

21726

6801, 68701, & 6803 MICROCOMPUTER PROGRAMMING & INTERFACING

By ANDREW C. STAUGAARD, JR.



BLACKSBURG CONTINUING EDUCATION SERIES™

edited by Larsen, Titus & Titus

The Blacksburg Continuing Education™ Series

The Blacksburg Continuing Education Series™ of books provide a laboratory—or experiment-oriented approach to electronic topics. Present and forthcoming titles in this series include:

- Basic Business Software
- Circuit Design Programs for the TRS-80
- DBUG: An 8080 Interpretive Debugger
- Design of Active Filters, With Experiments
- Design of Op-Amp Circuits, With Experiments
- Design of Phase-Locked Loop Circuits, With Experiments
- Design of Transistor Circuits, With Experiments
- Design of VMOS Circuits, With Experiments
- 8080/8085 Software Design (2 Volumes)
- 8085A Cookbook
- 555 Timer Applications Sourcebook, With Experiments
- Guide to CMOS Basics, Circuits, & Experiments
- How to Program and Interface the 6800
- Microcomputer—Analog Converter Software and Hardware Interfacing
- Microcomputer Interfacing With the 8255 PPI Chip
- NCR Basic Electronics Course, With Experiments
- NCR Data Communications Concepts
- NCR Data Processing Concepts Course
- NCR EDP Concepts Course
- PET Interfacing
- Programming and Interfacing the 6502, With Experiments
- 6502 Software Design
- 6801, 68701, and 6803 Microcomputer Programming and Interfacing
- 6809 Microcomputer Programming & Interfacing, With Experiments
- TEA: An 8080/8085 Co-Resident Editor/Assembler
- TRS-80 Interfacing (2 Volumes)

In most cases, these books provide both text material and experiments, which permit one to demonstrate and explore the concepts that are covered in the book. These books remain among the very few that provide step-by-step instructions concerning how to learn basic electronic concepts, wire actual circuits, test microcomputer interfaces, and program computers based on popular microprocessor chips. We have found that the books are very useful to the electronic novice who desires to join the "electronics revolution," with minimum time and effort.

Additional information about the "Blacksburg Group" is presented inside the rear cover.

Jonathan A. Titus, Christopher A. Titus, and David G. Larsen
"The Blacksburg Group"

Bug symbol trademark Nanofran, Inc., Blacksburg, VA 24060

720057
COMPUTER MART
OF ROYAL OAK
14.95

6801, 68701, & 6803

Microcomputer

Programming & Interfacing

by
Andrew C. Staugaard, Jr.

Howard W. Sams & Co., Inc.
4300 WEST 62ND ST. INDIANAPOLIS, INDIANA 46268 USA

Copyright © 1980 by Andrew C. Staugaard, Jr.

**FIRST EDITION
SECOND PRINTING—1981**

All rights reserved. No part of this book shall be reproduced, stored in a retrieval system, or transmitted by any means, electronic, mechanical, photocopying, recording, or otherwise, without written permission from the publisher. No patent liability is assumed with respect to the use of the information contained herein. While every precaution has been taken in the preparation of this book, the publisher assumes no responsibility for errors or omissions. Neither is any liability assumed for damages resulting from the use of the information contained herein.

**International Standard Book Number: 0-672-21726-0
Library of Congress Catalog Card Number: 80-51716**

Printed in the United States of America.

Preface

Welcome to the world of very-large-scale integration (VLSI) and, consequently, to the world of single-chip microcomputers. Within most of our lifetimes we have witnessed computer technology progress from those vacuum tube monstrosities of the early fifties, to the large solid-state mainframes that put man on the moon in the sixties, to the complete computer on a chip of today. Most of this progress has taken place within the past ten years. Dr. Gordon Moore, one of the original inventors of the transistor at Bell Telephone Laboratories and co-founder of the Intel Corporation, predicted in 1965 that "circuit complexity," and therefore capability, "will double each year." To date, his prediction has been correct and it is expected to hold at least through the mid-eighties.

Not too long ago, we saw the dawn of large scale integration that made the first generation microprocessors possible. At this time we figured we would never reach the limits of the photolithographic process that provided these first generation devices. How wrong we were! By the mid-seventies we were there, and needed new processing technology if we were to get into the realm of very large scale integration (VLSI) and "pack" a computer, complete with R/W memory, read only memory, parallel i/o, serial i/o, and possibly some timing capability onto that one piece of sand called a "chip." However, just when it seemed we had our backs against the wall, *electron beam lithography* was developed and hurled us into the future generations of microtechnology. To date, the electron beam lithography process has made possible VLSI devices such as the Motorola 6801 and Intel 8748 8-bit single chip microcomputers, as well as the 16-bit microprocessors like the Motorola 68000, Intel 8086, and Zilog Z-8000. So, where do we go from here? Again, the

end is not in sight, at least not within the immediate future. Circuit complexity, and therefore capability *will* double each year through the eighties to provide us with more computer power per given area, and at surprisingly lower prices. Products which never could possess a computer "intelligence" because the cost was prohibitive are now, and will be, marketed in the future with microcomputer control at minimum cost. I predict the 1980's will be a decade of *applications technology*.

Single-chip microcomputers, like the 6801, will be used in every area of industry. Industry is presently on a major campaign to retrofit machine tools and industrial processes with microcomputer technology. The automobile industry is incorporating microcomputer technology into their automobile designs in order to meet the stringent exhaust emission and fuel economy standards of the eighties. The consumer industry is continually applying this technology to provide us with new and more efficient products at lower cost. The end is not even in sight for the possible applications of the technology that exist *today*.

In this book we are going to provide you with a detailed discussion of the 6801 single-chip microcomputer, one of the most exciting products of this second generation microtechnology. As you will soon see, the 6801 and its various versions, the 68701 and 6803, are probably the most flexible single-chip devices available to date. They are 100% software compatible, bus compatible, and upward expandable with all the other 6800 family of microprocessors, memories and peripherals. Since the 6801 is software and bus compatible with the 6800, it is assumed that the reader has a basic understanding of the 6800. If a review is needed, consult *How To Program and Interface the 6800*, Howard W. Sams & Co., Inc., Indianapolis, Indiana 46268. The 6801 can be operated as a single stand-alone microcomputer with internal R/W memory, ROM (EPROM), parallel i/o, serial i/o and internal timing functions or expanded to a 64K external address system. This flexibility coupled with compatibility within the 6800 family is a major reason that both General Motors and the Ford Motor Company will be using Motorola processors very similar to the 6801 in their future automobile production.

This book is meant to be a tutorial type of text for a first exposure to the 6801, 68701, and 6803 or single-chip microcomputers in general. A set of objectives is provided before each chapter, with review questions and answers provided after each chapter. I am confident that you will also find this book extremely valuable as a "cookbook" type aid when working with the 6801, 68701 or 6803.

I would like to express my appreciation to Motorola Semiconductor Products at Phoenix, Arizona and Austin, Texas for their

technical assistance and permission to use their 6801, 68701, and 6803 documentation in this text. Last, but not least by any means, I wish to thank my wife, Janet, whose talent with the typewriter and ability to interpret my handwritten manuscript never ceases to amaze me. Also, I want to thank Sandy Trentini, whose talent with pen and ink are evident in many of the text illustrations.

ANDREW C. STAUGAARD, JR.

To my three boys: Ronnie, David, and Zane, who, along with the rest of their generation, will reap the benefits of this fantastic technology and who, I pray, will apply it for the betterment of mankind.

Contents

CHAPTER 1

FUNDAMENTAL 6801 CONCEPTS AND CHIP STRUCTURE	11
Introduction — Objectives — General 6801 Features — 6801 Expanded Chip Structure — Review Questions — Answers	

CHAPTER 2

6801 SOFTWARE	21
Introduction — Objectives — 6801 Programming Internal Register Format — 6801 Addressing Modes and Instruction Set — Examples — Review Questions — Answers	

CHAPTER 3

6801 FUNDAMENTAL OPERATING MODES AND PIN ASSIGNMENTS	40
Introduction — Objectives — Single-Chip Mode — Expanded Non-multiplexed Mode — Expanded Multiplexed Mode — Review Questions — Answers	

CHAPTER 4

MODE SELECTION AND PORT USAGE	55
Introduction — Objectives — 6801 Mode Selection and Additional Operating Modes — Peripheral Isolation During Mode Selection — 6801 Port Access and Control — Examples — Review Questions — Answers	

CHAPTER 5

USING THE 6801/68701/6803 ON-CHIP MEMORY	76
Introduction — Objectives — 6801 Memory Map — Using the On-	

**Chip R/W Memory — Using the On-Chip ROM — The 68701 —
The 6803 — Review Questions — Answers**

CHAPTER 6

How To Use the 6801 On-Chip Timer	101
Introduction — Objectives — Timer Structure — Timer Examples — Review Questions — Answers	

CHAPTER 7

How To Use the On-Chip Serial Communications Interface (SCI)	124
Introduction — Objectives — Serial Communications — Modes and Data Formats — SCI Internal Structure — SCI Operation — Review Questions — Answers	

CHAPTER 8

General Interfacing	144
Introduction — Objectives — Interfacing With Switches and Key- boards — Examples — Interfacing With Displays — Examples — Interfacing With Digital-to-Analog Converters (DACS) — Some Real Products — Example — Interfacing With Analog-to-Digital Converters — Some Real Products — Review Questions — Answers	

CHAPTER 9

6801 Applications	169
Introduction — Objectives — Serial Communication Via RS-232C — Temperature Monitoring Device — Automobile Applications — Re- mote Data Acquisition — Review Questions — Answers	

APPENDIX A

6801/68701/6803 Instruction Set	189
Nomenclature — Definitions of Executable Instructions	

APPENDIX B

Specifications Sheets	241
MC6801 — MC6801L1/MC6801P1 — MC6803/MC6803NR — MC- 1508L-8 — ICL7109 — NE5018 — MC14066B — MC14584B — MC6801/03 Port Expansion Application Note	

APPENDIX C

LILBUG MONITOR DESCRIPTION	327
MC6801 Overview — LILbug Monitor Overview — Commands —	
Monitor Options — Hardware Requirements	
INDEX	345

CHAPTER 1

Fundamental 6801 Concepts and Chip Structure

INTRODUCTION

To prepare you for more detailed discussions in subsequent chapters, in this chapter you will be given a brief discussion of the capabilities and chip structure of the 6801. As you will soon see, the beauty of the 6801 is its tremendous flexibility. For small dedicated applications, it can be operated as a complete single-chip microcomputer with limited R/W memory, ROM and programmable i/o, or it can be expanded to 64K address words for larger system applications. The 6801 combines onto one chip many functions which previously would have required multichip and board-level systems. Besides a 128-byte on-chip R/W memory and a 2K on-board ROM, the 6801 includes an on-chip clock, timer, and serial communications interface. A similar 6800-based system would require *nine* separate chips to provide the same system capabilities as one 6801. In addition, the 6801 is 100 percent bus-compatible, op-code and source-code compatible, software-compatible, and upward expandable with all the 6800 family of microprocessors, memories, peripherals, special purpose devices and support hardware and software. The 6801 utilizes the 6800 instruction set with all of its powerful addressing modes, and adds 11 new instructions to provide for a more efficient program execution. Now, let us begin with a discussion of the 6801 chip structure.

