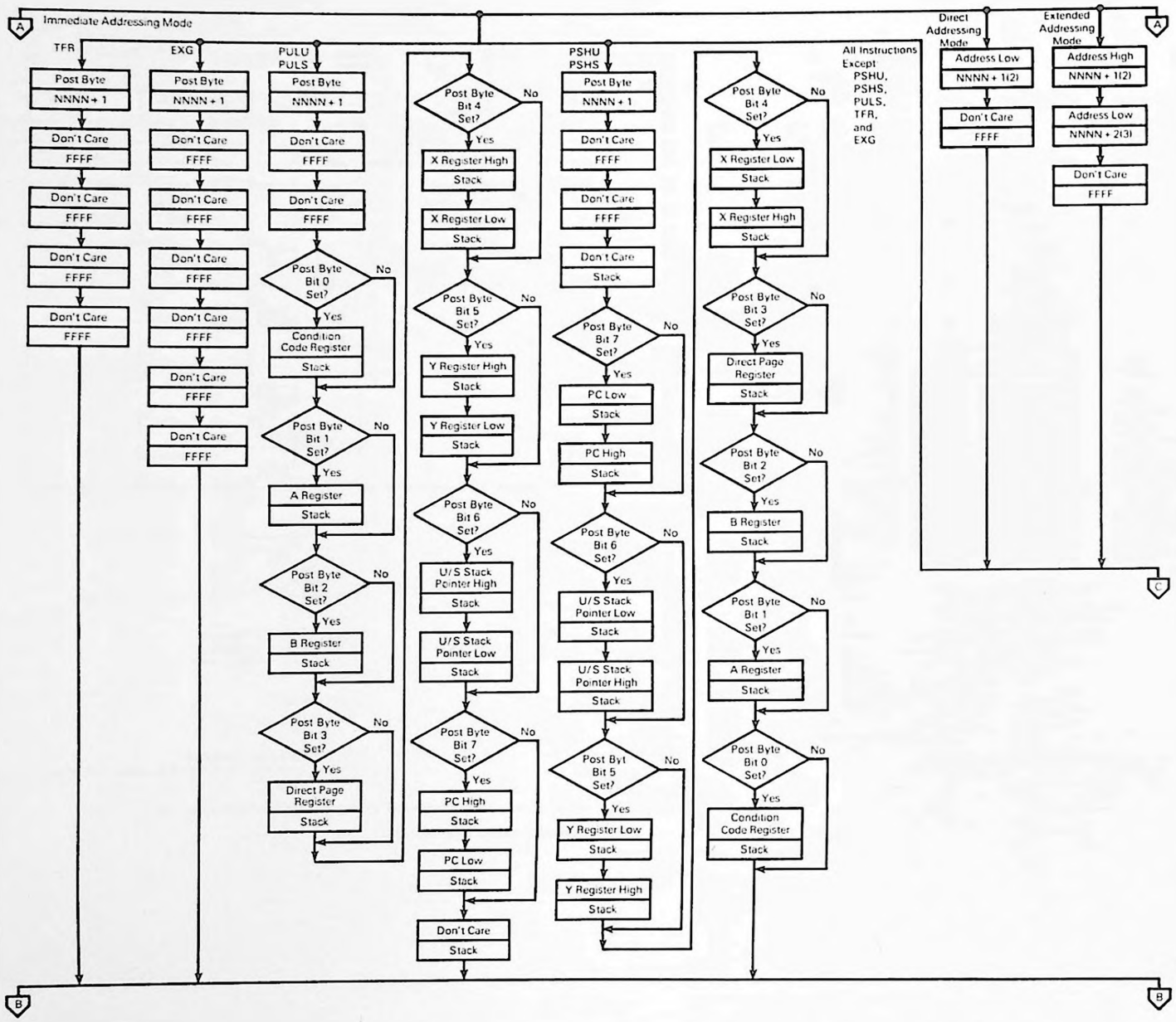
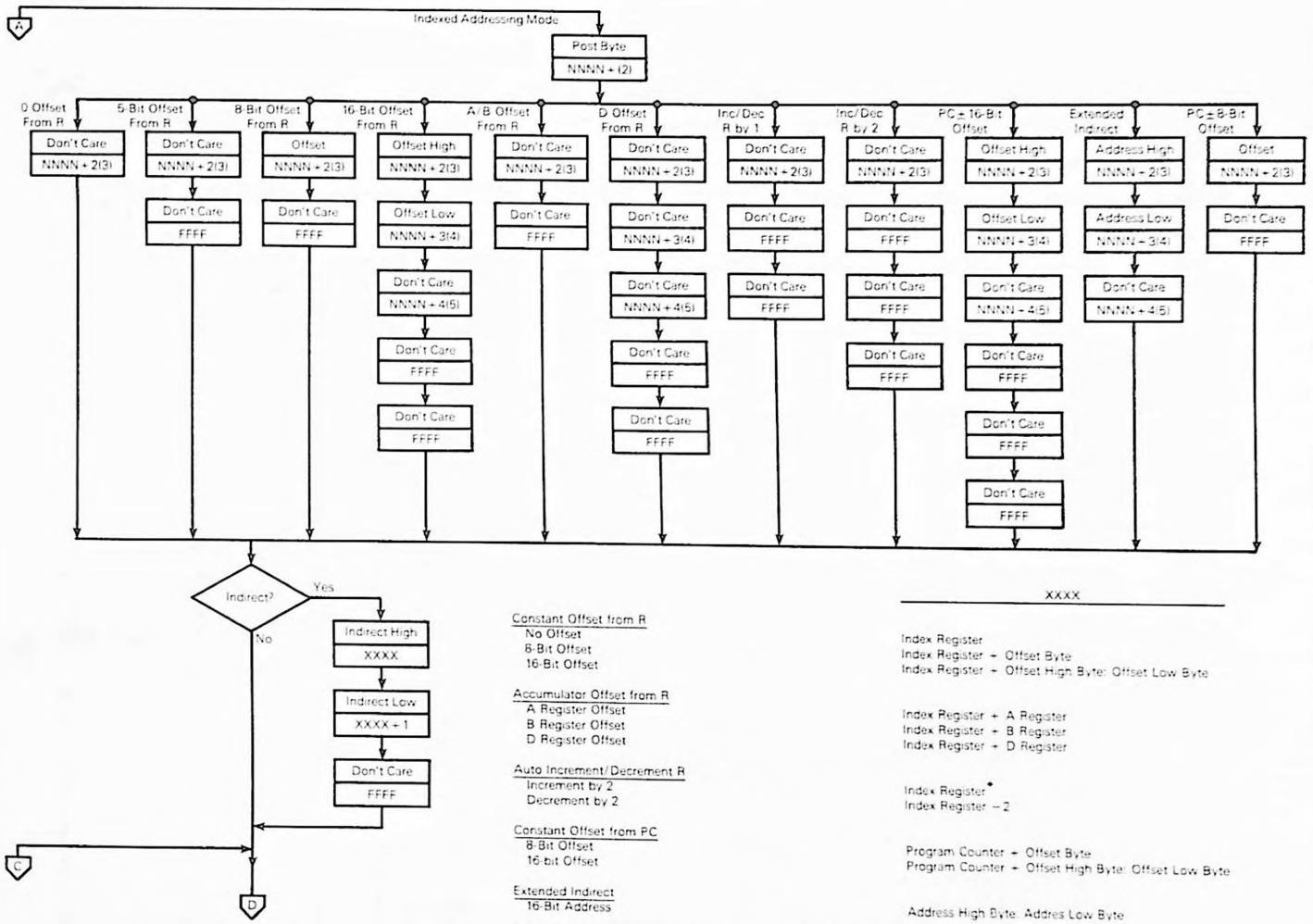
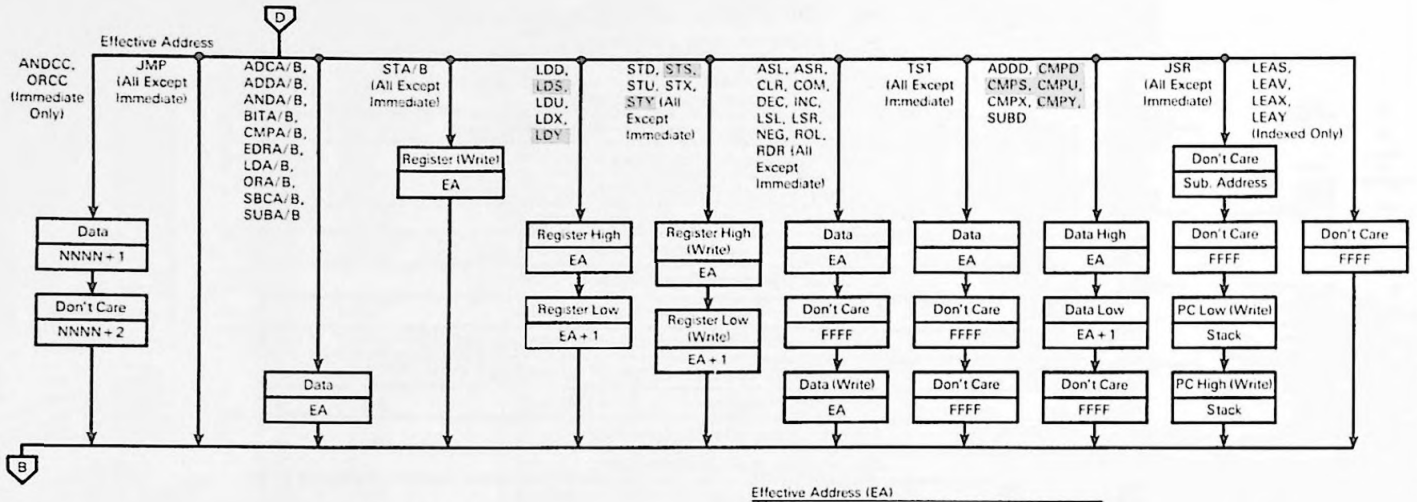


FIGURE 17 — CYCLE-BY-CYCLE PERFORMANCE (Sheet 4 of 5)



*The index register is incremented following the indexed access.

FIGURE 17 — CYCLE-BY-CYCLE PERFORMANCE (Sheet 5 of 5)



	Effective Address (EA)
Constant Offset from R	
No Offset	Index Register
5-Bit Offset	Index Register
8-Bit Offset	Index Register + Post Byte
16-Bit Offset	Index Register + Post Byte High, Post Byte Low
Accumulator Offset from R	
A Register Offset	Index Register + A Register
B Register Offset	Index Register + B Register
D Register Offset	Index Register + D Register
Auto Increment/Decrement R	
Increment by 1	Index Register*
Increment by 2	Index Register*
Decrement by 1	Index Register − 1
Decrement by 2	Index Register − 2
Constant Offset from PC	
8-Bit Offset	Program Counter + Offset Byte
16-Bit Offset	Program Counter + Offset High Byte, Offset Low Byte
Direct	Direct Page Register, Address Low
Extended	Address High, Address Low
Immediate	NNNN + 1

*The index register is incremented following the indexed access.

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 × 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

NOTE: D may be pushed (pulled) to either stack with PSHS, PSHU (PULS, PULU) instructions.

TABLE 6 — INDEX REGISTER/STACK POINTER INSTRUCTIONS

Instruction	Description
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 A, B, CC, DP, D, X, Y, U, or PC onto hardware stack
PSHU	Push A, B, CC, DP, D, X, Y, S, or PC onto user stack
PULS	Pull A, B, CC, DP, D, X, Y, U or PC from hardware stack
PULU	Pull A, B, CC, DP, D, X, Y, S or PC from hardware stack
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

Instruction	Description
SIMPLE BRANCHES	
BEQ, LBEQ	Branch if equal
BNE, LBNE	Branch if not equal
BMI, LBMI	Branch if minus
BPL, LBPL	Branch if plus
BCS, LBCS	Branch if carry set
BCC, LBCC	Branch if carry clear
BVS, LBVS	Branch if overflow set
BVC, LBVC	Branch if overflow clear
SIGNED BRANCHES	
BGT, LBGT	Branch if greater (signed)
BVS, LBVS	Branch if invalid 2's complement result
BGE, LBGE	Branch if greater than or equal (signed)
BEQ, LBEQ	Branch if equal
BNE, LBNE	Branch if not equal
BLE, LBLE	Branch if less than or equal (signed)
BVC, LBVC	Branch if valid 2's complement result
BLT, LBLT	Branch if less than (signed)
UNSIGNED BRANCHES	
BHI, LBHI	Branch if higher (unsigned)
BCC, LBCC	Branch if higher or same (unsigned)
BHS, LBHS	Branch if higher or same (unsigned)
BEQ, LBEQ	Branch if equal
BNE, LBNE	Branch if not equal
BLS, LBLS	Branch if lower or same (unsigned)
BCS, LBCS	Branch if lower (unsigned)
BLO, LBLO	Branch if lower (unsigned)
OTHER BRANCHES	
BSR, LBSR	Branch to subroutine
BRA, LBRA	Branch always
BRN, LBRN	Branch never

TABLE 8 — MISCELLANEOUS INSTRUCTIONS

Instruction	Description
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	Immed	5+	2	64	LSR		6+	2+
05	*				35	PULS	Immed	5+	2	65	*			
06	ROR		6	2	36	PSHU	Immed	5+	2	66	ROR		6+	2+
07	ASR		6	2	37	PULU	Immed	5+	2	67	ASR		6+	2+
08	ASL, LSL		6	2	38	*	—			68	ASL, LSL		6+	2+
09	ROL		6	2	39	RTS	Inherent	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	Inherent	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	≥4	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	Immed	8	2	4E	*				7E	JMP		4	3
1F	TFR	Immed	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	RORB		2	1	86	LDA		2	2
27	BEQ		3	2	57	ASRB		2	1	87	*			
28	BVC		3	2	58	ASLB, LSLB		2	1	88	EORA		2	2
29	BVS		3	2	59	ROLB		2	1	89	ADCA		2	2
2A	BPL		3	2	5A	DECB		2	1	8A	ORA		2	2
2B	BMI		3	2	5B	*				8B	ADDA		2	2
2C	BGE		3	2	5C	INCB		2	1	8C	CMPX	Immed	4	3
2D	BLT		3	2	5D	TSTB		2	1	8D	BSR	Relative	7	2
2E	BGT		3	2	5E	*				8E	LDX	Immed	3	3
2F	BLE	Relative	3	2	5F	CLRB	Inherent	2	1	8F	*			

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 MACHINE CODES (CONTINUED)

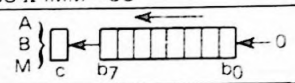
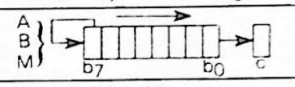
OP	Mnem	Mode	~	#
90	SUBA	Direct	4	2
91	CMPA	↑	4	2
92	SBCA		4	2
93	SUBD		6	2
94	ANDA		4	2
95	BITA		4	2
96	LDA		4	2
97	STA		4	2
98	EORA		4	2
99	ADCA		4	2
9A	ORA		4	2
9B	ADDA		4	2
9C	CMPX		6	2
9D	JSR		7	2
9E	LDX		5	2
9F	STX	Direct	5	2
A0	SUBA	Indexed	4+	2+
A1	CMPA	↑	4+	2+
A2	SBCA		4+	2+
A3	SUBD		6+	2+
A4	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

NOTE: All unused opcodes are both undefined and illegal

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

OP	Mnem	Mode	~	#
Page 2 and 3 Machine Codes				
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		5(6)	4
102E	LBGT	↓	5(6)	4
102F	LBLE	Relative	5(6)	4
103F	SWI2	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	SWI3	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

FIGURE 18 — PROGRAMMING AID

Instruction	Forms	Addressing Modes: Immediate Op	Immediate ~	Immediate #	Direct Op	Direct ~	Direct #	Indexed Op	Indexed ~	Indexed #	Extended Op	Extended ~	Extended #	Inherent Op	Inherent ~	Inherent #	Description	5 H	3 N	2 Z	1 V	0 C
ABX														3A	3	1	B + X → X (Unsigned)	•	•	•	•	•
ADC	ADCA	89	2	2	99	4	2	A9	4+	2+	B9	5	3				A + M + C → A	↕	↕	↕	↕	↕
	ADCB	C9	2	2	D9	4	2	E9	4+	2+	F9	5	3				B + M + C → B	↕	↕	↕	↕	↕
ADD	ADDA	8B	2	2	9B	4	2	AB	4+	2+	BB	5	3				A + M → A	↕	↕	↕	↕	↕
	ADDB	CB	2	2	DB	4	2	EB	4+	2+	FB	5	3				B + M → B	↕	↕	↕	↕	↕
	ADDD	C3	4	3	D3	6	2	E3	6+	2+	F3	7	3				D + M:M + 1 → D	•	↕	↕	↕	↕
AND	ANDA	84	2	2	94	4	2	A4	4+	2+	B4	5	3				A ∧ M → A	•	↕	↕	0	•
	ANDB	C4	2	2	D4	4	2	E4	4+	2+	F4	5	3				B ∧ M → B	•	↕	↕	0	•
	ANDCC	1C	3	2													CC ∧ IMM → CC					7
ASL	ASLA													48	2	1		8	↕	↕	↕	↕
	ASLB													58	2	1		8	↕	↕	↕	↕
	ASL				08	6	2	68	6+	2+	78	7	3					8	↕	↕	↕	↕
ASR	ASRA													47	2	1		8	↕	↕	•	↕
	ASRB													57	2	1		8	↕	↕	•	↕
	ASR				07	6	2	67	6+	2+	77	7	3					8	↕	↕	•	↕
BIT	BITA	85	2	2	95	4	2	A5	4+	2+	B5	5	3				Bit Test A (M ∧ A)	•	↕	↕	0	•
	BITB	C5	2	2	D5	4	2	E5	4+	2+	F5	5	3				Bit Test B (M ∧ B)	•	↕	↕	0	•
CLR	CLRA													4F	2	1	0 → A	•	0	1	0	0
	CLRB													5F	2	1	0 → B	•	0	1	0	0
	CLR				0F	6	2	6F	6+	2+	7F	7	3				0 → M	•	0	1	0	0
CMP	CMPA	81	2	2	91	4	2	A1	4+	2+	B1	5	3				Compare M from A	8	↕	↕	↕	↕
	CMPB	C1	2	2	D1	4	2	E1	4+	2+	F1	5	3				Compare M from B	8	↕	↕	↕	↕
	CMPD	10	5	4	10	7	3	10	7+	3+	10	8	4				Compare M:M + 1 from D	•	↕	↕	↕	↕
		83			93			A3			B3											
	CMPS	11	5	4	11	7	3	11	7+	3+	11	8	4				Compare M:M + 1 from S	•	↕	↕	↕	↕
		8C			9C			AC			BC											
	CMPU	11	5	4	11	7	3	11	7+	3+	11	8	4				Compare M:M + 1 from U	•	↕	↕	↕	↕
		83			93			A3			B3											
	CMPX	8C	4	3	9C	6	2	AC	6+	2+	BC	7	3				Compare M:M + 1 from X	•	↕	↕	↕	↕
	CMPY	10	5	4	10	7	3	10	7+	3+	10	8	4				Compare M:M + 1 from Y	•	↕	↕	↕	↕
		8C			9C			AC			BC											
COM	COMA													43	2	1	$\overline{A}$ → A	•	↕	↕	0	1
	COMB													53	2	1	$\overline{B}$ → B	•	↕	↕	0	1
	COM				03	6	2	63	6+	2+	73	7	3				$\overline{M}$ → M	•	↕	↕	0	1
CWAI		3C	≥20	2													CC ∧ IMM → CC Wait for Interrupt					7
DAA														19	2	1	Decimal Adjust A	•	↕	↕	0	↕
DEC	DECA													4A	2	1	A − 1 → A	•	↕	↕	↕	•
	DECB													5A	2	1	B − 1 → B	•	↕	↕	↕	•
	DEC				0A	6	2	6A	6+	2+	7A	7	3				M − 1 → M	•	↕	↕	↕	•
EOR	EORA	88	2	2	98	4	2	A8	4+	2+	B8	5	3				A ⊻ M → A	•	↕	↕	0	•
	EORB	C8	2	2	D8	4	2	E8	4+	2+	F8	5	3				B ⊻ M → B	•	↕	↕	0	•
EXG	R1, R2	1E	8	2													R1 → R2[2]	•	•	•	•	•
INC	INCA													4C	2	1	A + 1 → A	•	↕	↕	↕	•
	INCB													5C	2	1	B + 1 → B	•	↕	↕	↕	•
	INC				0C	6	2	6C	6+	2+	7C	7	3				M + 1 → M	•	↕	↕	↕	•
JMP					0E	3	2	6E	3+	2+	7E	4	3				EA[3] → PC	•	•	•	•	•
JSR					9D	7	2	AD	7+	2+	BD	8	3				Jump to Subroutine	•	•	•	•	•
LD	LDA	86	2	2	96	4	2	A6	4+	2+	B6	5	3				M → A	•	↕	↕	0	•
	LDB	C6	2	2	D6	4	2	E6	4+	2+	F6	5	3				M → B	•	↕	↕	0	•
	LDD	CC	3	3	DC	5	2	EC	5+	2+	FC	6	3				M:M + 1 → D	•	↕	↕	0	•
	LDS	10	4	4	10	6	3	10	6+	3+	10	7	4				M:M + 1 → S	•	↕	↕	0	•
		CE			DE			EE			FE											
	LDU	CE	3	3	DE	5	2	EE	5+	2+	FE	6	3				M:M + 1 → U	•	↕	↕	0	•
	LDX	8E	3	3	9E	5	2	AE	5+	2+	BE	6	3				M:M + 1 → X	•	↕	↕	0	•
	LDY	10	4	4	10	6	3	10	6+	3+	10	7	4				M:M + 1 → Y	•	↕	↕	0	•
		8E			9E			AE			BE											
LEA	LEAS							32	4+	2+							EA[3] → S	•	•	•	•	•
	LEAU							33	4+	2+							EA[3] → U	•	•	•	•	•
	LEAX							30	4+	2+							EA[3] → X	•	•	↕	•	•
	LEAY							31	4+	2+							EA[3] → Y	•	•	↕	•	•

LEGEND:

- OP Operation Code (Hexadecimal)
- ~ Number of MPU Cycles
- # Number of Program Bytes
- + Arithmetic Plus
- − Arithmetic Minus
- • Multiply
- $\overline{M}$ Complement of M
- → Transfer Into
- H Half-carry (from bit 3)
- N Negative (sign bit)
- Z Zero result
- V Overflow, 2's complement
- C Carry from ALU
- ↕ Test and set if true, cleared otherwise
- • Not Affected
- CC Condition Code Register
- : Concatenation
- ∨ Logical or
- ∧ Logical and
- ⊻ Logical Exclusive or

FIGURE 18 — PROGRAMMING AID (CONTINUED)

Instruction	Forms	Addressing Modes: Immediate Op	Immediate ~	Immediate #	Direct Op	Direct ~	Direct #	Indexed[1] Op	Indexed[1] ~	Indexed[1] #	Extended Op	Extended ~	Extended #	Inherent Op	Inherent ~	Inherent #	Description	5 H	3 N	2 Z	1 V	0 C
LSL	LSLA													48	2	1	A } ← 0	•	↕	↕	↕	↕
	LSLB													58	2	1	B } ← 0	•	↕	↕	↕	↕
	LSL				08	6	2	68	6+	2+	78	7	3				M } c b7 b0	•	↕	↕	↕	↕
LSR	LSRA													44	2	1	A } 0 →	•	0	↕	•	↕
	LSRB													54	2	1	B } 0 →	•	0	↕	•	↕
	LSR				04	6	2	64	6+	2+	74	7	3				M } 0 → b7 b0 c	•	0	↕	•	↕
MUL														3D	11	1	A × B → D (Unsigned)	•	•	↕	•	9
NEG	NEGA													40	2	1	$\overline{A}$ + 1 → A	8	↕	↕	↕	↕
	NEGB													50	2	1	$\overline{B}$ + 1 → B	8	↕	↕	↕	↕
	NEG				00	6	2	60	6+	2+	70	7	3				$\overline{M}$ + 1 → M	8	↕	↕	↕	↕
NOP														12	2	1	No Operation	•	•	•	•	•
OR	ORA	8A	2	2	9A	4	2	AA	4+	2+	BA	5	3				A V M → A	•	↕	↕	0	•
	ORB	CA	2	2	DA	4	2	EA	4+	2+	FA	5	3				B V M → B	•	↕	↕	0	•
	ORCC	1A	3	2													CC V IMM → CC				7	
PSH	PSHS	34	5+[4]	2													Push Registers on S Stack	•	•	•	•	•
	PSHU	36	5+[4]	2													Push Registers on U Stack	•	•	•	•	•
PUL	PULS	35	5+[4]	2													Pull Registers from S Stack	•	•	•	•	•
	PULU	37	5+[4]	2													Pull Registers from U Stack	•	•	•	•	•
ROL	ROLA													49	2	1	A } ← c b7 b0	•	↕	↕	↕	↕
	ROLB													59	2	1	B } ← c b7 b0	•	↕	↕	↕	↕
	ROL				09	6	2	69	6+	2+	79	7	3				M } ← c b7 b0	•	↕	↕	↕	↕
ROR	RORA													46	2	1	A } → c b7 b0	•	↕	↕	•	↕
	RORB													56	2	1	B } → c b7 b0	•	↕	↕	•	↕
	ROR				06	6	2	66	6+	2+	76	7	3				M } → c b7 b0	•	↕	↕	•	↕
RTI														3B	6/15	1	Return From Interrupt					7
RTS														39	5	1	Return from Subroutine	•	•	•	•	•
SBC	SBCA	82	2	2	92	4	2	A2	4+	2+	B2	5	3				A − M − C → A	8	↕	↕	↕	↕
	SBCB	C2	2	2	D2	4	2	E2	4+	2+	F2	5	3				B − M − C → B	8	↕	↕	↕	↕
SEX														1D	2	1	Sign Extend B into A	•	↕	↕	0	•
ST	STA				97	4	2	A7	4+	2+	B7	5	3				A → M	•	↕	↕	0	•
	STB				D7	4	2	E7	4+	2+	F7	5	3				B → M	•	↕	↕	0	•
	STD				DD	5	2	ED	5+	2+	FD	6	3				D → M:M + 1	•	↕	↕	0	•
	STS				10 DF	6	3	10 EF	6+	3+	10 FF	7	4				S → M:M + 1	•	↕	↕	0	•
	STU				DF	5	2	EF	5+	2+	FF	6	3				U → M:M + 1	•	↕	↕	0	•
	STX				9F	5	2	AF	5+	2+	BF	6	3				X → M:M + 1	•	↕	↕	0	•
	STY				10 9F	6	3	10 AF	6+	3+	10 BF	7	4				Y → M:M + 1	•	↕	↕	0	•
SUB	SUBA	80	2	2	90	4	2	A0	4+	2+	B0	5	3				A − M → A	8	↕	↕	↕	↕
	SUBB	C0	2	2	D0	4	2	E0	4+	2+	F0	5	3				B − M → B	8	↕	↕	↕	↕
	SUBD	83	4	3	93	6	2	A3	6+	2+	B3	7	3				D − M:M + 1 → D	•	↕	↕	↕	↕
SWI	SWI[6]													3F	19	1	Software Interrupt 1	•	•	•	•	•
	SWI2[6]													10 3F	20	2	Software Interrupt 2	•	•	•	•	•
	SWI3[6]													11 3F	20	1	Software Interrupt 3	•	•	•	•	•
SYNC														13	≥4	1	Synchronize to Interrupt	•	•	•	•	•
TFR	R1, R2	1F	6	2													R1 → R2[2]	•	•	•	•	•
TST	TSTA													4D	2	1	Test A	•	↕	↕	0	•
	TSTB													5D	2	1	Test B	•	↕	↕	0	•
	TST				0D	6	2	6D	6+	2+	7D	7	3				Test M	•	↕	↕	0	•

NOTES:

1. This column gives a base cycle and byte count. To obtain total count, add the values obtained from the INDEXED ADDRESSING MODE table, Table 2.
2. R1 and R2 may be any pair of 8 bit or any pair of 16 bit registers.
   The 8 bit registers are: A, B, CC, DP
   The 16 bit registers are: X, Y, U, S, D, PC
3. EA is the effective address.
4. The PSH and PUL instructions require 5 cycles plus 1 cycle for each byte pushed or pulled.
5. 5(6) means: 5 cycles if branch not taken, 6 cycles if taken (Branch instructions).
6. SWI sets I and F bits. SWI2 and SWI3 do not affect I and F.
7. Conditions Codes set as a direct result of the instruction.
8. Vaue of half-carry flag is undefined.
9. Special Case — Carry set if b7 is SET.

FIGURE 18 — PROGRAMMING AID (CONTINUED)

Branch Instructions

Instruction	Forms	Addressing Mode Relative OP	~ 5	#	Description	5 H	3 N	2 Z	1 V	0 C
BCC	BCC	24	3	2	Branch C = 0	•	•	•	•	•
	LBCC	10	5(6)	4	Long Branch	•	•	•	•	•
		24			C = 0					
BCS	BCS	25	3	2	Branch C = 1	•	•	•	•	•
	LBCS	10	5(6)	4	Long Branch	•	•	•	•	•
		25			C = 1					
BEQ	BEQ	27	3	2	Branch Z = 1	•	•	•	•	•
	LBEQ	10	5(6)	4	Long Branch	•	•	•	•	•
		27			Z = 0					
BGE	BGE	2C	3	2	Branch ≥ Zero	•	•	•	•	•
	LBGE	10	5(6)	4	Long Branch ≥ Zero	•	•	•	•	•
		2C								
BGT	BGT	2E	3	2	Branch > Zero	•	•	•	•	•
	LBGT	10	5(6)	4	Long Branch > Zero	•	•	•	•	•
		2E								
BHI	BHI	22	3	2	Branch Higher	•	•	•	•	•
	LBHI	10	5(6)	4	Long Branch Higher	•	•	•	•	•
		22								
BHS	BHS	24	3	2	Branch Higher or Same	•	•	•	•	•
	LBHS	10	5(6)	4	Long Branch Higher or Same	•	•	•	•	•
		24								
BLE	BLE	2F	3	2	Branch ≤ Zero	•	•	•	•	•
	LBLE	10	5(6)	4	Long Branch ≤ Zero	•	•	•	•	•
		2F								
BLO	BLO	25	3	2	Branch lower	•	•	•	•	•
	LBLO	10	5(6)	4	Long Branch Lower	•	•	•	•	•
		25								

Instruction	Forms	Addressing Mode Relative OP	~ 5	#	Description	5 H	3 N	2 Z	1 V	0 C
BLS	BLS	23	3	2	Branch Lower or Same	•	•	•	•	•
	LBLS	10	5(6)	4	Long Branch Lower or Same	•	•	•	•	•
		23								
BLT	BLT	2D	3	2	Branch < Zero	•	•	•	•	•
	LBLT	10	5(6)	4	Long Branch < Zero	•	•	•	•	•
		2D								
BMI	BMI	2B	3	2	Branch Minus	•	•	•	•	•
	LBMI	10	5(6)	4	Long Branch Minus	•	•	•	•	•
		2B								
BNE	BNE	26	3	2	Branch Z = 0	•	•	•	•	•
	LBNE	10	5(6)	4	Long Branch	•	•	•	•	•
		26			Z ≠ 0					
BPL	BPL	2A	3	2	Branch Plus	•	•	•	•	•
	LBPL	10	5(6)	4	Long Branch Plus	•	•	•	•	•
		2A								
BRA	BRA	20	3	2	Branch Always	•	•	•	•	•
	LBRA	16	5	3	Long Branch Always	•	•	•	•	•
BRN	BRN	21	3	2	Branch Never	•	•	•	•	•
	LBRN	10	5	4	Long Branch Never	•	•	•	•	•
		21								
BSR	BSR	8D	7	2	Branch to Subroutine	•	•	•	•	•
	LBSR	17	9	3	Long Branch to Subroutine	•	•	•	•	•
BVC	BVC	28	3	2	Branch V = 0	•	•	•	•	•
	LBVC	10	5(6)	4	Long Branch	•	•	•	•	•
		28			V = 0					
BVS	BVS	29	3	2	Branch V = 1	•	•	•	•	•
	LBVS	10	5(6)	4	Long Branch	•	•	•	•	•
		29			V = 1					

SIMPLE BRANCHES

	OP	~	#
BRA	20	3	2
LBRA	16	5	3
BRN	21	3	2
LBRN	1021	5	4
BSR	8D	7	2
LBSR	17	9	3

SIMPLE CONDITIONAL BRANCHES (Notes 1-4)

Test	True	OP	False	OP
N = 1	BMI	2B	BPL	2A
Z = 1	BEQ	27	BNE	26
V = 1	BVS	29	BVC	28
C = 1	BCS	25	BCC	24

SIGNED CONDITIONAL BRANCHES (Notes 1-4)

Test	True	OP	False	OP
r > m	BGT	2E	BLE	2F
r ≥ m	BGE	2C	BLT	2D
r = m	BEQ	27	BNE	26
r ≤ m	BLE	2F	BGT	2E
r < m	BLT	2D	BGE	2C

UNSIGNED CONDITIONAL BRANCHES (Notes 1-4)

Test	True	OP	False	OP
r > m	BHI	22	BLS	23
r ≥ m	BHS	24	BLO	25
r = m	BEQ	27	BNE	26
r ≤ m	BLS	23	BHI	22
r < m	BLO	25	BHS	24

NOTES:

1. All conditional branches have both short and long variations.
2. All short branches are 2 bytes and require 3 cycles.
3. All conditional long branches are formed by prefixing the short branch opcode with $10 and using a 16-bit destination offset.
4. All conditional long branches require 4 bytes and 6 cycles if the branch is taken or 5 cycles if the branch is not taken.
5. 5(6) means: 5 cycles if branch not taken, 6 cycles if taken.

INDEXED ADDRESSING MODES

Type	Forms	Nondirect Assembler Form	Nondirect Post-Byte Opcode	Nondirect + ~	Nondirect + #	Indirect Assembler Form	Indirect Post-Byte Opcode	Indirect + ~	Indirect + #
Constant Offset From R	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	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	1	[n, PCR]	1XX11101	8	2
Extended Indirect	16-Bit Address	–	–	–	–	[n]	10011111	5	2

R = X, Y, U, or S
X = Don't Care

RR: 00 = X 10 = U
01 = Y 11 = S

INDEXED ADDRESSING POSTBYTE REGISTER BIT ASSIGNMENTS

Post-Byte Register Bit 7	6	5	4	3	2	1	0	Indexed Addressing Mode
0	R	R	x	x	x	x	x	EA = , R + 5 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+ 8-Bit Offset
1	R	R	I	1	0	0	1	EA = , R + 16-Bit Offset
1	R	R	I	1	0	1	1	EA = , R + D Offset
1	x	x	I	1	1	0	0	EA = , PC + 8-Bit Offset
1	x	x	I	1	1	0	1	EA = , PC + 16-Bit Offset
1	R	R	1	1	1	1	1	EA = [, Address]

Addressing Mode Field

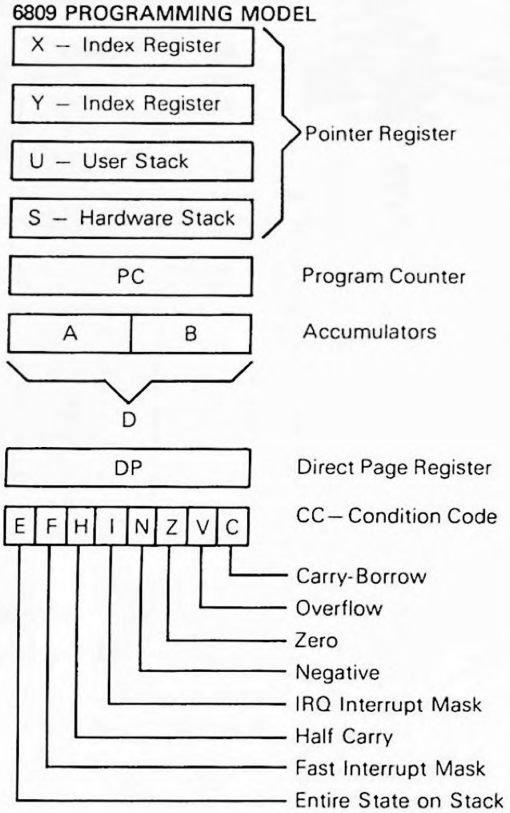
Indirect Field

(Sign bit when b_7 = 0)

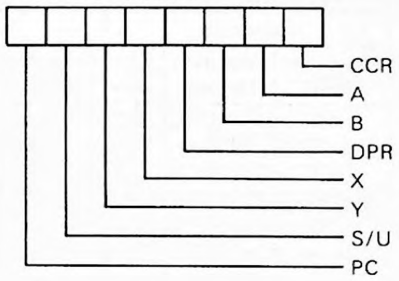
Register Field: RR
00 = X
01 = Y
10 = U
11 = S

X = Don't Care

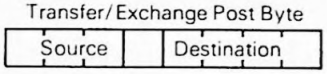
6809 PROGRAMMING MODEL



Push/Pull Post Byte



Transfer/Exchange Post Byte



Register Field

0000 = D (A-B)	0101 = PC
0001 = X	1000 = A
0010 = Y	1001 = B
0011 = U	1010 = CCR
0100 = S	1011 = DPR

6809 Stacking Order



6809 Vectors	
FFFE	Restart
FFFC	NMI
FFFA	SWI
FFF8	IRQ
FFF6	FIRQ
FFF4	SW12
FFF2	SW13
FFF0	Reserved

ORDERING INFORMATION

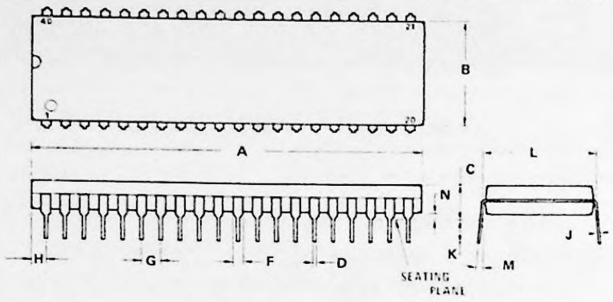
Package Type	Frequency	Temperature Range	Order Number
Ceramic L Suffix	1.0 MHz	0°C to 70°C	MC6809EL
	1.0 MHz	−40°C to 85°C	MC6809ECL
	1.5 MHz	0°C to 70°C	MC68A09EL
	1.5 MHz	−40°C to 85°C	MC68A09ECL
	2.0 MHz	0°C to 70°C	MC68B09EL
	2.0 MHz	−40°C to 85°C	MC68B09ECL
Plastic P Suffix	1.0 MHz	0°C to 70°C	MC6809EP
	1.0 MHz	−40°C to 85°C	MC6809ECP
	1.5 MHz	0°C to 70°C	MC68A09EP
	1.5 MHz	−40°C to 85°C	MC68A09ECP
	2.0 MHz	0°C to 70°C	MC68B09EP
	2.0 MHz	−40°C to 85°C	MC68B09ECP
Cerdip S Suffix	1.0 MHz	0°C to 70°C	MC6809ES
	1.0 MHz	−40°C to 85°C	MC6809ECS
	1.5 MHz	0°C to 70°C	MC68A09ES
	1.5 MHz	−40°C to 85°C	MC68A09ECS
	2.0 MHz	0°C to 70°C	MC68B09ES
	2.0 MHz	−40°C to 85°C	MC68B09ECS

BETTER PROGRAM

BETTER program processing is available on all types listed.
Add suffix letters to part number.

Level 1 add "S" Level 2 add "D" Level 3 add "DS
Level 1 "S" = 10 Temp Cycles−(−25 to 150°C).
Hi Temp testing at T_A max.
Level 2 "D" = 168 Hour Burn-in at 125°C
Level 3 "DS" = Combination of Level 1 and 2.

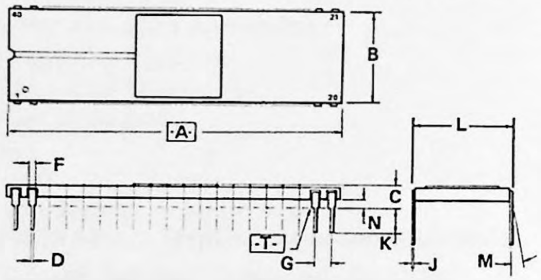
PACKAGE DIMENSIONS



DIM	MILLIMETERS MIN	MILLIMETERS MAX	INCHES MIN	INCHES MAX
A	51.69	52.45	2.035	2.065
B	13.72	14.22	0.540	0.560
C	3.94	5.08	0.155	0.200
D	0.36	0.56	0.014	0.022
F	1.02	1.52	0.040	0.060
G	2.54 BSC		0.100 BSC	
H	1.65	2.16	0.065	0.085
J	0.20	0.38	0.008	0 015
K	2.92	3.43	0.115	0.135
L	15.24 BSC		0.600 BSC	
M	0°	15°	0°	15°
N	0.51	1.02	0.020	0.040

NOTES:
1. POSITIONAL TOLERANCE OF LEADS (D), SHALL BE WITHIN 0.25 mm (0.010) AT MAXIMUM MATERIAL CONDITION, IN RELATION TO SEATING PLANE AND EACH OTHER.
2. DIMENSION L TO CENTER OF LEADS WHEN FORMED PARALLEL.
3. DIMENSION B DOES NOT INCLUDE MOLD FLASH.