OBJECTIVES

At the end of this chapter, you will be able to do the following:

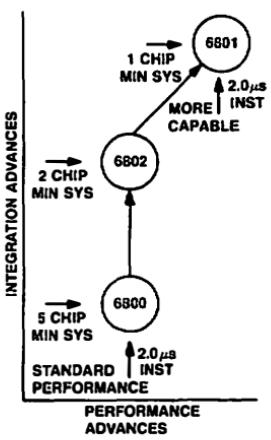
- Describe the internal functional parts of the 6801 microcomputer chip.
- Explain the functional differences between the 6801, 68701, and 6803 microcomputer chips.
- State the three basic modes of operation of the 6801 chip.
- Describe the function of each 6801 i/o port for each mode of operation.
- Describe the differences among the three basic chip modes.
- State the three timing functions of the on-board timer.
- Describe the capabilities of the on-board serial communications interface (SCI).
- Understand how an inexpensive 3.58-MHz tv color-burst crystal can be used to provide the frequency for the internal clock/oscillator driver.

GENERAL 6801 FEATURES

The 6801 is a product of very-large-scale integration (VLSI). It is considered part of the 6800 family since it is 100 percent software-compatible and bus-compatible with all the other 6800 family microprocessors, memories, and peripheral interface chips. As you will see in Chapter 2, the 6801 utilizes the standard 6800 CPU architecture, but it also allows for the software "combination" of the two 8-bit accumulators to form a single 16-bit accumulator. Because of Motorola's commitment to maintain compatibility within its chip family, the 6801 is completely *object code* compatible with the 6800. Object code compatibility means that all of the 6800 instruction op codes can be used with the 6801 to perform the same operations. In addition to the basic 6800 instruction set, there are 11 additional instructions, including an 8-bit by 8-bit multiply instruction that yields a 16-bit result. Fig. 1-1 summarizes the Motorola family evolution from the 6800 to the 6801, and points out some key advances provided with the 6801.

A functional 6801 chip layout is shown in Fig. 1-2. Note that the 6801 contains 128 bytes of on-chip R/W memory, of which the first 64 bytes can be retained in a power-down or standby situation by utilizing the V_{CC} Standby function of the chip. The 6801 also contains an on-chip 2K-byte mask-programmable ROM. Naturally, the ROM-based 6801 requires a relatively large and expensive custom order. However, there is also available the MC68701, an EPROM version of the 6801. In addition, if you desire to use the capabilities of the 6801 without either the ROM or EPROM, you can obtain the 6803, that is the version of the 6801 in which the ROM or EPROM function has been disabled. Detailed discussions of the 68701 and 6803 are provided in Chapter 5.

M6800 Microprocessor/Microcomputer Family Evolution



MC6801 ADVANCES

- MC6800 Software Compatible
 - Instruction Compatible
 - Addressing Compatible
- 2K Bytes ROM
 - Mask ROM
 - EPROM
 - No ROM
- 128 Bytes RAM
- 29 I/O Lines
- 3 Timer Functions
- Serial I/O
- New Instructions
- Faster Instructions
- On-Chip Clock
- MC6800 Bus Compatible
- Expandable With All M6800 Peripherals And Memory

ESM-2

Fig. 1-1. Family evolution of the M6800 microprocessor/microcomputer.

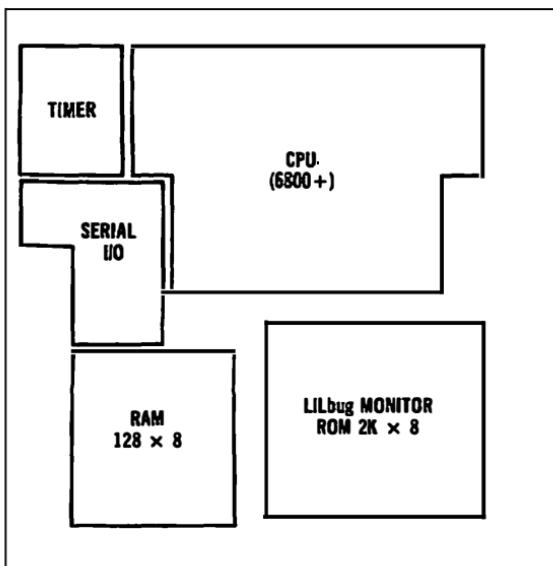


Fig. 1-2. Chip layout of a functional 6801 microcomputer.

Some other significant features of the 6801 include an on-chip clock, on-chip serial communications interface (SCI), an on-chip 16-bit timer and up to four i/o ports. The serial communications interface (SCI) provides full- and/or half-duplex operation in two formats, mark/space operation for standard interfacing, and biphase operation for use between processors. The serial communications interface also provides four different software-selectable bit-transfer rates. The 16-bit timer has three independent timing functions which may be used in applications that require very accurately timed periods. When the 6801 is operated as a single-chip microcomputer, up to four i/o ports are available, three 8-bit ports and one 5-bit port. Each port contains an associated data direction register (DDR), and each one operates in a manner that is similar to the 6821 peripheral interface adapter (PIA), which permits each i/o line to be separately programmed as either an input or an output line. As is the case with all the 6800 family chips, the 6801 operates from a single +5-V dc supply.

6801 EXPANDED CHIP STRUCTURE

You will now be given a brief explanation of each functional region within the 6801. A detailed discussion on the use of these functions will be provided in subsequent chapters.

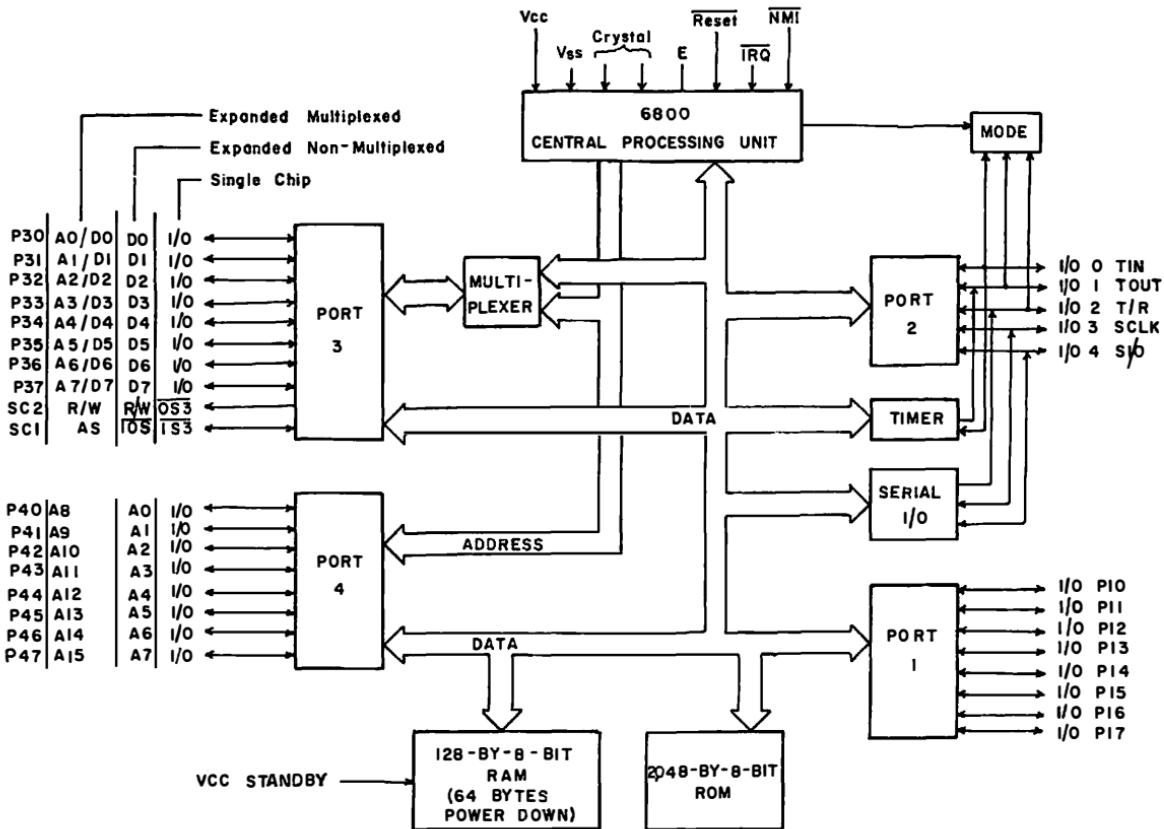
CPU

An expanded block diagram of the 6801 chip structure is shown in Fig. 1-3. At the top of the figure is the central processing unit (CPU). As was stated in the previous section, the 6801 CPU contains basically the same architecture as the 6800, and the basic 6800 instruction set is retained. Some additional instructions are implemented, and they will be discussed in Chapter 2. Note that the 6801 also has the same hardware interrupt capabilities as the 6800, with three hardware interrupts provided; these are interrupt request (IRQ), nonmaskable interrupt (NMI), and a reset interrupt. In addition, the 6801 provides separate interrupts for the internal timer and serial communications functions. These are discussed in subsequent chapters.

R/W Memory

The 6801 includes 128 bytes of on-chip R/W memory located at hex-addresses 0080-00FF. A nice feature of the R/W memory is that the first 64 bytes (0080-00BF) can be retained in power-down or standby situations by using the V_{CC} Standby function of the chip.

Fig. 1-3. Internal chip structure.



ROM

The on-chip ROM in the standard 6801 is a $2K \times 8$ mask-programmed ROM. The ROM is very similar to the $1K \times 8$ MC6830 ROM chip available from Motorola. In fact, Motorola has simply integrated the function of two MC6830 ROMs onto the 6801 chip. As stated earlier, there is also an EPROM version of the 6801 available, the 68701. The 68701 provides all the same functions of the 6801 except that 2K bytes of EPROM are provided rather than mask-programmed ROM. The 2K EPROM can actually be programmed using the internal R/W memory on the chip, making it self-programmable without requiring the use of a special programming device. This procedure will be discussed in a subsequent chapter.

A version of the 6801 that does not contain any ROM, the 6803, is also available. With the 6803, the ROM or EPROM functions have been disabled.

I/O Ports

The extreme flexibility of the 6801 is provided by the various ways in which its i/o ports can be used. There are four ports, port 1, port 2, port 3, and port 4. The function of each port is determined by one of three chip modes. The three chip modes that determine the port function are the *single-chip mode*, *expanded nonmultiplexed mode*, and *expanded multiplexed mode*. In the *single-chip mode*, the 6801 is totally self-reliant and uses only the on-chip resources previously discussed. No external address or data bus is provided. In the *expanded nonmultiplexed mode*, the 8-bit data bus is provided externally along with up to eight of the least-significant bits of address. These eight lines of address will provide a maximum of 256 external address locations. The full 16-bit address bus is provided with the *expanded multiplexed mode* to provide a maximum of 64K address locations, similar to the 6800. Each of these modes will be discussed in more detail in subsequent chapters. The various port functions are detailed below, and summarized in Table 1-1.

Port 1 is an 8-bit port which is always used as a parallel i/o port, regardless of the chip mode. It has an associated data direction register (DDR) and, therefore, each i/o line may be independently programmed for input or output. The use of the DDR is very similar to the use of DDRA or DDRB in the 6820/6821 peripheral interface adapter (PIA)¹. A "1" in the corresponding DDR bit will cause that i/o line to be an output. A "0" in the corresponding DDR bit will cause that i/o line to be an input.

Port 2 is a 5-bit port that can be used as a normal i/o port in all three modes. The port lines are configured as input or output with

an associated data direction register which is used in the same way as the port 1 DDR. This port also provides access to the SCI and 16-bit timer which are discussed in Chapters 6 and 7. Therefore, port 2 can be five lines of parallel i/o, serial i/o, timer, or any combination of these functions. As you will see later, port 2 is also used to configure the chip for any one of its three modes of operation.

Port 3 is an 8-bit port which takes on different functions for the different chip modes. In the single chip mode, port 3 will provide eight lines of parallel i/o similar to that of port 1. It has an associated DDR and two control lines which can be used as input and output strobes similar to the way in which the 6820/6821 PIA control lines (CA1, CA2, CB1, CB2) are utilized.¹

In the expanded nonmultiplexed mode, port 3 becomes the bi-directional data bus to provide data lines D0-D7 for external operations. This will allow the 6801 to be expanded into a multichip system, with up to 256 address words.

In the expanded multiplexed mode, this port becomes both a data bus, to provide data lines D0-D7, and an address bus, to provide the lower eight address lines, A0-A7. These two sets of signals are *multiplexed*, with address strobe (AS) signal being used to indicate which information is on the bus. A high address strobe signal indicates that an address is on the port; a low address strobe signal indicates that data is on the port. As you will see later, when the chip is used in this mode, latches are required to demultiplex the address information from the bus.

Port 4 is an 8-bit port whose lines can be configured for parallel i/o or used as address lines. In the single chip mode, all eight lines will provide parallel i/o as determined by the port 4 DDR.

In the expanded nonmultiplexed mode, port 4 is used to provide the *lower* eight address bus lines (A0-A7). Recall that in this mode, port 3 supplies the data bus lines. Therefore, a system could be expanded to include 256 external memory addresses using the expanded nonmultiplexed mode.

In the expanded multiplexed mode, port 4 is used to provide the *upper* eight address bus lines (A8-A15). Also, recall that in this mode the information at port 3 is multiplexed to provide the lower eight address lines (A0-A7) and the data bus (D0-D7). Therefore, in the expanded multiplexed mode, ports 3 and 4 combine to supply an external 16-bit address and 8-bit data bus, and thus allowing a 6801 system to be expanded to 64K address words and operate similar to a 6800 system.

¹ Staugaard, Andrew C., *How To Program and Interface the 6800* (Howard W. Sams and Co., Inc., Indianapolis, Indiana 46268, 1980).

Table 1-1. Port Function Summary

	Single-Chip Mode	Expanded Non-multiplexed Mode	Expanded Multiplexed Mode
Port 1 (8 Bits)	Parallel I/O	Parallel I/O	Parallel I/O
Port 2 (5 Bits)	Parallel I/O, Timer, SCI	Parallel I/O, Timer, SCI	Parallel I/O, Timer, SCI
Port 3 (8 Bits)	Parallel I/O	External Bi-directional Data Bus	MPX Lo address and Data Bus*
Port 4 (8 Bits)	Parallel I/O	Lo Address Bus	Hi Address Bus

*MPX mode requires use of AS signal for demultiplexing.

Timer

As shown in Fig. 1-3, the 6801 also includes an on-chip timer. The timer is effectively a 16-bit free-running counter. You can program it for one of four timing functions: variable pulse width measurement, continuous pulse width generation, single-shot pulse generation and period measurement. These different functions allow the 6801 to be used in applications that require very accurate time measurement. A more detailed explanation of these functions and use of the timer will be given in Chapter 6.

Serial Communications Interface (SCI)

The 6801 includes an on-chip SCI for serial i/o. As mentioned earlier, it is capable of full- and/or half-duplex operation in standard mark/space format for typical terminal/modem interfaces, or biphase format for use between processors. The SCI also contains a programmable bit rate generator to provide four software selectable rates. The use of the on-chip SCI is detailed in Chapter 7.

Clock

An internal clock/oscillator driver is part of the 6801 package. The clock circuit requires an external crystal to set the oscillator frequency. The 6801 has a maximum operational frequency of 1.25 MHz; however, the clock section contains an internal divide-by-four circuit so that a 5-MHz crystal is used to generate the 1.25-MHz clock signal. Actually, you can even use a readily available and inexpensive 3.58-MHz color-burst crystal to provide the oscillator frequency. With the divide-by-four circuitry, the 3.58-MHz crystal would provide a .895-MHz operating frequency for the 6801.

REVIEW QUESTIONS

- 1. The five major internal functional parts of the 6801 are:**

- 2. Describe the difference(s) between the 6801 and 6803.**

- 3. Describe the difference(s) between the 6801 and 68701.**

- 4. The three basic 6801 chip modes are: _____**
_____, and _____.

- 5. The 6801 is a product of what technology?**

- 6. The three external hardware interrupts provided with the 6801 are**
_____, _____, and _____.

- 7. The 6801 internal R/W memory is assigned to addresses _____**
through _____.

- 8. The 6801 contains how many i/o ports?**

- 9. Describe the function of port 1 for each chip mode.**

- 10. Describe the function of port 2 for each chip mode.**

- 11. Describe the function of port 3 for each chip mode.**

- 12. Describe the function of port 4 for each chip mode.**

13. The on-board timer is a _____-bit free-running counter.
14. The on-board SCI provides what type of i/o?
15. A _____-MHz crystal would be required to provide an 800-kHz operating frequency for the 6801.

ANSWERS

1. 6800 + CPU
128 byte R/W memory
2K ROM
16-bit timer
SCI (serial i/o)
2. The 6803 does not contain an internal ROM.
3. The 68701 contains a 2K EPROM in place of a 2K mask-programmed ROM.
4. single-chip, expanded nonmultiplexed, expanded multiplexed
5. Very-large-scale integration (VLSI).
6. reset, nonmaskable, interrupt request
7. 0080 through 0OFF
8. Four.
9. Port 1 provides eight parallel i/o lines for all modes.
10. Port 2 provides five parallel i/o lines, timer, serial i/o any combination thereof for all modes.
11. Port 3: single-chip mode—eight parallel i/o lines; expanded nonmultiplexed mode—data bus, D0-D7; expanded multiplexed mode—multiplexed to provide data bus, D0-D7, and address lines A0-A7.
12. Port 4: single-chip mode—eight parallel i/o lines, expanded nonmultiplexed mode—address lines A0-A7; expanded multiplexed mode—address lines A8-A15
13. 16
14. Serial
15. 3.2

CHAPTER 2

6801 Software

INTRODUCTION

In Chapter 1, you were briefly introduced to the physical structure and layout of the 6801 microcomputer chip. You were acquainted with the tremendous flexibility and capability of this *complete* computer-on-a-chip. However, all these functions are useless unless you can communicate with the chip and provide some sort of control for its operation. This control is implemented in the central processing unit (CPU) of the chip. In the case of the 6801, the CPU is simply an expanded 6800 CPU. This has been done purposely to provide software *compatibility* within the 6800 chip family. Once you are familiar with the 6800, you do not have to learn a totally new instruction set to use the newer generations of 6800 chips such as the 6801. You will find this same philosophy applied to the other new Motorola chips, such as the 6802 and 6809. In this chapter, you will see that the 6801 is completely op- and source-code compatible with the 6800, meaning that it uses the same instruction set. Therefore, it is assumed that you are familiar with the 6800 programming modes and instruction set. Consult *How To Program and Interface the 6800* (Howard W. Sams and Co., Inc.) for a complete discussion of the 6800, if a review is needed.

In addition, you will see that the only difference (in programming) between the 6801 and 6800 is the addition of a new accumulator in the 6801. This accumulator is referred to as accumulator D (ACCD). It is a 16-bit accumulator which, in fact, simply combines the contents of the two 8-bit accumulators, ACCA and ACCB. Eleven new instructions have been added to the basic 6800 instruction set to facilitate the use of the new accumulator and to provide

greater use of the index register. Most of these new instructions involve 16-bit operations, and thus provide for a more efficient program execution over similar 8-bit operations. Also, the instruction cycle times of many of the key instructions have been reduced to provide faster execution of a program in a 6801 over the same program in a 6800.

You are now ready to learn about programming the 6801, and its instruction set. You will find this easy and straightforward if you understand the 6800. Once this has been accomplished, in subsequent chapters you will learn how to configure the 6801 for a variety of applications.

OBJECTIVES

At the end of this chapter, you will be able to do the following:

- Describe the internal software architecture of the 6801.
- Understand the relationship between the three internal accumulators.
- Describe the six different addressing modes utilized by the 6801.
- State the seven general categories of 6801 instructions.
- Understand each of the following eleven new 6801 instructions: ABX, ADDD, ASLD, BRN, LDD, LSRD, MUL, PSHX, PULX, STD, SUBD.
- Give a program example of the use of the new multiply (MUL) instruction.
- Give an example of how and where to use the new logic shift right (LSRD) instruction.

6801 PROGRAMMING INTERNAL REGISTER FORMAT

The 6801 programming register format is shown in Fig. 2-1. Note that it is identical to the 6800, except for one change. The two 8-bit accumulators (ACCA and ACCB) can be temporarily "combined" through the use of special op codes to form one 16-bit accumulator (ACCD). The other registers are identical to those of the 6800 and perform the same functions. A brief review of each register shown in Fig. 2-1 follows.

Accumulators A, B, and D (ACCA, ACCB, ACCD)

Accumulators A and B are 8-bit storage registers that are used to hold operands before they are used in an operation, and also to hold the results of various operations. As with the 6800, data may be loaded into, transferred to, transferred from, added to, subtracted

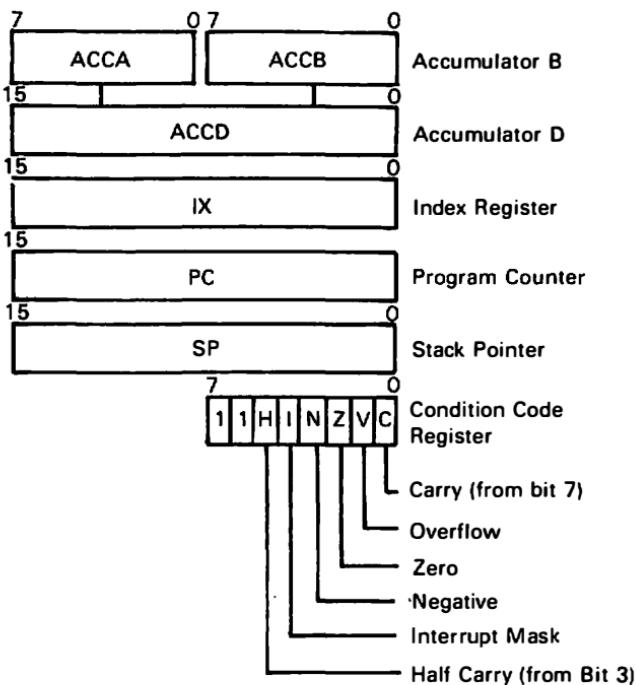


Fig. 2-1. The 6801 internal register programming format.

from, shifted, ANDed with, ORed with, XORED with, and compared to the contents of either accumulator. Accumulator D is a 16-bit register which combines accumulators A and B. It is used in the same way as accumulators A and B, but to perform 16-bit arithmetic operations. As you will soon see, there are seven new instructions associated with this accumulator. These instructions will allow 16-bit data to be transferred to, transferred from, shifted, added to, and subtracted from the content of accumulator D.