P SUFFIX
PLASTIC PACKAGE
CASE 711-03



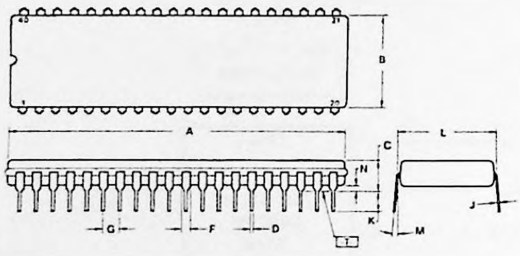
DIM	MILLIMETERS MIN	MILLIMETERS MAX	INCHES MIN	INCHES MAX
A	50.29	51.31	1.980	2.020
B	14.63	15.49	0.576	0.610
C	2.79	4.32	0.110	0.170
D	0.38	0.53	0.015	0.021
F	0.76	1.52	0.030	0.060
G	2.54 BSC		0.100 BSC	
J	0.20	0.33	0.008	0.013
K	2.54	4.57	0.100	0.180
L	14.99	15.65	0.590	0.616
M	–	10°	–	10°
N	1.02	1.52	0.040	0.060

NOTES:
1. DIMENSION -A- IS DATUM.
2. POSITIONAL TOLERANCE FOR LEADS:

⌖ | 0.25 (0.010) Ⓜ | T | A Ⓜ

3. -T- IS SEATING PLANE.
4. DIMENSION "L" TO CENTER OF LEADS WHEN FORMED PARALLEL.
5. DIMENSIONING AND TOLERANCING PER ANSI Y14.5, 1973.

L SUFFIX
CERAMIC PACKAGE
CASE 715-05



DIM	MILLIMETERS MIN	MILLIMETERS MAX	INCHES MIN	INCHES MAX
A	51.31	53.24	2.020	2.096
B	12.70	15.49	0.500	0.610
C	4.06	5.84	0.160	0.230
D	0.38	0.56	0.015	0.022
F	1.27	1.65	0.050	0.065
G	2.54 BSC		0.100 BSC	
J	0.20	0.30	0.008	0.012
K	3.18	4.06	0.125	0.160
L	15.24 BSC		0.600 BSC	
M	5°	15°	5°	15°
N	0.51	1.27	0.020	0.050

NOTES:
1. DIMENSION-A-IS DATUM.
2. POSITIONAL TOLERANCE FOR LEADS:

⌖ | ∅ 0.25 (0.010) Ⓜ | T | A Ⓜ

3. -T- IS SEATING PLANE.
4. DIMENSION L TO CENTER OF LEADS WHEN FORMED PARALLEL.
5. DIMENSION A AND B INCLUDES MENISCUS.

S SUFFIX
CERDIP PACKAGE
CASE 734-03

Motorola reserves the right to make changes to any products herein to improve reliability, function or design. Motorola does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights nor the rights of others.

MOTOROLA Semiconductor Products Inc.

3501 ED BLUESTEIN BLVD., AUSTIN, TEXAS 78721 • A SUBSIDIARY OF MOTOROLA INC.

AI3321-2 PRINTED IN USA 8-82 IMPERIAL LITHO 07803 10,000

MOTOROLA
SEMICONDUCTORS
P.O. BOX 20912 • PHOENIX, ARIZONA 85036

SN74LS783
MC6883

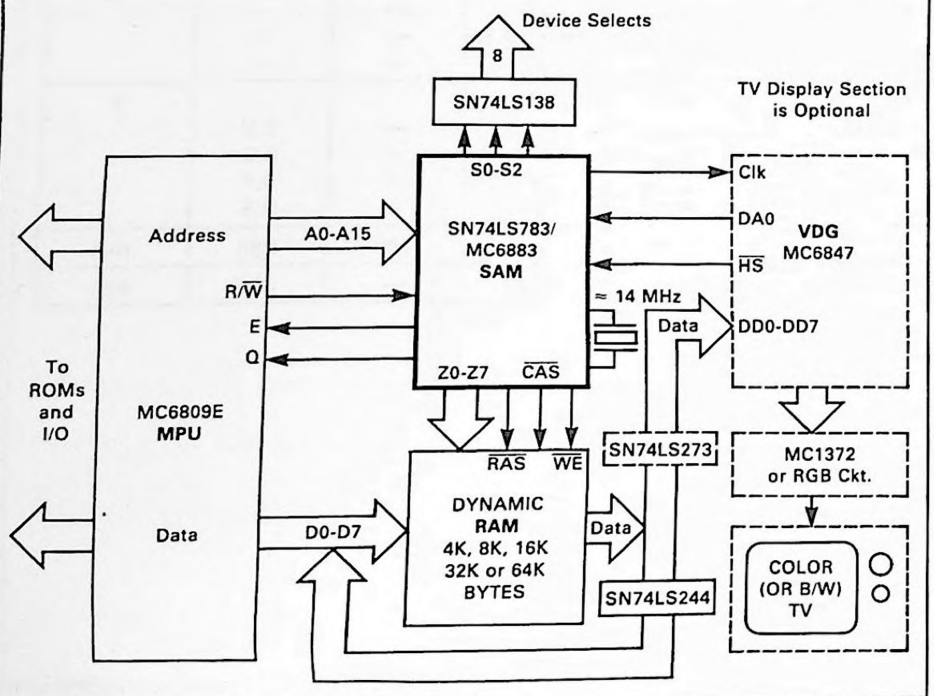
Advance Information

SYNCHRONOUS ADDRESS MULTIPLEXER

The SN74LS783/MC6883 brings together the MC6809E (MPU), the MC6847 (Color Video Display Generator) and dynamic RAM to form a highly effective, compact and cost effective computer and display system.

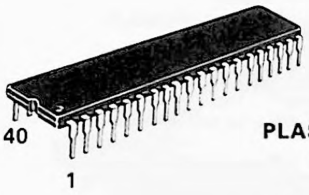
- MC6809E, MC6800, MC6801E, MC68000 and MC6847 (VDG) Compatible
- Transparent MPU/VDG/Refresh
- RAM size — 4K, 8K, 16K, 32K or 64K Bytes (Dynamic or Static)
- Addressing Range — 96K Bytes
- Single Crystal Provides All Timing
- Register Programmable:
  - VDG Addressing Modes
  - VDG Offset (0 to 64K)
  - RAM Size
  - Page Switch
  - MPU Rate (Crystal ÷ 16 or ÷ 8)
  - MPU Rate (Address Dependent or Independent)
- System "Device Selects" Decoded 'On Chip'
- Timing is Optimized for Standard Dynamic RAMs
- +5.0 V Only Operation
- Easy Synchronization of Multiple SAM Systems
- DMA Mode

SYSTEM BLOCK DIAGRAM

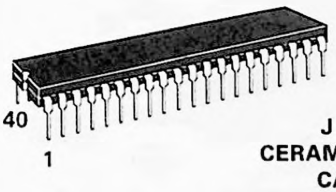


SYNCHRONOUS ADDRESS MULTIPLEXER

LOW POWER SCHOTTKY

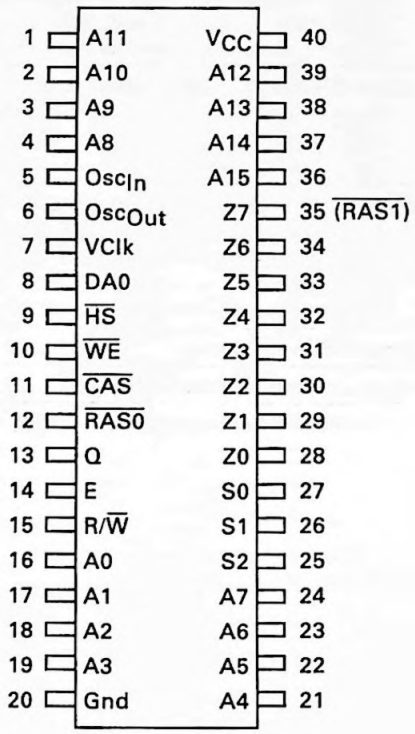


N SUFFIX
PLASTIC PACKAGE
CASE 711



J SUFFIX
CERAMIC PACKAGE
CASE 734

PIN ASSIGNMENT



This document contains information on a new product. Specifications and information herein are subject to change without notice.

© MOTOROLA INC., 1981

ADI-595
(Replacing NP-118)

MAXIMUM RATINGS (T_A = 25°C unless otherwise noted.)

Rating	Symbol	Value	Unit
Power Supply Voltage	V_{CC}	−0.5 to +7.0	Vdc
Input Voltage (Except Osc_{In})	V_I	−0.5 to 10	Vdc
Input Current (Except Osc_{In})	I_I	−30 to +5.0	mA
Output Voltage	V_O	−0.5 to +7.0	Vdc
Operating Ambient Temperature Range	T_A	0 to +70	°C
Storage Temperature Range	T_{stg}	−65 to +150	°C
Input Voltage Osc_{In}	$V_{IOsc_{In}}$	−0.5 to V_{CC}	Vdc
Input Current Osc_{In}	$I_{IOsc_{In}}$	−0.5 to +5.0	mA

RECOMMENDED OPERATING CONDITIONS

Rating	Symbol	Value	Unit
Power Supply Voltage	V_{CC}	4.75 to 5.25	Vdc
Operating Ambient Temperature Range	T_A	0 to +70	°C

DC CHARACTERISTICS (Unless otherwise noted specifications apply over recommended power supply and temperature ranges.)

Characteristic	Symbol	Min	Typ	Max	Units
Input Voltage — High Logic State	V_{IH}	2.0	—	—	V
Input Voltage — Low Logic State	V_{IL}	—	—	0.8	V
Input Clamp Voltage (V_{CC} = Min, I_{in} = −18 mA) All Inputs Except Osc_{In}	V_{IK}	—	—	−1.5	V
Input Current — High Logic State at Max Input Voltage	I_I				µA
(V_{CC} = Max, V_{in} = 5.25 V) VClk Input		—	—	200	
(V_{CC} = Max, V_{in} = 5.25 V) DA0 Input		—	—	100	
(V_{CC} = Max, V_{in} = 5.25 V) Osc_{Out} Input		—	—	250	
(V_{CC} = Max, V_{in} = 7.0 V) All Other Inputs Except Osc_{In}		—	—	100	
Input Current High Logic State (V_{CC} = Max, V_{in} = 2.7 V) All Inputs Except VClk, Osc_{In}*	I_{IH}	—	—	20	µA
Input Current — Low Logic State	I_{IL}				mA
(V_{CC} = Max, V_{in} = 0.4 V) DA0 Input		—	—	−1.2	
(V_{CC} = Max, V_{in} = 0.4 V) VClk Input		—	−30	−60	
(V_{CC} = Max, V_{in} = 0.4 V, Osc_{In} = Gnd) Osc_{Out} Input		—	—	−8	
(V_{CC} = Max, V_{in} = 0.4 V) All Other Inputs Except Osc_{In}		—	—	−.4	
Output Voltage — High Logic State					V
(V_{CC} = Min, I_{OH} = −1.0 mA) $\overline{RAS0}$, $\overline{RAS1}$, $\overline{CAS}$, $\overline{WE}$	$V_{OH(C)}$	3.0	—	—	
(V_{CC} = Min, I_{OH} = −0.2 mA) E, Q	$V_{OH(E)}$	V_{CC} − 0.75	—	—	
(V_{CC} = Min, I_{OH} = −0.2 mA) All Other Outputs	V_{OH}	2.7	—	—	
Output Voltage — Low Logic State					V
(V_{CC} = Min, I_{OL} = 8.0 mA) $\overline{RAS0}$, $\overline{RAS1}$, $\overline{CAS}$, $\overline{WE}$	$V_{OL(C)}$	—	—	0.5	
(V_{CC} = Min, I_{OL} = 4.0 mA) E, Q Outputs	$V_{OL(E)}$	—	—	0.5	
(V_{CC} = Min, I_{OL} = 0.8 mA) VClk Output	$V_{OL(V)}$	—	—	0.6	
(V_{CC} = Min, I_{OL} = 4.0 mA) All Other Outputs	V_{OL}	—	—	0.5	
Power Supply Current	I_{CC}	—	180	230	mA
Output Short-Circuit Current	I_{OS}	30	—	225	mA

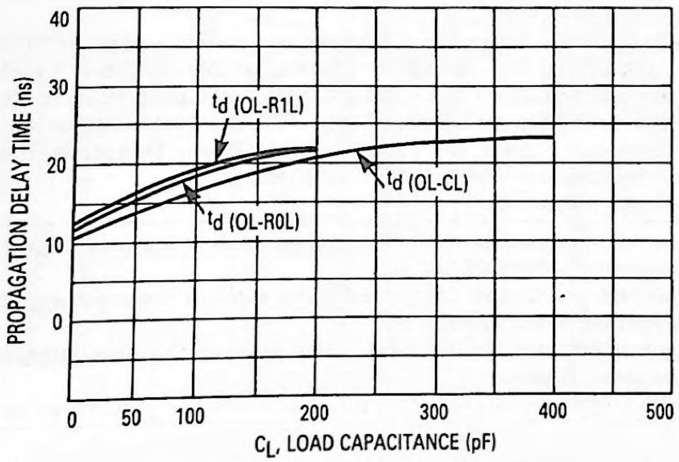
*Including Osc_{Out} (when Osc_{In} is grounded).

AC CHARACTERISTICS (4.75 V≤V_{CC}≤5.25 V and 0≤T_A≤70°C, unless otherwise noted).

Characteristic	Symbol	Min	Typ	Max	Units
Propagation Delay Times					ns
(See Circuit in Figure 9) Oscillator-In ↘ to Oscillator-Out ↗	$t_{d(OL-OH)}$	—	3.0	—	
Oscillator-In ↗ to Oscillator-Out ↘	$t_{d(OH-OL)}$	—	20	—	
(C_L = 195 pF) A0 thru A15 to Z0, Z1, Z2 thru Z7	$t_{d(A-Z)}$	—	28	—	
(C_L = 30 pF) A0 thru A15, R/W to S0, S1, S3	$t_{d(A-S)}$	—	18	—	
(C_L = 95 pF) Oscillator-Out ↘ to $\overline{RAS0}$ ↗	$t_{d(OL-R0H)}$	—	20	—	
(C_L = 95 pF) Oscillator-Out ↘ to $\overline{RAS0}$ ↘	$t_{d(OL-R0L)}$	—	18	—	
(C_L = 95 pF) Oscillator-Out ↘ to $\overline{RAS1}$ ↗	$t_{d(OL-R1H)}$	—	22	—	
(C_L = 95 pF) Oscillator-Out ↘ to $\overline{RAS1}$ ↘	$t_{d(OL-R1L)}$	—	20	—	
(C_L = 195 pF) Oscillator-Out ↘ to $\overline{CAS}$ ↗	$t_{d\ (OL-CH)}$	—	20	—	
(C_L = 195 pF) Oscillator-Out ↘ to $\overline{CAS}$ ↘	$t_{d(OL-CL)}$	—	20	—	
(C_L = 195 pF) Oscillator-Out ↘ to $\overline{WE}$ ↗	$t_{d(OL-WH)}$	—	22	—	
(C_L = 195 pF) Oscillator-Out ↘ to $\overline{WE}$ ↘	$t_{d(OL-WL)}$	—	40	—	
(C_L = 100 pF) Oscillator-Out ↘ to E ↗	$t_{d(OL-EH)}$	—	55	—	
(C_L = 100 pF) Oscillator-Out ↘ to E ↘	$t_{d(OL-EL)}$	—	25	—	
(C_L = 100 pF) Oscillator-Out ↘ to Q ↗	$t_{d(OL-QH)}$	—	55	—	
(C_L = 100 pF) Oscillator-Out ↘ to Q ↘	$t_{d(OL-QL)}$	—	25	—	
(C_L = 30 pF) Oscillator-Out ↗ to VClk ↗	$t_{d(OH-VH)}$	—	50	—	
(C_L = 30 pF) Oscillator-Out ↗ to VClk ↘	$t_{d(OH-VL)}$	—	65	—	
(C_L = 195 pF) Oscillator-Out ↘ to Row Address	$t_{d(OL-AR)}$	—	36	—	
(C_L = 195 pF) Oscillator-Out ↘ to Column Address	$t_{d(OL-AC)}$	—	33	—	
(C_L = 15 pF) Oscillator-Out ↘ to DA0 ↗ Earliest(1)	$t_{d(OL-DH)}$	—	−15	—	
(C_L = 15 pF) Oscillator-Out ↘ to DA0 ↗ Latest(1)	$t_{d(OL-DH)}$	—	+15	—	
(C_L = 95 pF on $\overline{RAS}$, C_L = 195 pF on $\overline{CAS}$) $\overline{CAS}$ ↘ to $\overline{RAS}$ ↗	$t_{d(CL-RH)}$	—	208	—	
Setup Time for A0 thru A15, R/$\overline{W}$ Rate = ÷16	$t_{su(A)}$	—	28	—	ns
Rate = ÷ 8		—	28		
Hold Time for A0 thru A15, R/$\overline{W}$ Rate = ÷16	$t_{h(A)}$	—	30	—	ns
Rate = ÷ 8		—	30	—	
Width of $\overline{HS}$ Low 2	$t_{WL(HS)}$	2.0	5.0	6.0	µs

Notes: 1. When using the SAM with an MC6847, the rising edge of DA0 is confined within the range shown in the timing diagrams (unless the synchronizing process is incomplete.) The synchronization process requires a maximum of 32 cycles of Osc_{Out} for completion.
2. $t_{WL(HS)}$ wider than 6.0 µs may yield more than 8 sequential refresh addresses.

FIGURE 1 — PROPAGATION DELAY TIMES VERSUS LOAD CAPACITANCE



PIN DESCRIPTION TABLE

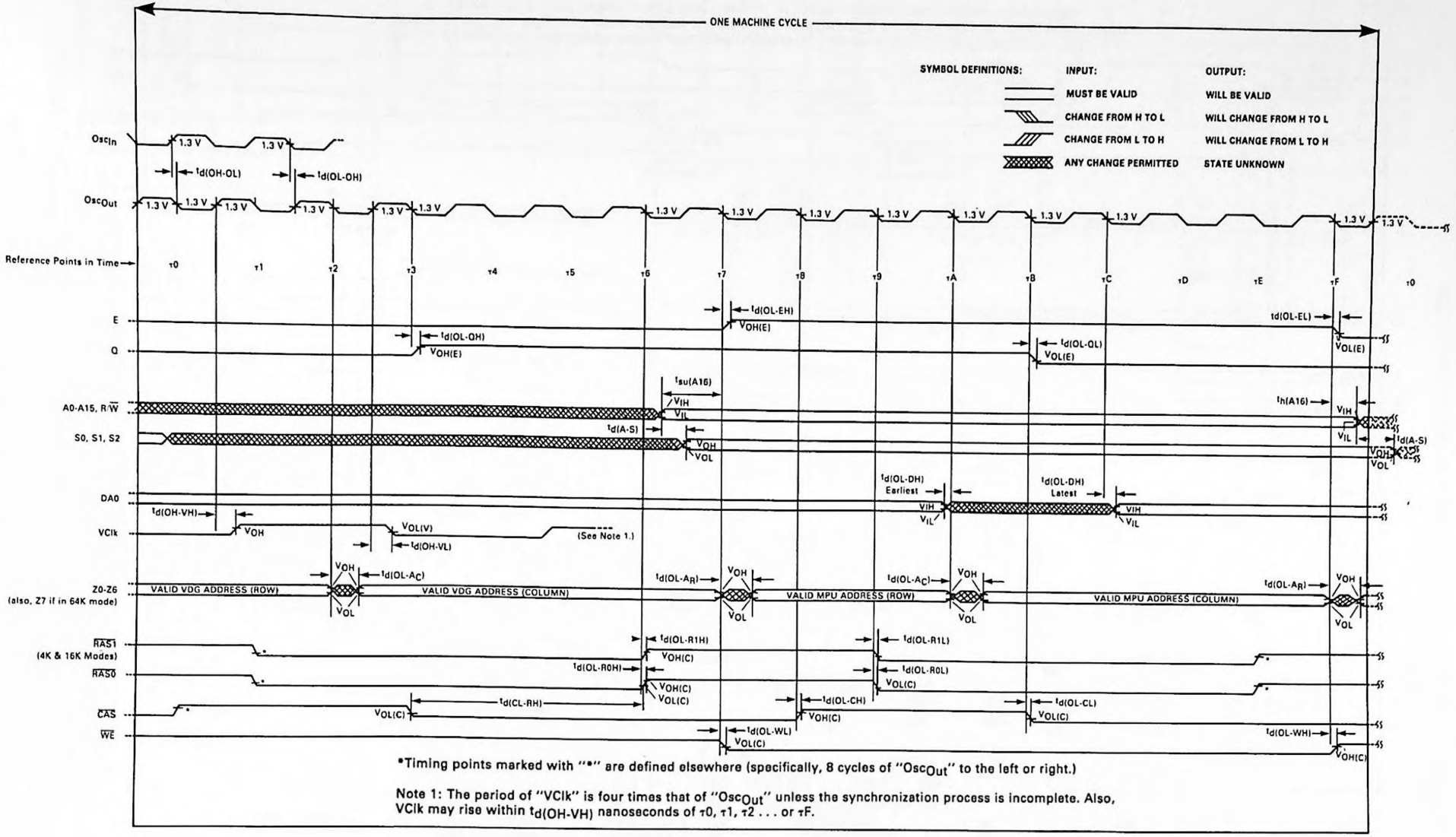
		Name	No.	Function
Input Pins	Power	V_{CC}	40	Apply + 5 volts ± 5%. SAM draws less than 230 mA.
		Gnd	20	Return Ground for +5 volts.
	MPU Address and Control	A15	36	Most Significant Bit.
		A14	37	
		A13	38	MPU address bits A0-A15. These 16 signals come directly from the MPU and are used to directly address up to 64K memory locations or to indirectly address up to 96K memory locations. (See pages 17 and 18 for memory maps). Each input is approximately equivalent to one low power Schottky load.
		A12	39	
		A11	1	
		A10	2	
		A9	3	
		A8	4	
		A7	24	
		A6	23	
		A5	22	
		A4	21	
		A3	19	
		A2	18	
		A1	17	
		A0	16	Least Significant Bit.
		R/$\overline{W}$	15	MPU READ or $\overline{WRITE}$. This signal comes directly from the MPU and is used to enable writing to the SAM control register, dynamic RAM (via $\overline{WE}$), and to enable device select #0.
		Osc_{In}	5	Apply 14.31818* MHz crystal and 2.5–30 pF trimmer to ground. See page 12.
	VDG Control	DA0	8	Display Address DA0. The primary function of this pin is to input the least significant bit of a 16-bit video display address. The more significant 15-bits are outputs from an internal 15-bit counter which is clocked by DA0. The secondary function of this pin is to indirectly input the logic level of the VDG "$\overline{FS}$" (field synchronization pulse) for vertical video address updating.
		$\overline{HS}$	9	Horizontal Synchronization. The primary function of this pin is to detect the falling edge of VDG "$\overline{HS}$" pulse in order to initiate eight dynamic RAM refresh cycles. The secondary function is to reset up to 4 least significant bits of the internal video address counter.
		VClk	7	VDG Clock. The primary function of this pin is to **output** a 3.579545 MHz square wave** to the VDG "Clk" pin. The secondary function resets the SAM when this VClk pin is pulled to logic "0" level, acting as an **input**.
Output Pins		Osc_{Out}	6	Apply 1.5 kΩ resistor to 14.31818* MHz crystal and 33 pF capacitor to ground. See page 12.
	Device Selects	S2	25	Most Significant Bit (Device Select Bits). The binary value of S2, S1, S0 selects one of eight "chunks" of MPU address space (numbers 0 through 7). Varying in length, these "chunks" provide efficient memory mapping for ROMs, RAMs, Input/Output devices, and MPU Vectors. (Requires 74LS 138-type demultiplexer).
		S1	26	
		S0	27	Least Significant Bit.
	MPU Clocks	E	14	E (Enable Clock) "E" and "Q" are 90° out of phase and are both used as MPU clocks for the MC6809E. For the MC6800 and MC6801E, only "E" is used. "E" is also used for many MC6800 peripheral chips.
		Q	13	Q (Quadrature Clock).
	RAM Address	Z7†	35	Most Significant Bit
		Z6†	34	First, the least significant address bits from the MPU or "VDG" are presented to Z0–Z5 (4K x 1 RAMs) or Z0–Z6 (16K x 1 RAMs) or Z0–Z7 (64K x 1 RAMs). Next, the most significant address bits from the MPU or "VDG" are presented to Z0–Z5 (4K x 1 RAMs) or Z0–Z6 (16K x 1 RAMs) or Z0–Z7 (64K x 1 RAMs). Note that for 4K x 1 and 16K x 1 RAMs, Z7 (Pin 35) is not needed for address information. Therefore, Pin 35 is used for a second row address select which is labeled ($\overline{RAS1}$).
		Z5†	33	
		Z4†	32	
		Z3†	31	
		Z2†	30	
		Z1†	29	
		Z0†	28	Least Significant Bit.
	RAM Control	$\overline{RAS1}$†	35	Row Address Strobe One. This pulse strobes the least significant 6,7 or 8 address bits into dynamic RAMs in Bank #1.
		$\overline{RAS0}$†	12	Row Address Strobe Zero. This pulse strobes the least significant 6,7 or 8 address bits into dynamic RAMs in Bank #0.
		$\overline{CAS}$†	11	Column Address Strobe. This pulse strobes the most significant 6,7 or 8 address bits into dynamic RAMs.
		$\overline{WE}$†	10	Write Enable. When low, this pulse enables the MPU to write into dynamic RAM.

*14.31818 MHz is 4 times 3.579545 MHz television color subcarrier. Other frequencies may be used. (See page 12.)

**When VDG and SAM are not yet synchronized the "square wave" will stretch (see page 10.)

† Due to fast transitions, ferrite beads in series with these outputs may be necessary to avoid high frequency (≈ 60 MHz) resonances.

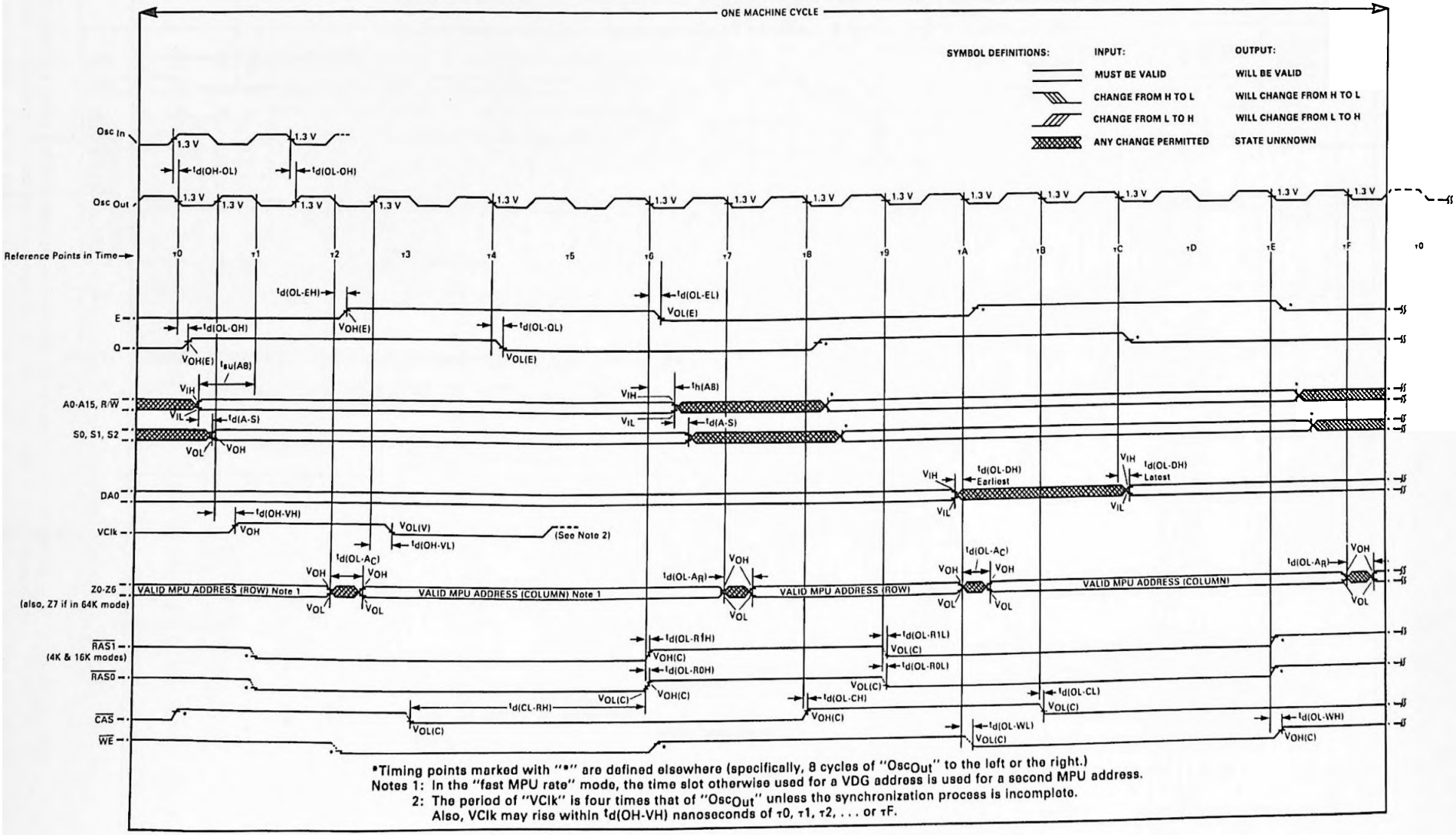
FIGURE 2 — TIMING WAVEFORMS for MPU RATE = SLOW



Timing points marked with "" are defined elsewhere (specifically, 8 cycles of "Osc_{Out}" to the left or right.)

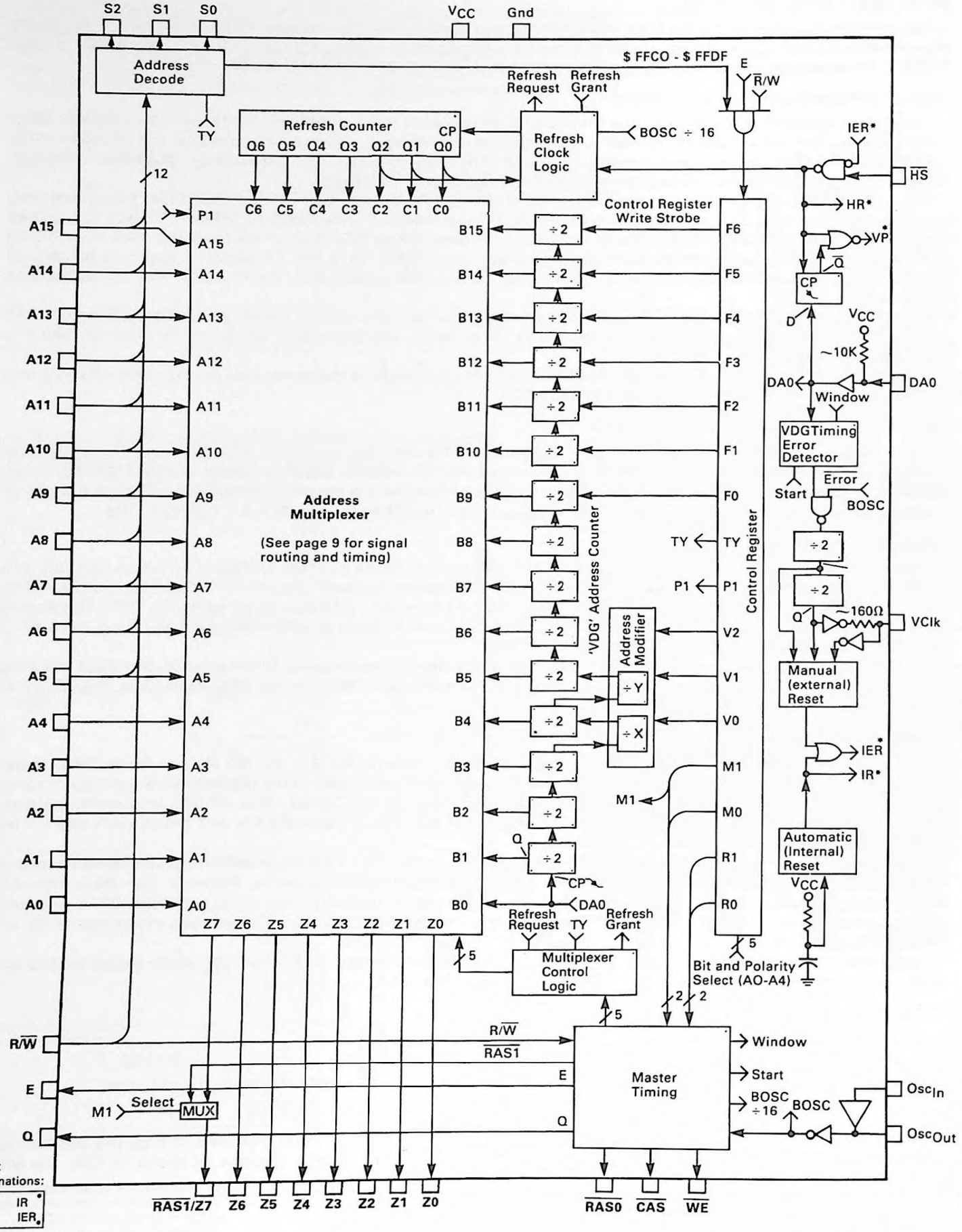
Note 1: The period of "VClk" is four times that of "Osc_{Out}" unless the synchronization process is incomplete. Also, VClk may rise within $t_{d(OH-VH)}$ nanoseconds of τ0, τ1, τ2 . . . or τF.

FIGURE 3— TIMING WAVEFORMS for MPU RATE = FAST



Timing points marked with "" are defined elsewhere (specifically, 8 cycles of "Osc_{Out}" to the left or the right.)
Notes 1: In the "fast MPU rate" mode, the time slot otherwise used for a VDG address is used for a second MPU address.
2: The period of "VClk" is four times that of "Osc_{Out}" unless the synchronization process is incomplete.
Also, VClk may rise within $t_{d(OH-VH)}$ nanoseconds of τ0, τ1, τ2, . . . or τF.

FIGURE 4 — SAM BLOCK DIAGRAM



See Text . . . Page 8.

SAM BLOCK DIAGRAM DESCRIPTION

MPU Addresses (A0 – A15):

These 16 signals come directly from the MPU and are used to directly address up to 64K memory locations (K = 1024) or to indirectly address up to 96K memory locations, by using a paging bit "P" (see pages 17 and 18 for memory maps.) Each input is approximately equivalent to one low power Schottky load.

VDG Address Counter (B0 – B15):

These 16 signals are derived from one input (DA0) which is the least significant bit of the VDG address. Most of the counter is simply binary. However, to duplicate the various addressing modes of the MC6847 VDG, ADDRESS MODIFIER logic is used. Selected by three VDG mode bits (V2, V1, and V0) from the SAM CONTROL REGISTER, eight address modifications are obtained as shown in Figure 5.

Also, notice that bits B9-B15 may be loaded from bits F0-F6 from the CONTROL REGISTER. This allows the starting address of the VDG display to be offset (in ½K increments) from $0000 to $FFFF†. B9-B15 are loaded when a VERTICAL PRE-LOAD(VP) pulse is generated. VP goes active (high) when $\overline{HS}$ from the VDG rises if DA0 is high (or a high impedance.) This condition should occur only while the TV electron beam is in vertical blanking and is simply implemented by connecting $\overline{FS}$ and $\overline{MS}$ together on the MC6847. The VP pulse also **clears** bits B1 – B8.

Finally, a HORIZONTAL RESET (HR) pulse may also affect the counter by clearing bits B1 – B3 or B1 – B4 when $\overline{HS}$ from the VDG is LOW (see Figure 5.) The HR pulse should occur only while the TV electron beam is in horizontal blanking.

In summary, DA0 clocks the VDG ADDRESS COUNTER; HR initializes the horizontal portion and VP initializes the vertical portion of the VDG ADDRESS COUNTER.

REFresh Address Counter (C0 – C6):

A seven bit binary counter with outputs labeled C0 – C6 supplies bursts of eight* sequential addresses triggered by a $\overline{HS}$ high to low transition. Thus, while the TV electron beam is in horizontal blanking, eight sequential addresses are accessed. Likewise, the next eight addresses are accessed during the next horizontal blanking period, etc. In this manner, all 128 addresses are refreshed in less than 1.1 milliseconds.

Address Multiplexer:

Occupying a large portion of the block diagram in Figure 4, is the address multiplexer which outputs bits Z0-Z7 (as addresses to dynamic RAM's.) Inputs to the address multiplexer include the VDG address (B0 – B15) the REFresh address (C0- C6) and the MPU address (A0 – A15) or (A0 – A14 plus one paging bit "P".) The paging bit "P" is one bit in the SAM CONTROL REGISTER that is used in place of A15 when memory map TYpe #0 is selected (via the SAM CONTROL REGISTER "TY" bit.)

Figure 6 shows which inputs are routed to Z0 – Z7 and **when** the routing occurs relative to one SAM machine cycle. Notice that Z7 and $\overline{RAS1}$ share the same pin. Z7 is selected if "M1" in the SAM CONTROL REGISTER IS HIGH (Memory size = 64K.)

Address Decode:

At the top left of Figure 4, is the Address Decode block. Outputs S2, S1, and S0 form a three bit encoded binary word(S). Thus S may be one of eight values (0 through 7) with each value representing a different range of MPU addresses. (To enable peripheral ROM's or I/O, decode the S2, S1, and S0 bits into eight seperate signals by using a 74LS138, 74LS155 or 74LS156. Notice that S2, S1, and S0 are **not** gated with any timing signals such as E or Q.)

Along with the A5 – A15 inputs is the MEMORY MAP TYpe bit (TY.) This bit is soft-programmable (as are **all** 16 bits in the SAM CONTROL REGISTER,) and selects one of two memory maps. Memory map #0 is intended to be used in systems that are primarily **ROM** based. Whereas, memory map #1 is intended for a primarily **RAM** based system with 64K contiguous RAM locations (minus 256 locations.) The various meanings of S2, S1, S0 are tabulated in Figure 16 (page 19) and again on pages 17 and 18.

In addition to S2, S1, and S0 outputs is a decode of $FFC0 through $FFDF which, when gated with E and $\overline{R}$/W, results in the write strobe for the SAM CONTROL REGISTER.

SAM Control Register

As shown in Figure 4, the CONTROL REGISTER has 16 "outputs":

VDG Addressing Modes:	V2, V1, V0	MPU Rate:	R1, R0
VDG Address OFFset:	F6, F5, F4, F3, F2, F1, F0	Memory Size (RAM):	M1, M0
32K Page Switch:	P	Memory Map TYpe:	TY

When the SAM is reset (see page 10,) all 16 bits are cleared. To **set** any one of these 16 bits, the MPU **writes** to a unique** **odd** address (within $FFC1 through $FFDF.) To **clear** any one of these 16 bits, the MPU

* If $\overline{HS}$ is held low longer than 8 µs, then the number of sequential addresses in one refresh "BURST" is proportional to the time interval during which $\overline{HS}$ is low.

** See pages 17 or 18 for specific addresses.

† In this document, the "$" symbol always preceeds hexidecimal characters.

simply writes to a unique** **even** address (within $FFCO through $FFDE.) Note that the **data** on the MPU data bus is irrelevant.

Inputs to the control register include A4, A3, A2, A1 (which are used to select **which one** of 16 bits is to be cleared or set), A0 (which determines the polarity . . . clear or set,) and R/W, E and $FFCO – $FFDF (which restrict the method, timing and addresses for changing one of the 16 bits.) For more detailed descriptions of the purposes of the 16 control bits, refer to related sections in the BLOCK DIAGRAM DESCRIPTION (pages 8 through 12) and the PROGRAMMING GUIDE (pages 14 through 18).