Index Register (IX)

The index register is a 16-bit register that can be used to store data or 16-bit addresses in conjunction with the indexed mode of memory addressing. This register may be incremented and decremented, as well as loaded from and copied into memory. Three new 6801 instructions will allow you to add to the index register, as well as push and pull its contents from a memory stack.

Program Counter (PC)

The program counter is a 16-bit counter that provides instruction sequencing.

Stack Pointer (SP)

The stack pointer register is a 16-bit register which always points to the next available location in a memory stack. The stack, which is located in R/W memory, is used to save the contents of the registers during the execution of subroutines and interrupt service routines.

Condition Codes Register (CCR)

As with the 6800, the 6801 CCR is an 8-bit status flag register. Bits six and seven of this register are not used and are permanently set to a logic "1" state. Bits zero through five are the carry (C), overflow (V), zero (Z), negative (N), interrupt (I) and half-carry (H) flags respectively. Changes in these flags occur as a result of arithmetic, logic and data handling operations. The flag status levels are used by the 6801 for conditional branch operations. As with the 6800, the 6801 instruction set includes instructions that allow you to set or clear the C, I, and V flags and to transfer data between the condition codes register and accumulator A.

6801 ADDRESSING MODES AND INSTRUCTION SET

The 6801 uses the same six addressing modes as the 6800. They are inherent (implied), immediate, direct, extended, indexed, and relative. The instruction format for each mode is shown in Fig. 2-2. Inherent instructions are simply one-byte instructions, such as halt,

ACCUMULATOR AND IMPLIED ADDRESSING	EXTENDED ADDRESSING	RELATIVE ADDRESSING
OP CODE	OP CODE ADDRESS (HI) ADDRESS (LO)	OP CODE RELATIVE ADDRESS
DIRECT ADDRESSING	IMMEDIATE ADDRESSING	INDEXED ADDRESSING
OP CODE ADDRESS 0-255	OP CODE OPERAND	OP CODE INDEX ADDRESS
	OP CODE OPERAND (HI) OPERAND (LO)	

Courtesy American Microsystems, Inc.

Fig. 2-2. 6801 addressing modes.

increment, and decrement, for which the meaning is obvious. Immediate instructions are those in which the operand immediately follows the instruction op code in memory. Direct addressing instructions require two bytes and are used when an operand is located in the lowest 256 bytes of memory (page zero). Extended addressing requires three bytes; it is utilized when an operand is

located in high memory, outside of page zero. Indexed addressing requires two bytes, the instruction and an *offset*. The offset is added to the contents of the index register to determine the address of the operand. Since the index register can be incremented or decremented, this mode of addressing allows the 6801 to perform arithmetic and data transfer operations very efficiently on data located in consecutive memory locations.

Relative addressing is used with the branch instructions of the 6801. If a branch is initiated, it will cause a new address to be loaded into the program counter, which will cause the program to continue execution at the new address. This new address is called the branch *destination*. Note in Fig. 2-2 that a branch instruction requires two bytes, the instruction and a *relative address*. To determine the branch destination, the 6801 adds the relative address to the current contents of the program counter. Since the 6801 uses twos-complement notation to represent negative numbers, the relative address can be positive or negative to provide both "forward" and "backward" branching. The biggest advantage of relative addressing is that it is independent of the absolute memory address. This advantage allows branching to be accomplished *relative* to the program counter contents; therefore, the proper branch destination is always achieved regardless of the location of the program in memory. However, the branching range is limited to +127 or -128 addresses.

The 6801 has 83 fundamental classes of instructions in its instruction set. These instructions, with their various addressing modes, comprise an instruction set that contains over 200 instructions. The 83 fundamental instructions are shown in Table 2-1. As stated earlier, most of the instructions can be used with several different addressing modes, which brings the total number of instructions to over 200. These instructions can be separated into seven general categories as follows: (1) arithmetic, (2) logic, (3) data handling, (4) data test, (5) condition code, (6) index register and stack pointer, and (7) jump and branch. Many of the arithmetic, logic, data-handling, and data-test instructions can be used with either immediate, direct, extended, or indexed addressing. A few of the instruction capabilities are worth reviewing. The 6801 has subroutine capabilities provided through the use of the jump-to-subroutine (JSR) and return-from-subroutine (RTS) instructions. Also, interrupts can be simulated for hardware and software testing through the use of software interrupt (SWI). Decimal adjust accumulator (DAA) instruction allows the 6801 to perform arithmetic operations on bcd numbers directly, without special conversion routines.

In addition to the existing 6800 instruction set, eleven new instructions are provided for the 6801. These new instructions are indi-

cated in Table 2-1 with an asterisk (*). Seven of these instructions were created to facilitate the use of the 16-bit accumulator, ACCD, with three provided to allow greater use of the index register contents. You will now be given an explanation of each new 6801 instruction. Refer to *How To Program and Interface the 6800* (Howard W. Sams and Co., Inc.) for a more detailed explanation and use of the 6800 instruction set.

ABX (Inherent)

$\text{IX} + \text{ACCB} \rightarrow \text{IX}$

The *unsigned* (8-bit straight binary) contents of accumulator B are added to the index register with the result being placed in the index register.

ADDD (Immediate, Direct, Extended, Indexed)

$\text{ACCD} + (\text{M:M+1}) \rightarrow \text{ACCD}$

The 16-bit contents of accumulator D are added to the two consecutive memory locations specified by M and M+1. The source for the memory address information depends upon which addressing mode is used. The result is placed in accumulator D. M will represent the high-order byte of the memory operand, with M+1 representing the low-order byte. The A accumulator is the high-order byte of the accumulator operand, with the B accumulator containing the low-order byte.

ASLD (Inherent)

$2\times(\text{ACCD}) \rightarrow \text{ACCD}$

All bits of accumulator D are shifted one place to the left. The most significant bit (MSB) is shifted into the C-flag of the condition code register (CCR), and a zero is loaded into the least significant bit (LSB). Note the times two ($2\times$) operation symbol. This symbol is used since each time the accumulator contents are shifted left, the result is the original contents multiplied by two. Executing ASLD twice would result in the original contents being multiplied by four, three times by eight, and so on. However, you must be careful when using this procedure for multiplication since you can lose the more significant bits when working with large numbers. After a bit is shifted left from the C-flag, it is lost, unless this condition is acted upon with subsequent instructions.

LDD (Immediate, Direct, Extended, Indexed)

$(\text{M: M+1}) \rightarrow \text{ACCD}$

The two consecutive memory locations specified by M and M+1 are loaded into accumulator D. Since ACCA and ACCB combine

Table 2-1. 6801/6803 Instruction Set— Alphabetic Sequence

ABA	Add Accumulators A and B	JMP	Jump
* ABX	Add to Index Register	JSR	Jump to Subroutine
ADC	Add with Carry	LDA	Load Accumulator (A or B)
ADD	Add	* LDD	Load Accumulator D
* ADDD	Add to Accumulator D	LDS	Load Stack Pointer
AND	Logical And	LDX	Load Index Register
ASL	Arithmetic Shift Left	LSR	Logical Shift Right
* ASLD	Arithmetic Shift Left, Accumulator D	* LSRD	Logic Shift Right, Accumulator D
ASR	Arithmetic Shift Right		
BCC	Branch If Carry Clear	* MUL	Multiply
BCS	Branch If Carry Set	NEG	Negate
* BEQ	Branch If Equal to Zero	NOP	No Operation
BGE	Branch If Greater or Equal Zero	ORA	Inclusive OR Accumulator (A or B)
BGT	Branch If Greater than Zero	PSH	Push Data
BHI	Branch If Higher	* PSHX	Push Index Register
BIT	Bit Test	PUL	Pull Data
BLE	Branch If Less or Equal	* PULX	Pull Index Register
BLS	Branch If Lower or Same		
BLT	Branch If Less than Zero	ROL	Rotate Left
BMI	Branch If Minus	ROR	Rotate Right
BNE	Branch If Not Equal to Zero	RTI	Return from Interrupt
* BRN	Branch Never	RTS	Return from Subroutine
BPL	Branch If Plus	SBA	Subtract Accumulators (B from A)
BRA	Branch Always	SBC	Subtract with Carry
BSR	Branch to Subroutine	SEC	Set Carry
BVC	Branch If Overflow Clear	SEI	Set Interrupt Mask
BVS	Branch If Overflow Set	SEV	Set Overflow
CBA	Compare Accumulators A and B	STA	Store Accumulator (A or B)
CLC	Clear Carry	* STD	Store Accumulator D
CLI	Clear Interrupt Mask	STS	Store Stack Pointer
CLR	Clear	STX	Store Index Register
CLV	Clear Overflow	SUB	Subtract
CMP	Compare	* SUBD	Subtract from Accumulator D
COM	Complement	SWI	Software Interrupt
CPX	Compare Index Register	TAB	Transfer Accumulators (A to B)
DAA	Decimal Adjust	TAP	Transfer Accumulator A to Condition Code Register
DEC	Decrement	TBA	Transfer Accumulators (B to A)
DES	Decrement Stack Pointer		
DEX	Decrement Index Register		
EOR	Exclusive OR		
INC	Increment		
INS	Increment Stack Pointer		
INX	Increment Index Register		

Table 2-1. Continued

TPA	Transfer Condition Code Register to Accumulator A	TXS	Transfer Index Register to Stack Pointer
TST	Test		
TSX	Transfer Stack Pointer to Index Register	WAI	Wait for Interrupt

to form ACCD, M is actually loaded into ACCA with M+1 being loaded into ACCB.

LSRD (Inherent)

$(ACCD) \div 2 \rightarrow ACCD$

All bits of accumulator D are shifted one place to the right. A zero is loaded into the MSB and the LSB is shifted into the C-flag of the condition code register. Note the divide-by-two ($\div 2$) operation symbol. This symbol is used since each time the accumulator contents are shifted right, the result is the original contents divided by two. Executing LSRD twice would result in the original contents being divided by four, three times by eight, and so on. Any remainder generated by the division is found in the C-flag. If this remainder is to be used, subsequent program steps must test the C-flag and act to save its state.

MUL (Inherent)

$ACCA * ACCB \rightarrow ACCD$

The unsigned eight bits of accumulator A are multiplied by the unsigned eight bits of accumulator B. An unsigned 16-bit result is generated and placed in accumulator D. The internal execution of this instruction is shown in Fig. 2-3. Note from the figure that the contents of accumulators A and B are multiplied in the arithmetic logic unit (alu) with the result being placed in accumulator D. Since accumulator A and B combine to form accumulator D, the most significant byte of the result is actually placed in accumulator A, and the least significant result byte is placed in accumulator B. Therefore, the previous contents (multiplier and multiplicand) of these accumulators are lost.

PSHX (Inherent)

$X_L \rightarrow M_{SP}, SP-1 \rightarrow SP$
 $X_H \rightarrow M_{SP}, SP-1 \rightarrow SP$

The contents of the index register are *pushed* onto the stack. The low byte of the index register is placed at the memory address specified by the stack pointer. The stack pointer is then decremented and the high byte of the index register is placed at this specified

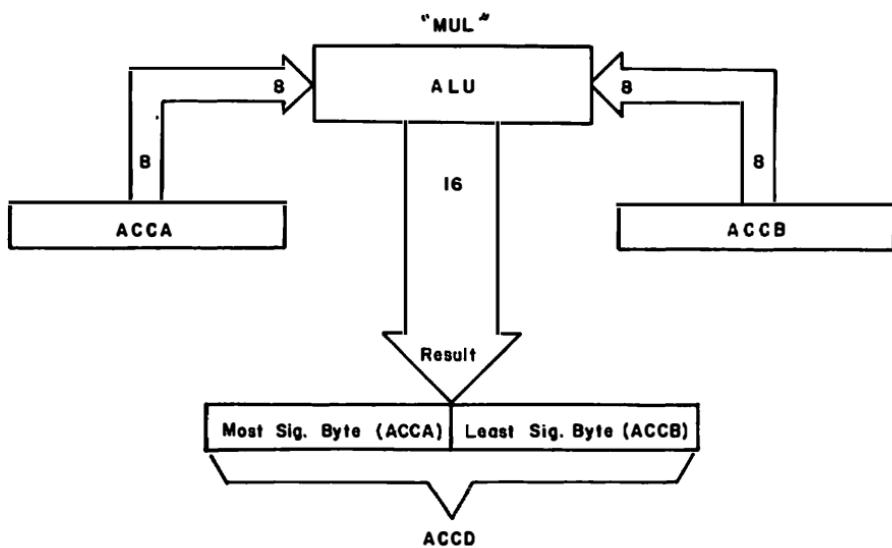


Fig. 2-3. Execution of the multiple (MUL) instruction.

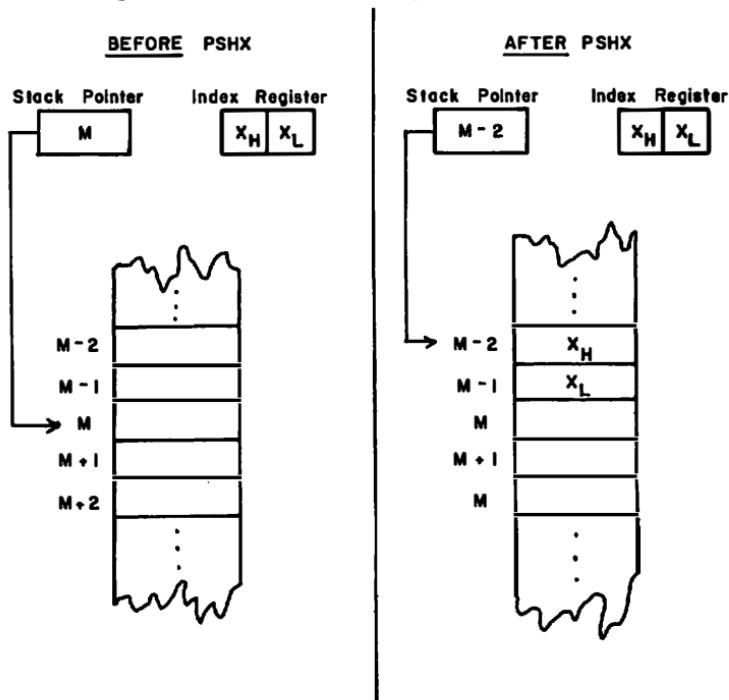


Fig. 2-4. Execution of the push index register (PSHX) instruction.

memory address. The stack pointer will then be decremented again to indicate the next *available* stack location. The conditions before and after executing this instruction are shown in Fig. 2-4.

PULX (Inherent)

$SP+1 \rightarrow SP, M_{SP} \rightarrow X_H$

$SP+1 \rightarrow SP, M_{SP} \rightarrow X_L$

The index register contents are *pulled* from the stack. The stack pointer is incremented to point to the address where the high byte of the index register is stored. The high byte is then pulled from the stack and loaded into the high byte of the index register. The stack pointer is then incremented again to point to the location of the low index register byte. This byte is then pulled from the stack and loaded into the low byte of the index register. The conditions before and after executing this instruction are shown in Fig. 2-5.

STD (Direct, Extended, Indexed)

$ACCD \rightarrow (M: M+1)$

The 16-bit contents of accumulator D are stored at the two consecutive memory locations specified by M and M+1. Actually, since ACCA and ACCB combine to form ACCD, ACCA is stored at memory location M and ACCB is stored at memory location M+1.

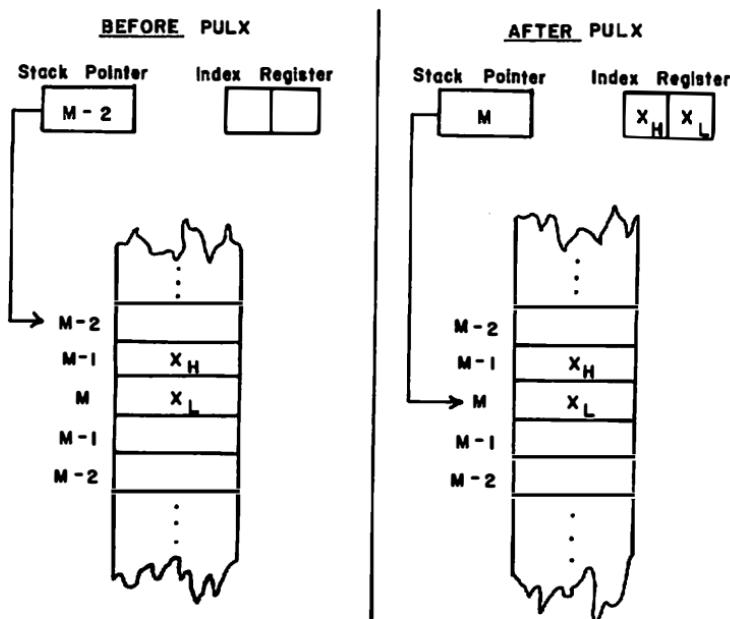


Fig. 2-5. Execution of the pull Index register (PULX) instruction.

Table 2-2. 6801/6803 Alphabetic Mnemonic/Op-Code Listing

Mnemonic	Immediate	Direct	Index	Extended	Inherent	Relative
ABA	—	—	—	—	1B	—
*ABX	—	—	—	—	3A	—
ADCA	89	99	A9	B9	—	—
ADCB	C9	D9	E9	F9	—	—
ADDA	8B	9B	AB	BB	—	—
ADDDB	CE	DB	EB	FB	—	—
*ADDD	C3	D3	E3	F3	—	—
ANDA	84	94	A4	B4	—	—
ANDB	C4	D4	E4	F4	—	—
ASL	—	—	68	78	—	—
ASLA	—	—	—	—	48	—
ASLB	—	—	—	—	58	—
*ASLD	—	—	—	—	05	—
ASR	—	—	67	77	—	—
ASRA	—	—	—	—	47	—
ASRB	—	—	—	—	57	—
BCC	—	—	—	—	—	24
BCS	—	—	—	—	—	25
BEQ	—	—	—	—	—	27
BGE	—	—	—	—	—	2C
BGT	—	—	—	—	—	2E
BHI	—	—	—	—	—	22
BITA	85	95	A5	B5	—	—
BITB	C5	D5	E5	F5	—	—
BLE	—	—	—	—	—	2F
BLS	—	—	—	—	—	23
BLT	—	—	—	—	—	2D
BMI	—	—	—	—	—	2B
BNE	—	—	—	—	—	26
BPL	—	—	—	—	—	2A
BRA	—	—	—	—	—	20
*BRN	—	—	—	—	—	21
BSR	—	—	—	—	—	8D
BVC	—	—	—	—	—	28
BVS	—	—	—	—	—	29
CBA	—	—	—	—	11	—
CLC	—	—	—	—	0C	—
CLI	—	—	—	—	0E	—
CLR	—	—	6F	7F	—	—
CLRA	—	—	—	—	4F	—
CLRB	—	—	—	—	5F	—
CLV	—	—	—	—	0A	—
CMPA	81	91	A1	B1	—	—
CMPB	C1	D1	E1	F1	—	—
COM	—	—	63	73	—	—
COMA	—	—	—	—	43	—
COMB	—	—	—	—	53	—
CPX	8C	9C	AC	BC	—	—
DAA	—	—	—	—	19	—
DEC	—	—	6A	7A	—	—
DECA	—	—	—	—	4A	—

Table 2-2. Continued

Mnemonic	Immediate	Direct	Index	Extended	Inherent	Relative
DECB	—	—	—	—	5A	—
DES	—	—	—	—	34	—
DEX	—	—	—	—	09	—
EORA	88	98	A8	B8	—	—
EORB	C8	D8	E8	F8	—	—
INC	—	—	6C	7C	—	—
INCA	—	—	—	—	4C	—
INCB	—	—	—	—	5C	—
INS	—	—	—	—	31	—
INX	—	—	—	—	08	—
JMP	—	—	6E	7E	—	—
JSR	—	9D	AD	BD	—	—
LDAA	86	96	A6	B6	—	—
LDAB	C6	D6	E6	F6	—	—
*LDD	CC	DC	EC	FC	—	—
LDS	8E	9E	AE	BE	—	—
LDX	CE	DE	EE	FE	—	—
LSR	—	—	64	74	—	—
LSRA	—	—	—	—	44	—
LSRB	—	—	—	—	54	—
*LSRD	—	—	—	—	04	—
*MUL	—	—	—	—	3D	—
NEG	—	—	60	70	—	—
NEGA	—	—	—	—	40	—
NEGB	—	—	—	—	50	—
NOP	—	—	—	—	01	—
ORAA	8A	9A	AA	BA	—	—
ORAB	CA	DA	EA	FA	—	—
PSHA	—	—	—	—	36	—
PSHB	—	—	—	—	37	—
*PSHX	—	—	—	—	3C	—
PULA	—	—	—	—	32	—
PULB	—	—	—	—	33	—
*PULX	—	—	—	—	38	—
ROL	—	—	69	79	—	—
ROLA	—	—	—	—	49	—
ROLB	—	—	—	—	59	—
ROR	—	—	66	76	—	—
RORA	—	—	—	—	46	—
RORB	—	—	—	—	56	—
RTI	—	—	—	—	3B	—
RTS	—	—	—	—	39	—
SBA	—	—	—	—	10	—
SBCA	82	92	A2	B2	—	—
SBCB	C2	D2	E2	F2	—	—
SEC	—	—	—	—	0D	—
SEI	—	—	—	—	0F	—
SEV	—	—	—	—	0B	—
STAA	—	97	A7	B7	—	—
STAB	—	D7	E7	F7	—	—

Table 2-2. Continued

Mnemonic	Immediate	Direct	Index	Extended	Inherent	Relative
*STD	—	DD	ED	FD	—	—
STS	—	9F	AF	BF	—	—
STX	—	DF	EF	FF	—	—
SUBA	80	90	A0	B0	—	—
SUBB	C0	D0	E0	F0	—	—
*SUBD	83	93	A3	B3	—	—
SWI	—	—	—	—	3F	—
TAB	—	—	—	—	16	—
TAP	—	—	—	—	06	—
TBA	—	—	—	—	17	—
TPA	—	—	—	—	07	—
TST	—	—	6D	7D	—	—
TSTA	—	—	—	—	4D	—
TSTB	—	—	—	—	5D	—
TSX	—	—	—	—	30	—
TXS	—	—	—	—	35	—
WAI	—	—	—	—	3E	—

*New Instructions

SUBD (Immediate, Direct, Extended, Indexed)

ACCD — (M:M+1) → ACCD

The contents of the two consecutive memory locations specified by M and M+1 are subtracted from accumulator D. The result is then placed in accumulator D.