** See pages 17 or 18 for specific addresses.

FIGURE 5 — VDG ADDRESS MODIFIER

Mode			Division Variables		Bits Cleared by HS (low)
V2	V1	V0	X	Y	
0	0	0	1	12	B1–B4
0	0	1	3	1	B1–B3
0	1	0	1	3	B1–B4
0	1	1	2	1	B1–B3
1	0	0	1	2	B1–B4
1	0	1	1	1	B1–B3
1	1	0	1	1	B1–B4
1	1	1	1	1	None (DMA MODE)

FIGURE 6 — SIGNAL ROUTING for ADDRESS MULTIPLEXER

Memory Size		Signal Source	Row/Column	Signals Routed to Z0–Z7								Timing (Figure 2)
M1	M0			Z7	Z6	Z5	Z4	Z3	Z2	Z1	Z0	
4K 0	0	MPU	ROW	*	A6	A5	A4	A3	A2	A1	A0	T7-TA
			COL	*	L	A11	A10	A9	A8	A7	A6	TA-TF
		VDG	ROW	*	B6	B5	B4	B3	B2	B1	B0	TF-T2
			COL	*	L	B11	B10	B9	B8	B7	B6	T2-T7
		REF	ROW	*	C6	C5	C4	C3	C2	C1	C0	TF-T2
			COL	*	L	L	L	L	L	L	L	T2-T7
16K 0	1	MPU	ROW	*	A6	A5	A4	A3	A2	A1	A0	T7-TA
			COL	*	A13	A12	A11	A10	A9	A8	A7	TA-TF
		VDG	ROW	*	B6	B5	B4	B3	B2	B1	B0	TF-T2
			COL	*	B13	B12	B11	B10	B9	B8	B7	T2-T7
		REF	ROW	*	C6	C5	C4	C3	C2	C1	C0	TF-T2
			COL	*	L	L	L	L	L	L	L	T2-T7
64K (dynamic) 1	0	MPU	ROW	A7	A6	A5	A4	A3	A2	A1	A0	T7-TA
			COL	P/A15***	A14	A13	A12	A11	A10	A9	A8	TA-TF
		VDG	ROW	B7	B6	B5	B4	B3	B2	B1	B0	TF-T2
			COL	B15	B14	B13	B12	B11	B10	B9	B8	T2-T7
		REF	ROW	L	C6	C5	C4	C3	C2	C1	C0	TF-T2
			COL	L	L	L	L	L	L	L	L	T2-T7
64K (static) 1	1	MPU	ROW	A7	A6	A5	A4	A3	A2	A1	A0	T7-**T9**
			COL	P/A15***	A14	A13	A12	A11	A10	A9	A8	**T9**-TF
		VDG	ROW	B7	B6	B5	B4	B3	B2	B1	B0	TF-**T1**
			COL	B15	B14	B13	B12	B11	B10	B9	B8	**T1**-T7
		REF	ROW	L	C6	C5	C4	C3	C2	C1	C0	TF-**T1**
			COL	L	L	L	L	L	L	L	L	**T1**-T7

Notes: "L" implies logical LOW level.
*Z7 functions as RAS1 and its level is address dependent. For example, when using two banks of 16K x 1 RAMs, RAS0 is active for addresses $0000 to $3FFF and RAS1 is active for addresses $4000 to $7FFF.
***If Map TYpe = 0, then page bit "P" is the output (otherwise A15).

Internal Reset

By lowering V_{CC} below 0.6 volts for at least one millisecond, a **complete** SAM reset is initiated and is completed within 500 nanoseconds after V_{CC} rises above 4.25 volts.

NOTE: In some applications, (for example, multiple "VDG-RAM" systems controlled by a single MPU) **multiple** SAM ICs can be synchronized as follows:

- Drive all SAM's from one external oscillator.
- Stop external oscillator.
- Lower V_{CC} below 0.6 volts for at least 1.0 millisecond.
- Raise V_{CC} to 5.0 volts.
- Start external oscillator.
- Wait at least 500 nanoseconds.

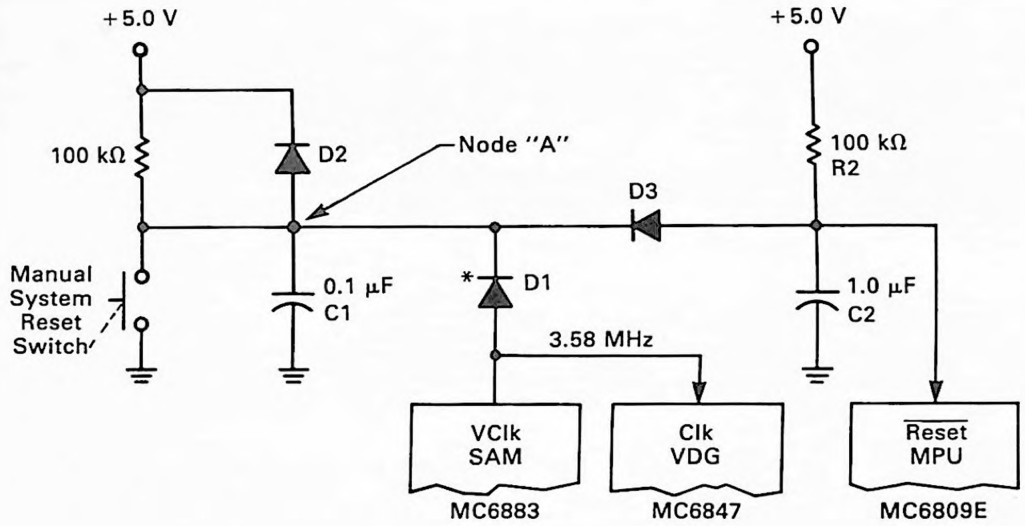
Now, the "E" clocks from all SAM's should be in-phase.

External Reset

When the VClk pin on SAM is forced below 0.8 volts for at least eight cycles of "oscillator-out", the SAM becomes **partially** reset. That is, all bits in the SAM control register are cleared. However, signals such as $\overline{RAS}$, $\overline{CAS}$, $\overline{WE}$, E or Q are **not** stopped (as they are with an **internal** reset), since the SAM must maintain dynamic RAM refresh even during this external reset period.

Figure 7 shows how VClk can be pulled low through diode D1 when node "A" is low.* When node "A" is high, only the backbiased capacitance of diode D1 loads the 3.58 MHz on VClk. Diode D2 helps discharge C1 (Power-on-Reset capacitor) when power is turned off. Diode D3 allows the MPU reset time constant R2C2 to be greater than the SAM reset time constant. Thereby, ensuring **release** of the SAM reset **prior to** attempting to program the SAM control register.

FIGURE 7 — EXTERNAL RESET CIRCUITRY



VDG Synchronization

In order for the VDG and MPU to share the same dynamic RAM (see page 13,) the **VDG clock must be stopped** until the VDG data fetch and MPU data fetch are synchronized as shown in Figure 12. Once synchronized, the VDG clock resumes its 3.579545 MHz rate and is not stopped again unless an extreme temperature change (or SAM reset) occurs. When stopped, the VDG clock remains stopped for **no more than** 32 Osc_{Out} cycles (approximately 2 microseconds.)

In the block diagram in Figure 4, DA0 enters a block labeled VDG Timing Error Detector. If DA0 rises **between** time reference points** τ_A and τ_C, then $\overline{Error}$ is high and VClk is the result of dividing BOSC (Buffered Osc_{Out} ≈ 14 MHz) by four. However, if DA0 rises **outside** the time Window τ_A to τ_C, then $\overline{Error}$ goes LOW and the VDG stops. A START pulse at time reference point τ_B (center of Window) restarts the VDG . . . properly synchronized.

*Use a diode with sufficiently low forward voltage drop to meet V_{IL} requirement at VClk.

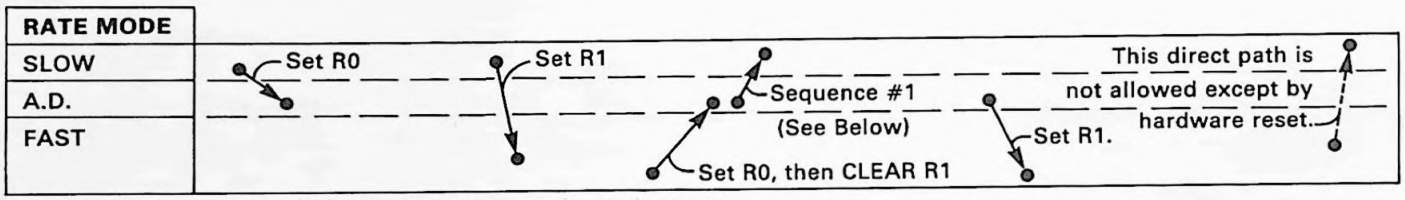
**See timing diagrams on page 5 and 6.

Changing the MPU Rate (by changing SAM control register bits R0, R1).

Two bits in the SAM control register determine the period of both "E" and "Q" MPU clocks. Three rate modes are implemented as follows:

RATE MODE	R1	R0	
SLOW	0	0	The frequency of "E" (and "Q") is f crystal ÷ 16. This rate mode is automatically selected when the SAM is reset. Note that system timing is least critical in this "SLOW" rate mode.
A.D. (Address Dependent)	0	1	The frequency of "E" (and "Q") is either f crystal ÷ 16 or f crystal ÷ 8, depending on the address the MPU is presenting.
FAST	1	X	The frequency of "E" (and "Q") is f crystal ÷ 8. This is accomplished by stealing the time that is normally used for VDG/REFRESH, and using this time for the MPU. Note: Neither VDG display nor dynamic RAM refresh are available in the "FAST" rate mode. (Both are available in SLOW and A.D. rate modes).

When changing between any two of the three rate modes, the following procedures must be followed to ensure that MPU timing specifications are met:



May be ANY address from $0000 to $7FFF.

SEQUENCE #1:

```
7D 00 00  TST #$0000 . . . Synchronizes STA instruction to write during T2-TG (See Figure #8).*
21 00     BRN 00
B7 FF D6  STA #$FFD6 . . . Clears bit R0
```

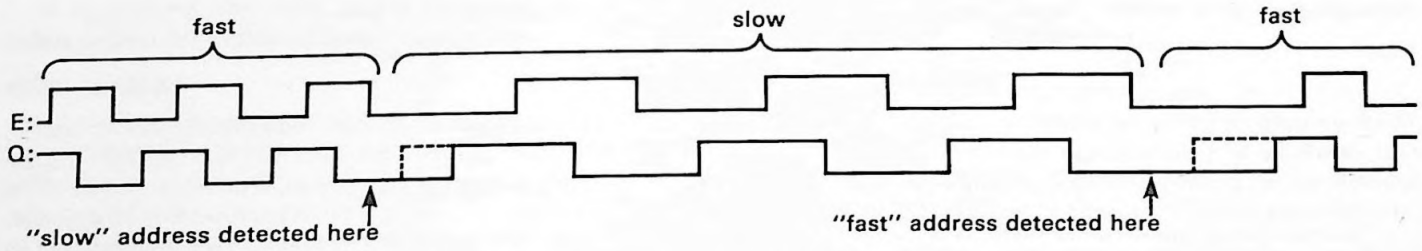
*Note: "TST" instruction affects MC6809E condition code register.

Changing the MPU Rate (In Address Dependent Mode)

When the SAM control register bits "R1", and "R0" are programmed to "0" and "1", respectively, the Address Dependent Rate Mode is selected. In this mode, the ÷ 16 MPU rate is automatically used when addressing within $0000 to $7FFF* or $FF00 to $FF1F ranges. Otherwise the ÷ 8 MPU rate is automatically used. (Refer to Figure 8 for sample "E" and "Q" waveforms yielding ÷ 8 to ÷ 16 and ÷ 16 to ÷ 8 rate changes). This mode often nearly doubles the MPU throughput while still providing transparent VDG and dynamic, RAM refresh functions. For example, since much of the MPU's time may be spent performing internal MPU functions (address = $FFFF)**, accessing ROM (address = $8000 to $FEFF) or accessing I/O (address = $FF20 — $FF5F), the faster f crystal ÷ 8 MPU rate may be used much of the time.

Note: The VDG operates normally when using the SLOW or A.D. rate modes. However, in the FAST rate mode, the VDG is not allowed access to the dynamic RAM.

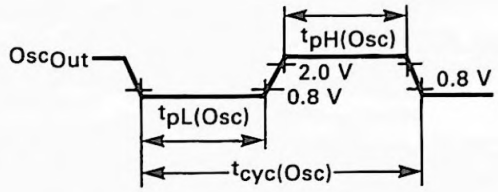
FIGURE 8 — RATE CHANGE E AND Q WAVEFORMS



*When using Memory Map 0, addresses $0000 to $7FFF may access Dynamic RAM.
**The MC6809 outputs $FFFF on A0–A15 when no other valid addresses are being presented.

Oscillator

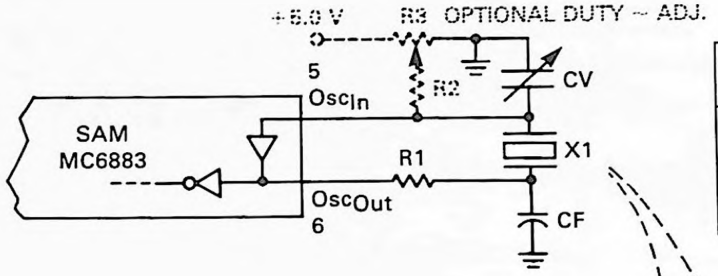
In Figure 4, an amplifier between Osc_{In} and Osc_{Out} provides the gain for oscillation (using a crystal as shown in Figure 9.) Alternately, Pin 5 (Osc_{In}) may be grounded while Pin 6 (Osc_{Out}) may be driven at low-power Schottky levels as shown in Figure 10. Also, see V_{IH}, V_{IL} on page 2.



AC Specifications*

	Max	Typ	Min	Units
$t_{pH(Osc)}$	—	30	22	ns
$t_{pL(Osc)}$	—	30	22	ns
$t_{cyc(Osc)}$	—	70	62.4	ns

FIGURE 9 — CRYSTAL OSCILLATOR



Suggested Component Values

Freq. MHz	CV*	CF*	R1*	R2*	R3*	X1
14.31818	2.5–30 pF	33 pF	1.5 kΩ	~ 100K	10K	*
16.0000	2.5–30 pF	33 pF	1.5 kΩ	~ 100K	10K	*

Recommended Crystal Parameters

	14.31818 MHz**	16.0000 MHz**
R_S	10 Ω ± 2.0 Ω	10 Ω ± 2.0 Ω
CO	5.0 pF ± 1.5 pF	6.0 pF ± 1.0 pF
C1	0.0245 pF ± 15%	0.0319 pF ± 15%
L1	5.05 mH	3.1 mH
Q	50K ± 10K	40K ± 10K

Calibration Tolerance: 0.002% at 26°C
Temperature Tolerance: 0.001% 0°C to 70°C

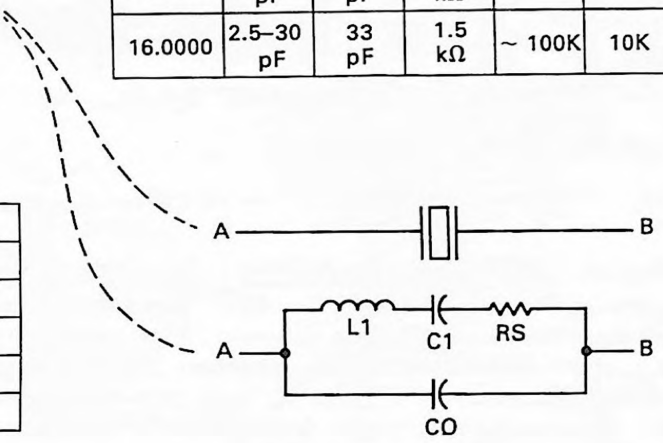
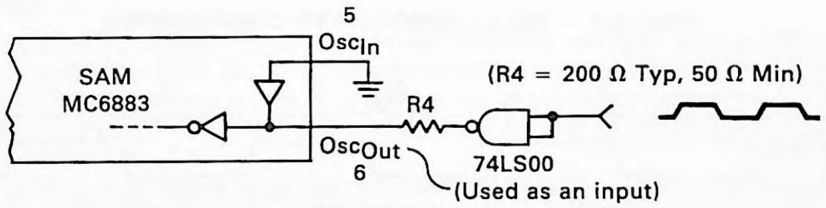


FIGURE 10 — TTL CLOCK INPUT



Typical input capacitances are 3.0 pF for Pin 5 and 5.5 pF for Pin 6.

*Optimum values depend on characteristics of the crystal (X1). For many applications, VClk must be 3.579545 MHz ± 50 Hz! Hence, Osc_{Out} must be made similarly "drift resistant" (by balancing temperature coefficients of X1, CV, CF, R1, R2 and R3).

**Specifically cut for MC6883 are International Crystal Manufacturing, Inc. Crystals (#167568 for 14.31818 MHz or #167569 for 16.0 MHz). However, other crystals may be used.

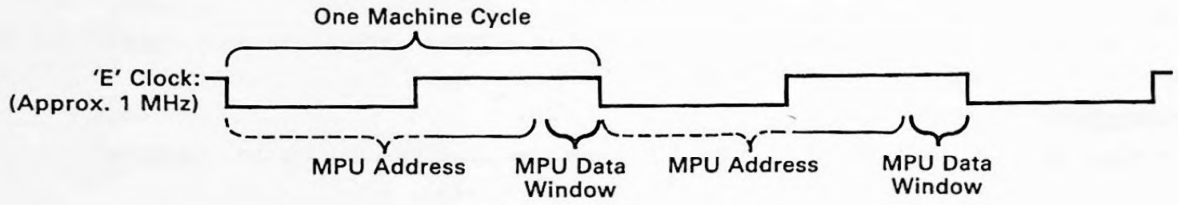
Video or No Video

Although the MC6883 may be used as a dynamic RAM controller **without** a video display*, most applications are likely to include a MC6847 video display generator (VDG). Therefore, this document emphasizes MC6883 with MC6847 systems.

Shared RAM (with interleaved DMA)

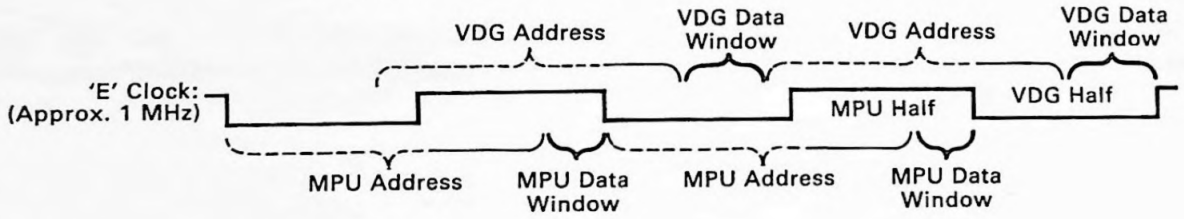
To minimize the number of RAM and interface chips, both the MPU and VDG share common dynamic RAM. Yet, the use of common RAM creates an apparent difficulty. That is, the MPU and VDG must both access the RAM without contention. This difficulty is overcome by taking advantage of the timing and architecture of Motorola MPU's (MC6800, MC6801E, MC6809E, MC68000). Specifically, **all** MPU accesses of external memory **always** occur in the **latter half** of the machine cycle, as shown below:

FIGURE 11 — MOTOROLA MPU TIMING



Similarly, the MC6847 (non-interlaced) VDG transfers a data byte in a half machine cycle (E or Φ2). Thus, when properly positioned, VDG and MPU RAM accesses interleave without contention as shown below:

FIGURE 12 — MOTOROLA MPU WITH VDG TIMING



This Interleaved Direct Memory Access (IDMA) is synchronized via the MC6883 by centering the VDG data window half-way between MPU data windows.**

The result is a shared RAM system without MPU/VDG RAM access contention, with both MPU and VDG running uninterrupted at normal operating speed, each transparent to the other.

RAM Refresh

Dynamic RAM refresh is accomplished by accessing eight*** sequential addresses every 64*** microseconds until 128 consecutive addresses have been accessed. To avoid RAM access contention between REFRESH and MPU, each of the 128 refresh accesses occupies the "VDG half" of the interleaved DMA (IDMA). Furthermore, refresh accesses occur only during the television retrace period (at which time the VDG doesn't need to access RAM).

In summary, the VDG, MPU and MC6883's Refresh Counter all transparently access the common dynamic RAM without contention or interruption.

Why IDMA?

Use of the interleaved direct memory access results in fast modification to variable portions of display RAM, by the MPU, without any distracting flashes on the screen (due to RAM access contention.) In addition, the MPU is not slowed down nor stopped by the MC6883; thereby, assuring accurate software timing loops without costly additional hardware timers. Furthermore, additional hardware and software to give "access permission" to the MPU is eliminated since the MPU may access RAM at **any** time.

* Only 1 pin, (DA0) out of 40 pins is dedicated to the video display.
** See VDG synchronization (page 10) for more detail.
*** When not using a MC6847, $\overline{HS}$ may be wired low for continuous transparent refresh.

"Systems On Silicon" Concept

Total Timing

For most applications, the SAM can supply complete system timing from its on-chip precision 14.31818 MHz oscillator. This includes buffered MPU clocks (E and Q), VDG clock, color subcarrier (3.58 MHz), row address select ($\overline{RAS}$), column address select ($\overline{CAS}$) and write enable ($\overline{WE}$).

Total Address Decode

For most applications, the SAM plus a "1 of 8 decoder" chip completely decodes I/O, ROM and RAM chip selects without wasting memory address space and without needlessly chopping-up contiguous address space. Chip selects are positioned in address space to allow three types of memory (RAM, local ROM and cartridge ROM) independent room for growth. For example, RAM may grow from address $0000-up, cartridge ROM may grow from address $FEFF-down and local ROM may grow from $FBFF-down. Alternately, if the application requires minimum ROM and maximum contiguous RAM, a second choice of two memory maps places RAM from $0000 to $FEFF. (See pages 17 and 18.)

In both memory maps all I/O, MPU vectors, SAM control registers, and some reserved address spaces are efficiently contained between addresses $FF00 and $FFFF.

How Much RAM?

Using nine SAM pins (Z0 – Z7 and $\overline{RAS0}$) the following combinations require no additional address logic.

FIGURE 13 — RAM CONFIGURATIONS

Address: MSB LSB	Chip Select:	
Z5Z4Z3Z2Z1Z0	$\overline{RAS0}$	One or two banks of 4K x 8 (like MCM4027's)
Z5Z4Z3Z2Z1Z0	$\overline{RAS1}$ (=Z7)	
Z6Z5Z4Z3Z2Z1Z0	$\overline{RAS0}$	One or two banks of 16K x 8 (like MCM4116's)
Z6Z5Z4Z3Z2Z1Z0	$\overline{RAS1}$ (=Z7)	
Z7Z6Z5Z4Z3Z2Z1Z0	$\overline{RAS0}$	One bank of 64K x 8 (like MCM6665's)

PROGRAMMING GUIDE

SAM — Programmability

The SAM contains a 16-bit control register which allows the MC6809E to program the SAM for the following options:

VDG Addressing Mode	3-bits
VDG Address Offset	7-bits
32K Page Switch	1-bit
MPU Rate	2-bits
Memory Size	2-bits
Map Type	1-bit

Note that when the SAM is **reset** by first applying power or by manual hardware reset,† all control register bits are **cleared** (to a logic "0").

VDG Addressing Mode

Three bits (V2, V1, V0) control the sequence of DISPLAY ADDRESSES generated by the SAM (which are used to scan dynamic RAM for video information). For example, if you wish to display Dynamic RAM data as INTERNAL ALPHANUMERICS VIDEO, you should program‡ the MC6847 for the INTERNAL ALPHANUMERICS MODE **and** CLEAR BITS V2, V1 and V0 in the SAM. The table on the following page summarizes the available modes:

† See Figure 7 for manual reset circuit.

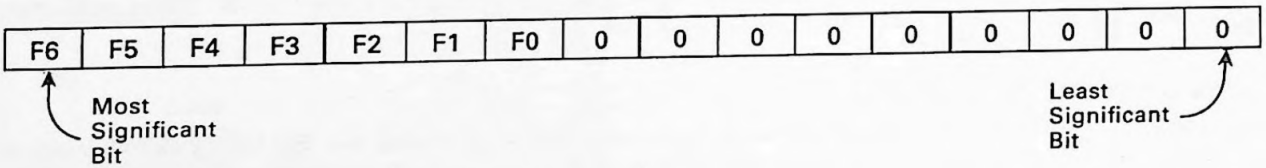
‡ Typically, part of a PIA (MC6821) at location $FF22 is used to control MC6847 modes. (See MC6847 Data Sheet.)

Mode Type	MC6847 Mode					SAM Mode		
	G/$\overline{A}$	GM2	GM1	GM0 EXT/$\overline{I}$	CSS	V2	V1	V0
Internal Alphanumerics	0	X	X	0	X	0	0	0
External Alphanumerics	0	X	X	1	X	0	0	0
OSemigraphics — 4	0	X	X	0	X	0	0	0
Semigraphics — 6	0	X	X	1	X	0	0	0
Semigraphics — 8*	0	X	X	0	X	0	1	0
Semigraphics — 12*	0	X	X	0	X	1	0	0
Semigraphics — 24*	0	X	X	0	X	1	1	0
Full Graphics — 1C	1	0	0	0	X	0	0	1
Full Graphics — 1R	1	0	0	1	X	0	0	1
Full Graphics — 2C	1	0	1	0	X	0	1	0
Full Graphics — 2R	1	0	1	1	X	0	1	1
Full Graphics — 3C	1	1	0	0	X	1	0	0
Full Graphics — 3R	1	1	0	1	X	1	0	1
Full Graphics — 6C	1	1	1	0	X	1	1	0
Full Graphics — 6R	1	1	1	1	X	1	1	0
Direct Memory Access†	X	X	X	X	X	1	1	1

*S8, S12, & S24 modes are **not** described in the MC6847 Data Sheet. See appendix "A".
†DMA is identical to 6R except as shown in Figure 5 on page 9.

VDG Address Offset

Seven bits (F6, F5, F4, F3, F2, F1 and F0) determine the **Starting Address** for the video display. The "Starting Address" is defined as "the address corresponding to data displayed in the **Upper Left** corner of the TV screen". The "Starting Address" is shown below in binary:



Note that the "Starting Address" may be placed anywhere within the 64K address space with a resolution of ½K (the size of one alphanumeric page).
The F6–F0 bits take effect during the TV vertical synchronization pulse (i.e., when $\overline{FS}$ from MC6847 is low).

Page Switch

One bit (P1) is used "in place of" A15 from the MC6809E in order to refer access within $0000-$7FFF to one of two 32K byte **pages** of RAM. If the system does **not** use more than 32K bytes of RAM, P1 can be ignored.**

**When using 4K x 1 RAMS, two banks of eight IC's are allowed. This accounts for Addresses $0000-1FFF. Also, this same RAM can be addressed at $2000-$3FFF, $4000-$5FFF and $6000-$7FFF.

MPU Rate

Two bits (R1, R0) control the clock rate to the MC6809E MPU. The options are:

RATE (FREQUENCY OF "E" CLOCK)	R1	R0
0.9 MHz (Crystal Frequency ÷ 16) Slow	0	0
0.9/1.8 MHz (Address Dependent Rate)	0	1
1.8 MHz (Crystal Frequency ÷ 8) Fast	1	X
(Typical Crystal Frequency = 14.31818 MHz)		

In the "address dependent rate" mode, accesses to $0000-$7FFF and $FF00-$FF1F are slowed to 0.9 MHz (crystal frequency ÷ 16) and all other addresses are accessed at 1.8 MHz (crystal frequency ÷ 8.)

Memory Size

Two bits (M1 and M0) determine RAM memory size. The options are:

SIZE	M1	M0
One or two banks of 4K × 1 dynamic RAMs	0	0
One or two banks of 16K × 1 dynamic RAMs	0	1
One bank of 64K × 1 dynamic RAMs	1	0
Up to 64K static RAM*	1	1

*Requires a latch for demultiplexing the RAM address.

IMPORTANT!

Note: Be sure to program the SAM for the correct memory size **before** using RAM (i.e., for a subroutine stack).

Map Type

One bit (TY) is used to select between two memory map configurations.

Refer to pages 17, 18 and 19 for details. When using Map Type "TY = 1", only the "Slow" MPU rate may be used. Future versions of the SAM may allow use of all rates.

Writing To The SAM Control Register

Any bit in the control register (CR) may be set by writing to a specific unique address. Each bit has two unique addresses . . . writing to the **even** # address **clears** the bit and writing to the **odd** # address **sets** the bit. (Data on the data bus is irrelevant in this procedure.) The specific addresses are tabulated on pages 17 and 18.

If desired, a short routine may be written to program the SAM CR "a word at a time". For example, the following routine copies "B" bits from "A" register to SAM CR addresses beginning with address "X".

SAM1	46		ROR	A
	24	06	BCC	SAM2
	30	01	INX	(LEAX1,X)
	A7	80	STA	O,X+
	20	02	BRA	SAM3
SAM2	A7	81	STA	O,X++
SAM3	5A		DEC	B
	26	F2	BNE	SAM1
	39		RTS	

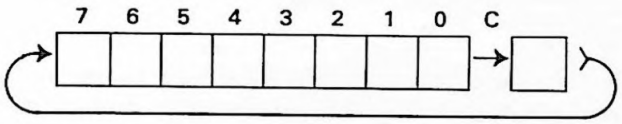
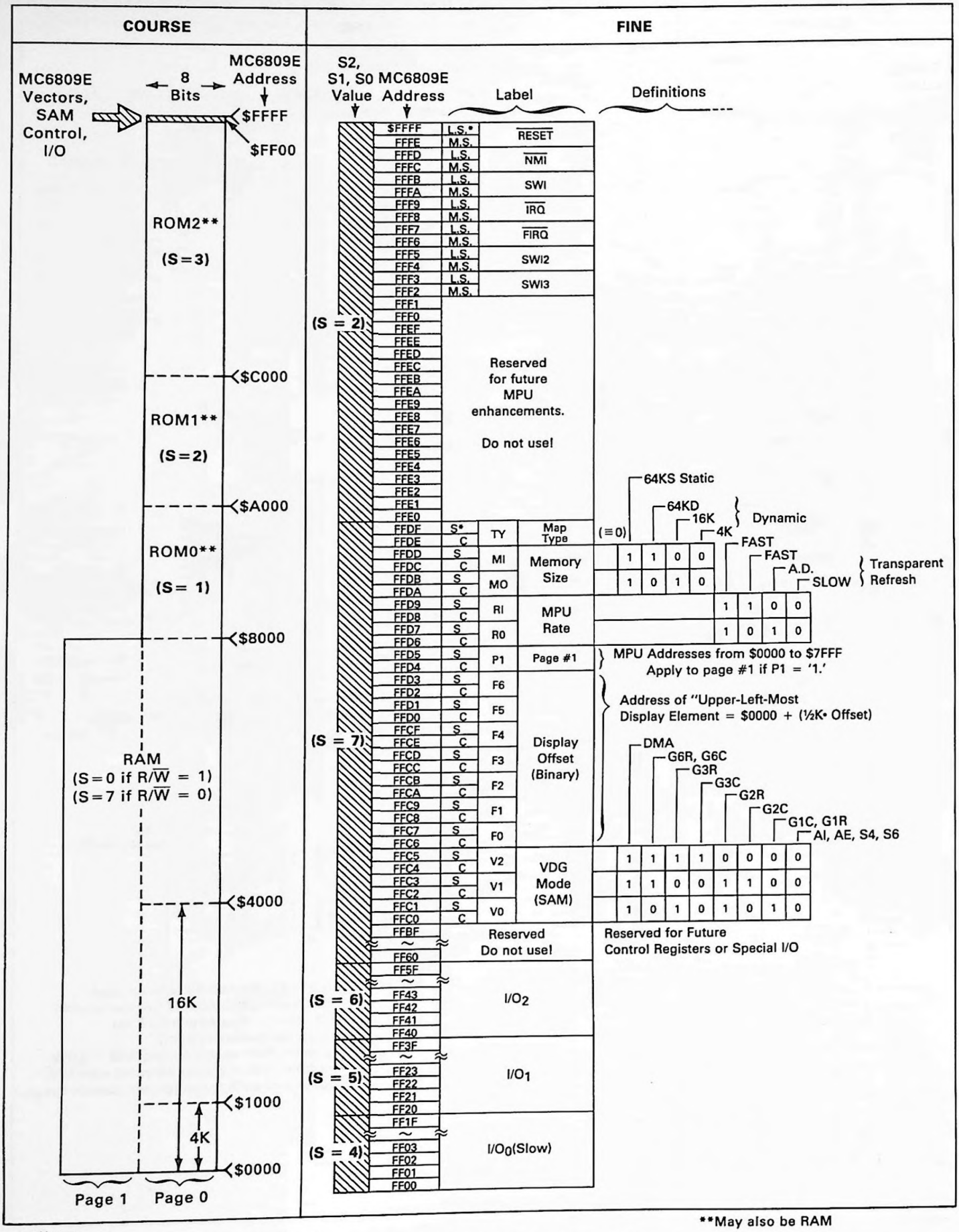


FIGURE 14 — MEMORY MAP (TYPE #0)



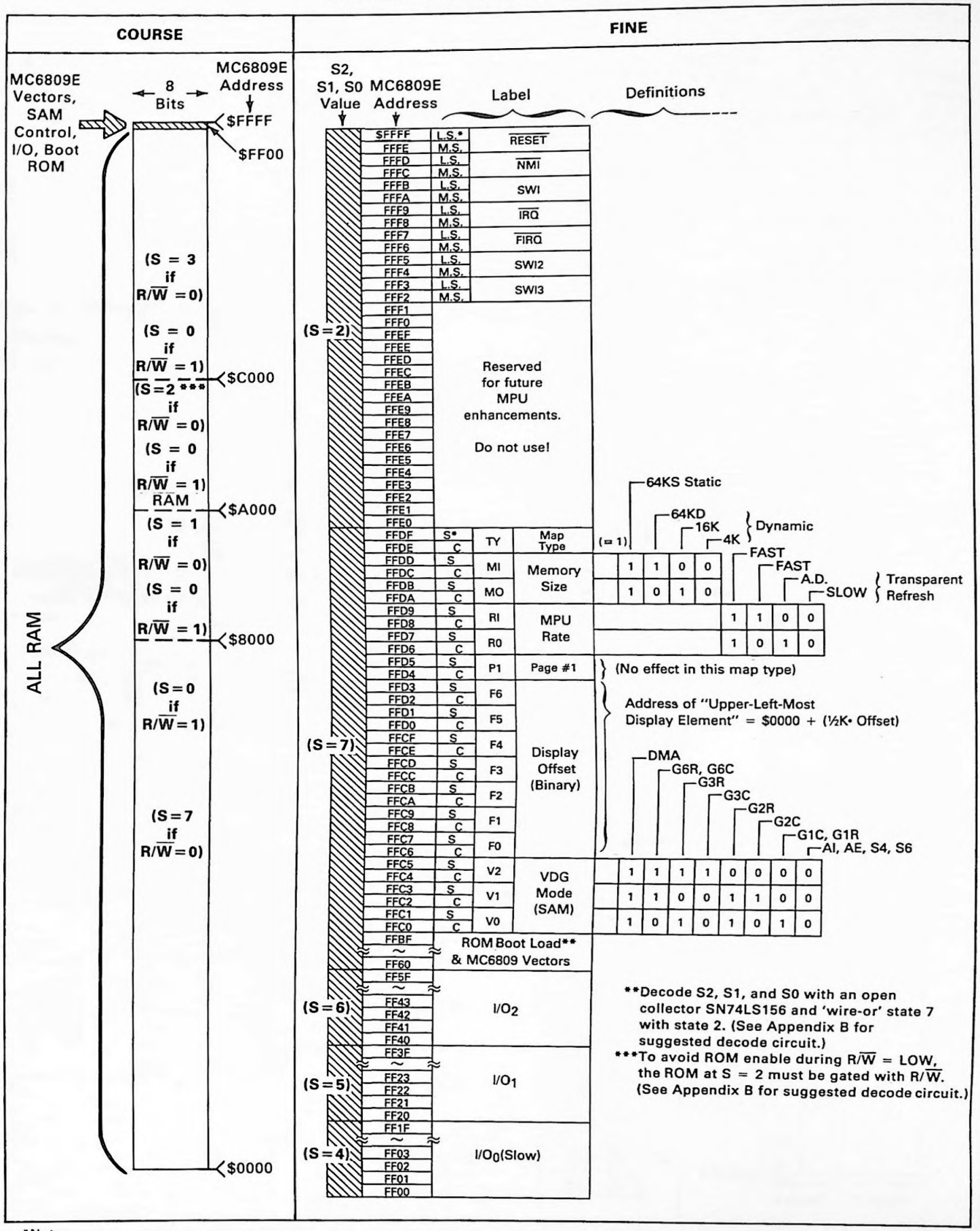
**May also be RAM

*Note:
M.S. = Most Significant
L.S. = Least Significant

S = Set Bit
C = Clear Bit
(All bits are cleared when SAM is reset.)

S = Device Select value = 4 x S2 + 2 x S1 + 1 x S0

FIGURE 15 — MEMORY MAP (TYPE #1)



*Note:
M.S. ≡ Most Significant
L.S. ≡ Least Significant
S ≡ Set Bit
C ≡ Clear Bit
(All bits are cleared when SAM is reset.)
S = Device Select value = 4 x S2 + 2 x S1 + 1 x S0

FIGURE 16 — MEMORY ALLOCATION TABLE
(Also, see the memory MAPs on pages 17 and 18.)

Type # 0: (Primarily for ROM based systems)

Address Range	S=4(S2)+2 (S1)+S0 S Value	Intended Use
$FFF2 to FFFF	2	MC6809E Vectors: $\overline{\text{Reset}}$, $\overline{\text{NMI}}$, SWI, $\overline{\text{IRQ}}$, $\overline{\text{FIRQ}}$, SWI2, SWI3.
FFE0 to FFF1	2	Reserved for future MPU enhancements.
FFC0 to FFDF	7	SAM Control Register: V0, – V2, F0 – F6, P, R0, R1, M0, M1, TY.
FF60 to FFBF	7	Reserved for future control register enhancements.
FF40 to FF5F	6	I/O_2: Input/Output (PIAs, ACIAs, etc.) To subdivide, use A0 – A4.
FF20 to FF3F	5	I/O_1: Input/Output (PIAs, ACIAs, etc.) To subdivide, use A0 – A4.
FF00 to FF1F	4	I/O_0: Input/Output (PIAs, ACIAs, etc.) To subdivide, use A0 – A4.
C000 to FEFF	3	ROM2: 16K addresses. External cartridge ROM*.
A000 to BFFF	2	ROM1: 8K addresses. Internal ROM*. Note that MC6809E vector addresses select this ROM*.
8000 to 9FFF	1	ROM0: 8K addresses. Internal ROM*.
0000 to 7FFF	0 if R/$\overline{W}$ = 1 7 if R/$\overline{W}$ = 0	RAM: 32K addresses. RAM shared by MPU and VDG.

*Not restricted to ROM. For example, RAM or I/O may be used here.

Type # 1: (Primarily for RAM based systems)

Address Range	S=4(S2)+2 (S1)+S0 S Value	Intended Use
$FFF2 to FFFF	2	MC6809E Vectors: $\overline{\text{Reset}}$, $\overline{\text{NMI}}$, SWI, $\overline{\text{IRQ}}$, $\overline{\text{FIRQ}}$, SWI2, SWI3.
FFE0 to FFF1	2	Reserved for future MPU enhancements.
FFC0 to FFDF	7	SAM Control Register: V0 – V2, F0 – F_6, P, R0, R1, M0, M1, TY.
FF60 to FFBF	7	Small ROM: Boot load program and initial MC6809 vectors.
FF40 to FF5F	6	I/O_2: Input/Output (PIAs, ACIAs, etc.) To subdivide, use A0-A4.
FF20 to FF3F	5	I/O_1: Input/Output (PIAs, ACIAs, etc.) To subdivide, use A0 – A4.
FF00 to FF1F	4	I/O_0: Input/Output (PIAs, ACIAs, etc.) To subdivide, use A2 – A4.
0000 to FEFF	0 if R/$\overline{W}$ = 1	RAM: 64K(–256) addresses, shared by MPU and VDG. (If R/$\overline{W}$ = 0 then S = 3 for $C000–$FEFF; S = 2 for $A000–$BFFF; S = 1 for $8000–$9FFF and S = 7 for $0000–$7FFF.)

APPENDIX A

VDG/SAM Video Display System Offers 3 New Modes

by

Paul Fletcher

There are three new modes created when the VDG and SAM are used together in a video display system. These modes offer alphanumeric compatibility with 8 color low-to-high resolution graphics, 64Hx64V, 64Hx96V, 64Hx192V. The new modes S8, S12, and S24 are created by placing the VDG in the Alpha Internal mode and having the SAM in a 2K, 3K or 6K full color graphics mode. In all modes the VDG's S/$\overline{A}$ and Inv. pins are connected to data bits DD7 and DD6 to allow switching on the fly between Alpha and Semigraphics and between inverted and non-inverted alpha. This method is used in most VDG systems to obtain maximum flexibility.