BRN (Relative)

Branch Never

The branch-never instruction is actually a two-byte no-operation (NOP) instruction. The 6801 will cycle through the instruction byte and relative address without altering its execution. On the surface you might think this is a meaningless instruction and wonder why it was included in the 6801 instruction set. However, you can *bury* an instruction op code as the relative address. For example, when the 6801 first cycles through the branch instruction, it will read the buried op code as a relative address and no operation will result. Then, later in the program, you can jump back to the buried op code for its execution.

As indicated previously, these instructions can be used with various addressing modes. A complete 6801 instruction op-code listing is provided in Table 2-2, and the instruction execution time in MPU cycles is shown in Table 2-3. Also, a complete explanation of each instruction in the 6801 instruction set is provided in Appendix A. If you compare the number of MPU cycles required to execute the 6801 instructions (Table 2-3) with those for the 6800 instructions,

you will find that the number of cycles required by the 6801 instructions has been reduced for some of the key instructions. This has been done to decrease the execution time for the 6801 when compared with a similar 6800 operation.

**Table 2-3. 6801/6803 Instruction Execution Times
in Machine Cycles**

	ACCX	IMMEDIATE	DIRECT	EXTENDED	INDEXED	INHERENT	RELATIVE	ACCX	IMMEDIATE	DIRECT	EXTENDED	INDEXED	INHERENT	RELATIVE
ABA	●	●	●	●	●	●	●	INX	●	●	●	●	●	3
ABX	●	●	●	●	●	●	●	JMP	●	●	●	●	●	3
ADC	●	2	3	4	4	4	●	JSR	●	●	●	●	●	3
ADD	●	●	2	3	4	4	●	LDA	●	●	●	●	●	6
ADDD	●	●	4	5	6	6	●	LDI	●	3	3	4	4	5
AND	●	2	3	4	4	4	●	LDS	●	3	4	5	5	5
ASL	2	●	●	●	6	6	●	LDX	●	3	4	5	6	6
ASLD	●	●	●	●	●	●	3	LSR	2	●	●	●	●	●
ASR	2	●	●	●	6	6	●	LSRD	●	●	●	●	3	●
BCC	●	●	●	●	●	●	●	MUL	●	●	●	●	●	10
BCS	●	●	●	●	●	●	3	NEG	2	●	●	●	●	●
BEQ	●	●	●	●	●	●	3	NOP	●	●	●	●	●	2
BGE	●	●	●	●	●	●	3	ORA	●	2	3	4	4	●
BGT	●	●	●	●	●	●	3	PSH	3	●	●	●	●	●
BHI	●	●	●	●	●	●	3	PSHX	●	●	●	●	●	4
BIT	●	2	3	4	4	4	●	PUL	4	●	●	●	●	●
BLE	●	●	●	●	●	●	3	PULK	●	●	●	●	●	5
BLS	●	●	●	●	●	●	3	ROL	2	●	●	6	6	●
BLT	●	●	●	●	●	●	3	ROR	2	●	●	6	6	●
BMI	●	●	●	●	●	●	3	RTI	●	●	●	●	●	10
BNE	●	●	●	●	●	●	3	RTS	●	●	●	●	●	5
BPL	●	●	●	●	●	●	3	SBA	●	●	●	●	●	2
BRA	●	●	●	●	●	●	3	SBC	●	2	3	4	4	●
BSR	●	●	●	●	●	●	●	SEC	●	●	●	●	●	2
BVC	●	●	●	●	●	●	●	SEI	●	●	●	●	●	2
BVS	●	●	●	●	●	●	●	SEV	●	●	●	●	●	2
CBA	●	●	●	●	●	●	●	STA	●	●	3	4	4	●
CLC	●	●	●	●	●	●	●	STD	●	●	4	5	5	●
CLI	●	●	●	●	●	●	●	STS	●	●	4	5	5	●
CLR	2	●	●	●	6	6	●	STX	●	●	4	5	5	●
CLV	●	●	●	●	●	●	●	SUB	●	2	3	4	●	●
CMP	●	2	3	4	4	4	●	SUBD	●	4	5	6	6	●
COM	2	●	●	●	6	6	●	SWI	●	●	●	●	●	12
CPX	●	4	5	6	6	6	●	TAB	●	●	●	●	●	2
DAA	●	●	●	●	●	●	2	TAP	●	●	●	●	●	2
DEC	2	●	●	●	6	6	●	TBA	●	●	●	●	●	2
DES	●	●	●	●	●	●	●	TPA	●	●	●	●	●	2
DEX	●	●	●	●	●	●	3	TST	2	●	●	6	6	●
EOR	●	2	3	4	4	4	●	TSX	●	●	●	●	3	●
INC	2	●	●	●	6	6	●	TXS	●	●	●	●	●	3
INS	●	●	●	●	●	●	3	WAI	●	●	●	●	9	●

There are two other subtle differences between the 6801 and 6800 instruction sets. They involve the jump to subroutine (JSR) and compare index register (CPX) instructions. With the 6801, the JSR instruction can be used in the direct addressing mode where this is not possible with the 6800. See Table 2-2 for the proper op code. The CPX instruction can now be used with any conditional branch instruction in the 6801, where its use in the 6800 is limited to those branch instructions involving the Z-flag.

The following are examples of how some of the new 6801 instructions might be used.

Example 2-1: Using the Multiply (MUL) Instruction

This program shows how to use the MUL instruction.

Hex Address	Hex Contents	Mnemonic/Contents	Operation
0080	86	LDAA #	
0081	20	20	20 → ACCA
0082	C6	LDAB #	
0083	0A	0A	0A → ACCB
0084	3D	MUL	ACCA * ACCB → ACCD
0085	FD	STD \$\$	
0086	00	00	ACCD → Mc0:Mc1
0087	C0	C0	
0088	3E	WAI	Stop
.	.	.	.
00C0	—	—	Most Significant Result Byte
00C1	—	—	Least Significant Result Byte

The program multiplies the contents of ACCA and ACCB, then stores the 16-bit result at memory locations 00C0 and 00C1. Accumulator A is located with 20 and accumulator B with 0A. The MUL instruction then multiplies these values and places the 16-bit result, 0140, in accumulator D. Note, since accumulator D is the combined contents of accumulators A and B, accumulator A should now contain 01 and accumulator B will contain 40. The previous accumulator contents are lost. Finally, the result is stored at memory locations 00C0 and 00C1 with the store accumulator D (STD) instruction. In practice, the most significant result byte, 01, located in accumulator A is stored at memory location 00C0 and the least significant result byte, 40, located in accumulator B is stored at memory location 00C1. This program would not require the use of any external memory since it operates within the 128 bytes of on-chip memory of the 6801. Note that the program begins at address 0080, the beginning of the internal R/W memory.

Fig. 2-6 shows 16 push-button switches interfaced to the 6801 via pull-up resistors. If one of the switches is depressed, a logic "0" will be present on its interface line. When interfacing with switches, three functions must be performed: detect switch closure, provide switch debouncing and then, in the case of multiple switches, decode the switch column or matrix to determine exactly which switch is closed (Reference Chapter 8).

The following program will detect switch closure and decode the switch column shown in Fig. 2-6. To make the program com-

plete, a debouncing subroutine should be added between the switch closure detection and decoding routines.

Example 2-2: Decoding a Switch Column

	Hex Address	Hex Contents	Mnemonic/Contents	Operation
Detect Switch Closure	0080	DC	LDD \$	(Mo:Mo7) ACCD
	0081	06	06	(read switch data)
	0082	81	CMPA #	Compare ACCA Immediate to FF
	0083	FF	FF	
	0084	26	BNE	Branch If Z-flag cleared
	0085	06	06	(to address 008C)
	0086	C1	CMPB #	Compare ACCB Immediate to FF
	0087	FF	FF	
	0088	26	BNE	Branch If Z-flag cleared
	0089	02	02	(to address 008C)
Decode Switch Column	008A	20	BRA	Branch Always
	008B	F4	F4	(to address 0080)
	008C	CE	LDX #	
	008D	00	00	Clear the Index register
	008E	00	00	
	008F	04	LSRD	Logic Shift Right - ACCD
	0090	08	INX	Increment the Index register
	0091	25	BCS	Branch If Carry Set
	0092	FC	FC	(to address 008F)
	0093	DF	STX \$	Store Index register
	0094	BF	BF	count at 00C0
	0095	3E	WAI	Stop
.				
.				
.				
00C0				
Index register count = switch #				

The program assumes that two of the 6801 ports have been configured as input ports and assigned addresses 0006 and 0007. The program begins by reading the 16-bit switch data into accumulator D. If a switch has been depressed, one of the accumulator bits will be a logic "0." All of the other bits should be "1" and to detect the zero bit, the program first compares accumulator A, then accumulator B, to FF. If during either compare operation the zero bit is present, the Z-flag will be cleared and the program will branch to the switch decoding routine. If no switch closure is detected, the program will branch back to the beginning and continue to read the switch data until a closure is detected. Once a closure has been detected and debounced, you must determine exactly which switch is closed. This requires you to determine the logic zero bit position. The zero bit can be decoded by rotating the accumulator D contents into the

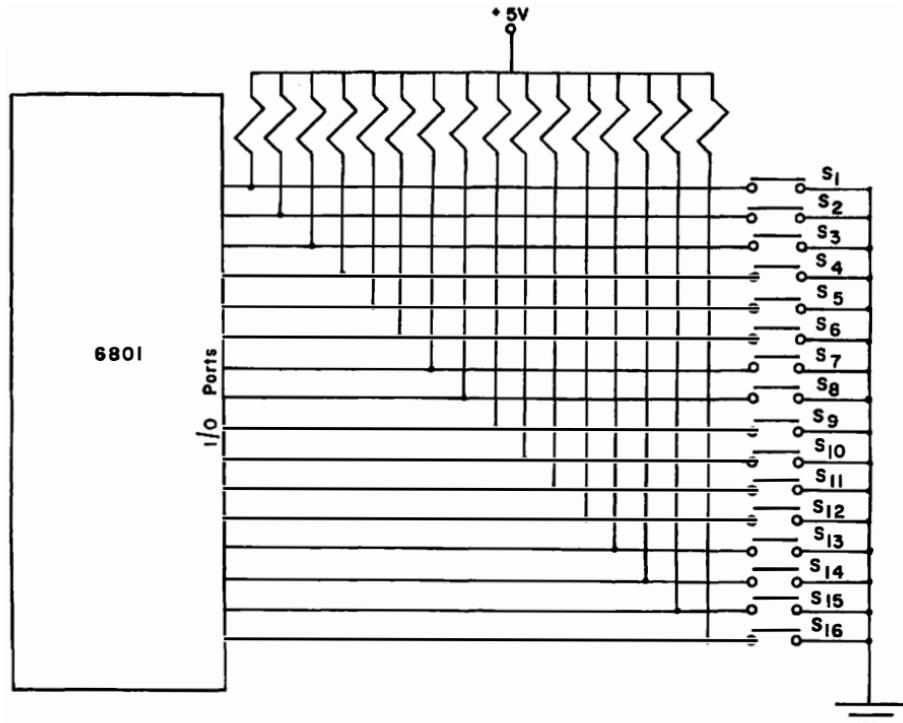


Fig. 2-6. Interfacing a switch column to the 6801.

carry flag of the condition code register until that flag is cleared. The 6801 can then determine which switch is closed by counting the number of rotations that it takes to detect the cleared, or logic zero bit position. This program rotates accumulator D into the C-flag using the LSRD instruction. After each rotation, the index register is incremented to count the number of rotations necessary to clear the C-flag. This count corresponds to the switch number. The proper switch number is then stored at memory location 00C0. Again, this routine would not require the use of any external memory since it operates within the 128 bytes of on-chip memory of the 6801. Also, no external PIA would be required for the switch interfacing since the 6801 on-chip i/o ports are being utilized. A discussion of how to configure these ports for this type of operation will follow in subsequent chapters.

We will standardize on the following method to indicate instruction addressing:

= immediate addressing

\$ = direct addressing

\$\$ = extended addressing

X = indexed addressing
no symbol = inherent addressing

This method was used in the previous examples and will be used throughout the text.

REVIEW QUESTIONS

1. Accumulator D is a _____-bit accumulator.
2. Accumulators A and B combine to form accumulator D where accumulator _____ forms the high byte and accumulator _____ forms the low byte.
3. Bits zero through five comprise the six flags of the condition code register.

List them in their respective order. _____, _____,

4. The six modes of addressing utilized by the 6801 are: _____,

_____ → _____ → _____ →

5. What is the advantage of relative addressing?

6. The 6801 instruction set can be broken down into seven general categories.

They are: _____, _____, _____,

_____ → _____ → _____ →

7. List the seven new instructions which are associated with the new accumulator (ACCD). _____, _____,

_____ → _____ → _____ →

8. List the three new instructions that are associated with the index register.

_____ → _____ → _____ →

9. Describe the MUL instruction.

10. $2x(\text{ACCD}) \rightarrow \text{ACCD}$ is the operation symbol for the _____ instruction.

11. $\text{ACCD} \div 2 \rightarrow \text{ACCD}$ is the operation symbol for the _____ instruction.

12. When detecting switch closure in a switch column, a _____ instruction is used.

13. When decoding switch closure in a switch column, a _____ instruction is used.
14. The 6801 has _____ fundamental instructions. (How many?)
15. The add to index register (ABX) instruction adds _____ to the index register and stores the result in the index register.

ANSWERS

1. 16
2. A, B
3. carry (C), overflow (V), zero (Z), negative (N), interrupt (I), half carry (H)
4. inherent, immediate, direct, extended, indexed, relative
5. Program position independence
6. arithmetic, logic, data handling, data test, condition code, index register and stack pointer, jump and branch
7. add to accumulator D (ADDD)
arithmetic shift left—accumulator D (ASLD)
load accumulator D (LDD)
logic shift right—accumulator D (LSRD)
multiply (MUL)
store accumulator D (STD)
subtract from accumulator D (SUBD)
8. add to index register (ABX)
push index register (PSHX)
pull index register (PULX)
9. The multiply (MUL) instruction multiplies the contents of accumulator A times accumulator B and stores the *unsigned* result in accumulator D.
10. arithmetic shift left—accumulator D (ASLD)
11. logic shift right—accumulator D (LSRD)
12. compare
13. logic shift right
14. 83
15. accumulator B

CHAPTER 3

6801 Fundamental Operating Modes and Pin Assignments

INTRODUCTION

Now that you have become acquainted with the basic 6801 chip structure and software, you are ready to understand why the 6801 is referred to as a "superchip." The power of the 6801 lies in its extreme flexibility. As stated earlier, the 6801 can operate as a single stand-alone microcomputer with limited R/W memory, ROM and i/o, or it can be expanded into a 64K address system. This tremendous flexibility allows you to use the 6801 for a wide variety of applications that are made possible by three fundamental operating modes. You can select one of three operating modes which will best suit *your* application. As mentioned in Chapter 1, the three fundamental operating modes are *single-chip*, *expanded nonmultiplexed*, and *expanded multiplexed*.

The *single-chip* mode will allow you to use the 6801 as a stand-alone microcomputer. All that you need to supply to the 6801 are power and an external crystal to provide the clock frequency. In this mode, you will use the 128 bytes of on-chip R/W memory and the 2K on-chip ROM. In addition, you will have access to four independent, programmable, i/o ports along with the internal timer and serial communications interface. This mode is ideal for small dedicated applications such as process controllers, appliances, toys, games, etc.

In the *expanded nonmultiplexed* mode, the 6801 provides you with access to the internal data bus and the lower eight address lines and thus allows expansion to 256 address words. In this mode, you

sacrifice two i/o ports; however, you will still have access to one 8-bit i/o port, the internal 16-bit timer, and serial communications interface.

Finally, in the *expanded multiplexed* mode you will gain access to the data bus and the complete 16-line address bus. This will allow you to expand the 6801 to a 64K address system. By using a *multiplexing* scheme between the data and address buses, you will still have access to one 8-bit i/o port, the internal timer, and serial communications interface.

You might wonder how all of this can be achieved with one 40-pin chip. When you select a particular operating mode, you are configuring the chip's pins, so that depending upon the mode selected, a particular pin could be configured as i/o, data or address. Therefore, you might say that the chip pin assignments are programmable. Now, let us dispense with the rhetoric and get into the *flexible* world of the 6801.

OBJECTIVES

At the end of this chapter you will be able to do the following:

- Define each of the three fundamental 6801 operating modes.
- Describe the crystal requirements for the 6801.
- Connect the V_{cc} Standby pin to a battery backup to provide the retainable R/W memory function.
- Describe the 6801 external hardware interrupt capabilities.
- Describe the function of each port versus the 6801 chip mode.
- Suggest applications for the 6801.

SINGLE-CHIP MODE

As stated in the introduction to this chapter, there are three basic functional modes of the 6801. These modes will allow you to operate the 6801 as a stand-alone microcomputer with the 128 bytes of R/W memory and 2K bytes of ROM in the chip, or to expand to a 64K address system. You will now be given a functional pin-by-pin description of the 6801 in each of its three basic modes. Our discussion will include figures that serve as functional operating diagrams as well as provide the 6801 pin assignments for each basic mode. A complete discussion of how to configure the 6801 in its various modes is provided in Chapter 4.

The single-chip mode will allow you to operate the 6801 as a complete one-chip microcomputer with 128 bytes of R/W memory, 2K bytes of ROM and four i/o ports. In addition, you will also have access to the on-board serial communications interface and the 16-

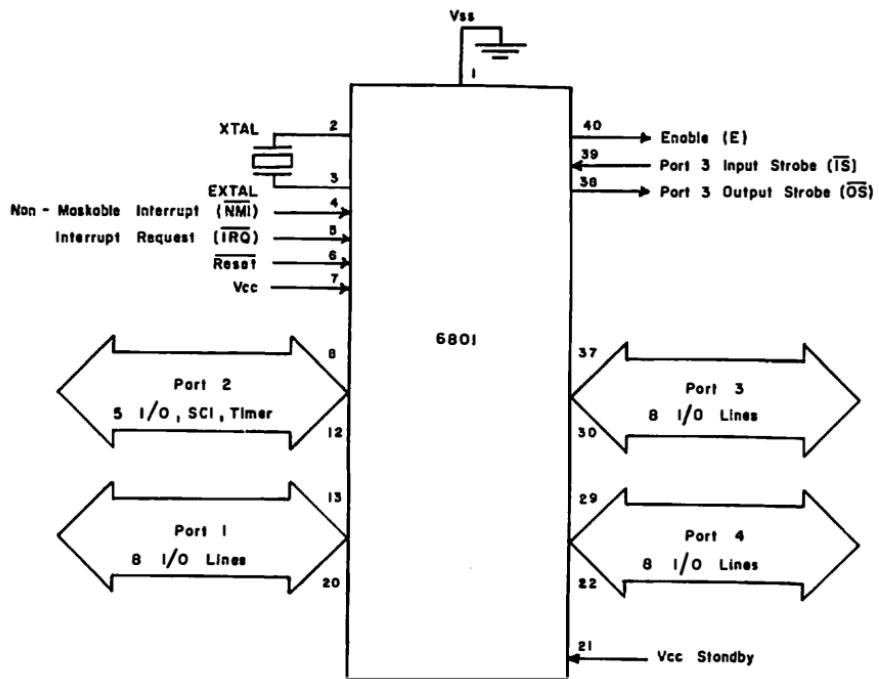


Fig. 3-1. The 6801 in single-chip mode.

bit timer via one of the i/o ports. Fig. 3-1 shows the 6801 pin configurations for the single-chip mode. The following is an explanation of the function of each pin when configured in this mode.

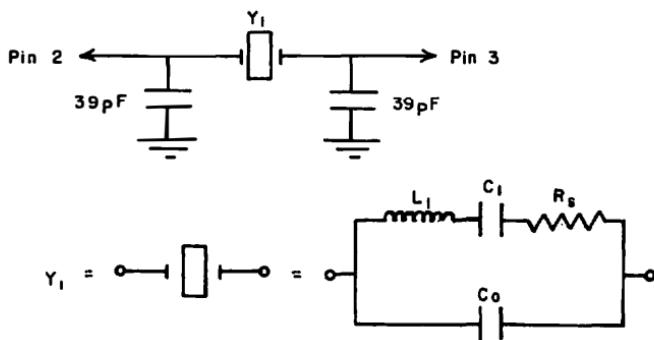
V_{SS} (Pin 1)

Pin 1 is used to ground the chip.

XTAL 1 and EXTAL 2 (Pins 2 and 3)

These pins are used to connect a parallel-resonant fundamental crystal to provide crystal control of the internal oscillator's frequency. The 6801 is basically a 1-MHz chip which contains an internal divide-by-four circuit such that a 4-MHz crystal may be used for a cost-effective system. Actually, you could even connect an inexpensive 3.58-MHz color burst crystal to these pins. This would provide an operating frequency of .895 MHz, which is fast enough for most applications. The required crystal circuit is shown in Fig. 3-2. Note that two 39-pF capacitors need to be connected from the crystal's pins to ground. This will ensure optimum chip performance. The vendor specifications for the crystal are shown in the table in

Fig. 3-2. Note with a 5-MHz crystal, it is possible to operate the 6801 at a frequency of 1.25 MHz. For optimum performance, Motorola recommends that LC networks *not* be used in place of a crystal. To minimize distortion, the crystal should be mounted as close as possible to the input pins, XTAL and EXTAL.