The three modes divide the standard 8*12 dot box used by the VDG for the standard alpha and semigraphics modes into eight 4*3 dot boxes for the S8 mode, twelve 4*2 dot boxes for the S12 mode, and twenty-four 4*1 dot boxes for the S24 mode. Figure 17 shows the arrangement of these boxes. One byte is needed to control two horizontally consecutive boxes. It therefore takes four bytes for the S8, six bytes for the S12, and 12 bytes for the S24 mode to control the entire 8*12 dot box. These two horizontally consecutive boxes have four combinations of luminance controlled by bits B0 – B3. For convenience B2 should be made equal to B0 and B3 should be made equal to B1. This eliminates a screen placement problem which would cause other codes to change patterns when moved vertically on the screen. The illuminated boxes can be one of eight colors which are controlled by B4 – B6 (see Figure 18). The bytes needed to control all the boxes in the 8*12 dot box must be spaced 32 address spaces apart in the display RAM because of the addressing scheme orginally used in the VDG and duplicated by the SAM. This means to place an alphanumeric character on the TV screen it requires 4, 6, or 12 bytes depending on the mode used. These bytes are placed 32 memory locations apart in the display RAM (see Figure 18). This multiple byte format allows the mixing of character rows of different characters in the same 8*12 dot box creating new characters and symbols. It also allows overlining and underlining in eight colors by switching to semigraphics at the correct time.

These new modes optimize the memory versus screen density tradeoffs for RF performance on color TVs. This could make them the most versatile of all the modes depending on the users creativity and the software sophistication.

APPENDIX B

Memory Decode for "MAP TYPE = 1"

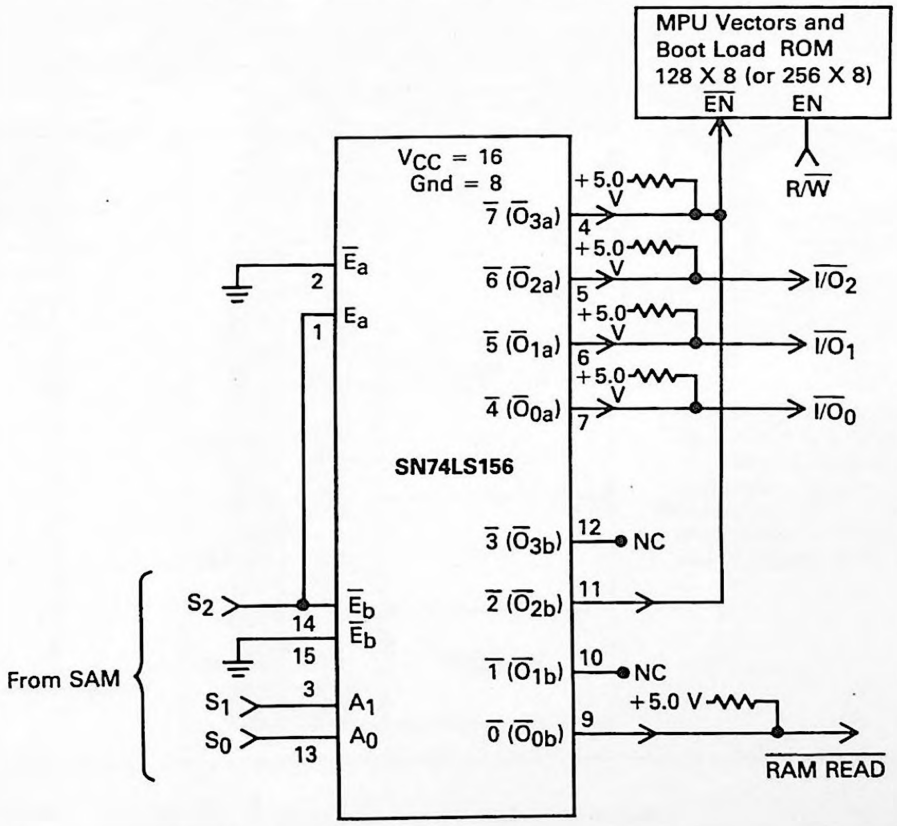
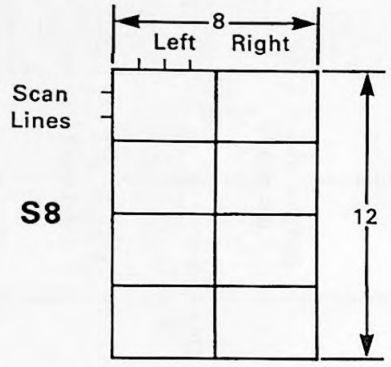
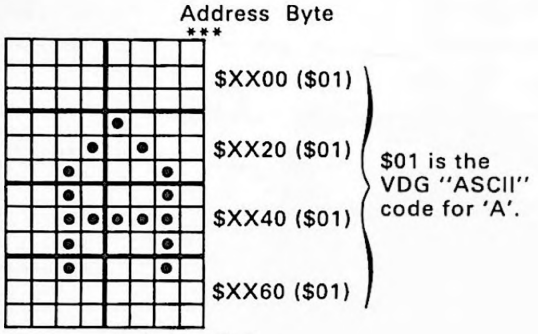
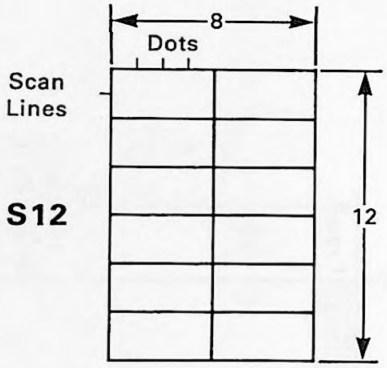


FIGURE 17 — DISPLAY MODES S8, S12, S24
Bit/Visible Dot Correlation



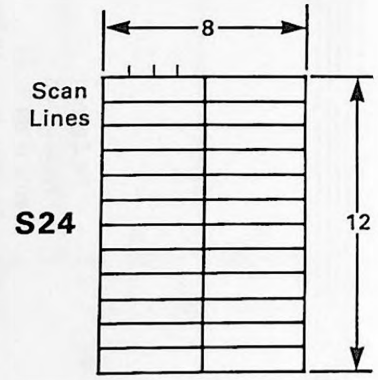


- Alphanumeric Compatible



Left	Right ***	
Red	Red	$XX00 ($BF)
Blue	Off	$XX20 ($AA)
Off	Green	$XX40 ($85)
Orange	Orange	$XX60 ($FF)
Off	Off	$XX80 ($80)
Yellow	Yellow	$XXA0 ($9F)

- Options: One of 8 colors for L or R or both. Off = Black

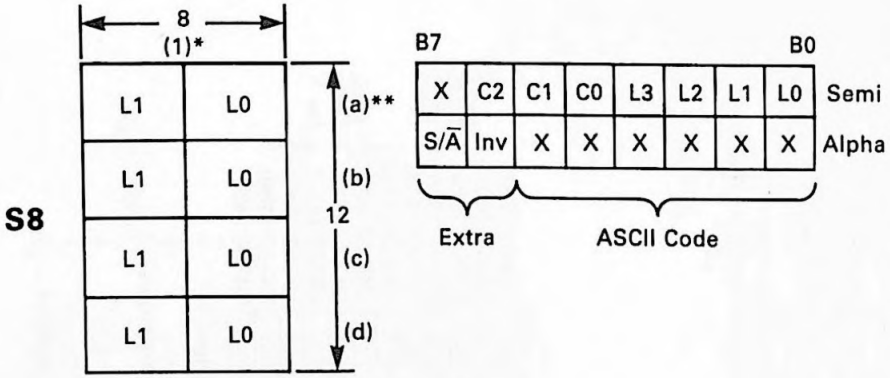


***	
Blue Blue	$XX00 ($AF)
Black Black	$XX20 ($80)
Black Black	$XX40 ($80)
● ● ● ● ● ●	$XX60 ($14) — VDG Code for T
● ●	$XX80 ($18)
● ●	$XXA0 ($18)
●	$XXC0 ($18)
● ●	$XXE0 ($18) — VDG Code for X
● ●	$X100 ($18)
● ●	$X120 ($18)
Black Black	$X140 ($80)
Green Green	$X160 ($8F)

- Underline, Overline
- Mix Character Dot Rows

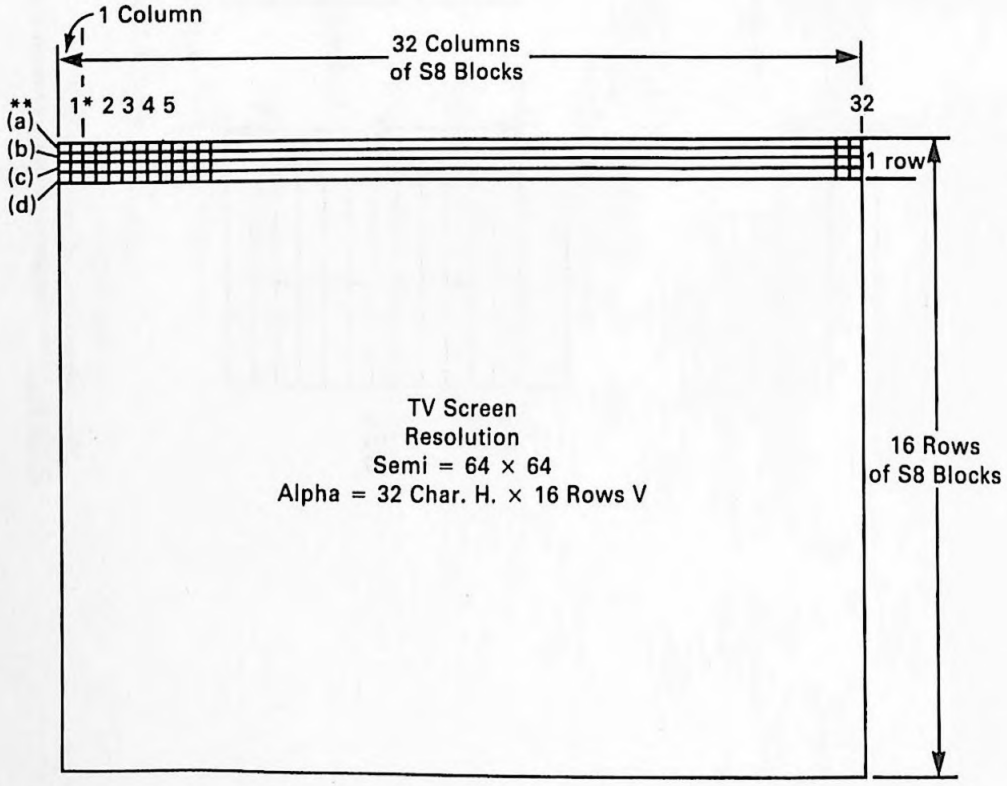
*** Characters will always remain in standard VDG positions.

FIGURE 18 — S8 DISPLAY FORMAT EXAMPLES



LX	C2	C1	C0	Color
0	X	X	X	Black
1	0	0	0	Green
1	0	0	1	Yellow
1	0	1	0	Blue
1	0	1	1	Red
1	1	0	0	Buff
1	1	0	1	Cyan
1	1	1	0	Magenta
1	1	1	1	Orange

B3,B1	B2,B0		4	
0	0	=	Off	Off
0	1	=	Off	Color
1	0	=	Color	Off
1	1	=	Color	



S8 Screen Memory Map

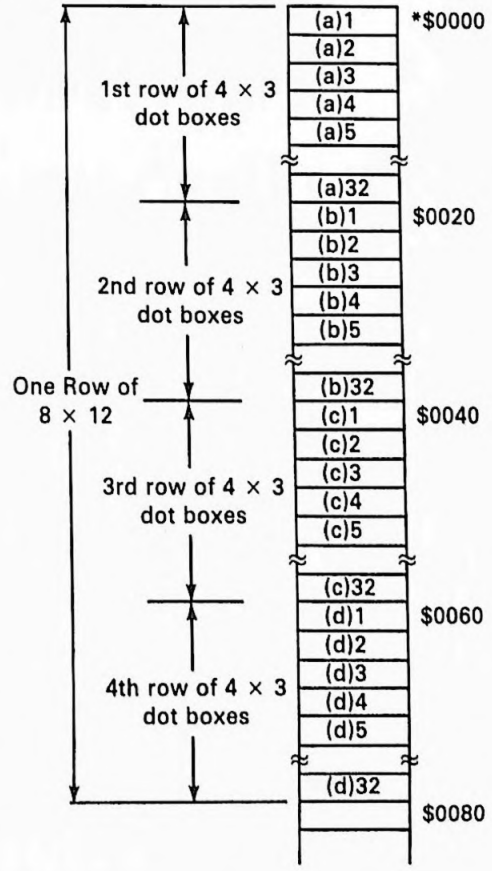


FIGURE 19 — EXAMPLE of MC6809E, MC6883 and MC6847 COMPUTER

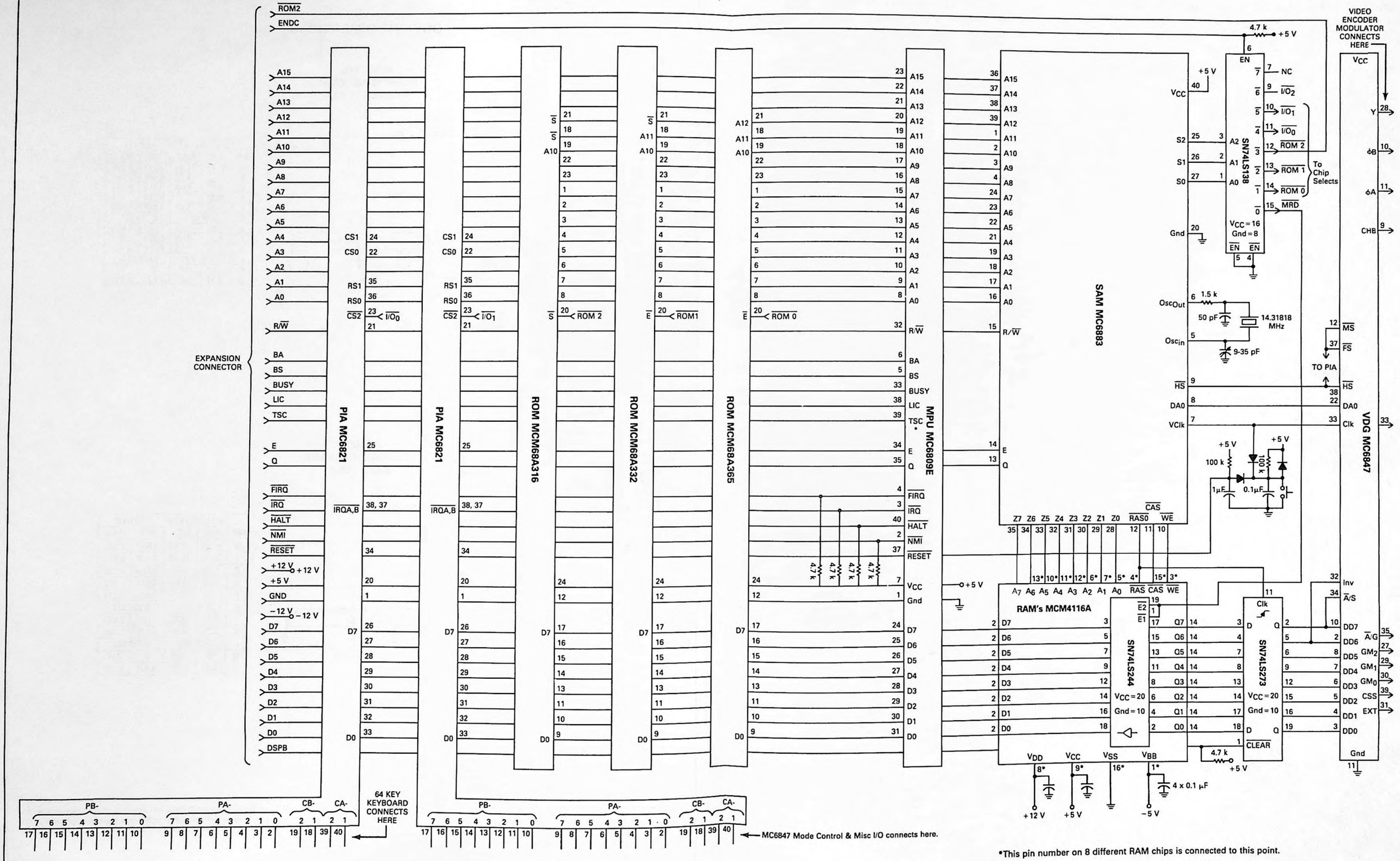


FIGURE 20 — EQUIVALENT OF OSCILLATOR INPUT AND OUTPUT

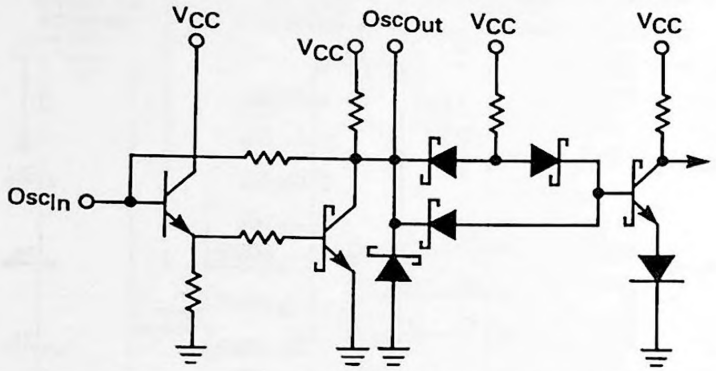


FIGURE 21 — DA0 INPUT

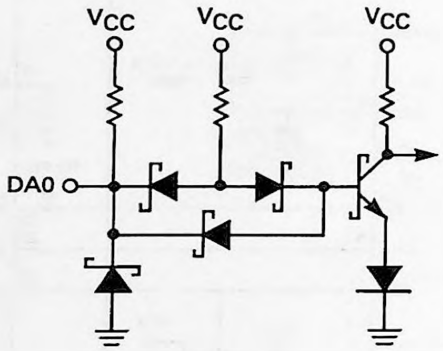


FIGURE 22 — VClk INPUT/OUTPUT

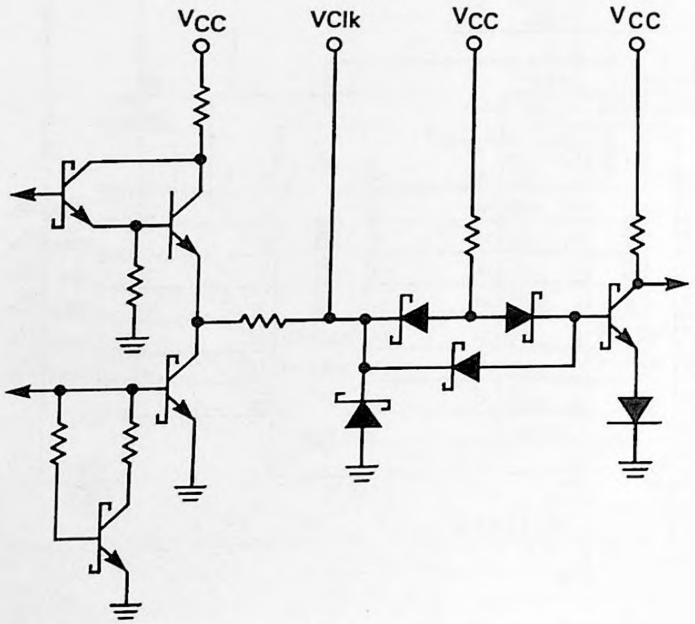


FIGURE 23 — E AND Q OUTPUTS

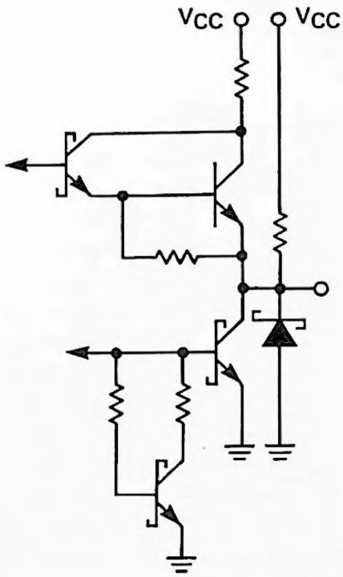


FIGURE 24 — TYPICAL INPUT

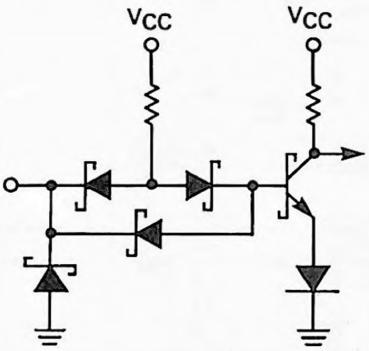
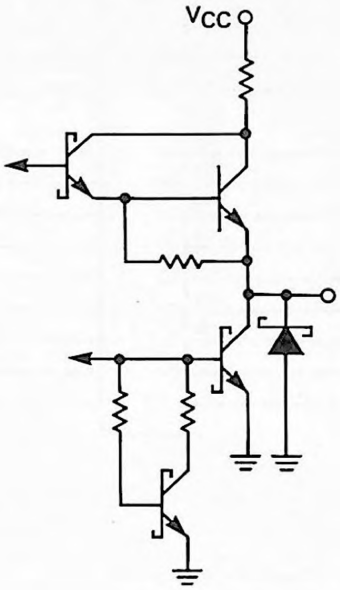
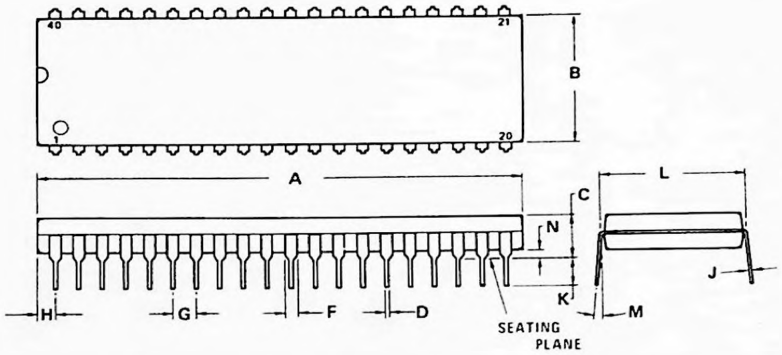


FIGURE 25 — TYPICAL OUTPUT



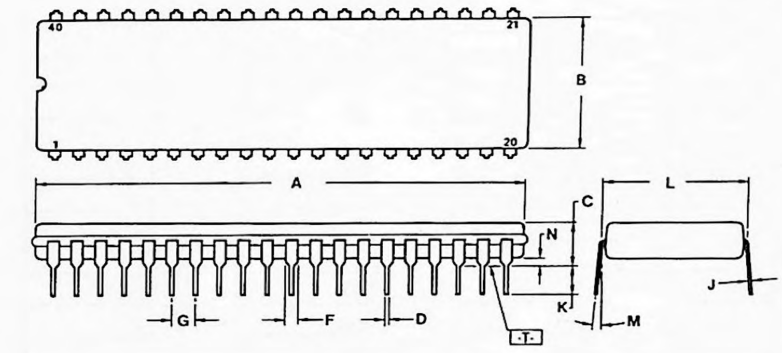
OUTLINE DIMENSIONS

N SUFFIX
PLASTIC PACKAGE
CASE 711-03



DIM	MILLIMETERS		INCHES	
	MIN	MAX	MIN	MAX
A	51.69	52.45	2.035	2.065
B	13.72	14.22	0.540	0.560
C	3.94	5.08	0.155	0.200
D	0.36	0.56	0.014	0.022
F	1.02	1.52	0.040	0.060
G	2.54 BSC		0.100 BSC	
H	1.65	2.16	0.065	0.085
J	0.20	0.38	0.008	0.015
K	2.92	3.43	0.115	0.135
L	15.24 BSC		0.600 BSC	
M	0°	15°	0°	15°
N	0.51	1.02	0.020	0.040

J SUFFIX
CERAMIC PACKAGE
CASE 734-04



DIM	MILLIMETERS		INCHES	
	MIN	MAX	MIN	MAX
A	51.31	53.24	2.020	2.096
B	12.70	15.49	0.500	0.610
C	4.06	5.84	0.160	0.230
D	0.38	0.56	0.015	0.022
F	1.27	1.65	0.050	0.065
G	2.54 BSC		0.100 BSC	
J	0.20	0.30	0.008	0.012
K	3.18	4.06	0.125	0.160
L	15.24 BSC		0.600 BSC	
M	5°	15°	5°	15°
N	0.51	1.27	0.020	0.050

Motorola reserves the right to make changes to any products herein to improve reliability, function or design. Motorola does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights nor the rights of others.

MOTOROLA Semiconductor Products Inc.
BOX 20912 • PHOENIX, ARIZONA 85036 • A SUBSIDIARY OF MOTOROLA INC.

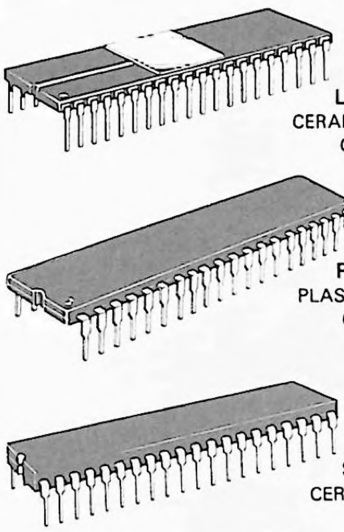
13717-2 PRINTED IN USA 3-82 IMPERIAL LITHO C03855 2,000
ADI-595

MOTOROLA
SEMICONDUCTORS
3501 ED BLUESTEIN BLVD., AUSTIN, TEXAS 78721

MC6847
Non-Interlace
MC6847Y
Interlace

MOS
(N-CHANNEL, SILICON-GATE)

VIDEO DISPLAY GENERATOR



L SUFFIX
CERAMIC PACKAGE
CASE 715

P SUFFIX
PLASTIC PACKAGE
CASE 711

S SUFFIX
CERDIP PACKAGE
CASE 734

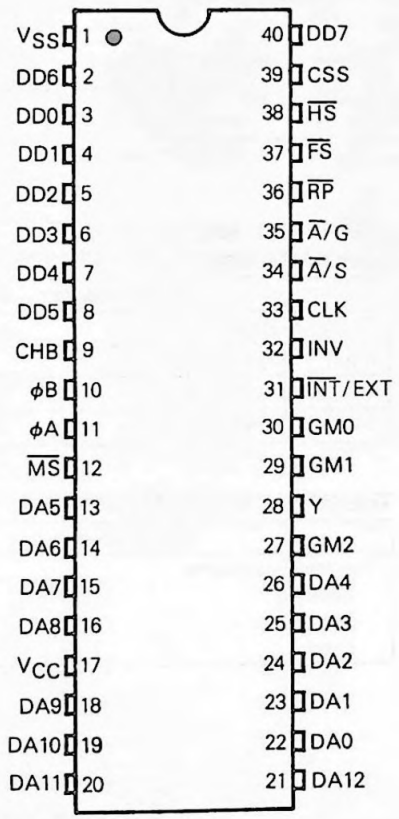
MC6847/MC6847Y VIDEO DISPLAY GENERATOR (VDG)

The video display generator (VDG) provides a means of interfacing the M6800 microprocessor family (or similar products) to a standard color or black and white NTSC television receiver. Applications of the VDG include video games, process control displays, home computers, education, communications, and graphics applications.

The VDG reads data from memory and produces a video signal which will allow the generation of alphanumeric or graphic displays. The generated video signal may be modulated to either channel 3 or 4 by using the compatible MC1372 (TV chroma and video modulator). This modulated signal is suitable for reception by a standard unmodified television receiver. A typical TV game is shown in Figure 1.

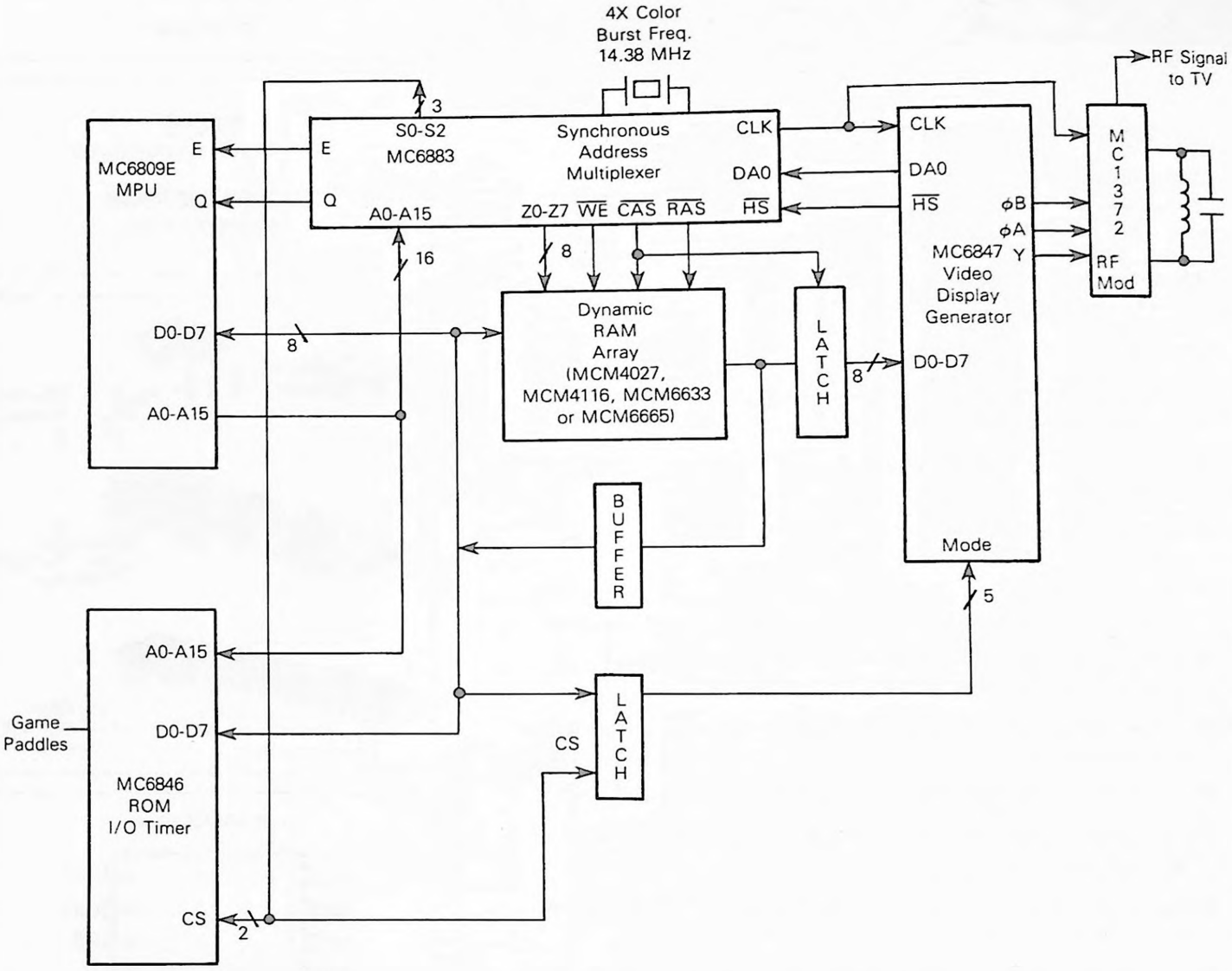
- Compatible with the M6800 Family, the M68000 Family, and Other Microprocessor Families
- Generates Four Different Alphanumeric Display Modes, Two Semigraphic Modes, and Eight Graphic Display Modes
- The Alphanumeric Modes Display 32 Characters Per Line by 16 Lines Using Either the Internal ROM or an External Character Generator
- Alphanumeric and Semigraphic Modes May Be Mixed on a Character-by-Character Basis
- Alphanumeric Modes Support Selectable Inverse on a Character-by-Character Basis
- Internal ROM May Be Mask Programmed with a Custom Pattern
- Full Graphic Modes Offer 64 × 64, 128 × 64, 128 × 96, 128 × 192, or 256 × 192 Densities
- Full Graphic Modes Use One of Two 4-Color Sets or One of Two 2-Color Sets
- Compatible with the MC1372 and MC1373 Modulators Via Y, R-Y (ϕA), and B-Y (ϕB) Interface
- Compatible with the MC6883 (74LS783) Synchronous-Address Multiplexer
- Available in Either an Interlace (NTSC Standard) or Non-interlace Version

PIN ASSIGNMENT



©MOTOROLA INC., 1981

FIGURE 1 — BLOCK DIAGRAM OF A TV GAME USING THE VDG AND THE MC6809E MPU



ELECTRICAL SPECIFICATIONS
ABSOLUTE MAXIMUM RATINGS

Characteristics	Symbol	Value	Unit
Supply Voltage	V_{CC}	−0.3 to +7.0	V
Input Voltage Any Pin	V_{in}	−0.3 to +7.0	V
Operating Temperature	T_A	0 to +70	°C
Storage Temperature	T_{stg}	−65 to +150	°C

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. Reliability of operation is enhanced if unused inputs are tied to an appropriate logic voltage (e.g., either V_{SS} or V_{CC}).

THERMAL CHARACTERISTICS

Characteristics	Symbol	Value	Unit
Thermal Resistance Ceramic Plastic Cerdip	θ_{JA}	 50 100 60	°C/W

DC (STATIC) CHARACTERISTICS (V_{CC} = 5.0 V ±5%, V_{SS} = 0.0 V, T_A = 0°C to 70°C unless otherwise noted)

Characteristic		Symbol	Min	Typ	Max	Unit
Input High Voltage CLK Other Inputs		V_{IH}	 V_{SS} + 2.4 V_{SS} + 2.0	 – –	 V_{CC} V_{CC}	V
Input Low Voltage CLK Other Inputs		V_{IL}	 V_{SS} – 0.3 V_{SS} – 0.3	 – –	 V_{SS} + 0.4 V_{SS} + 0.8	V
Input Leakage Current, Force 5.25 V on Pin Under Test, V_{CC} = 5.5 V CLK, GM0-GM2, INV, $\overline{INT}$/EXT, $\overline{MS}$, V_{SS}, DD0-DD7, $\overline{A}$/S, $\overline{A}$/G		I_{in}	–	–	2.5	μA
Three-State (Off State) Input Current DA0-DA12 Force 2.4 V and 0.4 V on Pin Under Test		I_{OL}	–	–	±10	μA
Output High Voltage (C_{Load} = 30 pF, I_{Load} = – 100 μA	$\overline{RP}$, $\overline{HS}$, $\overline{FS}$	V_{OH}	2.4	–	–	V
Output High Voltage (C_{Load} = 55 pF, I_{Load} = – 100 μA)	DA0-DA12	V_{OH}	2.4	–	–	V
Output Low Voltage (C_{Load} = 30 pF, I_{Load} = 1.6 mA)	RP, HS, FS	V_{OL}	–	–	V_{SS} + 0.4	V
Output Low Voltage (C_{Load} = 55 pF, I_{Load} = 1.6 mA)	DA0-DA12	V_{OL}	–	–	V_{SS} + 0.4	V
Output High Current (Sourcing) (V_{OH} = 2.4 V)	All Outputs (Except φA, φB, Y, and CHB)	I_{OH}	– 100	–	–	μA
Output Low Current (Sinking) (V_{OL} = 0.4 V)	All Outputs (Except φA, φB, Y, and CHB)	I_{OL}	1.6	–	–	mA
Input Capacitance (V_{in} = 0, T_A = 25°C, f = 1.0 MHz)	All Inputs	C_{in}	–	–	7.5	pF
Internal Power Dissipation (Measured at T_A = 0 to 70°C)		P_{INT}	–	–	600	mW
Chroma φA Voltage (Figure 3) (C_{Load} = 20 pF, R_{Load} = 100 kΩ) (Note 1)		V_{IH} V_R V_{OL}	1.8 1.34 0.8	2.0 1.5 1.0	2.2 1.66 1.2	V
Chroma φB Voltage (Figure 3) (C_{Load} = 20 pF, R_{Load} = 100 kΩ) (Note 1)		V_{IH} V_R V_{OL} V_{Burst}	1.8 1.34 0.80 1.07	2.0 1.5 1.0 1.25	2.2 1.66 1.2 1.43	V
Luminance Y Voltage (Figure 3) (C_{Load} = 20 pF, R_{Load} = 100 kΩ) (Voltage Synchronization) (Voltage Blank) (Voltage Black) (Voltage White Low) (Voltage White Medium) (Voltage White High) (Note 1)		 V_S V_{Blank} V_{Black} V_{WL} V_{WM} V_{WH}	 0.9 0.63 0.58 0.51 0.40 0.27	 1.0 0.77 0.72 0.65 0.54 0.42	 1.1 0.9 0.83 0.75 0.65 0.53	V
Chroma Bias Voltage (C_{Load} = 20 pF, R_{Load} = 100 kΩ)		V_R	0.27 V_{CC}	0.3 V_{CC}	0.33 V_{CC}	V
Resistor % of V_{SS} Tracking (Analog Outputs Linearity Error)		R_T	–	1.0	3.0	%

NOTE 1: The specified minimum and maximum number reflect performance of the VDG of the specified temperature range. Overlapping voltage levels will not occur. Refer to Figure 2.

POWER CONSIDERATIONS

The average chip-junction temperature, T_J, in °C can be obtained from: (1)

$$T_J = T_A + (P_D \bullet \theta_{JA})$$

Where:

$T_A \equiv$ Ambient Temperature, °C

$\theta_{JA} \equiv$ Package Thermal Resistance, Junction-to-Ambient, °C/W

$P_D \equiv P_{INT} + P_{PORT}$

$P_{INT} \equiv I_{CC} \times V_{CC}$, Watts — Chip Internal Power

$P_{PORT} \equiv$ Port Power Dissipation, Watts — User Determined

For most applications $P_{PORT} \ll P_{INT}$ and can be neglected. P_{PORT} may become significant if the device is configured to drive Darlington bases or sink LED loads.

An approximate relationship between P_D and T_J (if P_{PORT} is neglected) is: (2)

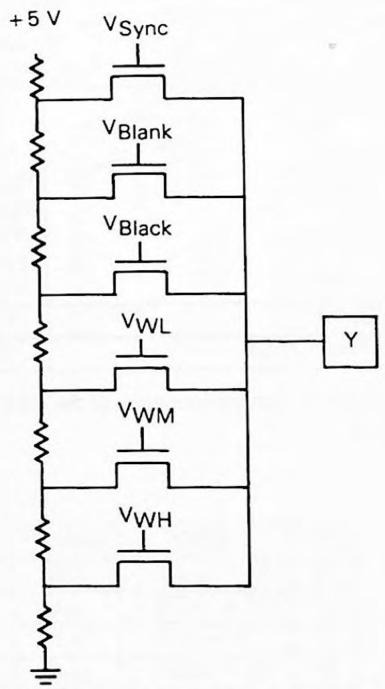
$$P_D = K \div (T_J + 273°C)$$

Solving equations 1 and 2 for K gives: (3)

$$K = P_D \bullet (T_A + 273°C) + \theta_{JA} \bullet P_D^2$$

Where K is a constant pertaining to the particular part. K can be determined from equation 3 by measuring P_D (at equilibrium) for a known T_A. Using this value of K the values of P_D and T_J can be obtained by solving equations (1) and (2) iteratively for any value of T_A.

FIGURE 2 — PSEUDO ANALOG LUMINANCE RESISTOR CHAIN



NOTE: The chrominance output chain is similar in design to the luminance chain.

AC (DYNAMIC) CHARACTERISTICS (V_{CC} = 5.0 V ±5%, T_A = 0°C to 70°C) (Load Circuit of Figure 3)

Characteristic		Symbol	Min	Max	Unit	Figure
CLK (Frequency (3.579545 Color Burst Frequency)		f	3.579535	3.597555	MHz	4
CLK Duty Cycle		CLK_{dc}	45	55	%	4
Clock Rise Time		t_{CLKr}	–	50	ns	4
Clock Fall Time		t_{CLKf}	–	50	ns	4
Clock Pulse Width		PW_{CLK}	120	160	ns	4
Horizontal Display Address Delay from Counter	DA0-DA3	t_{HDAD}	–	490	ns	4, 5, 6
	DA4	t_{HDA4D}	–	550	ns	5, 6
Horizontal Display Address Hold Time		t_{HDAH}	0	–	ns	4, 5, 6
		t_{HDA4H}	0	–		
Display Data Setup Time	CSS, INV, $\overline{A}$/S, $\overline{INT}$/EXT, DD0-DD7	t_{DDS}	70	–	ns	4, 5, 6
Display Data Hold Time	CSS, INV, $\overline{A}$/S, $\overline{INT}$/EXT, DD0-DD7	t_{DDH}	140	–	ns	4, 5, 6
Horizontal Sync ($\overline{HS}$) Delay	Fall	t_{DHSf}	–	550	ns	7
	Rise	t_{DHSr}	–	740		
Row Preset ($\overline{RP}$) Delay	Fall	t_{DRPf}	–	660	ns	7
	Rise	t_{DRDr}	–	540		
Vertical Display Address Delay from Counter	DA5-DA12	t_{VDAD}	–	6.0	µs	7
Vertical Display Address Hold Time		t_{VDAH}	–	220	ns	7
Field Sync ($\overline{FS}$) Delay	Fall	t_{DFSf}	–	520	ns	8
	Rise	t_{DFSr}	–	600		
Memory Select Low to Display Address High-Impedance		t_{DMST}	–	80	ns	9
Memory Select High to Display Address Valid		t_{DMSV}	–	400	ns	9
Chroma Rise and Fall Times						
(ϕA Rise Time)		$t_{rC\phi A}$	–	100	ns	12
		$t_{r\phi A}$	–	100		
(ϕA Fall Time)		$t_{fC\phi A}$	–	100		
		$t_{f\phi A}$	–	100		
(ϕB Rise Time)		$t_{rC\phi B}$	–	100		
		$t_{r\phi B}$	–	100		
(ϕB Fall Time)		$t_{fC\phi B}$	–	100		
		$t_{f\phi B}$	–	100		
Color Burst Rise Time on ϕB Output		t_{CBr}	–	100	ns	12
Color Burst Fall Time on ϕB Output		t_{CBf}	–	100	ns	12
Chroma Phase Delay (Measured with Respect to "Y" Output)						
ϕA		t_{YA}	–50	140	ns	11
ϕB		t_{YB}	–50	140		
Luminance Rise Time		t_{ry}	–	100	ns	12
Luminance Fall Time		t_{fy}	–	100	ns	12
Horizontal Sync Rise Time on Y Output		t_{Hr}	–	100	ns	12
Horizontal Sync Fall Time on Y Output		t_{Hf}	–	100	ns	12
Horizontal Blanking Rise Time on Y Output		t_{HBr}	–	100	ns	12
Horizontal Blanking Fall Time on Y Output		t_{HBf}	–	100	ns	12
Front Porch Duaration Time (7 × 1/f)		t_{FP}	1.8	2.4	µs	12
Back Porch Duration Time (17.5 × 1/f)		t_{BP}	4.5	5.1	µs	12
Left Border Duration Time (29.5 × 1/f)		t_{LB}	7.5	8.3	µs	12
Right Border Duration Time (28 × 1/f)		t_{RB}	7.5	8.3	µs	12
Color Burst Duration Time (10.5 × 1/f)		t_{CB}	2.7	3.2	µs	12

FIGURE 3 — TEST LOADS

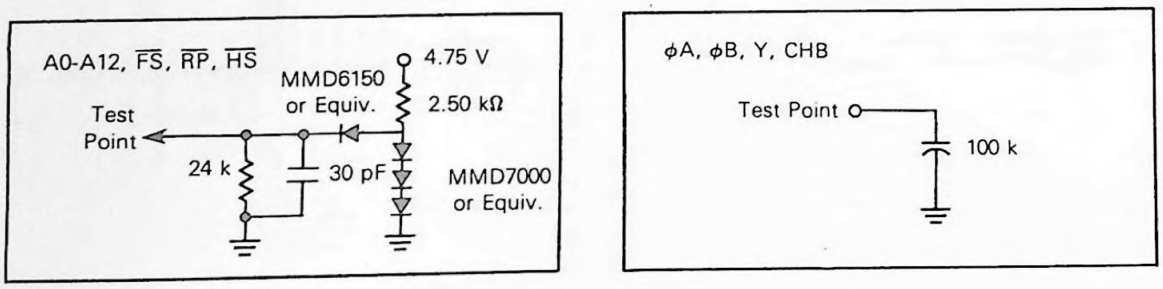
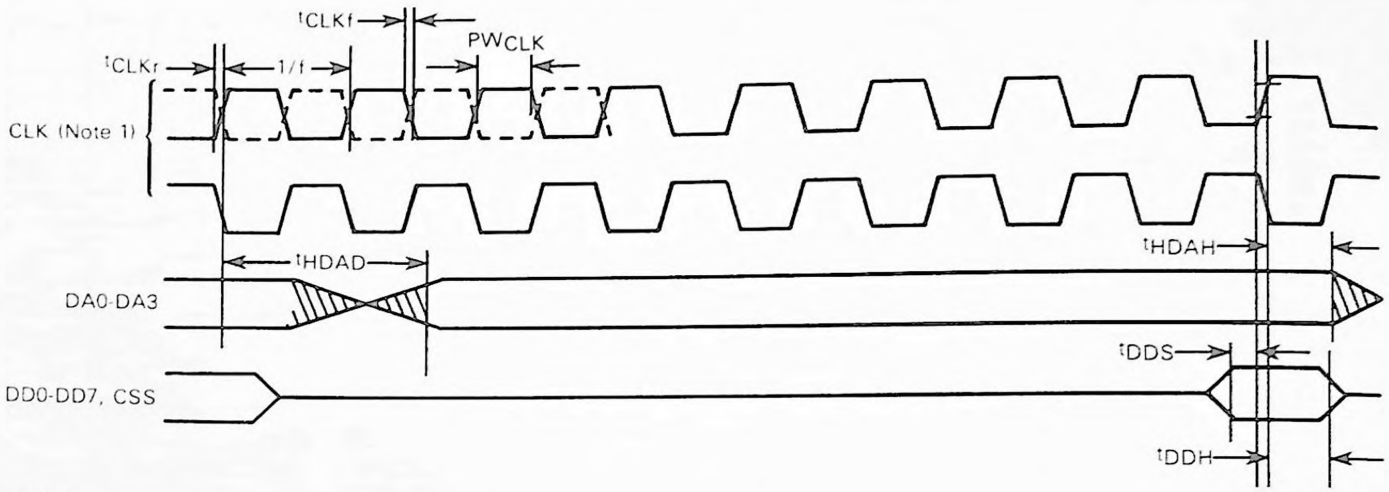
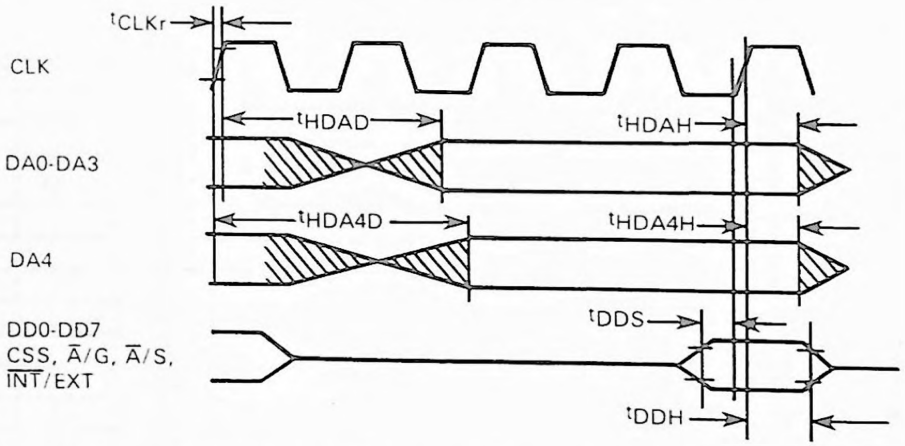


FIGURE 4 — CLOCK AND LONG CYCLE HORIZONTAL ACCESS TIMING



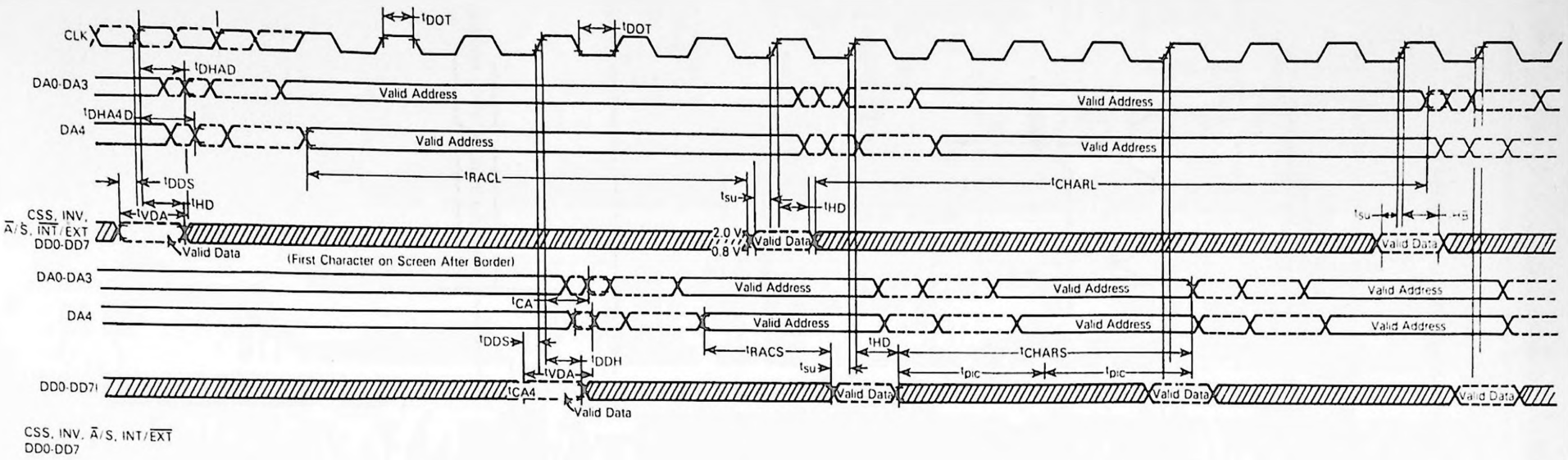
NOTES:
1. The VDG may power-up using either the rising or falling edge of the clock (dotted line).
2. Transitions of DA4-DA12 occur outside the display area. DA0-DA3 access the 16 bytes of data displayed during each scan line in the display area.
3. Long cycle timing applies to CG1, RG1, RG2, and RG3 modes (see Table 3). $\overline{A}/G$ is high; AS, $\overline{INT}/EXT$, and INV input levels do not affect the VDG in long cycle modes.
4. Usable RAM access time for the long cycle may be calculated using the following equation:
   $t_{RACL} = 8 \cdot 1/f_{max} - t_{HDAD_{max}} - t_{DDS_{max}} - t_{CLK}$
   If address and data buffers are used, the access time must be adjusted accordingly.
5. All timing is measured to and from a low voltage of 0.8 volts and a high voltage of 2.0 volts unless otherwise specified.

FIGURE 5 — SHORT CYCLE HORIZONTAL ACCESS TIMING



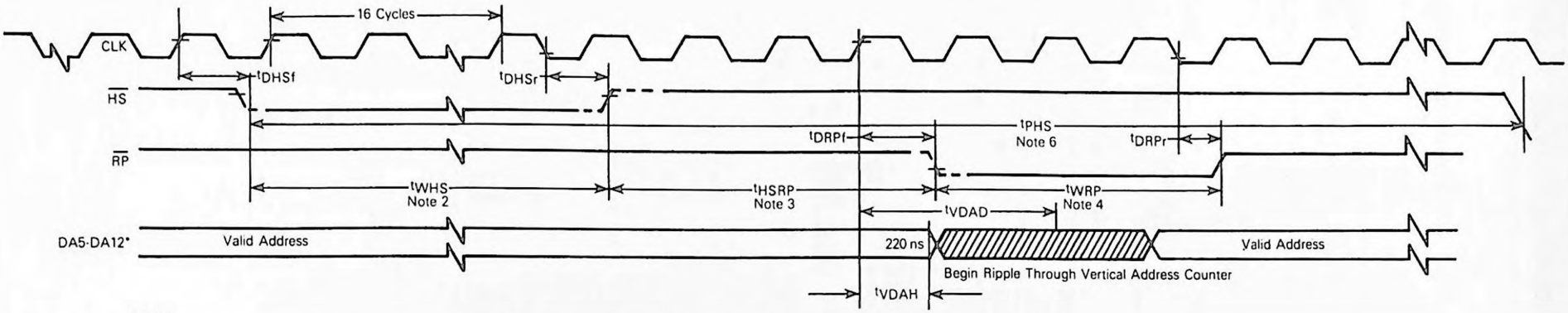
NOTES:
1. The VDG may power-up using either the rising or falling edge of the clock as shown in Figure 4.
2. Transitions of DA5-DA12 occur outside the display area. DA0-DA4 access the 32 bytes of data displayed during each scan line in the display area.
3. Short cycle timing applies to the four alphanumeric modes, two semigraphic modes, and to the CG2, CG3, CG6, RG6 modes (see Table 3). For the four graphic modes, $\overline{A}/G$ is high and the $\overline{A}/S$, $\overline{INT}/EXT$, and INV input levels do not affect the VDG.
4. Usable RAM access time for the short cycle may be calculated using the following equation:
   $t_{RACS} = 4 \cdot 1/f_{max} - t_{HDA4D_{max}} - t_{DDS_{max}} - t_{CLKr}$
   If address and data buffers are used, the access time must be adjusted accordingly.
5. All timing is measured to and from a low voltage of 0.8 volts and a high voltage of 2.0 volts unless otherwise specified.

FIGURE 6 — HORIZONTAL ADDRESS AND VALID DATA SETUP AND HOLD TIMING
(Timing Relationships Shown From Beginning of Line)



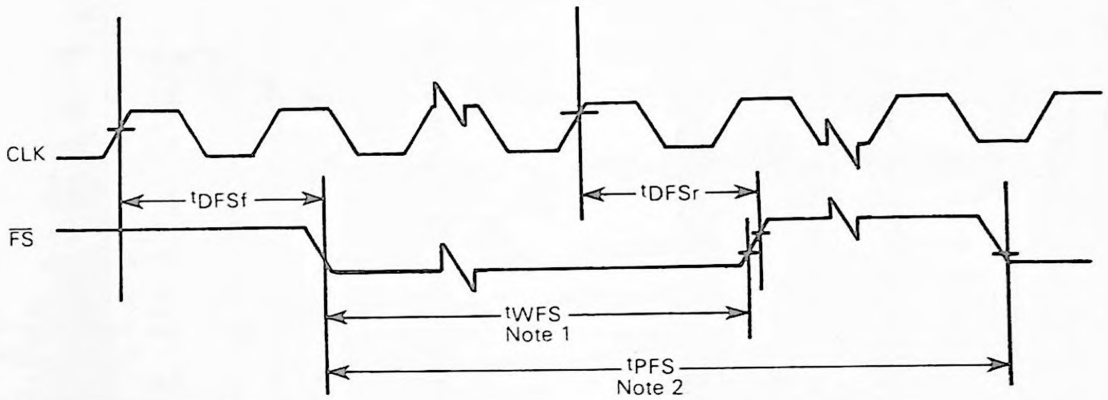
*Long element/access modes: CG1, RG1, RG2, RG3
**Short element/access mode: CG2, CG3, CG6, RG6, Alphanumerics, Semigraphics

FIGURE 7 — VERTICAL ADDRESS, ROW PRESET AND HORIZONTAL SYNCHRONIZATION TIMING



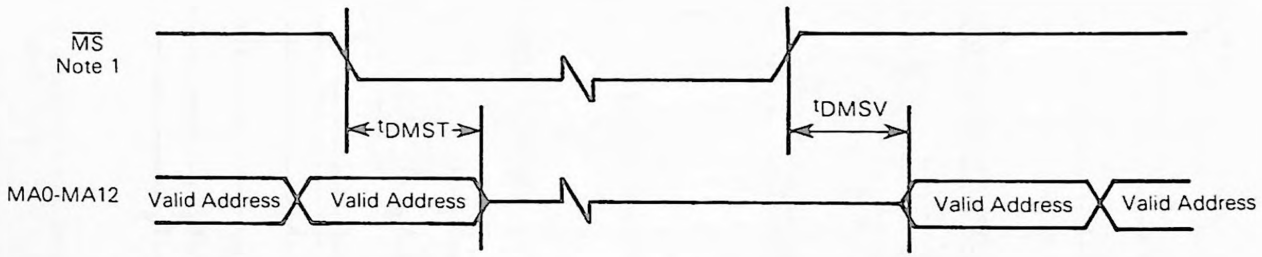
NOTES:
1. All timing is measured to and from a low voltage of 0.8 volts and a high voltage of 2.0 volts unless otherwise specified.
2. HS pulse width may be determined by $t_{WHS} = 16.5 \cdot 1/f - t_{DHSF} + t_{DHSr}$.
3. HS to RP may be determined by $t_{HSRP} = 3.5 \cdot 1/f - t_{DHSr} + t_{DRPf}$.
4. RP pulse width may be determined by $t_{WRP} = 3.5 \cdot 1/f - t_{DRPf} + t_{DRPr}$.
5. DA5-DA12 will change during the inactive portion of the display.
6. $t_{PHS} = 227.5 \cdot 1/f$.
7. $t_{DOT} = \frac{1}{2} f$.

FIGURE 8 — FIELD SYNC ($\overline{FS}$) TIMING



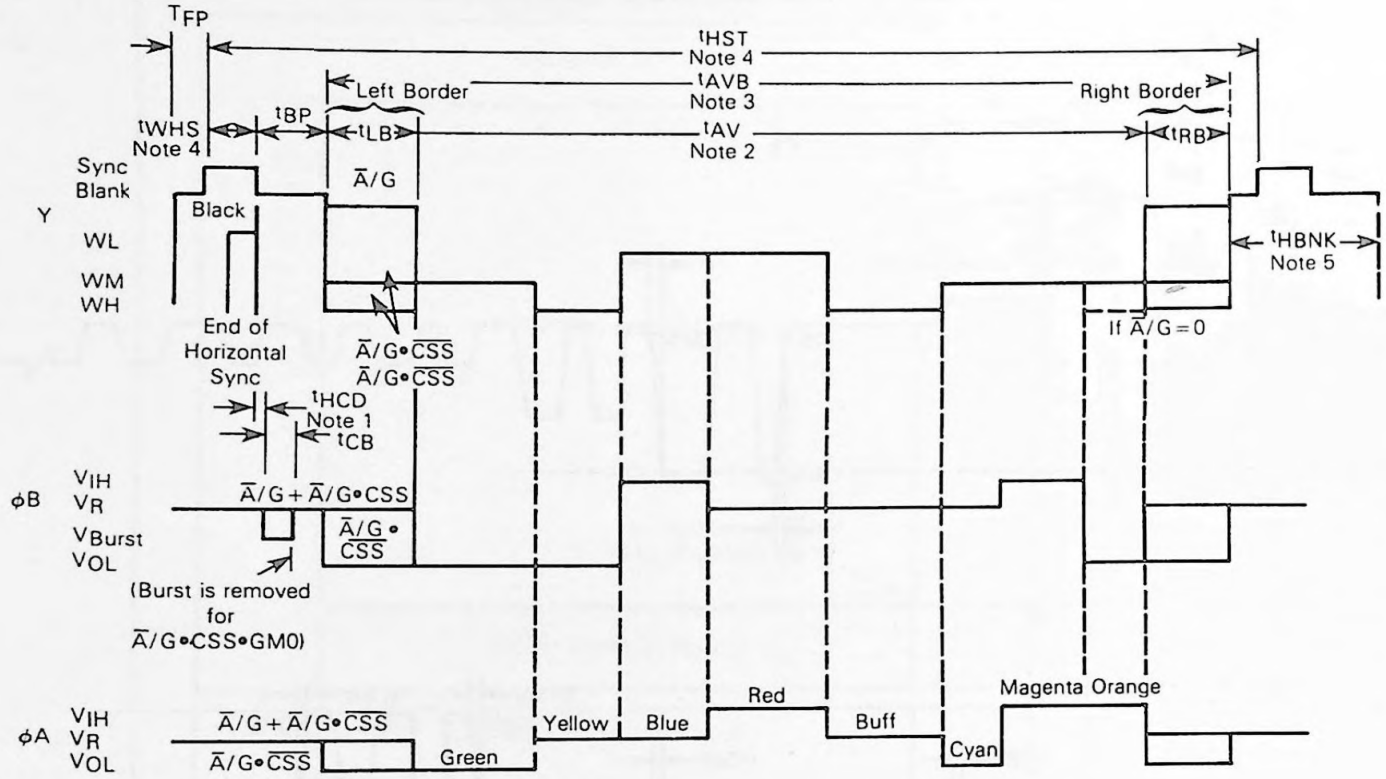
NOTES:
1. $t_{WFS} = 32 \cdot t_{PHST} = 32 \cdot (277.5 \cdot 1/f)$
2. $t_{PFS} = 262 \cdot t_{PHST} = 262 \cdot (227.5 \cdot 1/f)$ for MC6847
   $t_{PFS} = 262.5 \cdot t_{PHST} = 262.5 \cdot (227.5 \cdot 1/f)$ for MC6847Y

FIGURE 9 — MEMORY SELECT ($\overline{MS}$) TIMING



NOTES:
1. MS is asserted asynchronously with respect to CLK.

FIGURE 10 — VIDEO AND CHROMINANCE OUTPUT WAVEFORM RELATIONSHIPS



NOTES:
1. $t_{HCD} = 3.5 \cdot 1/f$
2. $t_{AV} = 128 \cdot 1/f$
3. $t_{AVB} = 185.5 \cdot 1/f$
4. Refer to Figure 7
5. $t_{HBNK} = 42 \cdot 1/f$

FIGURE 11 — CHROMA PHASE DELAY

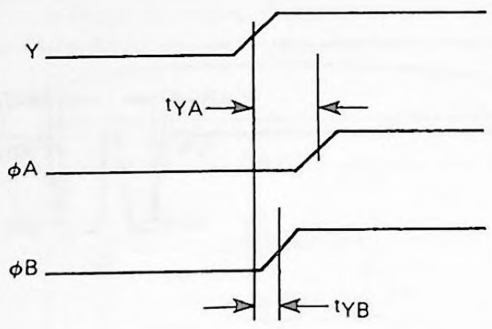


FIGURE 12 — TIMING DIAGRAMS
VIDEO RISE AND FALL TIMES (Illustrates Beginning of One Horizontal Line)

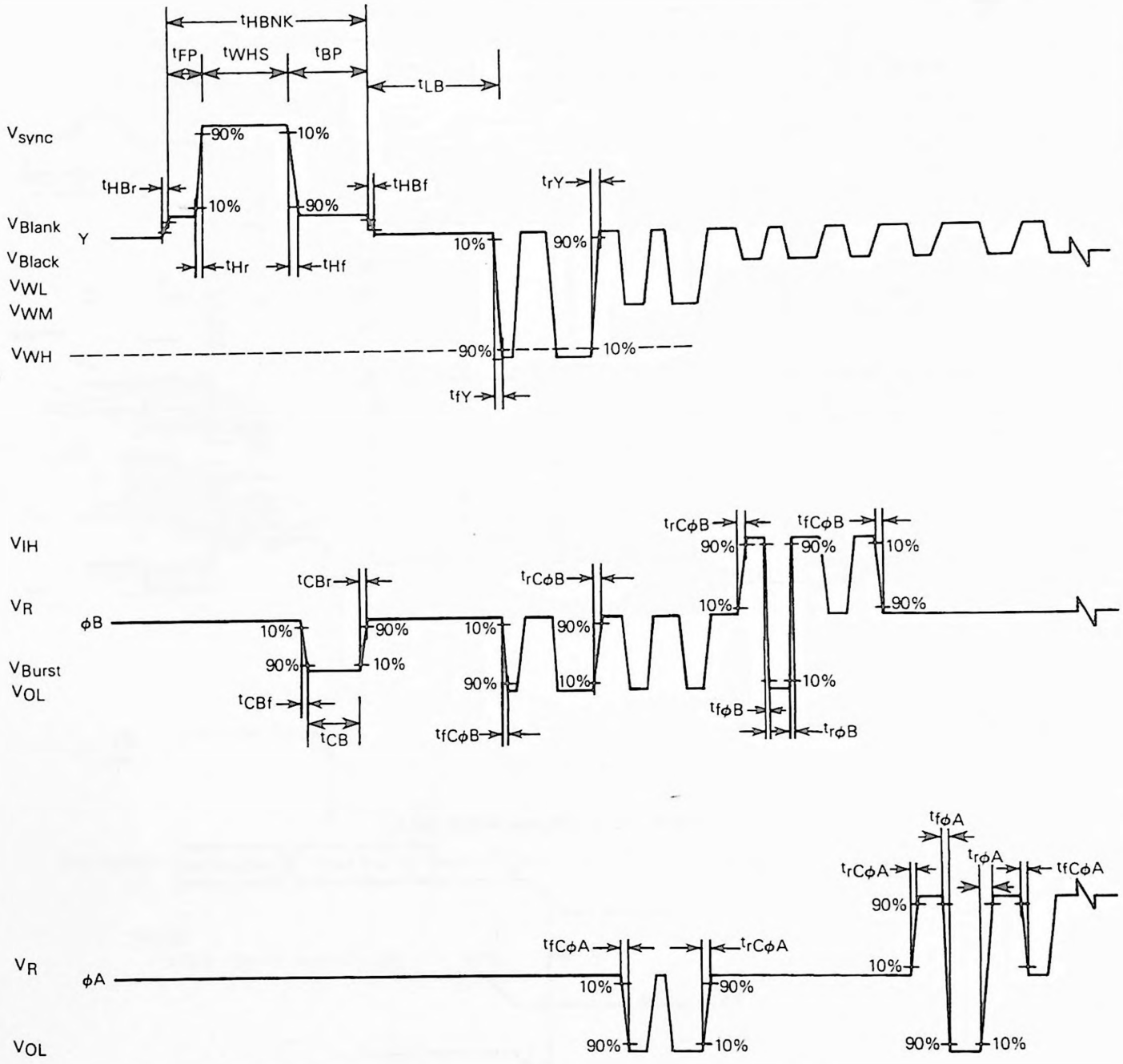


FIGURE 13 — DISPLAY AREA TIMING

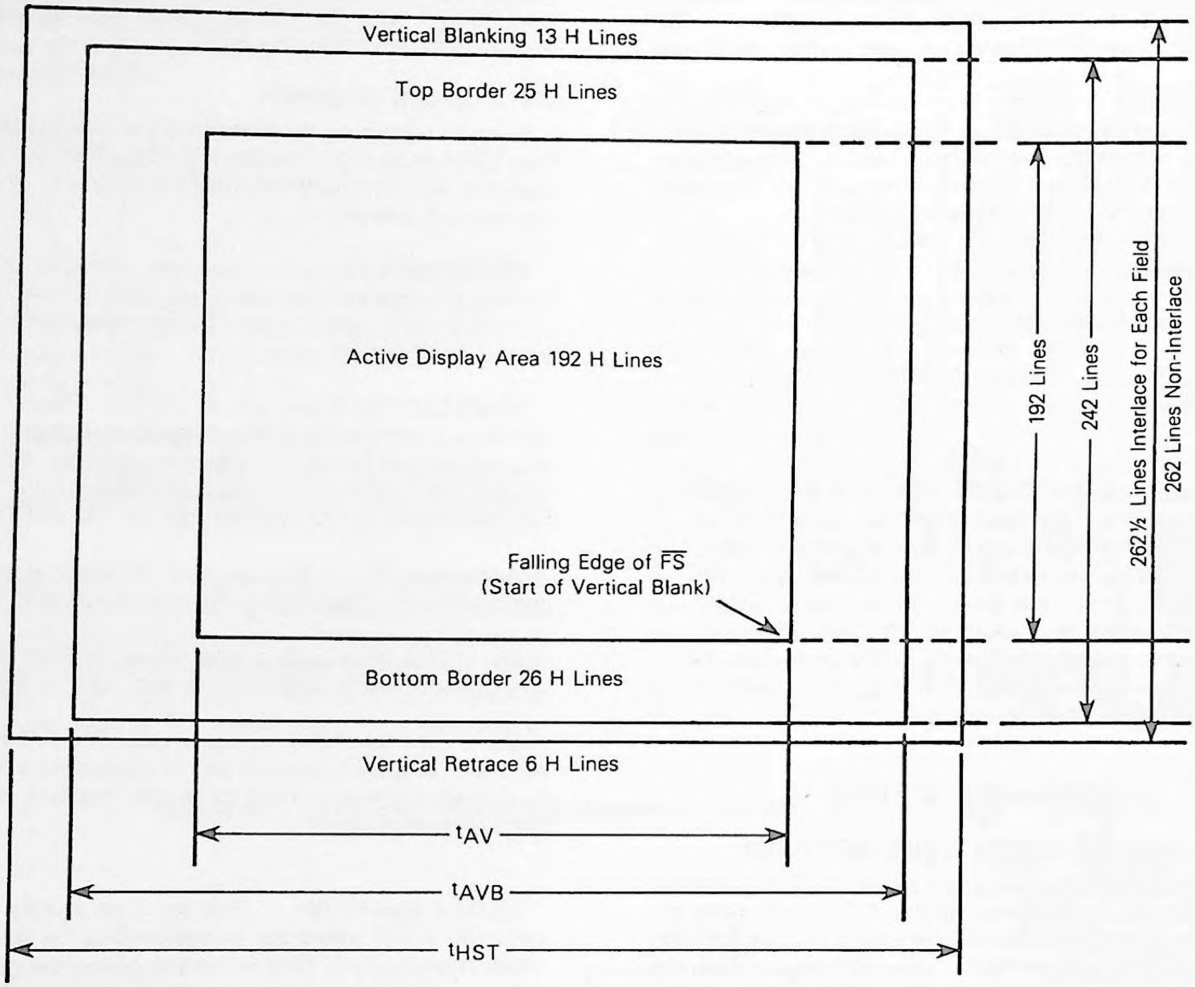
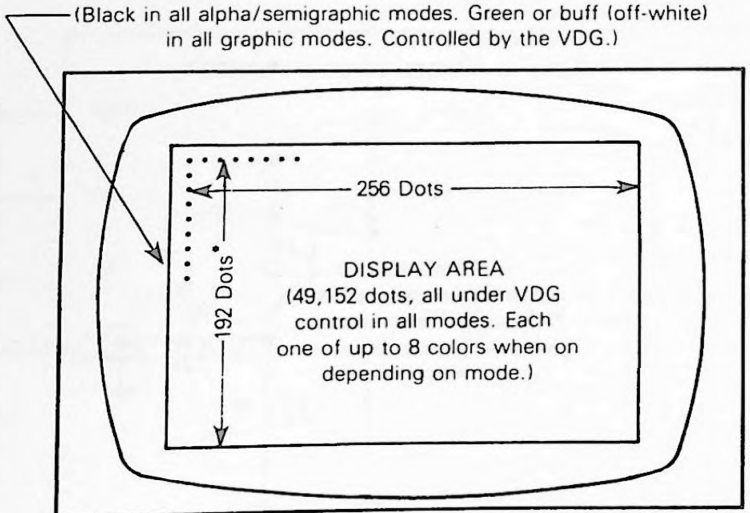


FIGURE 14 — TYPICAL FORMAT OF THE TELEVISION SCREEN



*One on each non-interlaced line; for interlace, the lines of the odd field are copied into the even field thus doubling the number of displayed dots.

VIDEO DISPLAY GENERATOR DESCRIPTION

The MC6847/MC6847Y video display generators provide a simple interface for display of digital information on a color monitor or standard color/black and white television receiver.

Television transmissions in North and South America and Japan conform to the National Television System Committee (NTSC) standards. This system is based on a field repetition rate of 60 fields per second. There are 525 interlaced lines per frame or one-half this number per field.

The MC6847 scans one field of 262 lines 60 times per second. The MC6847 non-interlace VDG is recommended for use in systems (i.e., TV games and personal computers) where absolute NTSC compatibility is not required. If NTSC compatibility is required, perhaps for caption overlays on broad-case signals, then the MC6847Y interlace VDG is recommended.

NOTE

A system with the MC6847 VDG and the MC1372 video modulator forms a transmitter, transmitting at 61.2 MHz (channel 3) or 67.25 MHz (channel 4) depending on component values chosen. This being a Class I TV device, care must be taken to meet FCC requirements Part 15, Subpart H. However, if the composite video output from the MC1372 were to drive the television directly, Section 15.7 of the FCC specification must be adhered to.

SIGNAL DESCRIPTION

DISPLAY ADDRESS OUTPUT LINES (DA0-DA12)

Thirteen address lines are used by the VDG to scan the display memory as shown in Figures 4-7. The starting address of the display memory is located at the upper left corner of the display screen. As the television sweeps from the left to right and top to bottom, the VDG increments the RAM display address. The timing for two accesses starting at the beginning of the line is shown in Figure 6. These lines are TTL compatible and may be forced into a high-impedance state whenever MS (pin 12) goes low. A0-A3 change during the active display area. A4 changes during the active display area in the alphanumerics, semigraphics, CG2, CG3, CG6, and RG6 modes. A5-A12 do not toggle within the active display area but instead, ripple through the address during border and blanking time.

DATA INPUTS (DD0-DD7)

Eight TTL compatible data lines are used to input data from RAM to be processed by the VDG. The data is then interpreted and transformed into luminance (Y) and chroma outputs (ϕA and ϕB).

POWER INPUTS — V_{CC} requires +5 volts ±5%. V_{SS} requires zero volts and is normally ground. The tolerance and current requirements of the VDG are specified in the Electrical Characteristics.

VIDEO OUTPUTS (ϕA, ϕB, Y, CHB) — These four analog outputs are used to transfer luminance and color information to a standard NTSC color television receiver, either via the MC1372 RF modulator or via drivers directly into Y, ϕA, ϕB television video inputs (see Figures 10, 11, and 12).

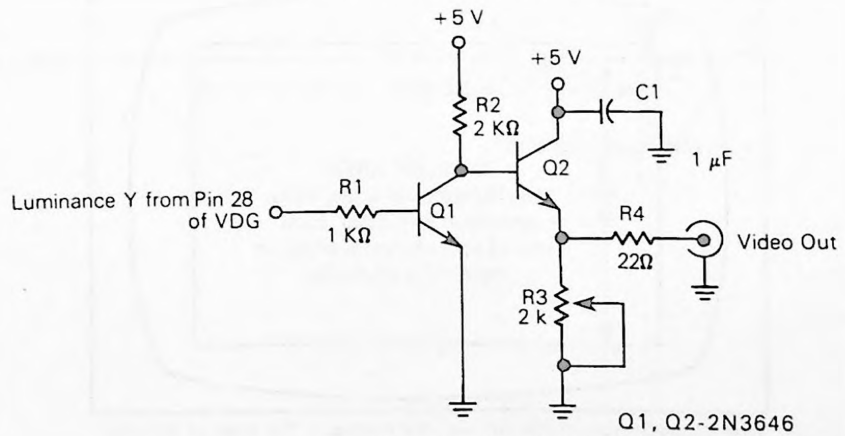
Luminance (Y) — This six level analog output contains composite sync, blanking and four levels of video luminance.

ϕA — This three level analog output is used in combination with ϕB and Y outputs to specify one of eight colors.

ϕB — This four level output is used in combination with ϕA and Y outputs to specify one of eight colors. Additionally, one analog level is used to specify the time of the color burst reference signal.

Chroma Bias (CHB) — This pin is an analog output and provides a DC reference corresponding to the quiescent value of ϕA and ϕB. CHB is used to guarantee good thermal tracking and minimize the variation between the MC1372 and MC6847. This pin, when pulled low, resets certain registers within the chip. In a user's system, this pin should not normally be used as an input. It is used mainly to enhance test capabilities within the factory.

FIGURE 15 — VIDEO TO B&W MONITOR



Q1, Q2-2N3646

Contributed by Ted Hanwick, Honeywell, Inc.

SYNCHRONIZING INPUTS ($\overline{MS}$, CLK)

THREE-STATE CONTROL — ($\overline{MS}$) is a TTL compatible input which, when low, forces the VDG address lines into a high-impedance state, as shown in Figure 9. This may be done to allow other devices (such as an MPU) to address the display memory (RAM).

CLOCK (CLK) — The VDG clock input (CLK) requires a 3.579545 MHz (standard color burst) TV crystal frequency square wave. The duty cycle of this clock must be between 45 and 55% since it controls the width of alternate dots on the television screen. The MC1372 RF modulator may be used to supply the 3.579545 MHz clock and has provisions for a duty cycle adjustment. The VDG will power-up using either the rising or falling edge of the clock. The dotted line on the CLK signal in Figure 4 indicates this characteristic of latching in data on either clock edge.

SYNCHRONIZING OUTPUTS ($\overline{FS}$, $\overline{HS}$, $\overline{RP}$)

Three TTL compatible outputs provide circuits, exterior to the VDG, with timing references to the following internal VDG states:

FIELD SYNC ($\overline{FS}$) — The high-to-low transition of the $\overline{FS}$ output coincides with the end of active display area (see Figure 8). During this time interval, an MPU may have total access to the display RAM without causing undesired flicker on the screen. The low-to-high transition of $\overline{FS}$ coincides with the trailing edge of the vertical synchronization pulse.

HORIZONTAL SYNC ($\overline{HS}$) — The $\overline{HS}$ pulse coincides with the horizontal synchronization pulse furnished to the television receiver by the VDG (see Figure 7). The high-to-low transition of the $\overline{HS}$ output coincides with the leading edge of the horizontal synchronization pulse and the low-to-high transition coincides with the trailing edge.

ROW PRESET ($\overline{RP}$) — If desired, an external character generator ROM may be used with the VDG. However, an external four bit counter must be added to supply row addresses. The counter is clocked by the $\overline{HS}$ signal and is cleared by the $\overline{RP}$ signal. $\overline{RP}$ pulses occur in all alphanumeric and semigraphics modes; no pulses are output in the full graphic modes. $\overline{RP}$ occurs after the first valid 12 lines. Therefore, use an $\overline{FS}$ clocked preloadable counter such as a 74LS161 as shown in Figures 7, 14, and 23.

MODE CONTROL LINES INPUT ($\overline{A}$/G, $\overline{A}$/S, $\overline{INT}$/EXT, GM0, GM1, GM2, CSS, INV)

Eight TTL compatible inputs are used to control the operating mode of the VDG. $\overline{A}$/S $\overline{INT}$/EXT, CSS, and INV may be changed on a character-by-character basis. The CSS pin is used to select between two possible alphanumeric colors when the VDG is in the alphanumeric mode and between two color sets when the VDG is in the Semigraphics 6 or full graphic modes. Table 1 illustrates the various modes that can be obtained using the mode control lines. There are two different types of memory access concerning these modes, they are a short and a long access cycle, which differ by a

FIGURE 16 — EXTERNAL CHARACTER GENERATOR ROW COUNTER FOR MC6847

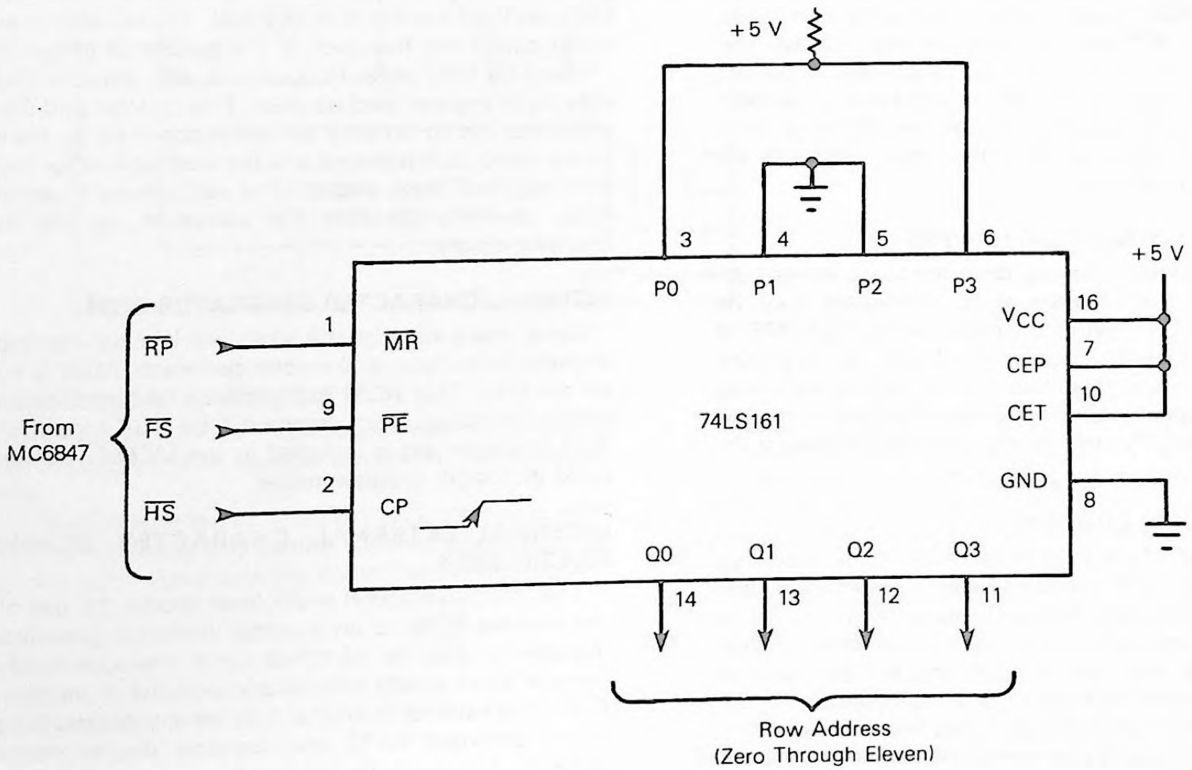


TABLE 1 — MODE CONTROL LINES (INPUTS)

$\overline{A}/G$	$\overline{A}/S$	$\overline{INT}/EXT$	INV	GM2	GM1	GM0	Alpha/Graphic Mode Select	# of Colors
0	0	0	0	X	X	X	Internal Alphanumerics	2
0	0	0	1	X	X	X	Internal Alphanumerics Inverted	
0	0	1	0	X	X	X	External Alphanumerics	
0	0	1	1	X	X	X	External Alphanumerics Inverted	
0	1	0	X	X	X	X	Semigraphics 4 (SG4)	8
0	1	1	X	X	X	X	Semigraphics 6 (SG6)	8
1	X	X	X	0	0	0	64 × 64 Color Graphics One (CG1)	4
1	X	X	X	0	0	1	128 × 64 Resolution Graphics One (RG1)	2
1	X	X	X	0	1	0	128 × 64 Color Graphics Two (CG2)	4
1	X	X	X	0	1	1	128 × 96 Resolution Graphics Two (RG2)	2
1	X	X	X	1	0	0	128 × 96 Color Graphics Three (CG3)	4
1	X	X	X	1	0	1	128 × 192 Resolution Graphics Three (RG3)	2
1	X	X	X	1	1	0	128 × 192 Color Graphics Six (CG6)	4
1	X	X	X	1	1	1	256 × 192 Resolution Graphics Six (RG6)	2

shift of one full 3.58 MHz cycle. One of the differences between these access times, in the short access time frame, is a shift of one full 3.58 MHz cycle from the corresponding normal long access time frame, as shown in Figure 6. The modes using short access times read memory twice as often as the long access modes.

OPERATION OF THE VDG

A simplified block diagram of the VDG is shown in Figure 17a and a detailed block diagram is shown in Figure 17b.

The externally generated 3.58 MHz color burst clock drives the VDG. Referring to Figures 11 and 12, note that the horizontal screen span from blanking to blanking is 193.1 clock periods ($\approx 53.95\ \mu s$). The display window is offset from the left-hand edge by 283 periods and lasts for 128 periods ($35.75\ \mu s$). Of the 242 lines on the vertical screen from blanking to blanking, 192 lines are used for the display. The display window is offset from the top by 25 lines. Under the constraint of the master clock, the smallest display element possible for the VDG is half period of the 3.58 MHz clock wide by one scan line high. All other display elements are multiples of this basic size.

DISPLAY MEMORY ADDRESS DRIVERS

The address drivers normally drive the video refresh address into the display memory so characters may be displayed on the CRT. When the memory select pin (MS) is pulled low by an external decoder, the driver outputs go to a high-impedance state so external three-state drivers may switch the MPU produced address onto the display memory address bus; the MPU may directly manipulate data in the display memory.

VIDEO TIMING AND CONTROL

This subsystem of the VDG includes the mode decoding, timing generation, and associated row counter logic, and uses the 3.58 MHz color frequency to generate horizontal and vertical timing information (via linear shift register counters), which the video and chroma encoder uses to generate color video information. The horizontal timing for the VDG is summarized in Figure 7. Ten and one-half cycles of the 3.58 MHz subcarrier are transmitted on the back porch of every horizontal blanking period. This color burst is suppressed during vertical sync and equalizing intervals. Color burst is also suppressed in the most dense two color graphic modes. This leads to some interesting rainbow effects on the display which is frequency and pattern dependent. The vertical timing for the VDG is given in Figure 18. Vertical retrace is initiated by the luminance signal being brought to the blanking level. The vertical blanking period begins with three lines of equalizing pulses followed by three lines of serrated vertical sync pulses followed by three more lines of equalizing pulses. The remaining vertical blanking period contains the normal horizontal sync pulses. The equalizing and serration pulses are at half line frequency. Notice the difference in spacing between the last horizontal sync pulse and the first equalizing pulse in even and odd fields. It is the half line difference between fields that produces the interlaced picture in a frame. Vertical timing between fields for the non-interlaced VDG, on the other hand, is identical. The equalizing and serration pulses are, however, at the horizontal frequency.

The 3.58 MHz color frequency is also used to clock the video shift register load counter. This counter and the video shift clock inhibit circuitry derive the dot-clock for the output of the video shift registers and the load signals for the video shift registers' input latches. The vertical and horizontal address counters generate the addresses for the external display memory.

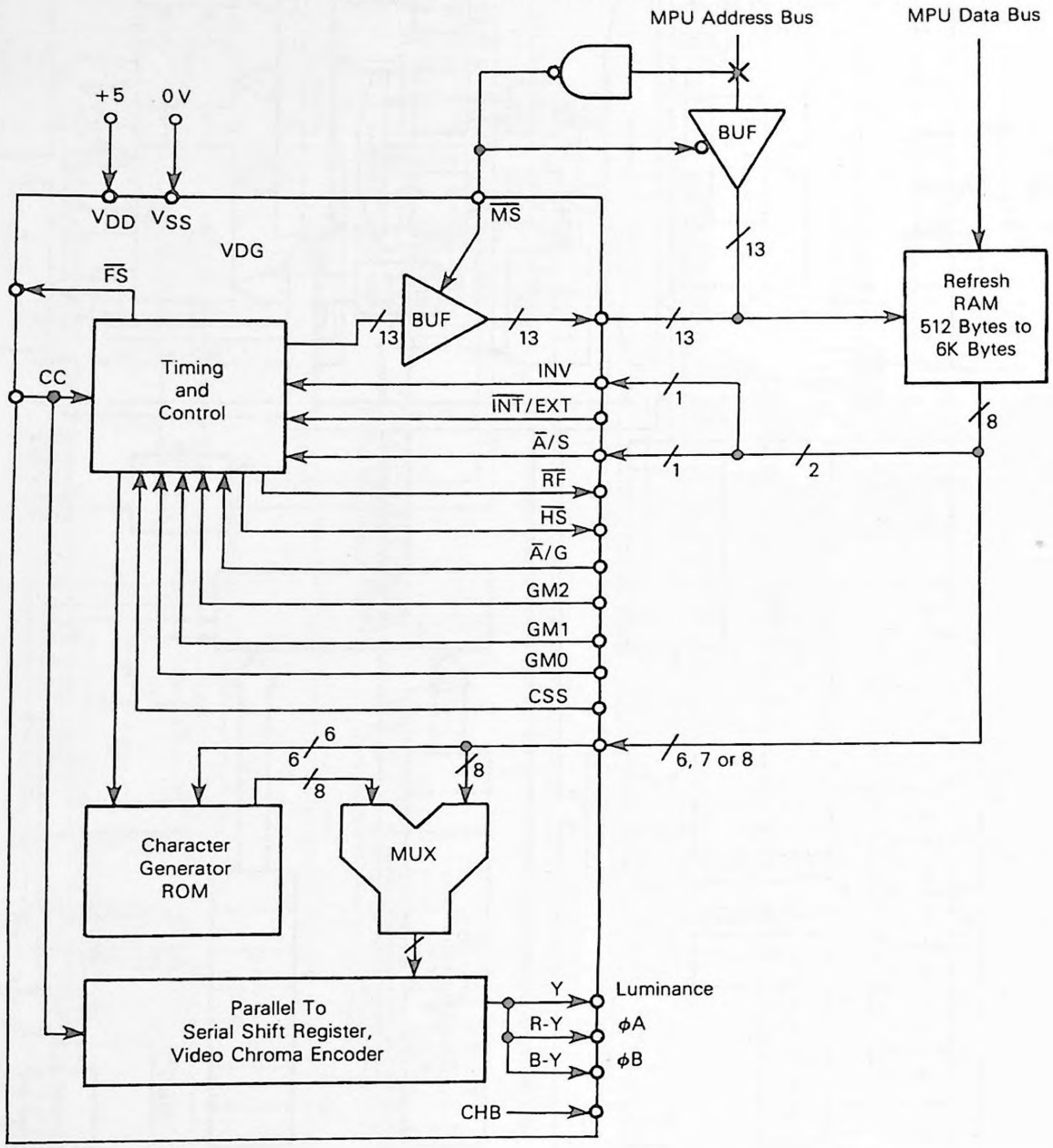
INTERNAL CHARACTER GENERATOR ROM

Since many uses of the VDG will involve the display of alphanumeric data, a character-generator ROM is included on the chip. This ROM will generate 64 standard 5 × 7 dot matrix characters from standard 6-bit ASCII input. A standard character set is included in the MC6847 although the ROM is custom programmable.

INTERNAL/EXTERNAL CHARACTER GENERATOR MULTIPLEXER

The internal/external multiplexer allows the use of either the internal ROM or an external character generator. This multiplexer may be switched on a character-by-character basis to allow mixed internal and external characters on the CRT. The external character may be any desired dot-pattern in the standard 8 × 12 one-character display matrix, thus allowing the maximum 256 × 192 screen density.

FIGURE 17a — SIMPLIFIED VDG BLOCK DIAGRAM



VIDEO AND COLOR SUBSYSTEM

The 8-bit output of the internal/external multiplexer is serialized in an 8-bit shift register clocked at the dot-clock frequency.

The luminance information from the shift register is summed with the horizontal and vertical sync signals to produce a composite video signal less the chrominance information, called Y. The luminance signal, Y, and the two chrominance outputs, ϕA (R-Y) and ϕB (B-Y), can be combined (modulated) by an MC1372 into a composite video signal with color. Figures 8, 9, 10, and 16 show the relationship between the luminance and chrominance signals and the resultant color.

FIGURE 17b — DETAILED VDG BLOCK DIAGRAM

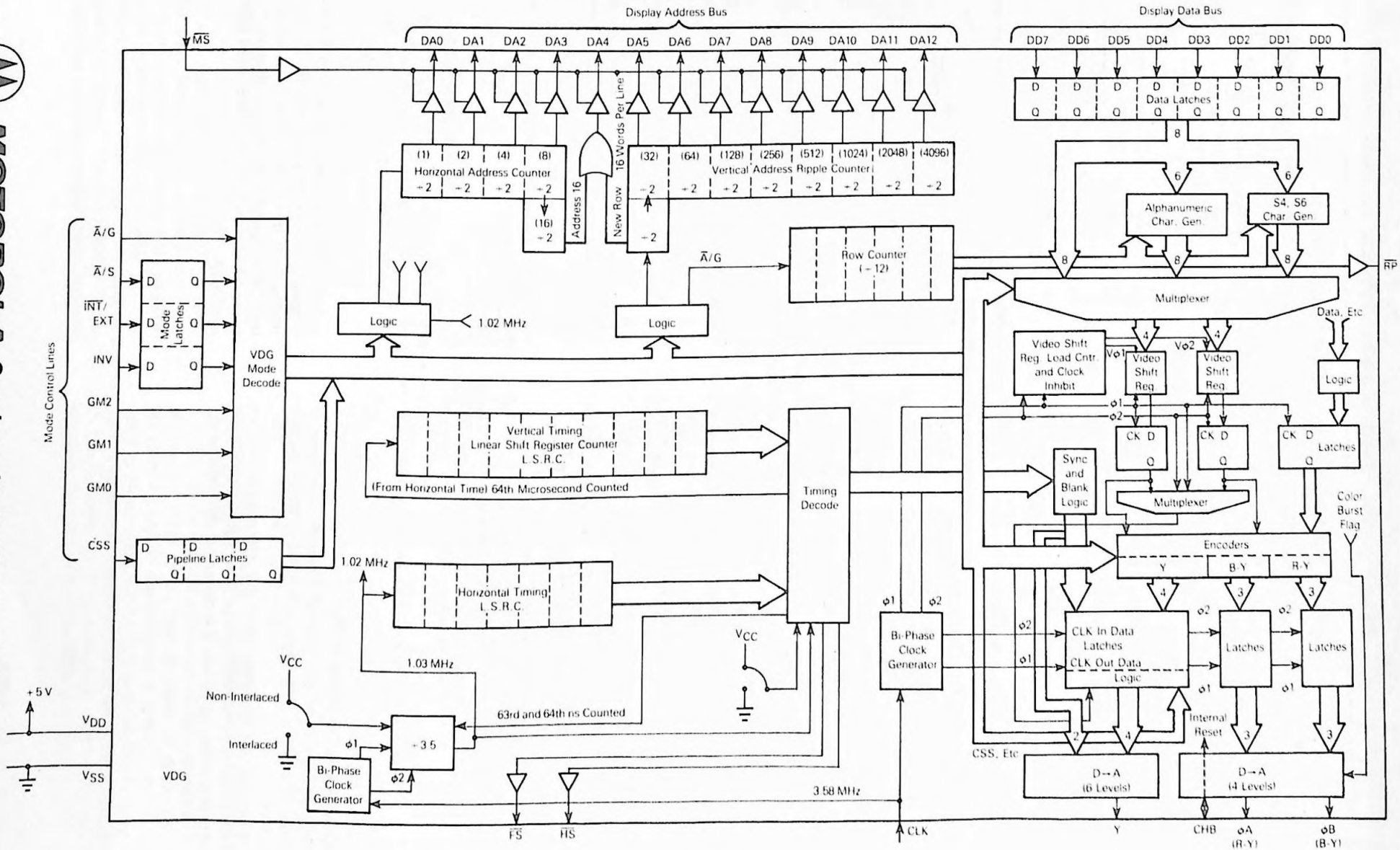
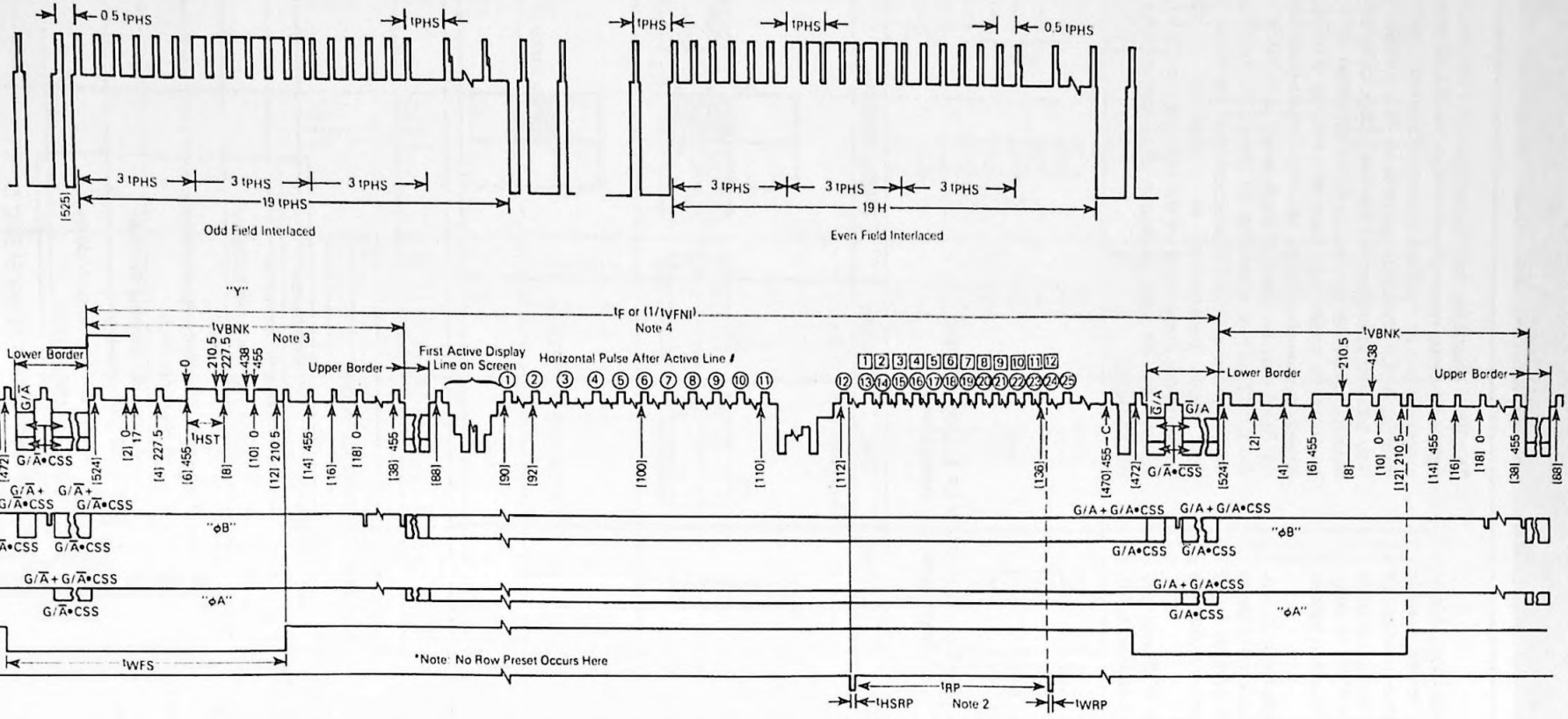


FIGURE 18 — NON-INTERLACE VERTICAL TIMING
(For Interlace Vertical Timing Use Inserts)



NOTES:

1. [ ] # of 31.468 kHz clocks
   (1/31.468 kHz) = 31.7783 μs
   Non-Parentheses = # of 3.58 MHz Clocks
   1/3.58 MHz = 279.366 ns
   Time marks 455 and 0 are the same points in time.

   Example Timing Calculations:
   t_{HST} = (227.5-0) × 279.366 ns = 63.5μs
   = (455-227.5) × 279.366 ns = 63.5 μs
   t_{WHS} = (455-438) × 279.366 ns = 4.75 μs
   Lower Border = (524-472) × 31.7783 μs-t_{HBNK}
   = 1.6525 ms-11.6 ns = 1.64 ms

   Upper Border = (88 – 38) × 31.7783 μs – t_{HBNK} =
   = 1.5889 μs-11.6 μs = 1.58 ms

2. t_{RP} = 12 horizontal scan lines
3. t_{VBNK} = 20•t_{PHS} = 20•(227.5•1/f)
4. t_F = 262•t_{PHS} = 262•(227.5•1/f) for Non-Interlace.
   t_F = 262.5•T_{PHS} = 262.5•(227.5•1/f) for Interlace.

*t_{HBNK} could actually be considered as part of the border – especially for purposes of writing to the screen. The same holds true for the upper border.

DISPLAY MODES

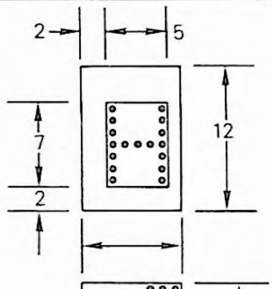
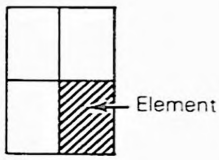
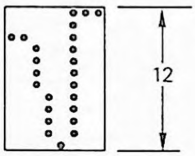
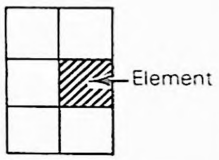
There are two major display modes in the VDG. Major mode 1 contains four alphanumeric and two limited graphic modes. Major mode 2 contains eight graphic modes. Of these, four are full color graphic and four restricted color graphic modes. The mode selection for the VDG is summarized in Table 2. The mnemonics of these fourteen modes are explained in the following sections.

In major mode 1 the display window is divided into 32 columns by 16 character element rows thus requiring 512 bytes of memory. Each character element is 8 half periods by 12 scan lines in size as shown in Figure 19. The area outside the display window is black.

The VDG has a built-in character generator ROM containing the 64 ASCII characters in a 5 × 7 format (see Figure 20). The 5 × 7 character font is positioned two columns to the right and three rows down within the 8 × 12 character element. Six bits of the 8-bit data word are typically used for the internal ASCII character generator. The remaining two bits may be used to implement inverse video, color switching, or external character generator ROM selection on a character-by-character basis. For those who wish to display lower case letters, special characters, or even limited-graphics, an external ROM may be used. If such external ROM is used, all of the 8 × 12 picture elements, or pixels, in the character element can be utilized. Characters may be either green on a dark green background or orange on a dark orange background, depending on the state of the CSS pin. The invert pin can be used to display dark characters on a bright background.

TABLE 2 — SUMMARY OF MAJOR MODES

Major Mode 1 — Alpha Modes

Title	Memory	Display Elements	Colors	Title	Memory	Display Elements	Colors
Alphanumerics (Internal)	512 × 8		2	Semigraphic 4	512 × 8		8
Alphanumerics (External)	512 × 8		2	Semigraphic 6	512 × 8		4

Major Mode 2 — Graphics Modes

Title	Memory	Colors	Comments
64 × 64 Color Graphic	1 k × 8	4	Matrix 64 × 64 Elements
128 × 64 Graphics*	1 k × 8	2	Matrix 128 Elements Wide by
128 × 64 Color Graphic	2 k × 8	4	64 Elements High
128 × 96 Graphics*	1.5 k × 8	2	Matrix 128 Elements Wide by
128 × 96 Color Graphic	3 k × 8	4	96 Elements High
128 × 192 Graphics*	3 k × 8	2	Matrix 128 Elements Wide by
128 × 192 Color Graphic	6 k × 8	4	192 Elements High
256 × 192 Graphics	6 k × 8	2	Matrix 256 Elements Wide by 192 Elements High

*Graphics mode turns on or off each element. The color may be one of two.

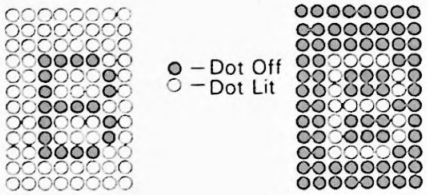
TABLE 3 — DETAILED DESCRIPTION OF VDG MODES

VDG Pins									Color			TV Screen		VDG Data Bus	Comments
MS	G/A	S/A	EXT/INT	GM2	GM1	GM0	CSS	INV	Character Color	Background	Border	Display Mode	Detail		
1	0	0	0	X	X	X	0 1	0 1 0 1	Green Black Orange Black	Black Green Black Orange	Black Black	32 Characters per row 16 Characters in rows	8 wide × 12 high Internal Alphanumerics	extra, ASCII Code	The ALPHANUMERIC INTERNAL mode uses an internal character generator (which contains the following five dot by seven dot characters: SP ! " # $ % & ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; < = > ? @ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [\ ] ↑ ←). The six bit ASCII code leaves two bits free and these may be externally connected to the mode pins (G/A, S/A, EXT/INT, GM2, GM1, GM0, CSS or INV).
1	0	0	1	X	X	X	0 1	0 1 0 1	Green Black Orange Black	Black Green Black Orange	Black Black	32 Characters per row 16 Characters in rows	8 wide × 12 high	One Row of Custom Characters	The ALPHANUMERIC EXTERNAL mode uses an external character generator as well as a row counter. Thus, custom character fonts or graphic symbol sets with up to 256 different 8 × 12 dot "characters" may be displayed.
1	0	1	0	X	X	X	X	X	Lx C2 C1 C0 Color 0 X X X Black 1 0 0 0 Green 1 0 0 1 Yellow 1 0 1 0 Blue 1 0 1 1 Red 1 1 0 0 Buff 1 1 0 1 Cyan 1 1 1 0 Magenta 1 1 1 1 Orange		Black	64 Display elements per row 32 Display elements in rows in rows	4 wide × 6 high L_3 L_2 / L_1 L_0 = One Element	extra, C_2 C_1 C_0 L_3 L_2 L_1 L_0	The SEMIGRAPHICS FOUR mode uses an internal "course graphics" generator in which a rectangle (eight dots by twelve dots) is divided into four equal parts. The luminance of each part is determined by a corresponding bit on the VDG data bus. The color of illuminated parts is determined by three bits.
1	0	1	1	X	X	X	0 1	X	Lx C1 C0 Color 0 X X Black 0 0 1 Green 0 1 0 Yellow 0 1 1 Blue 1 0 0 Red 1 0 1 Buff 1 1 0 Black 1 1 1 Cyan 1 1 1 Magenta 1 1 1 Orange		Black	64 Display elements per row 48 Display elements in rows	4 wide × 4 high L_5 L_4 / L_3 L_2 / L_1 L_0 = One Element	C_1 C_0 L_5 L_4 L_3 L_2 L_1 L_0	The SEMIGRAPHIC SIX mode is similar to the SEMIGRAPHIC FOUR mode with the following differences. The eight dot by twelve dot rectangle is divided into six equal parts. Color is determined by the two remaining bits.
1	1	X	X	0	0	0	0 1	X	C1 C0 Color 0 0 Green 0 1 Yellow 1 0 Blue 1 1 Red 0 0 Buff 0 1 Cyan 1 0 Magenta 1 1 Orange		Green Buff	64 Display elements per row 64 Display elements	4 wide E_3 E_2 E_1 E_0	C_1 C_0 C_1 C_0 C_1 C_0 C_1 C_0	The COLOR GRAPHICS ONE mode uses a maximum of 1024 bytes of display RAM in which one pair of bits specifies one picture element.
1	1	X	X	0	0	1	0 1	X	Lx 0 1 0 1	Color Black Green Black Buff	Green Buff	128 Display elements per row 64 Display elements in rows	3 wide L_7 L_6 L_5 L_4 L_3 L_2 L_1 L_0	L_7 L_6 L_5 L_4 L_3 L_2 L_1 L_0	The RESOLUTION GRAPHICS ONE mode uses a maximum of 1024 bytes of display RAM in which one bit specifies one picture element.
1	1	X	X	0	1	0	0 1	X	Same color as Color Graphics One		Green Buff	128 Display elements per row 64 Display elements in rows	3 wide E_3 E_2 E_1 E_0	C_1 C_0 C_1 C_0 C_1 C_0 C_1 C_0	The COLOR GRAPHICS TWO mode uses a maximum of 2048 bytes of display RAM in which one pair of bits specifies one picture element.
1	1	X	X	0	1	1	0 1	X	Same color as Resolution Graphics One		Green Buff	128 Display elements per row 96 Display elements in rows	2 wide L_7 ... L_0	L_7 L_6 L_5 L_4 L_3 L_2 L_1 L_0	The RESOLUTION GRAPHICS TWO mode uses a maximum of 1536 bytes of display RAM in which one bit specifies one picture element.
1	1	X	X	1	0	0	0 1	X	Same color as Color Graphics One		Green Buff	126 Display elements per row 96 Display elements in rows	2 wide E_3 E_0	C_1 C_0 C_1 C_0 C_1 C_0 C_1 C_0	The COLOR GRAPHICS THREE mode uses a maximum of 3072 bytes of display RAM in which one pair of bytes specifies one picture element.
1	1	X	X	1	0	1	0 1	X	Same color as Resolution Graphics One		Green Buff	128 Display elements per row 192 Display elements in rows	2 wide L_7 ... L_0	L_7 L_6 L_5 L_4 L_3 L_2 L_1 L_0	The RESOLUTION GRAPHICS THREE mode uses a maximum of 3072 bytes of display RAM in which one bit specifies one picture element.
1	1	X	X	1	1	0	0 1	X	Same color as Color Graphics One		Green Buff	128 Display elements per row 192 Display Elements in rows	2 wide E_3 E_0	C_1 C_0 C_1 C_0 C_1 C_0 C_1 C_0	The COLOR GRAPHICS SIX mode uses a maximum of 6144 bytes of display RAM in which one pair of bits specifies one picture element.
1	1	X	X	1	1	1	0 1	X	Same color as Resolution Graphics One		Green Buff	256 Display elements per row 192 Display elements in rows	1 wide L_7 ... L_0	L_7 L_6 L_5 L_4 L_3 L_2 L_1 L_0	The RESOLUTION GRAPHICS SIX mode uses a maximum of 6144 bytes of display RAM in which one bit specifies one picture element.

AD0048

FIGURE 19 — ALPHANUMERIC MODE (INTERNAL)

512 Characters (32 × 16)
Typical Character



Inverted
Black Character
Orange or Green
Background (Selectable)

Normal
Black Background
Orange or Green
Character (Selectable)

Character Source:
Internal — 6 Bit ASCII Generator ROM On Chip or User Definable
External — Users ROM

The two limited graphic modes are Semigraphics 4 and Semigraphics 6. In Semigraphics 4, the 8 × 12 dot character block is divided into four pixels (each pixel is four half-clocks by six scan lines). The four low-order bits (DD0-DD3) of each incoming byte of data select one of sixteen possible illumination patterns while the next three bits (DD4-DD6) determine the color of the illuminated elements. The most significant bit is unused. Figure 21 shows the color and pattern selections. In Semigraphics 6 the 8 × 12 dot character block is divided into six pixels, each four half-clocks by four scan lines. The six low-order bits of each byte of incoming data select one of 64 possible illumination patterns while the CSS input and the high-order data bits (DD6-DD7) determine the color of the illuminated elements.

The display window in major mode 2 (full graphics) has a less rigorous format than in major mode 1. The display elements vary from one scan line to three scan lines in height. The length of the display element is either eight or sixteen half-periods wide. Each display element is divided into four or eight pixels. The former corresponds to a full color mode while the latter a restricted color mode, like the semigraphics modes, represents illumination data. When it is high the pixel is illuminated with the color chosen by the color set select (CSS) pin. When it is low the pixel is black. In the full color modes, pairs of data bits choose one of four colors in one of two color sets defined by the CSS pin. Depending on the state of the CSS pin, the area outside the display window is either green or buff. The display formats and color selection for this major mode are summarized in Figure 19.

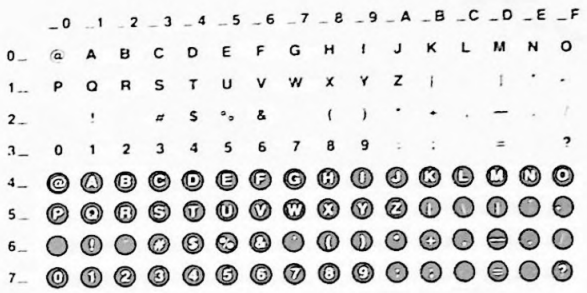
THE 64 × 64 COLOR GRAPHICS ONE (CG1) MODE — The 64 × 64 color graphics mode generates a display matrix of 64 elements wide by 64 elements high. Each element may be one of four colors. A 1k × 8 display memory is required. The display RAM is accessed 16 times per horizontal line. Each pixel equals four half-clocks by three scan lines.

THE 128 × 64 RESOLUTION GRAPHICS ONE (RG1) MODE — The 128 × 64 graphics mode generates a matrix 128 elements wide by 64 elements high. Each element may be either ON or OFF. However, the entire display may be one of two colors, selected by using the color set select pin. A 1k × 8 display memory is required. The display RAM is accessed 16 times per horizontal line. Each pixel equals two half-clocks by three scan lines.

FIGURE 20 — AVAILABLE ALPHANUMERICS

MD4 = INV = D4
MD7 = AS = D7

O = Inverted Character — Illuminated Background, Dark Character



THE 128 × 64 COLOR GRAPHICS TWO (CG2) MODE — The 128 × 64 color graphics mode generates a display matrix 128 elements wide by 64 elements high. Each element may be one of four colors. A 2k × 8 display memory is required. The display RAM is accessed 32 times per horizontal line. Each pixel equals two half-clocks by three scan lines.

THE 128 × 96 RESOLUTION GRAPHICS TWO (RG2) MODE — The 128 × 96 graphics mode generates a display matrix 128 elements wide by 96 elements high. Each element may be either ON or OFF. However, the entire display may be one of two colors selected by using the color set select pin. A 1.5k × 8 display memory is required. The display RAM is accessed 16 times per horizontal line. Each pixel equals two half-clocks by two scan lines.

THE 128 × 96 COLOR GRAPHICS THREE (CG3) MODE — The 128 × 96 color graphics mode generates a display 128 elements wide by 96 elements high. Each element may be one of four colors. A 3k × 8 display memory is required. The display RAM is accessed 32 times per horizontal line. Each pixel equals two half-clocks by two scan lines.

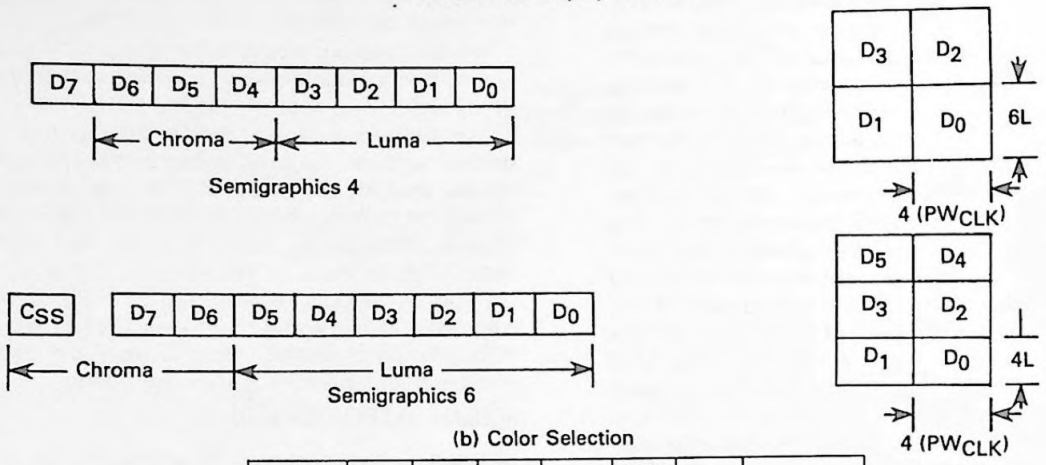
THE 128 × 192 RESOLUTION GRAPHICS THREE (RG3) MODE — The 128 × 192 graphics mode generates a display matrix 128 elements wide by 192 elements high. Each element may be either ON or OFF, but the ON element may be one of two colors selected with the color set select pin. A 3k × 8 display memory is required. The display RAM is accessed 16 times per horizontal line. Each pixel equals two half-clocks by one scan line.

THE 128 × 192 COLOR GRAPHICS SIX (CG6) MODE — The 128 × 192 color graphics mode generates a display 128 elements wide by 192 elements high. Each element may be one of four colors. A 6k × 8 display memory is required. The display RAM is accessed 32 times per horizontal line. Each pixel equals two half-clocks by one scan line.

THE 256 × 192 RESOLUTION GRAPHICS SIX (RG6) MODE — The 256 × 193 graphics mode generates a display 256 elements wide by 192 elements high. Each element may be either ON or OFF, but the ON element may be one of two colors selected with the color set select pin. A 6k × 8 display memory is required. The display RAM is accessed 32 times per horizontal line. Each pixel equals one half-clock by one scan line.

FIGURE 21 — SEMIGRAPHIC MODE ENCODING

(a) Data and Display Formats



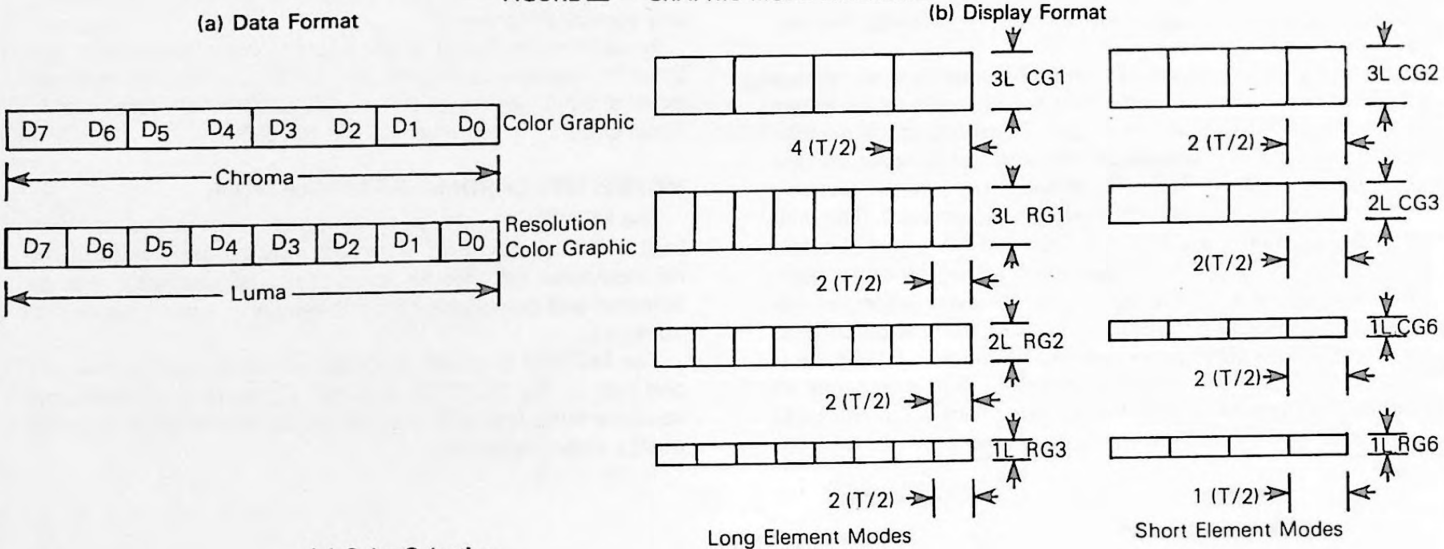
(b) Color Selection

Luma DN	SG4 D6	SG4 D5	SG4 D4	CSS	SG6 D7	SG6 D6	Color
0	X	X	X	X	X	X	Black
1	0	0	0	0	0	0	Green
1	0	0	1	0	0	1	Yellow
1	0	1	0	0	1	0	Blue
1	0	1	1	0	1	1	Red
1	1	0	0	1	0	0	Buff
1	1	0	1	1	0	1	Cyan
1	1	1	0	1	1	0	Magenta
1	1	1	1	1	1	1	Orange

FIGURE 22 — GRAPHIC MODE ENCODING

(a) Data Format

(b) Display Format



(c) Color Selection

CSS	Border	Resolution Color Mode DN	Resolution Color Mode Color	Color Mode DN+1	Color Mode DN	Color Mode Color
0	Green	0	Black	0	0	Green
0	Green	1	Green	0	1	Yellow
0	Green	1	Green	1	0	Blue
0	Green	1	Green	1	1	Red
1	Buff	0	Black	0	0	Buff
1	Buff	1	Buff	0	1	Cyan
1	Buff	1	Buff	1	0	Magenta
1	Buff	1	Buff	1	1	Orange

TYPICAL SYSTEM IMPLEMENTATION

The block diagram in Figure 23 shows how the VDG is related to other functional blocks in a typical system (non-6883). A negative row preset signal ($\overline{RP}$) generated by the VDG initializes the row scan counter for the external character generator once every twelve scan lines, while the negative horizontal sync ($\overline{HS}$) acts as clock to this counter. The negative field sync ($\overline{FS}$) generates an interrupt to the MPU, signifying that the display memory can be updated without interference with the VDG display function. This signal must not be confused with the system vertical sync signal. Field sync is activated by the end of the vertical display window and deactivated by the trailing edge of vertical sync. This gives the MPU a total of thirty-two scan lines or 2.03 ms to update the display memory. The MPU acknowledges the interrupt request from the VDG by bringing the negative memory select input (MS) to the VDG low. This puts the address bus output from the VDG into high-impedance state, thus relinquishing bus control to the MPU. The timing relationship of horizontal sync, row preset, and field sync are shown in Figures 7, 8, and 13.

The display memory is an element-by-element map of the display window on the screen. The VDG addresses the display memory storage locations in succession and translates their contents into luminance and chrominance levels. The frequency of address update is dependent on the length of the display element. Recall that display elements in major mode 1 are four periods and major mode 2 are either four or eight periods of the master clock. Data from the display memory is latched on every address transition. Hence, the data for the first display element must be stable four or eight periods before the horizontal display window depending on the display mode selected. This timing requirement is illustrated in Figure 6.

Examination of Figures 21 and 22 reveal that all display elements within major mode 1 are similar while those within major mode 2 are largely dissimilar. Therefore, mode switching between alphanumeric modes and semigraphic modes can be carried out freely. Care must be taken, however, when performing mode switching in major mode 2. The only compatible modes are between CG1 and RG1, and between CG6 and RG6. Minor mode switching within the same major mode in a given element row can be achieved as long as it is between compatible modes. It should be quite apparent that major mode switching on an element-by-element basis is impractical. It can be achieved, however, at the expense of added component count. The element formats in the VDG lend themselves to major mode switching between element rows. The presence of row preset in major mode 1 serves as a flag for the beginning of a new element row. Detection of this signal can initiate a major mode switch from 1 to 2.

Display memory size is a function of the display density. Quite often a graphic display contains shapes that are several times larger than that of the display elements in the VDG. This is particularly true of certain video games. Much of the display consists of a fixed background. The vertical size of a display element can be doubled or quadrupled by simply ignoring the lowest order or the first two low order vertical addresses, respectively, from the VDG. Reduction of address lines naturally leads to reduction in memory size. Another method of memory reduction is to store objects or object fragments in ROM and store their display addresses in the RAM portion of display memory. Here, the larger the object fragment, the greater the memory saving.

ASSOCIATED DEVICES

MC6883 — SYNCHRONOUS ADDRESS MULTIPLEXER (SAM)

This device, a linear bipolar companion to the MC6800 or MC6809E (external clock inputs), is primarily a VDG transparent-access controller. It allows the microprocessor to load and store to VDG display memory ("screen RAM") without waiting for a blank screen interval. Figure 1 shows a typical system using the SAM and the MC6809E. The inherent interleaved direct memory accesses (IDMA) which occur, continuously keep the VDG updated with the proper data (independently of mode), as well as keeping the dynamic memory (used as system memory with the MC6833) refreshed. This is done through a IDMA process as well, during the time the VDG does not need display data (horizontal and vertical sync times).

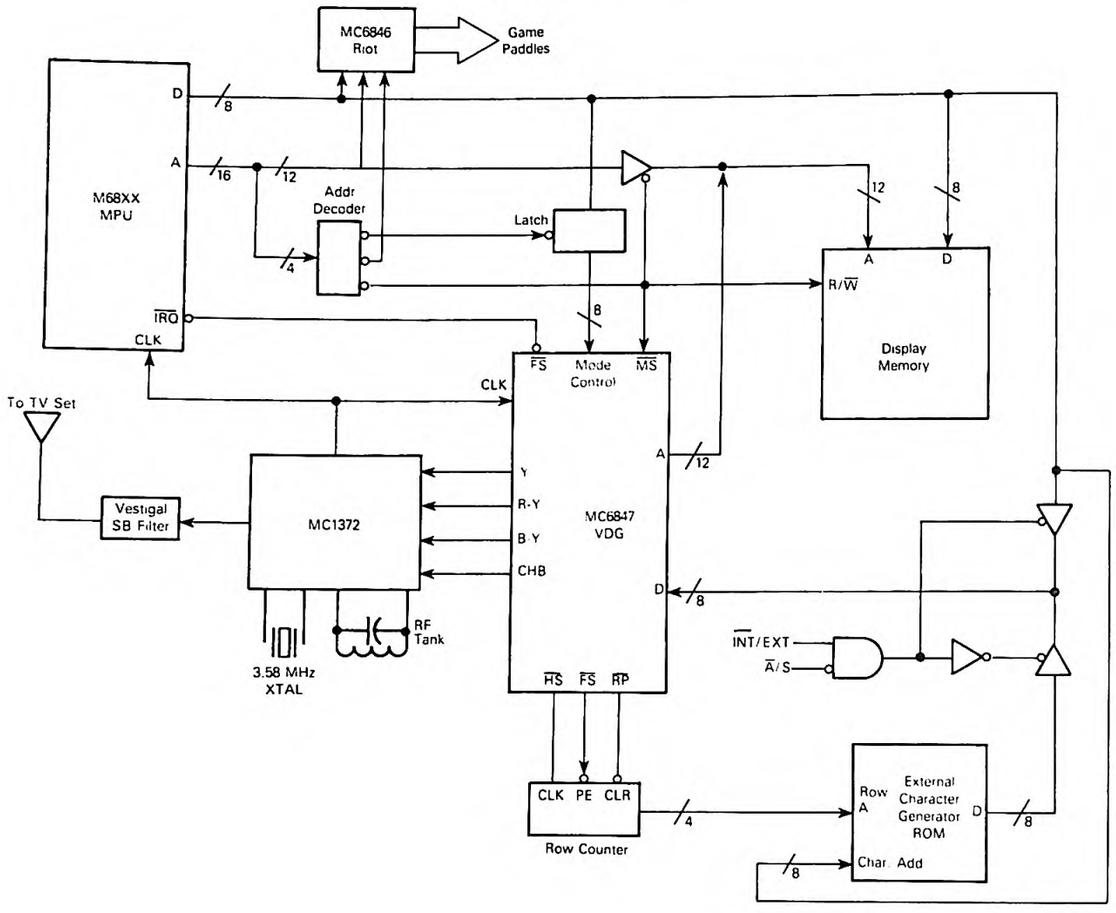
In addition to being a transparent memory access and dynamic memory controller, the SAM also functions as an external clock generator for the MC6800/6809E (slight additional circuitry is required for the MC6800).

MC1372/1373 CHROMA/RF MODULATOR

The MC1372 is a chrominance phase-shift modulator with built in RF up-converter. The part may be used without the RF modulator for chroma only, or the RF oscillator may be defeated and composite chrominance and luminance can be obtained.

The MC1373 is an RF modulator only (similar to the second half of the MC1372) and can be used to up-modulate separate luma and chroma signals at the receiver for high quality video reception.

FIGURE 23 — TYPICAL VDG SYSTEM



APPENDIX A
CUSTOM MC6847 ORDERING INFORMATION

The following information is required when ordering a custom MCU. This information may be transmitted to Motorola in the following media:

PROM(s) MCM2716s or MCM2708s

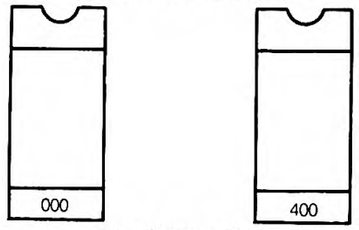
MDOS disk file

To initiate a ROM pattern for the MCU it is necessary to first contact your local field service office, local sales person, or your local Motorola representative.

PROMs — The MCM2708 or MCM2716 type PROMs, programmed with the customer program (positive logic sense for address and data), may be submitted for pattern generation. The PROMs must be clearly marked to indicate which PROM corresponds to which address space (000-3FF HEX), (400-7FF) or (000-7FF). See Figure 24 for recommended marking procedure.

After the PROM(s) are marked they should be placed in conductive IC carriers and securely packed. Do not use styrofoam.

FIGURE 24 — PROM MARKING



xxx = Customer ID

VERIFICATION MEDIA

All original pattern media (PROMs or Floppy Disk) are filed for contractual purposes and are not returned. A computer listing of the ROM code will be generated and returned along with a listing verification form. The listing should be thoroughly checked and the verification form completed, signed, and returned to Motorola. The signed verification form constitutes the contractual agreement for creation of the customer mask. If desired, Motorola will program a blank

2716 EPROM (supplied by the customer) from the data file used to create the custom mask to aid in the verification process.

ROM VERIFICATION UNITS

Ten MC6847s containing the customer's ROM pattern will be sent for program verification. These units will have been made using the custom mask but are for the purpose of ROM verification only. For expediency they are usually unmarked, packaged in ceramic, and tested only at room temperature and 5 volts. These RVUs are included in the mask charge and are not production parts.

FLEXIBLE DISKS

The disk media submitted must be single-sided, single-density, 8-inch, MDOS compatible floppies. The customer must write the binary file name and company name on the disk with a felt-tip pen. The floppies are not to be returned by Motorola as they are used for archival storage. The minimum MDOS system files must be on the disk as well as the absolute binary object file (filename.LO type of file). An object file made from a memory dump using the ROLLOUT command is also admissable. Consider submitting a source listing as well as the following files: filename.LX (EXORciser® loadable format) and filename.SA (ASCII Source Code). These files will of course be kept confidential and are used 1) to speed up the process in house if any problems arise, and 2) to speed up our customer to factory interface if a user finds any software errors and needs assistance quickly from the factory representatives.

MDOS is Motorola's Disk Operating System available on development systems such as EXORcisers, or EXORsets, etc.

FIGURE A-2

Customer Name _____

Address _____

City _____

Phone (_____) _____ State _____ Zip _____

Contact Ms/Mr _____

Customer Part Number _____

Pattern Media
2708 PROM
2716 PROM

MDOS Disk
(Note 2) _____

Other (NOTE: Other media requires prior factory approval)

Signature _____

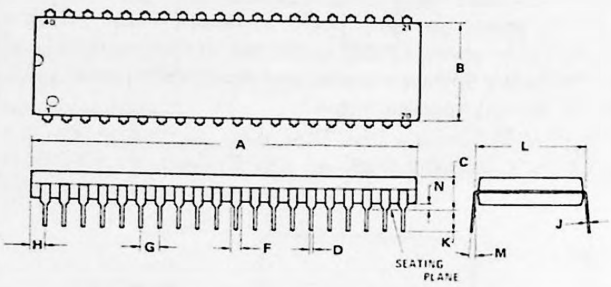
Title _____

® Registered trademark of Motorola Inc.

EXORsets is a trademark of Motorola Inc.

OUTLINE DIMENSIONS

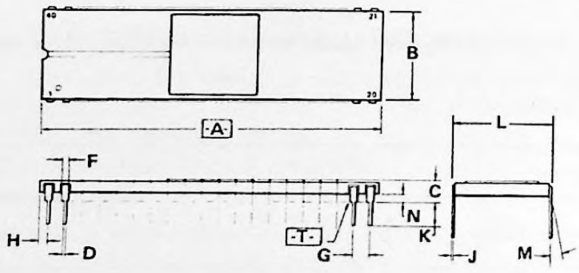
P SUFFIX
PLASTIC PACKAGE
CASE 711-03



DIM	MILLIMETERS		INCHES	
	MIN	MAX	MIN	MAX
A	51.69	52.45	2.035	2.065
B	13.72	14.22	0.540	0.560
C	3.94	5.08	0.155	0.200
D	0.36	0.56	0.014	0.022
F	1.02	1.52	0.040	0.060
G	2.54 BSC		0.100 BSC	
H	1.65	2.16	0.065	0.085
J	0.20	0.38	0.008	0.015
K	2.92	3.43	0.115	0.135
L	15.24 BSC		0.600 BSC	
M	0°	15°	0°	15°
N	0.51	1.02	0.020	0.040

NOTES:
1. POSITIONAL TOLERANCE OF LEADS (D), SHALL BE WITHIN 0.25 mm (0.010) AT MAXIMUM MATERIAL CONDITION, IN RELATION TO SEATING PLANE AND EACH OTHER.
2. DIMENSION L TO CENTER OF LEADS WHEN FORMED PARALLEL.
3. DIMENSION B DOES NOT INCLUDE MOLD FLASH.

L SUFFIX
CERAMIC PACKAGE
CASE 715-04



DIM	MILLIMETERS		INCHES	
	MIN	MAX	MIN	MAX
A	50.29	51.31	1.980	2.020
B	14.63	15.49	0.576	0.610
C	3.05	4.32	0.120	0.170
D	0.38	0.53	0.015	0.021
F	0.76	1.40	0.030	0.055
G	254 BSC		0.100 BSC	
J	0.20	0.33	0.008	0.013
K	2.54	4.19	0.100	0.165
L	14.99	15.65	0.590	0.616
M	–	10°	–	10°
N	1.02	1.52	0.040	0.060

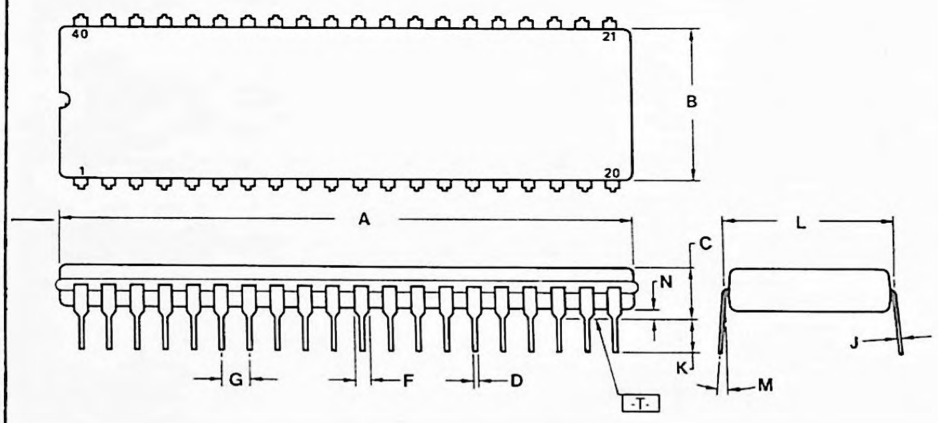
NOTES:
1. DIMENSION [A] IS DATUM.
2. POSITIONAL TOLERANCE FOR LEADS:

   ⌖ 0.25 (0.010) Ⓜ T A Ⓜ

3. [T] IS SEATING PLANE.
4. DIMENSION "L" TO CENTER OF LEADS WHEN FORMED PARALLEL.
5. DIMENSIONING AND TOLERANCING PER ANSI Y14.5, 1973.

OUTLINE DIMENSIONS (CONTINUED)

S SUFFIX
CERDIP PACKAGE
CASE 734-03



DIM	MILLIMETERS MIN	MILLIMETERS MAX	INCHES MIN	INCHES MAX
A	51.31	53.24	2.020	2.096
B	12.70	15.49	0.500	0.610
C	4.06	5.84	0.160	0.230
D	0.38	0.56	0.015	0.022
F	1.27	1.65	0.050	0.065
G	2.54 BSC		0.100 BSC	
J	0.20	0.30	0.008	0.012
K	3.18	4.06	0.125	0.160
L	15.24 BSC		0.600 BSC	
M	5°	15°	5°	15°
N	0.51	1.27	0.020	0.050

NOTES:
1. DIMENSION-A-IS DATUM.
2. POSITIONAL TOLERANCE FOR LEADS:
   ⌖ | Ø 0.25 (0.010) Ⓜ | T | A Ⓜ
3. -T- IS SEATING PLANE.
4. DIMENSION L TO CENTER OF LEADS WHEN FORMED PARALLEL.
5. DIMENSION A AND B INCLUDES MENISCUS.

Motorola reserves the right to make changes to any products herein to improve reliability, function or design. Motorola does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights nor the rights of others.

MOTOROLA Semiconductor Products Inc.
3501 ED BLUESTEIN BLVD., AUSTIN, TEXAS 78721 • A SUBSIDIARY OF MOTOROLA INC.

11724-3 PRINTED IN USA 7-81 IMPERIAL LITHO 95760 10,000

DS9023

MOTOROLA
SEMICONDUCTORS
3501 ED BLUESTEIN BLVD., AUSTIN, TEXAS 78721

MC6821 (1.0 MHz)
MC68A21 (1.5 MHz)
MC68B21 (2.0 MHz)

PERIPHERAL INTERFACE ADAPTER (PIA)

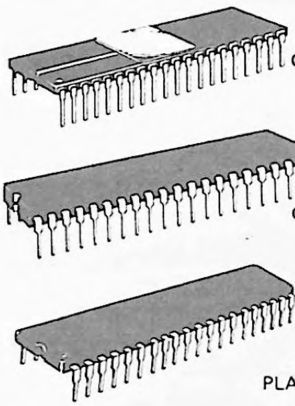
The MC6821 Peripheral Interface Adapter provides the universal means of interfacing peripheral equipment to the M6800 family of microprocessors. This device is capable of interfacing the MPU to peripherals through two 8-bit bidirectional peripheral data buses and four control lines. No external logic is required for interfacing to most peripheral devices.

The functional configuration of the PIA is programmed by the MPU during system initialization. Each of the peripheral data lines can be programmed to act as an input or output, and each of the four control/interrupt lines may be programmed for one of several control modes. This allows a high degree of flexibility in the overall operation of the interface.

- 8-Bit Bidirectional Data Bus for Communication with the MPU
- Two Bidirectional 8-Bit Buses for Interface to Peripherals
- Two Programmable Control Registers
- Two Programmable Data Direction Registers
- Four Individually-Controlled Interrupt Input Lines; Two Usable as Peripheral Control Outputs
- Handshake Control Logic for Input and Output Peripheral Operation
- High-Impedance Three-State and Direct Transistor Drive Peripheral Lines
- Program Controlled Interrupt and Interrupt Disable Capability
- CMOS Drive Capability on Side A Peripheral Lines
- Two TTL Drive Capability on All A and B Side Buffers
- TTL-Compatible
- Static Operation

MOS
(N-CHANNEL, SILICON-GATE, DEPLETION LOAD)

PERIPHERAL INTERFACE ADAPTER



L SUFFIX
CERAMIC PACKAGE
CASE 715

S SUFFIX
CERDIP PACKAGE
CASE 734

P SUFFIX
PLASTIC PACKAGE
CASE 711

MAXIMUM RATINGS

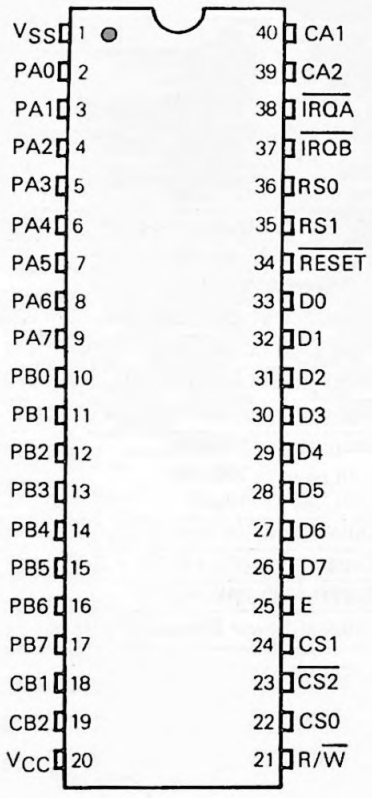
Characteristics	Symbol	Value	Unit
Supply Voltage	V_{CC}	−0.3 to +7.0	V
Input Voltage	V_{in}	−0.3 to +7.0	V
Operating Temperature Range MC6821, MC68A21, MC68B21 MC6821C, MC68A21C, MC68B21C	T_A	T_L to T_H 0 to 70 −40 to +85	°C
Storage Temperature Range	T_{stg}	−55 to +150	°C

THERMAL CHARACTERISTICS

Characteristic	Symbol	Value	Unit
Thermal Resistance Ceramic Plastic Cerdip	θ_{JA}	 50 100 60	°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. Reliability of operation is enhanced if unused inputs are tied to an appropriate logic voltage (i.e., either V_{SS} or V_{CC}).

PIN ASSIGNMENT



©MOTOROLA INC., 1981

POWER CONSIDERATIONS

The average chip-junction temperature, T_J, in °C can be obtained from: (1)

$$T_J = T_A + (P_D \bullet \theta_{JA})$$

Where:

$T_A \equiv$ Ambient Temperature, °C

$\theta_{JA} \equiv$ Package Thermal Resistance, Junction-to-Ambient, °C/W

$P_D \equiv P_{INT} + P_{PORT}$

$P_{INT} \equiv I_{CC} \times V_{CC}$, Watts — Chip Internal Power

$P_{PORT} \equiv$ Port Power Dissipation, Watts — User Determined

For most applications $P_{PORT} \ll P_{INT}$ and can be neglected. P_{PORT} may become significant if the device is configured to drive Darlington bases or sink LED loads.

An approximate relationship between P_D and T_J (if P_{PORT} is neglected) is: (2)

$$P_D = K \div (T_J + 273°C)$$

Solving equations 1 and 2 for K gives: (3)

$$K = P_D \bullet (T_A + 273°C) + \theta_{JA} \bullet P_D^2$$

Where K is a constant pertaining to the particular part. K can be determined from equation 3 by measuring P_D (at equilibrium) for a known T_A. Using this value of K the values of P_D and T_J can be obtained by solving equations (1) and (2) iteratively for any value of T_A.

DC ELECTRICAL CHARACTERISTICS (V_{CC} = 5.0 Vdc ±5%, V_{SS} = 0, $T_A = T_L$ to T_H unless otherwise noted).

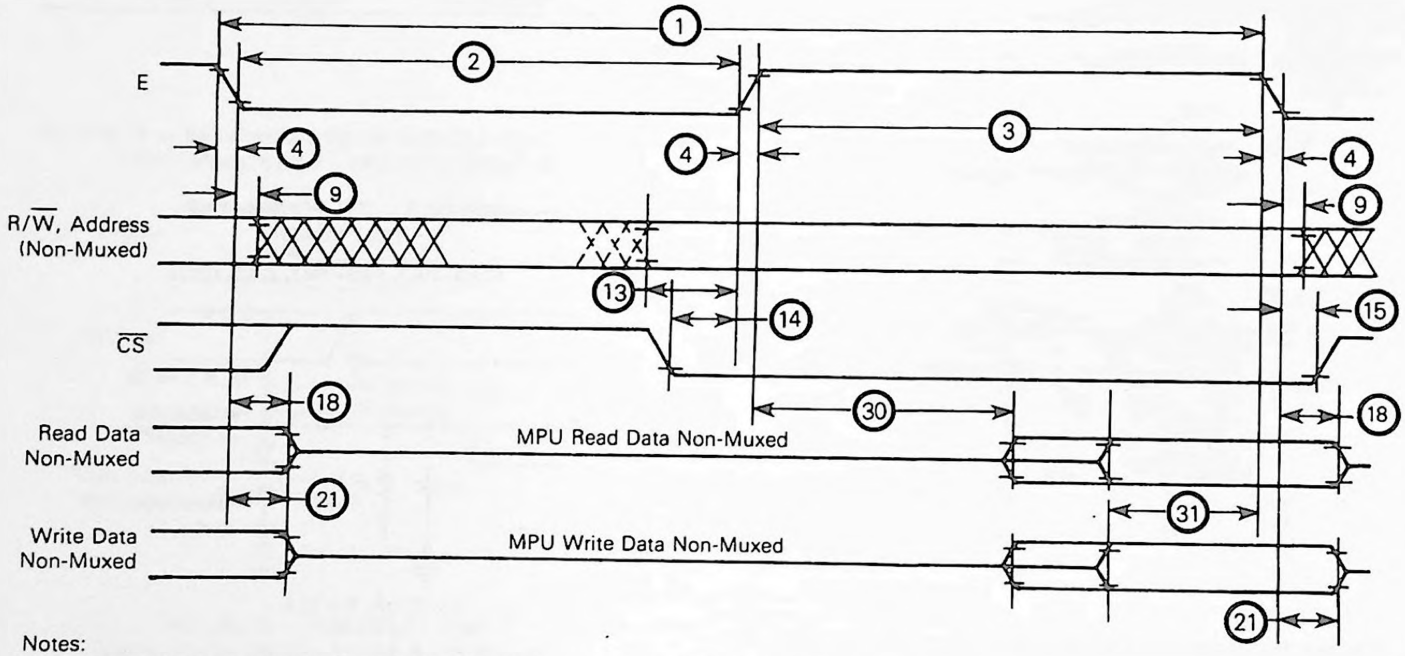
Characteristic		Symbol	Min	Typ	Max	Unit
BUS CONTROL INPUTS (R/$\overline{W}$, Enable, $\overline{RESET}$, RS0, RS1, CS0, CS1, CS2)						
Input High Voltage		V_{IH}	$V_{SS}+2.0$	–	V_{CC}	V
Input Low Voltage		V_{IL}	$V_{SS}-0.3$	–	$V_{SS}+0.8$	V
Input Leakage Current (V_{in} = 0 to 5.25 V)		I_{in}	–	1.0	2.5	μA
Capacitance (V_{in} = 0, T_A = 25°C, f = 1.0 MHz)		C_{in}	–	–	7.5	pF
INTERRUPT OUTPUTS ($\overline{IRQA}$, $\overline{IRQB}$)						
Output Low Voltage (I_{Load} = 3.2 mA)		V_{OL}	–	–	$V_{SS}+0.4$	V
Three-State Output Leakage Current		I_{OZ}	–	1.0	10	μA
Capacitance (V_{in} = 0, T_A = 25°C, f = 1.0 MHz)		C_{out}	–	–	5.0	pF
DATA BUS (D0-D7)						
Input High Voltage		V_{IH}	$V_{SS}+2.0$	–	V_{CC}	V
Input Low Voltage		V_{IL}	$V_{SS}-0.3$	–	$V_{SS}+0.8$	V
Three-State Input Leakage Current (V_{in} = 0.4 to 2.4 V)		I_{IZ}	–	2.0	10	μA
Output High Voltage (I_{Load} = −205 μA)		V_{OH}	$V_{SS}+2.4$	–	–	V
Output Low Voltage (I_{Load} = 1.6 mA)		V_{OL}	–	–	$V_{SS}+0.4$	V
Capacitance (V_{in} = 0, T_A = 25°C, f = 1.0 MHz)		C_{in}	–	–	12.5	pF
PERIPHERAL BUS (PA0-PA7, PB0-PB7, CA1, CA2, CB1, CB2)						
Input Leakage Current (V_{in} = 0 to 5.25 V)	R/$\overline{W}$, $\overline{RESET}$, RS0, RS1, CS0, CS1, $\overline{CS2}$, CA1, CB1, Enable	I_{in}	–	1.0	2.5	μA
Three-State Input Leakage Current (V_{in} = 0.4 to 2.4 V)	PB0-PB7, CB2	I_{IZ}	–	2.0	10	μA
Input High Current (V_{IH} = 2.4 V)	PA0-PA7, CA2	I_{IH}	−200	−400	–	μA
Darlington Drive Current (V_O = 1.5 V)	PB0-PB7, CB2	I_{OH}	−1.0	–	−10	mA
Input Low Current (V_{IL} = 0.4 V)	PA0-PA7, CA2	I_{IL}	–	−1.3	−2.4	mA
Output High Voltage (I_{Load} = −200 μA) (I_{Load} = −10 μA)	PA0-PA7, PB0-PB7, CA2, CB2 PA0-PA7, CA2	V_{OH}	$V_{SS}+2.4$ $V_{CC}-1.0$	– –	– –	V
Output Low Voltage (I_{Load} = 3.2 mA)		V_{OL}	–	–	$V_{SS}+0.4$	V
Capacitance (V_{in} = 0, T_A = 25°C, f = 1.0 MHz)		C_{in}	–	–	10	pF
POWER REQUIREMENTS						
Internal Power Dissipation (Measured at $T_A = T_L$)		P_{INT}	–	–	550	mW

BUS TIMING CHARACTERISTICS (See Notes 1 and 2)

Ident. Number	Characteristic	Symbol	MC6821		MC68A21		MC68B21		Unit
			Min	Max	Min	Max	Min	Max	
1	Cycle Time	t_{cyc}	1.0	10	0.67	10	0.5	10	μs
2	Pulse Width, E Low	PW_{EL}	430	–	280	–	210	–	ns
3	Pulse Width, E High	PW_{EH}	450	–	280	–	220	–	ns
4	Clock Rise and Fall Time	t_r, t_f	–	25	–	25	–	20	ns
9	Address Hold Time	t_{AH}	10	–	10	–	10	–	ns
13	Address Setup Time Before E	t_{AS}	80	–	60	–	40	–	ns
14	Chip Select Setup Time Before E	t_{CS}	80	–	60	–	40	–	ns
15	Chip Select Hold Time	t_{CH}	10	–	10	–	10	–	ns
18	Read Data Hold Time	t_{DHR}	20	50*	20	50*	20	50*	ns
21	Write Data Hold Time	t_{DHW}	10	–	10	–	10	–	ms
30	Output Data Delay Time	t_{DDR}	–	290	–	180	–	150	ns
31	Input Data Setup Time	t_{DSW}	165	–	80	–	60	–	ns

*The data bus output buffers are no longer sourcing or sinking current by t_{DHR}max (High Impedance).

FIGURE 1 — BUS TIMING



Notes:
1. Voltage levels shown are $V_L \leq 0.4$ V, $V_H \geq 2.4$ V, unless otherwise specified.
2. Measurement points shown are 0.8 V and 2.0 V, unless otherwise specified.

PERIPHERAL TIMING CHARACTERISTICS (V_{CC} = 5.0 V ±5%, V_{SS} = 0 V, $T_A = T_L$ to T_H unless otherwise specified)

Characteristic	Symbol	MC6821		MC68A21		MC68B21		Unit	Reference Fig. No.
		Min	Max	Min	Max	Min	Max		
Data Setup Time	t_{PDS}	200	–	135	–	100	–	ns	6
Data Hold Time	t_{PDH}	0	–	0	–	0	–	ns	6
Delay Time, Enable Negative Transition to CA2 Negative Transition	t_{CA2}	–	1.0	–	0.670	–	0.500	µs	3, 7, 8
Delay Time, Enable Negative Transition to CA2 Positive Transition	T_{RS1}	–	1.0	–	0.670	–	0.500	µs	3, 7
Rise and Fall Times for CA1 and CA2 Input Signals	t_r, t_f	–	1.0	–	1.0	–	1.0	µs	8
Delay Time from CA1 Active Transition to CA2 Positive Transition	t_{RS2}	–	2.0	–	1.35	–	1.0	µs	3, 8
Delay Time, Enable Negative Transition to Data Valid	t_{PDW}	–	1.0	–	0.670	–	0.5	µs	3, 9, 10
Delay Time, Enable Negative Transition to CMOS Data Valid PA0-PA7, CA2	t_{CMOS}	–	2.0	–	1.35	–	1.0	µs	4, 9
Delay Time, Enable Positive Transition to CB2 Negative Transition	t_{CB2}	–	1.0	–	0.670	–	0.5	µs	3, 11, 12
Delay Time, Data Valid to CB2 Negative Transition	t_{DC}	20	–	20	–	20	–	ns	3, 10
Delay Time, Enable Positive Transition to CB2 Positive Transition	t_{RS1}	–	1.0	–	0.670	–	0.5	µs	3, 11
Control Output Pulse Width, CA2/CB2	PW_{CT}	500	–	375	–	250	–	ns	3, 11
Rise and Fall Time for CB1 and CB2 Input Signals	t_r, t_f	–	1.0	–	1.0	–	1.0	µ	12
Delay Time, CB1 Active Transition to CB2 Positive Transition	t_{RS2}	–	2.0	–	1.35	–	1.0	µs	3, 12
Interrupt Release Time, $\overline{IRQA}$ and $\overline{IRQB}$	t_{IR}	–	1.60	–	1.10	–	0.85	µs	5, 14
Interrupt Response Time	t_{RS3}	–	1.0	–	1.0	–	1.0	µs	5, 13
Interrupt Input Pulse Time	PW_I	500	–	500	–	500	–	ns	13
$\overline{RESET}$ Low Time*	t_{RL}	1.0	–	0.66	–	0.5	–	µs	15

*The $\overline{RESET}$ line must be high a minimum of 1.0 µs before addressing the PIA.

FIGURE 2 — BUS TIMING TEST LOADS

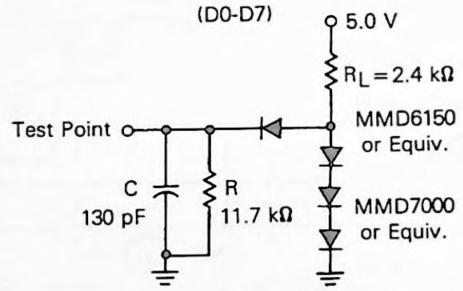
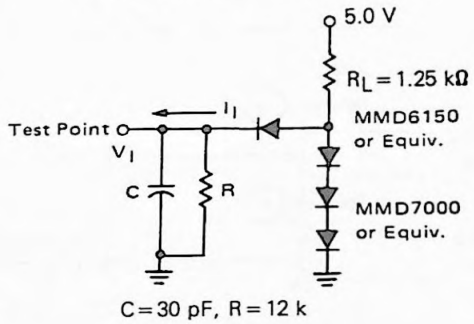


FIGURE 3 — TTL EQUIVALENT TEST LOAD

(PA0–PA7, PB0–PB7, CA2, CB2)



C = 30 pF, R = 12 k

FIGURE 4 — CMOS EQUIVALENT TEST LOAD

(PA0-PA7, CA2)

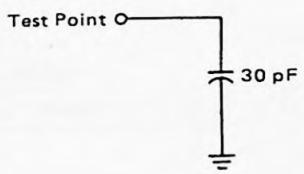


FIGURE 5 — NMOS EQUIVALENT TEST LOAD

($\overline{IRQ}$ Only)

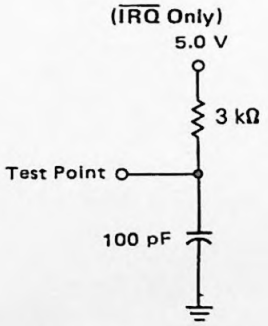


FIGURE 6 — PERIPHERAL DATA SETUP AND HOLD TIMES
(Read Mode)

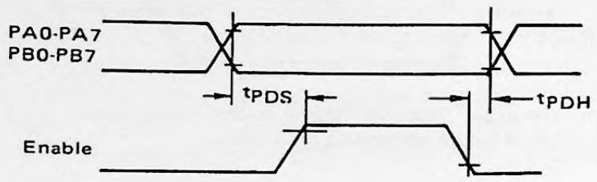
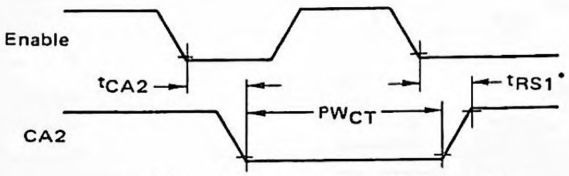


FIGURE 7 — CA2 DELAY TIME
(Read Mode; CRA-5 = CRA3 = 1, CRA-4 = 0)



*Assumes part was deselected during the previous E pulse.

FIGURE 8 — CA2 DELAY TIME
(Read Mode; CRA-5 = 1, CRA-3 = CRA-4 = 0)

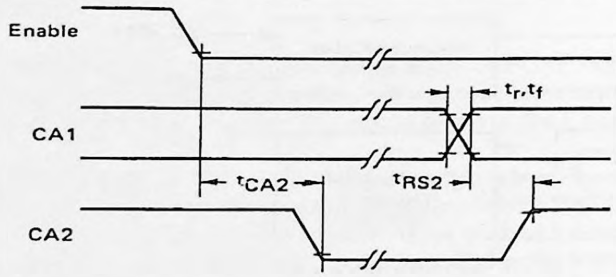


FIGURE 9 — PERIPHERAL CMOS DATA DELAY TIMES
(Write Mode; CRA-5 = CRA-3 = 1, CRA-4 = 0)

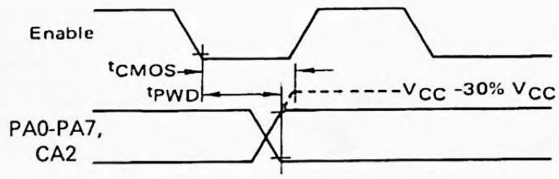
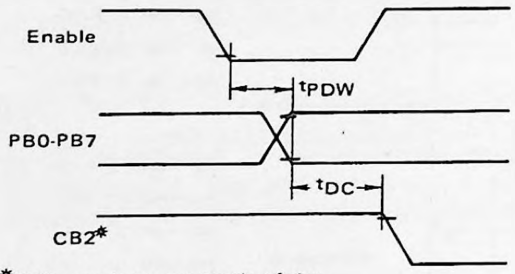
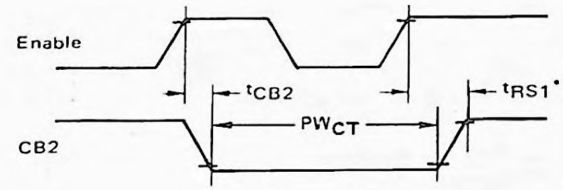


FIGURE 10 — PERIPHERAL DATA AND CB2 DELAY TIMES
(Write Mode; CRB-5 = CRB-3 = 1, CRB-4 = 0)



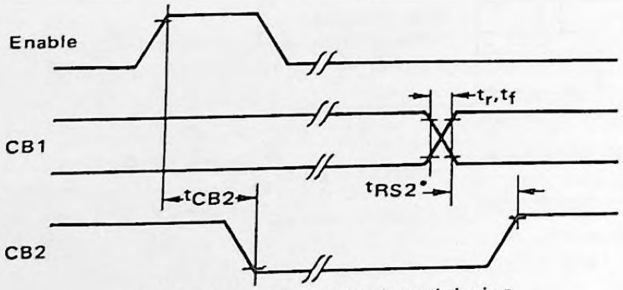
*CB2 goes low as a result of the positive transition of Enable.

FIGURE 11 — CB2 DELAY TIME
(Write Mode; CRB-5 = CRB-3 = 1, CRB-4 = 0)



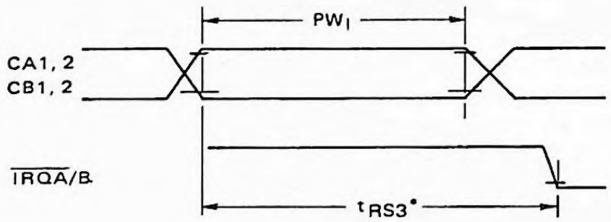
*Assumes part was deselected during the previous E pulse.

FIGURE 12 — CB2 DELAY TIME
(Write Mode; CRB-5 = 1, CRB-3 = CRB-4 = 0)



*Assumes part was deselected during any previous E pulse.

FIGURE 13 — INTERRUPT PULSE WIDTH AND $\overline{IRQ}$ RESPONSE



*Assumes Interrupt Enable Bits are set.

Note: Timing measurements are referenced to and from a low voltage of 0.8 volts and a high voltage of 2.0 volts, unless otherwise noted.

FIGURE 14 — $\overline{IRQ}$ RELEASE TIME

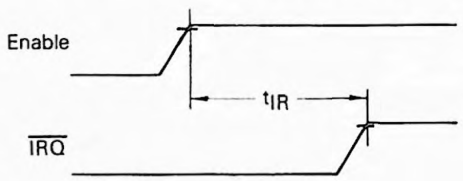
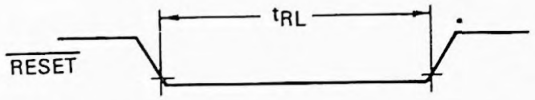


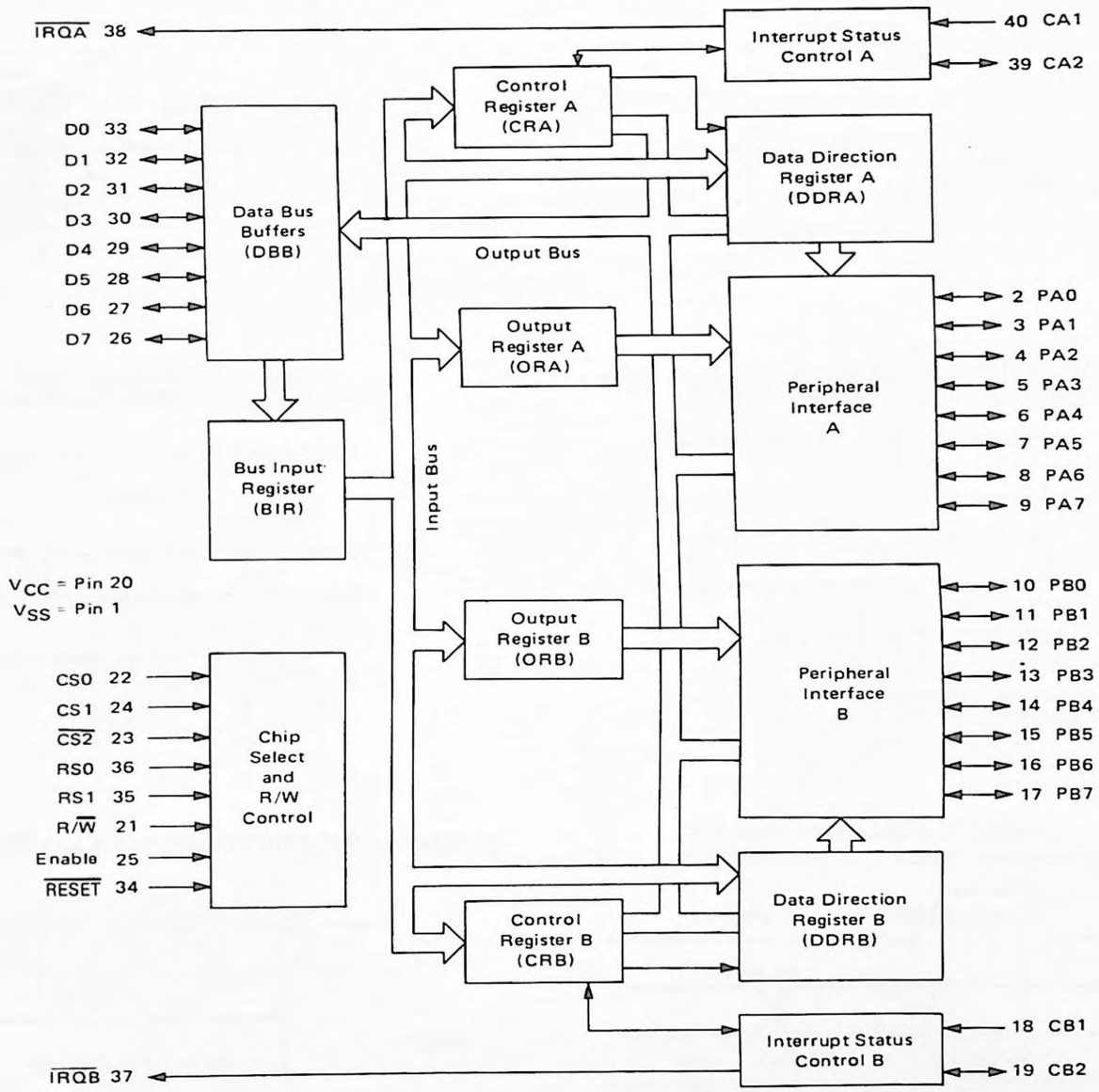
FIGURE 15 — $\overline{RESET}$ LOW TIME



*The $\overline{RESET}$ line must be a V_{IH} for a minimum of 1.0 μs before addressing the PIA.

Note: Timing measurements are referenced to and from a low voltage of 0.8 volts and a high voltage of 2.0 volts, unless otherwise noted.

FIGURE 16 — EXPANDED BLOCK DIAGRAM



PIA INTERFACE SIGNALS FOR MPU

The PIA interfaces to the M6800 bus with an 8-bit bidirectional data bus, three chip select lines, two register select lines, two interrupt request lines, a read/write line, an enable line and a reset line. To ensure proper operation with the MC6800, MC6802, or MC6808 microprocessors, VMA should be used as an active part of the address decoding.

Bidirectional Data (D0-D7) — The bidirectional data lines (D0-D7) allow the transfer of data between the MPU and the PIA. The data bus output drivers are three-state devices that remain in the high-impedance (off) state except when the MPU performs a PIA read operation. The read/write line is in the read (high) state when the PIA is selected for a read operation.

Enable (E) — The enable pulse, E, is the only timing signal that is supplied to the PIA. Timing of all other signals is referenced to the leading and trailing edges of the E pulse.

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

$\overline{\text{RESET}}$ — The active low $\overline{\text{RESET}}$ line is used to reset all register bits in the PIA to a logical zero (low). This line can be used as a power-on reset and as a master reset during system operation.

Chip Selects (CS0, CS1, and $\overline{CS2}$) — These three input signals are used to select the PIA. CS0 and CS1 must be high and $\overline{CS2}$ must be low for selection of the device. Data transfers are then performed under the control of the enable and read/write signals. The chip select lines must be stable for the duration of the E pulse. The device is deselected when any of the chip selects are in the inactive state.

Register Selects (RS0 and RS1) — The two register select lines are used to select the various registers inside the PIA. These two lines are used in conjunction with internal Control Registers to select a particular register that is to be written or read.

The register and chip select lines should be stable for the duration of the E pulse while in the read or write cycle.

Interrupt Request ($\overline{\text{IRQA}}$ and $\overline{\text{IRQB}}$) — The active low Interrupt Request lines ($\overline{\text{IRQA}}$ and $\overline{\text{IRQB}}$) act to interrupt the MPU either directly or through interrupt priority circuitry. These lines are "open drain" (no load device on the chip). This permits all interrupt request lines to be tied together in a wire-OR configuration.

Each Interrupt Request line has two internal interrupt flag bits that can cause the Interrupt Request line to go low. Each flag bit is associated with a particular peripheral interrupt line. Also, four interrupt enable bits are provided in the PIA which may be used to inhibit a particular interrupt from a peripheral device.

Servicing an interrupt by the MPU may be accomplished by a software routine that, on a prioritized basis, sequentially reads and tests the two control registers in each PIA for interrupt flag bits that are set.

The interrupt flags are cleared (zeroed) as a result of an MPU Read Peripheral Data Operation of the corresponding data register. After being cleared, the interrupt flag bit cannot be enabled to be set until the PIA is deselected during an E pulse. The E pulse is used to condition the interrupt control lines (CA1, CA2, CB1, CB2). When these lines are used as interrupt inputs, at least one E pulse must occur from the inactive edge to the active edge of the interrupt input signal to condition the edge sense network. If the interrupt flag has been enabled and the edge sense circuit has been properly conditioned, the interrupt flag will be set on the next active transition of the interrupt input pin.

PIA PERIPHERAL INTERFACE LINES

The PIA provides two 8-bit bidirectional data buses and four interrupt/control lines for interfacing to peripheral devices.

Section A Peripheral Data (PA0-PA7) — Each of the peripheral data lines can be programmed to act as an input or output. This is accomplished by setting a "1" in the corresponding Data Direction Register bit for those lines which are to be outputs. A "0" in a bit of the Data Direction Register causes the corresponding peripheral data line to act as an input. During an MPU Read Peripheral Data Operation, the data on peripheral lines programmed to act as inputs appears directly on the corresponding MPU Data Bus lines. In the input mode, the internal pullup resistor on these lines represents a maximum of 1.5 standard TTL loads.

The data in Output Register A will appear on the data lines that are programmed to be outputs. A logical "1" written into the register will cause a "high" on the corresponding data line while a "0" results in a "low." Data in Output Register A may be read by an MPU "Read Peripheral Data A" operation when the corresponding lines are programmed as outputs. This data will be read property if the voltage on the peripheral data lines is greater than 2.0 volts for a logic "1" output and less than 0.8 volt for a logic "0" output. Loading the output lines such that the voltage on these lines does not reach full voltage causes the data transferred into the MPU on a Read operation to differ from that contained in the respective bit of Output Register A.

Section B Peripheral Data (PB0-PB7) — The peripheral data lines in the B Section of the PIA can be programmed to act as either inputs or outputs in a similar manner to PA0-PA7. They have three-state capabiity, allowing them to enter a high-impedance state when the peripheral data line is used as an input. In addition, data on the peripheral data lines

PB0-PB7 will be read properly from those lines programmed as outputs even if the voltages are below 2.0 volts for a "high" or above 0.8 V for a "low". As outputs, these lines are compatible with standard TTL and may also be used as a source of up to 1 milliampere at 1.5 volts to directly drive the base of a transistor switch.

Interrupt Input (CA1 and CB1) — Peripheral input lines CA1 and CB1 are input only lines that set the interrupt flags of the control registers. The active transition for these signals is also programmed by the two control registers.

Peripheral Control (CA2) — The peripheral control line CA2 can be programmed to act as an interrupt input or as a peripheral control output. As an output, this line is compatible with standard TTL; as an input the internal pullup resistor on this line represents 1.5 standard TTL loads. The function of this signal line is programmed with Control Register A.

Peripheral Control (CB2) — Peripheral Control line CB2 may also be programmed to act as an interrupt input or peripheral control output. As an input, this line has high input impedance and is compatible with standard TTL. As an output it is compatible with standard TTL and may also be used as a source of up to 1 milliampere at 1.5 volts to directly drive the base of a transistor switch. This line is programmed by Control Register B.

INTERNAL CONTROLS

INITIALIZATION

A $\overline{\text{RESET}}$ has the effect of zeroing all PIA registers. This will set PA0-PA7, PB0-PB7, CA2 and CB2 as inputs, and all interrupts disabled. The PIA must be configured during the restart program which follows the reset.

There are six locations within the PIA accessible to the MPU data bus: two Peripheral Registers, two Data Direction Registers, and two Control Registers. Selection of these locations is controlled by the RS0 and RS1 inputs together with bit 2 in the Control Register, as shown in Table 1.

Details of possible configurations of the Data Direction and Control Register are as follows:

TABLE 1 — INTERNAL ADDRESSING

		Control Register Bit		
RS1	RS0	CRA-2	CRB-2	Location Selected
0	0	1	X	Peripheral Register A
0	0	0	X	Data Direction Register A
0	1	X	X	Control Register A
1	0	X	1	Peripheral Register B
1	0	X	0	Data Direction Register B
1	1	X	X	Control Register B

X = Don't Care

PORT A-B HARDWARE CHARACTERISTICS

As shown in Figure 17, the MC6821 has a pair of I/O ports whose characteristics differ greatly. The A side is designed to drive CMOS logic to normal 30% to 70% levels, and incorporates an internal pullup device that remains connected even in the input mode. Because of this, the A side requires more drive current in the input mode than Port B. In contrast, the B side uses a normal three-state NMOS buffer which cannot pullup to CMOS levels without external resistors. The B side can drive extra loads such as Darlingtons without problem. When the PIA comes out of reset, the A port represents inputs with pullup resistors, whereas the B side (input mode also) will float high or low, depending upon the load connected to it.

Notice the differences between a Port A and Port B read operation when in the output mode. When reading Port A, the actual pin is read, whereas the B side read comes from an output latch, ahead of the actual pin.

CONTROL REGISTERS (CRA and CRB)

The two Control Registers (CRA and CRB) allow the MPU to control the operation of the four peripheral control lines CA1, CA2, CB1, and CB2. In addition they allow the MPU to enable the interrupt lines and monitor the status of the interrupt flags. Bits 0 through 5 of the two registers may be written or read by the MPU when the proper chip select and register select signals are applied. Bits 6 and 7 of the two registers are read only and are modified by external interrupts occurring on control lines CA1, CA2, CB1, or CB2. The format of the control words is shown in Figure 18.

DATA DIRECTION ACCESS CONTROL BIT (CRA-2 and CRB-2)

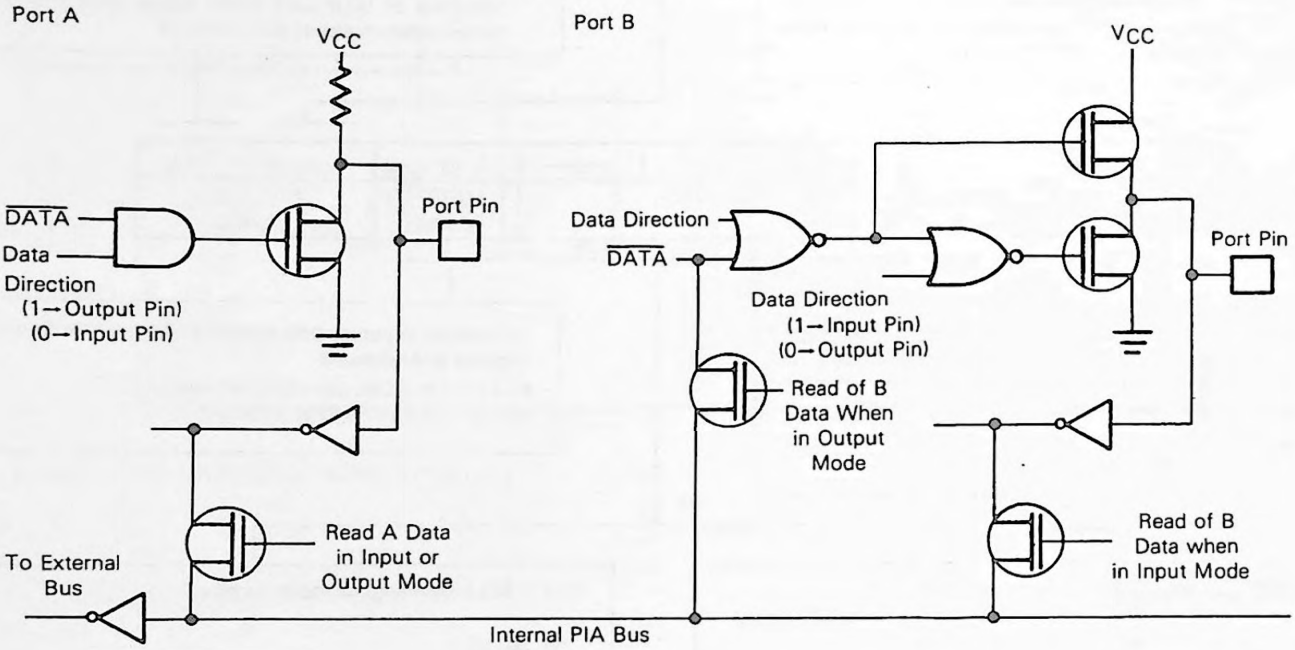
Bit 2, in each Control Register (CRA and CRB), determines selection of either a Peripheral Output Register or the corresponding Data Direction E Register when the proper register select signals are applied to RS0 and RS1. A "1" in bit 2 allows access of the Peripheral Interface Register, while a "0" causes the Data Direction Register to be addressed.

Interrupt Flags (CRA-6, CRA-7, CRB-6, and CRB-7) — The four interrupt flag bits are set by active transitions of signals on the four Interrupt and Peripheral Control lines when those lines are programmed to be inputs. These bits cannot be set directly from the MPU Data Bus and are reset indirectly by a Read Peripheral Data Operation on the appropriate section.

Control of CA2 and CB2 Peripheral Control Lines (CRA-3, CRA-4, CRA-5, CRB-3, CRB-4, and CRB-5) — Bits 3, 4, and 5 of the two control registers are used to control the CA2 and CB2 Peripheral Control lines. These bits determine if the control lines will be an interrupt input or an output control signal. If bit CRA-5 (CRB-5) is low, CA2 (CB2) is an interrupt input line similar to CA1 (CB1). When CRA-5 (CRB-5) is high, CA2 (CB2) becomes an output signal that may be used to control peripheral data transfers. When in the output mode, CA2 and CB2 have slightly different loading characteristics.

Control of CA1 and CB1 Interrupt Input Lines (CRA-0, CRB-1, CRA-1, and CRB-1) — The two lowest-order bits of the control registers are used to control the interrupt input lines CA1 and CB1. Bits CRA-0 and CRB-0 are used to enable the MPU interrupt signals $\overline{IRQA}$ and $\overline{IRQB}$, respectively. Bits CRA-1 and CRB-1 determine the active transition of the interrupt input signals CA1 and CB1.

FIGURE 17 — PORT A AND PORT B EQUIVALENT CIRCUITS



ORDERING INFORMATION

MC68A21CP

- Motorola Integrated Circuit
- M6800 Family
- Blanks = 1.0 MHz
- A = 1.5 MHz
- B = 2.0 MHz
- Device Designation In M6800 Family
- Temperature Range
  - Blank = 0° → +70°C
  - C = −40° → +85°C
- Package
  - P = Plastic
  - S = Cerdip
  - L = Ceramic

Speed	Device	Temperature Range
1.0 MHz	MC6821P,L,S	0 to 70°C
	MC6821CP,CL,CS	−40 to +85°C
1.5 MHz	MC68A21P,L,S	0 to +70°C
	MC68A21CP,CL,CS	−40 to +85°C
2.0 MHz	MC68B21P,L,S	0 to +70°C

BETTER PROGRAM

Better program processing is available on all types listed. Add suffix letters to part number.

Level 1 add "S" Level 2 add "D" Level 3 add "DS"

Level 1 "S" = 10 Temp Cycles — (−25 to 150°C); Hi Temp testing at T_A max.
Level 2 "D" = 168 Hour Burn-in at 125°C
Level 3 "DS" = Combination of Level 1 and 2.

FIGURE 18 — CONTROL WORD FORMAT

Determine Active CA1 (CB1) Transition for Setting Interrupt Flag IRQA(B)1 — (bit 7)

b1 = 0: IRQA(B)1 set by high-to-low transition on CA1 (CB1)

b1 = 1: IRQA(B)1 set by low-to-high transition on CA1 (CB1).

CA1 (CB1) Interrupt Request Enable/Disable

b0 = 0: Disables IRQA(B) MPU Interrupt by CA1 (CB1) active transition.[1]

b0 = 1: Enable IRQA(B) MPU Interrupt by CA1 (CB1) active transition.

1. IRQA(B) will occur on next (MPU generated) positive transition of b0 if CA1 (CB1) active transition occurred while interrupt was disabled.

IRQA(B) 1 Interrupt Flag (bit 7)

Goes high on active transition of CA1 (CB1); Automatically cleared by MPU Read of Output Register A(B). May also be cleared by hardware Reset.

Control Register

b7	b6	b5	b4	b3	b2	b1	b0
IRQA(B)1 Flag	IRQA(B)2 Flag	CA2 (CB2) Control			DDR Access	CA1 (CB1) Control	

IRQA(B)2 Interrupt Flag (bit 6)

When CA2 (CB2) is an input, IRQA(B) goes high on active transition CA2 (CB2); Automatically cleared by MPU Read of Output Register A(B). May also be cleared by hardware Reset.

CA2 (CB2) Established as Output (b5 = 1): IRQA(B) 2 = 0, not affected by CA2 (CB2) transitions.

Determines Whether Data Direction Register Or Output Register is Addressed

b2 = 0: Data Direction Register selected.

b2 = 1: Output Register selected.

CA2 (CB2) Established as Output by b5 = 1

(Note that operation of CA2 and CB2 output functions are not identical)

b5 b4 b3

1 0 → CA2

b3 = 0: **Read Strobe with CA1 Restore**
CA2 goes low on first high-to-low E transition following an MPU read of Output Register A; returned high by next active CA1 transition, as specified by bit 1.

b3 = 1: **Read Strobe with E Restore**
CA2 goes low on first high-to-low E transition following an MPU read of Output Register A; returned high by next high-to-low E transition during a deselect.

→ CB2

b3 = 0: **Write Strobe with CB1 Restore**
CB2 goes low on first low-to-high E transition following an MPU write into Output Register B; returned high by the next active CB1 transition as specified by bit 1. CRB-b7 must first be cleared by a read of data.

b3 = 1: **Write Strobe with E Restore**
CB2 goes low on first low-to-high E transition following an MPU write into Output Register B; returned high by the next low-to-high E transition following an E pulse which occurred while the part was deselected.

b5 b4 b3

1 1 → **Set/Reset CA2 (CB2)**

CA2 (CB2) goes low as MPU writes b3 = 0 into Control Register.

CA2 (CB2) goes high as MPU writes b3 = 1 into Control Register.

CA2 (CB2) Established as Input by b5 = 0

b5 b4 b3

0 → **CA2 (CB2) Interrupt Request Enable/Disable**

b3 = 0: Disables IRQA(B) MPU Interrupt by CA2 (CB2) active transition.*

b3 = 1: Enables IRQA(B) MPU Interrupt by CA2 (CB2) active transition.

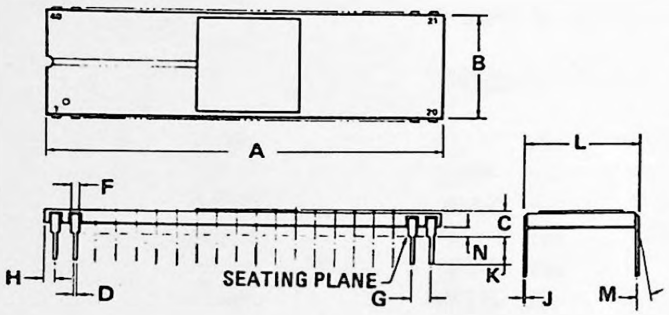
*IRQA(B) will occur on next (MPU generatted) positive transition of b3 if CA2 (CB2) active transition occurred while interrupt was disabled.

→ **Determines Active CA2 (CB2) Transition for Setting Interrupt Flag IRQA(B)2 — (Bit b6)**

b4 = 0: IRQA(B)2 set by high-to-low transition on CA2 (CB2).

b4 = 1: IRQA(B)2 set by low-to-high transition on CA2 (CB2).

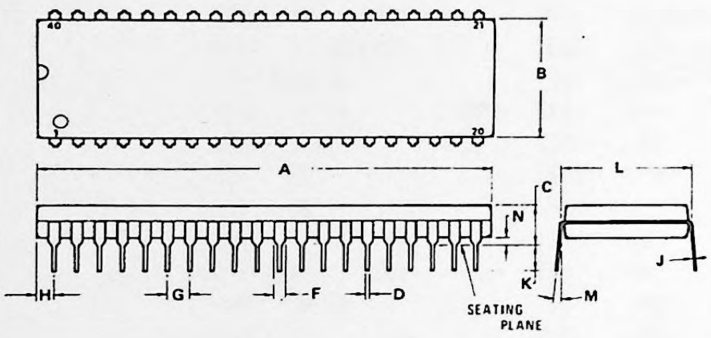
PACKAGE DIMENSIONS



L SUFFIX
CERAMIC PACKAGE
CASE 715-04

DIM	MILLIMETERS MIN	MILLIMETERS MAX	INCHES MIN	INCHES MAX
A	50.29	51.31	1.980	2.020
B	14.94	15.34	0.588	0.604
C	3.05	4.06	0.120	0.160
D	0.38	0.53	0.015	0.021
F	0.76	1.40	0.030	0.055
G	2.54 BSC		0.100 BSC	
H	0.76	1.78	0.030	0.070
J	0.20	0.33	0.008	0.013
K	2.54	4.19	0.100	0.165
L	14.99	15.49	0.590	0.610
M	–	10°	–	10°
N	1.02	1.52	0.040	0.060

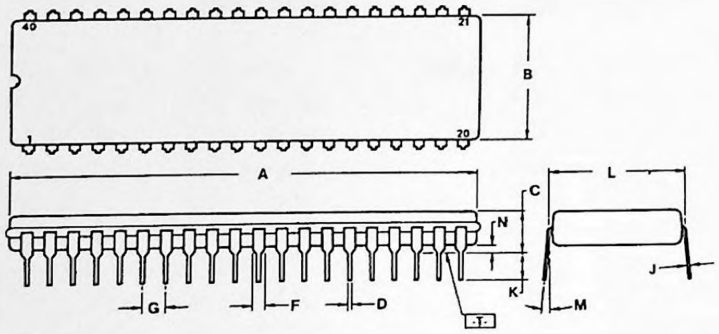
NOTES:
1. LEADS, TRUE POSITIONED WITHIN 0.25 mm (0.010) DIA (AT SEATING PLANE), AT MAX MAT'L CONDITION.
2. DIMENSION "L" TO CENTER OF LEADS WHEN FORMED PARALLEL.



P SUFFIX
PLASTIC PACKAGE
CASE 711-03

DIM	MILLIMETERS MIN	MILLIMETERS MAX	INCHES MIN	INCHES MAX
A	51.69	52.45	2.035	2.065
B	13.72	14.22	0.540	0.560
C	3.94	5.08	0.155	0.200
D	0.36	0.56	0.014	0.022
F	1.02	1.52	0.040	0.060
G	2.54 BSC		0.100 BSC	
H	1.65	2.16	0.065	0.085
J	0.20	0.38	0.008	0.015
K	2.92	3.43	0.115	0.135
L	15.24 BSC		0.600 BSC	
M	0°	15°	0°	15°
N	0.51	1.02	0.020	0.040

NOTES:
1. POSITIONAL TOLERANCE OF LEADS (D), SHALL BE WITHIN 0.25 mm (0.010) AT MAXIMUM MATERIAL CONDITION, IN RELATION TO SEATING PLANE AND EACH OTHER.
2. DIMENSION L TO CENTER OF LEADS WHEN FORMED PARALLEL.
3. DIMENSION B DOES NOT INCLUDE MOLD FLASH.



S SUFFIX
CERDIP PACKAGE
CASE 734-03

DIM	MILLIMETERS MIN	MILLIMETERS MAX	INCHES MIN	INCHES MAX
A	51.31	53.24	2.020	2.096
B	12.70	15.49	0.500	0.610
C	4.06	5.84	0.160	0.230
D	0.38	0.56	0.015	0.022
F	1.27	1.65	0.050	0.065
G	2.54 BSC		0.100 BSC	
J	0.20	0.30	0.008	0.012
K	3.18	4.06	0.125	0.160
L	15.24 BSC		0.600 BSC	
M	5°	15°	5°	15°
N	0.51	1.27	0.020	0.050

NOTES:
1. DIMENSION-A-IS DATUM.
2. POSITIONAL TOLERANCE FOR LEADS:
   ⌖ | Ø 0.25 (0.010) Ⓜ | T | A Ⓜ
3. [T] IS SEATING PLANE.
4. DIMENSION L TO CENTER OF LEADS WHEN FORMED PARALLEL.
5. DIMENSION A AND B INCLUDES MENISCUS.

Motorola reserves the right to make changes to any products herein to improve reliability, function or design. Motorola does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights nor the rights of others.

MOTOROLA Semiconductor Products Inc.

3501 ED BLUESTEIN BLVD., AUSTIN, TEXAS 78721 • A SUBSIDIARY OF MOTOROLA INC.

A10959-7 PRINTED IN USA 11-82 IMPERIAL LITHO 009350 5,000

D89435-R2

A P P E N D I X H

```
*************************************************************

* SOUND DRIVER ROUTINE

*************************************************************

FF20=             DA      EQU     $FF20   ADDRESS OF THE D/A CONVERTER
A976=             AUDON   EQU     $A976   ENABLE SOUND ANALOG MULTIPLEXER
A9A2=             SELMUX  EQU     $A9A2   CONNECT D/A TO OUTPUT OF THE ANALOG MULTIPLEXER
A7D3=             WAIT    EQU     $A7D3   BASIC'S DELAYING ROUTINE
000A=             DELAY   EQU     10
000A=             CYCLES  EQU     10      # OF CYCLES FOR THIS SOUND

3000               ORG $3000

3000  34 70       START   PSHS    U,X,Y
3002  BD   A976           JSR     AUDON   TURN ON AUDIO (ANALOG MULTIPLEXER)
3005  5F                   CLRB            ENABLE 0,0 INPUT TO ANALOG MULTIPLEXER
3006  BD   A9A2           JSR     SELMUX  SET UP D/A TO PASS THROUGH THE ANA MUX
3009  33 8D 001F           LEAU    TABLE,PCR       POINT Y TO AMPLITUDE TABLE
300D 10AE C1               LDY     ,U++    HOW MANY TIMES TO REPEAT THE PATTERN
3010  EF E3       LOOP2   STU     ,--S    SAVE START OF DATA ON THE STACK
3012  A6 C0       LOOP    LDA     ,U+     GET AMPLITUDE (0-63)
3014  48                   LSLA
3015  48                   LSLA            MOVE IT TO THE SIX HIGH ORDER BITS
3016  8A     02            ORA     #2      KEEP RS232 MARKING
3018  B7   FF20           STA     DA      SEND TO D/A CONVERTER
301B  AE C1                LDX     ,U++    GET DELAY - IS IT 0 (END OF TABLE)?
301D  27     05            BEQ     LOOP1   YES
301F  BD   A7D3           JSR     WAIT    DELAY FOR A WHILE IN BASIC
3022  20     EE            BRA     LOOP    KEEP ON GOING
3024  EE E1       LOOP1   LDU     ,S++    POINT U TO START OF CYCLE DATA AGAIN
3026  31 3F                LEAY    -1,Y    DECREMENT CYCLE
3028  26     E6            BNE     LOOP2   KEEP GOING IF <>0
302A  35 F0                PULS    U,X,Y,PC

302C 000A         TABLE   FCB     0,CYCLES        NUMBER OF CYCLES OF THE SOUND
302E 00000A               FCB     0,0,DELAY       0 DATA POINT
3031 01000A               FCB     1,0,DELAY       1 DATA POINT
3034 01000A               FCB     1,0,DELAY       2 DATA POINT
3037 02000A               FCB     2,0,DELAY       3 DATA POINT
303A 04000A               FCB     4,0,DELAY       4 DATA POINT
303D 06000A               FCB     6,0,DELAY       5 DATA POINT
3040 09000A               FCB     9,0,DELAY       6 DATA POINT
3043 0D000A               FCB     13,0,DELAY      7 DATA POINT
3046 11000A               FCB     17,0,DELAY      8 DATA POINT
3049 16000A               FCB     22,0,DELAY      9 DATA POINT
304C 1C000A               FCB     28,0,DELAY      10 DATA POINT
304F 23000A               FCB     35,0,DELAY      11 DATA POINT
3052 2C000A               FCB     44,0,DELAY      12 DATA POINT
3055 3F000A               FCB     63,0,DELAY      13 DATA POINT
3058 3F0000               FCB     63,0,0  END OF TABLE FLAG
```

ERROR MESSAGES AND THEIR NUMBERS AS USED INTERNALLY

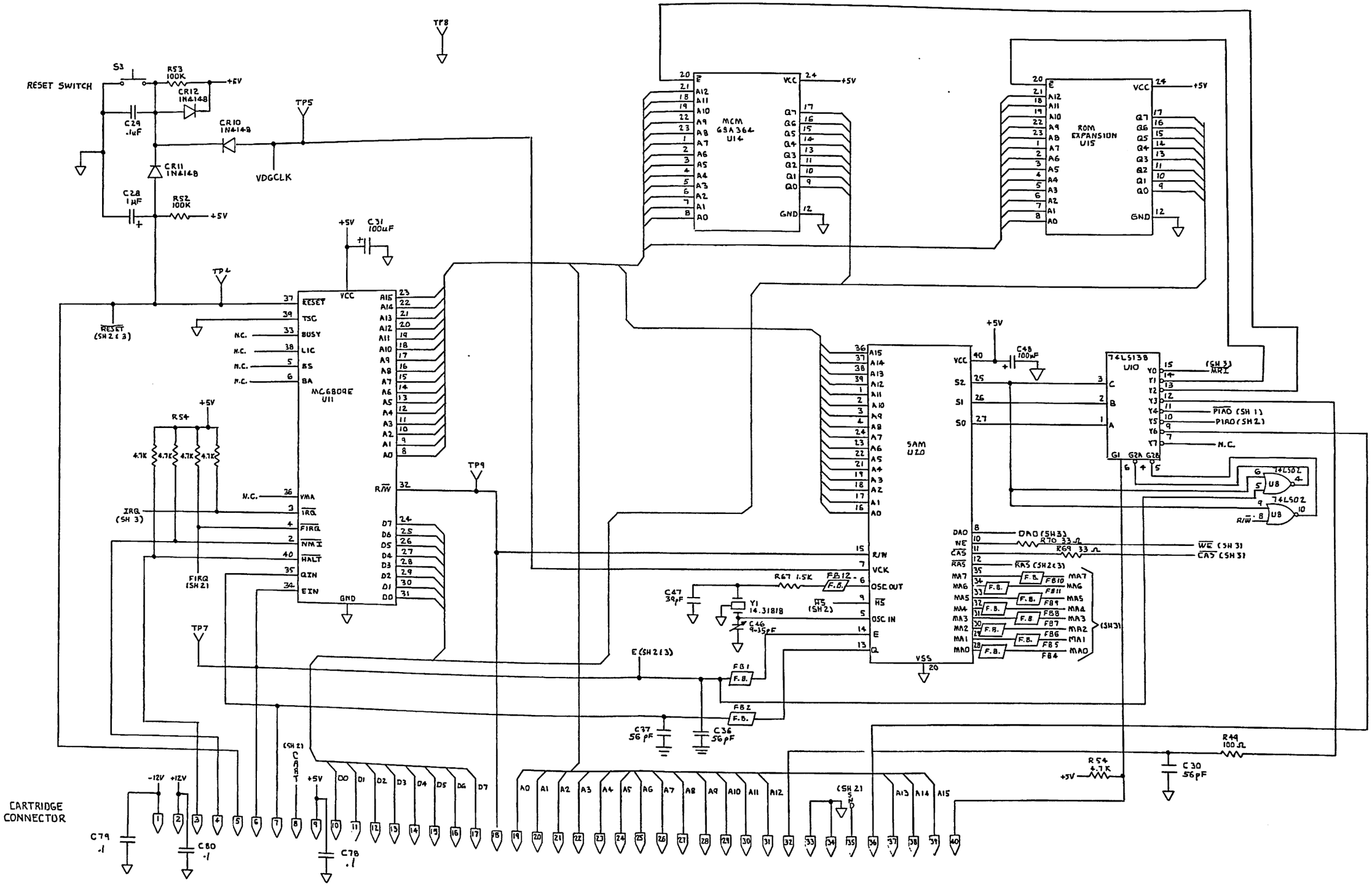
```
ABAF 4E 46      LABAF   FCC     'NF'        NEXT WITHOUT FOR 0
ABB1 53 4E              FCC     'SN'        SYNTAX ERROR 1
ABB3 52 47              FCC     'RG'        RETURN WITHOUT GOSUB 2
ABB5 4F 44              FCC     'OD'        OUT OF DATA 3
ABB7 46 43              FCC     'FC'        ILLEGAL FUNCTION CALL 4
ABB9 4F 56              FCC     'OV'        OVERFLOW 5
ABBB 4F 4D              FCC     'OM'        OUT OF MEMORY 6
ABBD 55 4C              FCC     'UL'        UNDEFINED LINE NUMBER 7
ABBF 42 53              FCC     'BS'        BAD SUBSCRIPT 8
ABC1 44 44              FCC     'DD'        REDIMENSIONED ARRAY 9
ABC3 2F 30              FCC     '/0'        DIVISION BY ZERO 10
ABC5 49 44              FCC     'ID'        ILLEGAL DIRECT STATEMENT 11
ABC7 54 4D              FCC     'TM'        TYPE MISMATCH 12
ABC9 4F 53              FCC     'OS'        OUT OF STRING SPACE 13
ABCB 4C 53              FCC     'LS'        STRING TOO LONG 14
ABCD 53 54              FCC     'ST'        STRING FORMULA TOO COMPLEX 15
ABCF 43 4E              FCC     'CN'        CAN'T CONTINUE 16
ABD1 46 44              FCC     'FD'        BAD FILE DATA 17
ABD3 41 4F              FCC     'AO'        FILE ALREADY OPEN 18
ABD5 44 4E              FCC     'DN'        DEVICE NUMBER ERROR 19
ABD7 49 4F              FCC     'IO'        I/O ERROR 20
ABD9 46 4D              FCC     'FM'        BAD FILE MODE 21
ABDB 4E 4F              FCC     'NO'        FILE NOT OPEN 22
ABDD 49 45              FCC     'IE'        INPUT PAST END OF FILE 23
ABDF 44 53              FCC     'DS'        DIRECT STATEMENT IN FILE 24

                * ADDITIONAL ERROR MESSAGES ADDED BY EXTENDED BASIC

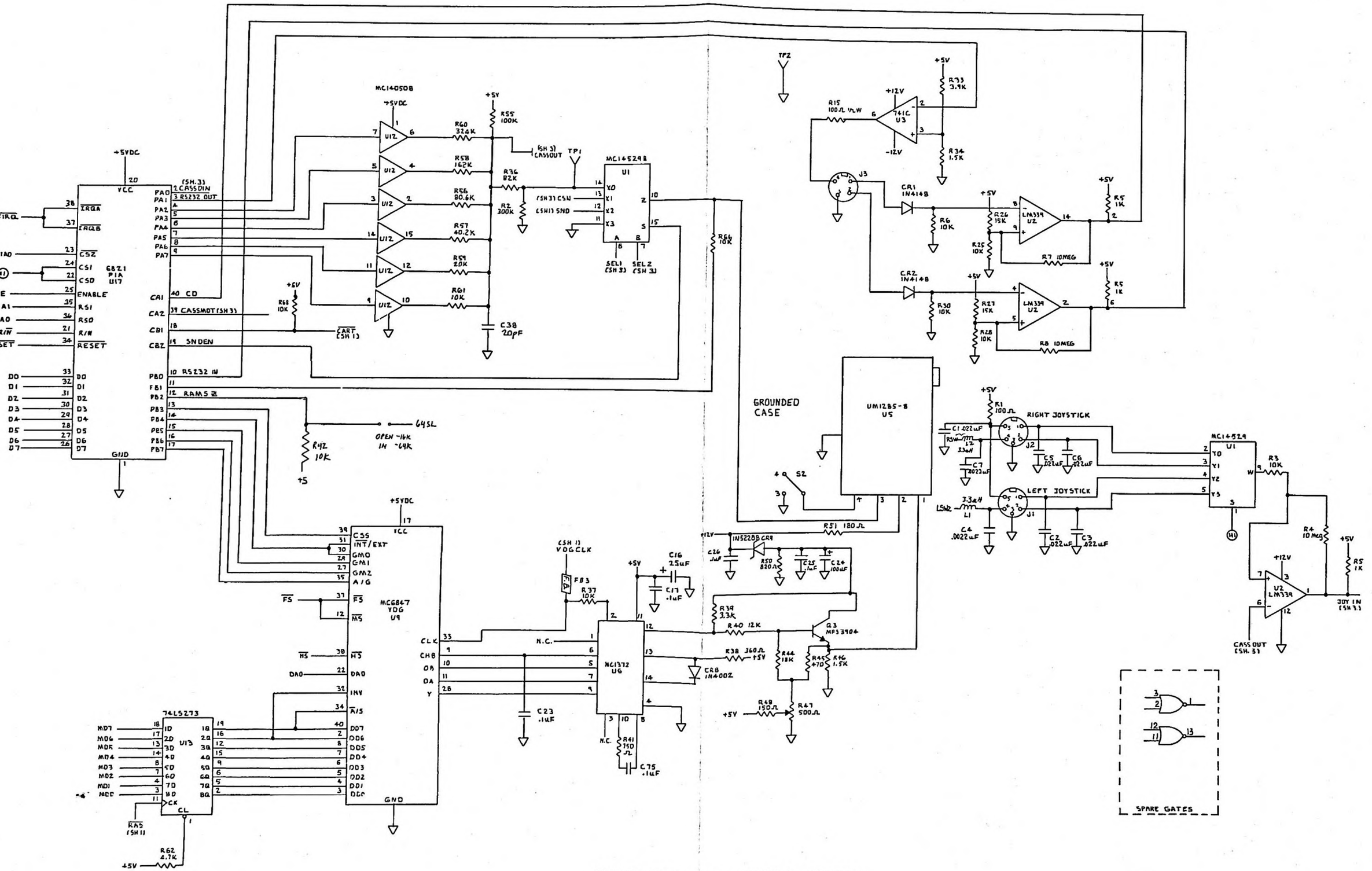
890B 55 46              FCC     'UF'        25 UNDEFINED FUNCTION (FN) CALL
890D 4E 45              FCC     'NE'        26 DLOAD ERROR

                * ADDITIONAL ERROR MESSAGES FOR DISK BASIC

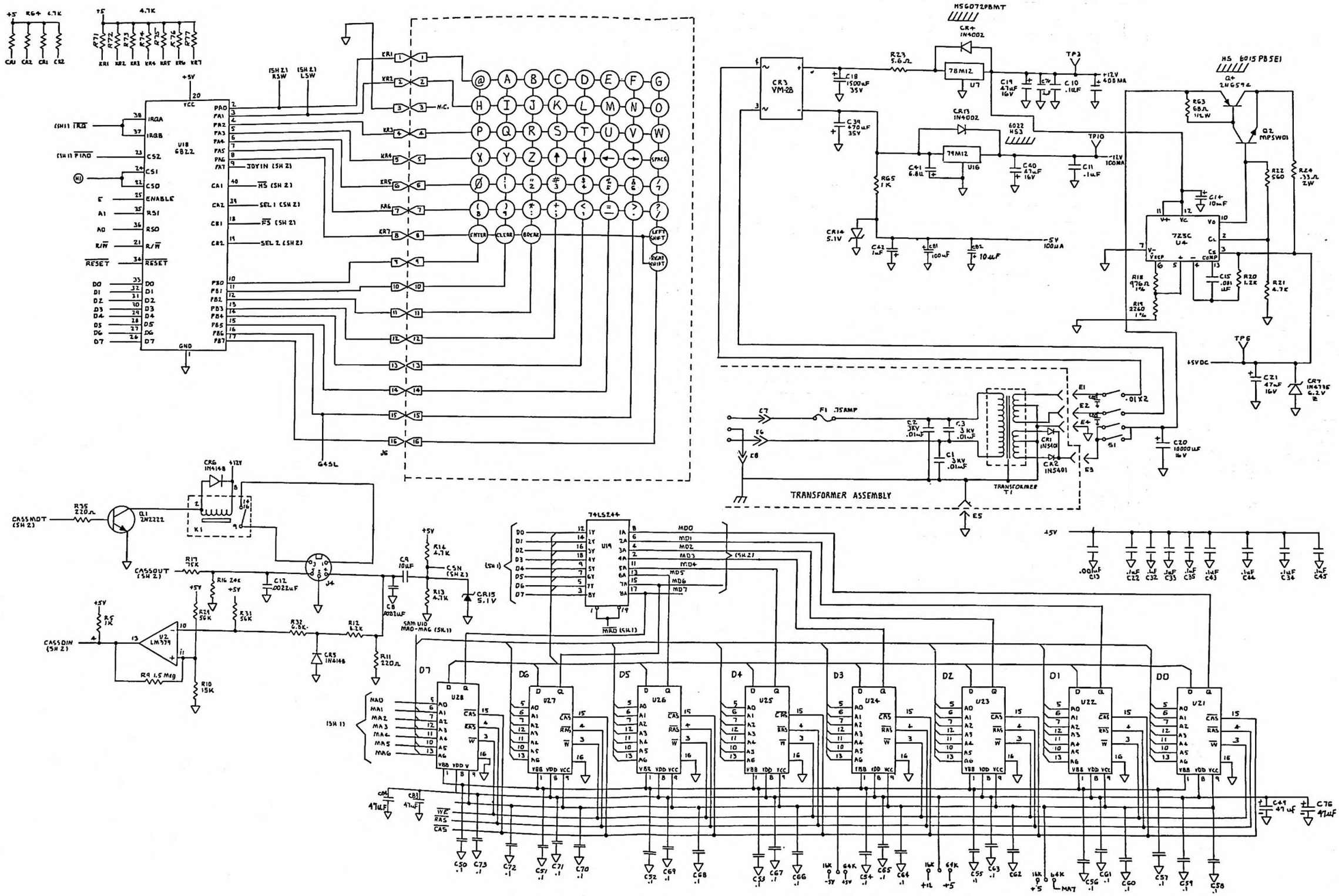
C278 42 52              FCC     'BR'        27 BAD RECORD NUMBER
C27A 44 46              FCC     'DF'        28 DISK FULL
C27C 4F 42              FCC     'OB'        29 OUT OF BUFFER SPACE
C27E 57 50              FCC     'WP'        30 WRITE PROTECTED
C280 46 4E              FCC     'FN'        31 BAD FILE NAME
C282 46 53              FCC     'FS'        32 BAD FILE STRUCTURE
C284 41 45              FCC     'AE'        33 FILE ALREADY EXISTS
C286 46 4F              FCC     'FO'        34 FIELD OVERFLOW
C288 53 45              FCC     'SE'        35 SET TO NON-FIELDED STRING
C28A 56 46              FCC     'VF'        36 VERIFICATION
C28C 45 52              FCC     'ER'        37 WRITE/INPUT PAST END OF RECORD
```



APPENDIX I : COLOR COMPUTER SCHEMATICS



COLOR COMPUTER SCHEMATIC – SHEET 2



COLOR COMPUTER SCHEMATIC – SHEET 3