Y_1 Crystal Frequency	C_1	R_g	C_0	C_L	Q
3.5 MHz	.015 pF	60 μ H	3.5 pF	25 pF	40K
4.1 MHz	.025 pF	50 μ H	6.5 pF	25 pF	30K
5.0 MHz	.027 pF	30 μ H	6.5 pF	25 pF	40K

Fig. 3-2. 6801 crystal circuit and crystal specifications table.

You may also drive the 6801 with an external TTL or CMOS clock signal. If this is desired, the signal is applied to pin 3 (EXTAL), with pin 2 (XTAL) grounded. In either case, the clock frequency will be divided by four for internal operations.

Nonmaskable Interrupt (Pin 4)

The nonmaskable interrupt (NMI) is a hardware interrupt controlled by an external device. As with the 6800, the 6801 uses a vectored interrupt scheme. When a high-to-low transition is provided at pin 4, the 6801 will automatically look at address FFFC:FFF4 for the nonmaskable interrupt service routine vector. This vector *points* to the beginning of the nonmaskable interrupt service routine. As the name implies, it cannot be masked-out, and is always active, accepting interrupts at any time. The sequence of events shown in Fig. 3-3 will be initiated when pin 4 goes through a high-to-low transition.

Interrupt Request (Pin 5)

Interrupt request (IRQ) is a hardware interrupt controlled by an external device. It can be masked-out with the I-flag of the condition code register. The sequence of events shown in Fig. 3-4 will

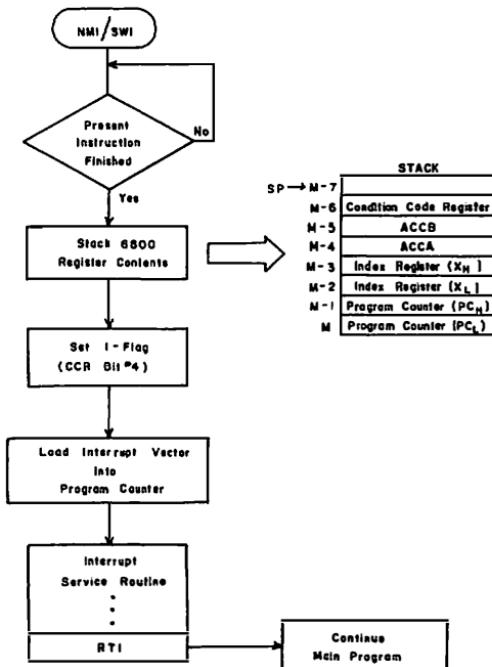


Fig. 3-3. Nonmaskable interrupt (NMI) sequence of events.

be initiated when pin 5 goes from a high to a low state. The IRQ interrupt vector is located at addresses FFF8:FFF9.

RESET (Pin 6)

The reset function is also a hardware interrupt. The reset interrupt vector is located at address FFFE:FFFF. The reset interrupt service routine, normally located in ROM, will normally provide for system restart and initialization. The reset interrupt sequence of events is shown in Fig. 3-5.

The 6801 interrupt vector map is given in Table 3-1. Note that the NMI, IRQ and RESET vector assignments are the same as for the 6800. However, the 6801 provides four additional vectors for the timer and serial i/o function of port 2. Therefore, the total vector map comprises addresses FFF0 through FFFF. The timer and serial i/o vectors are discussed separately in Chapters 6 and 7.

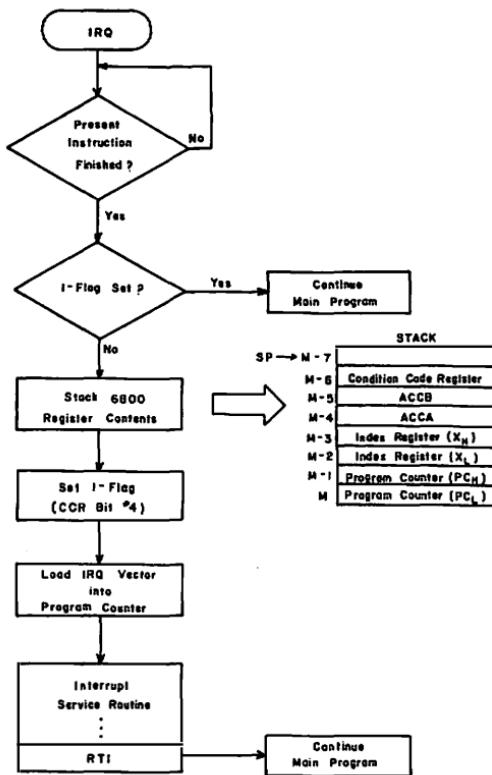


Fig. 3-4. Interrupt request (IRQ) sequence.

Table 3-1. 6801 Interrupt Vector Map

	Vector Address	Interrupt Vector
(highest priority)	FFFF:FFFF FFFC:FFFD FFFA:FFFB FFF8:FFF9 FFF6:FFF7 FFF4:FFF5 FFF2:FFF3 FFF0:FFF1	RESET Nonmaskable Interrupt (NMI) Software Interrupt Interrupt Request (IRQ1) Timer Input Capture (IRQ2) Timer Output Compare (IRQ2) Timer Overflow (IRQ2) Serial Communications (IRQ2)

V_{CC} (Pin 7)

As with all of the 6800 family, the 6801 is a 5-volt chip. Therefore, this pin is connected to a +5-V dc power supply. Consult the 6801 specifications in Appendix B for the supply voltage requirements.

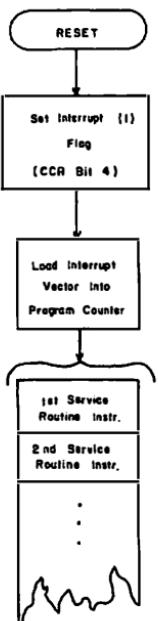


Fig. 3-5. Reset Interrupt sequence of events.

Port 2 (Pins 8–12)

Port 2 consists of five lines that are designated as P20 through P24. These lines are associated with pins 8 through 12, respectively. In all modes, this port can be configured as parallel i/o through the use of an associated data direction register (DDR) located at address 0001. In addition, this port is also used to gain access to the serial communications interface for serial i/o, and also to access the 16-bit timer. As you will see in Chapters 6 and 7, P20 and P21 are used for the timer with P22, P23, and P24 used for serial i/o. Serial i/o timer, or any combination thereof, will be discussed in subsequent chapters.

Port 1 (Pins 13–20)

Port 1 consists of eight parallel i/o lines that we shall designate as P10 through P17, where P10 stands for port 1, bit 0; P11 stands for port 1, bit 1; etc. These eight lines use pins 13 through 20, respectively. As you will see in the other 6801 modes, port 1 is *always* an i/o port. The i/o lines are independently programmable for either input or output through the use of an associated data direction register (DDR) located at address 0000. The procedure is similar to that used with the 6820/6821 PIA.¹ The port lines are three-state buffered and TTL compatible. When used as outputs, these lines can drive one Schottky TTL load and 30 pF, Darlington trans-

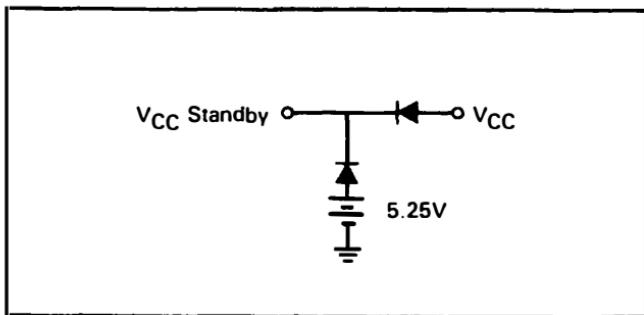


Fig. 3-6. Battery backup connection diagram.

sistors, or CMOS devices using external pull-up resistors. The port initialization procedure and addressing will be discussed in subsequent chapters.

$V_{CC\text{ Standby}}$ (Pin 21)

The $V_{CC\text{ Standby}}$ function requires a 4.0- to 5.25-volt dc voltage source. This source will supply power to the first 64 bytes of R/W memory located at addresses 0080 to 00BF. Maximum current drain is approximately 8 mA at 5.25 V (42 mW). Using this function, you can retain memory in the first 64 bytes of R/W memory when a loss of the main power supply (V_{CC}) is experienced due to power failure or a standby condition. A battery backup connection diagram is given in Fig. 3-6. If the battery backup option is not desired, you must connect this pin to V_{CC} to provide power for the first 64 bytes of R/W memory.

Port 4 (Pins 22–29)

In the single-chip mode, port 4 is very much like port 1. It has eight parallel i/o lines that we shall designate as P40 through P47. Here, P40 is associated with pin 29, P41 with pin 28 and so on, ending with P47 at pin 22. These lines can be programmed independently as either input or output with the use of an associated DDR at address 0005. Port 4 can drive one Schottky TTL load and 90 pF and is the only port with internal pull-up resistors. Therefore, you can interface this port directly to CMOS devices. However, external pull-up resistors to more than five volts cannot be used.

Port 3 (Pins 30–37)

In this mode, port 3 is also utilized as an i/o port. Its eight parallel i/o lines will be designated as P30 through P37 with P30 being associated with pin 37, P31 with pin 36 and so on, ending with P37 at pin 30. These eight lines also have an associated DDR at

address 0004 to allow them to be independently programmed as either input or output. There are also two control lines associated with this port in this mode, an input strobe and an output strobe. The port lines are three-state buffered and TTL compatible. When used as outputs, these lines can drive one Schottky TTL load and 90 pF. CMOS devices require the use of external pull-up resistors.

Port 3 Output Strobe (Pin 38)

The port 3 output strobe (\overline{OS}) is similar to the CA2 and CB2 control lines on the 6820/6821 PIA. However, this pin always provides an output strobe pulse and it *cannot* be programmed for input.¹ Pin 38 is normally high, but will go low for one clock cycle when a read or write operation is performed from, or to, port 3. This will provide a strobe for complete or partial handshaking between the 6801 and an i/o device. The use of this strobe is discussed in Chapter 4.

Port 3 Input Strobe (Pin 39)

The port 3 input strobe (\overline{IS}) is similar to the CA1 and CB1 control lines on the 6820/6821 PIA.¹ It is used in conjunction with port 3 to generate an interrupt request to the 6801. When an i/o device requires service via port 3, it can generate an interrupt request to the 6801 by providing a high-to-low transition at pin 39.

In the single chip mode, the combined use of the input and output strobes can provide complete handshaking with an i/o device via port 3. The use of this strobe is discussed in Chapter 4.

Enable (Pin 40)

In all the 6801 modes, the Enable (E) line is used to provide a synchronization pulse for communication between the 6801 and its external devices. This line is used in a manner as the ϕ_2 clock pulse on the 6800, and it should be part of any external chip decoding scheme.

EXPANDED NONMULTIPLEXED MODE

The expanded nonmultiplexed mode provides you with external access to the 6801's data bus, D0 through D7, and address lines A0 through A7. Therefore, in this mode, you can expand the 6801 to 256 address words. The functional configuration and pin assignments for this mode are shown in Fig. 3-7. The functions of the first 21 pins are identical to that of the single-chip mode. Note that ports 1 and 2 take on the same functions as in the single-chip mode, with

¹ Staugaard, Andrew C., *How To Program and Interface the 6800* (Howard W. Sams and Co., Inc., Indianapolis, Indiana 46268, 1980).

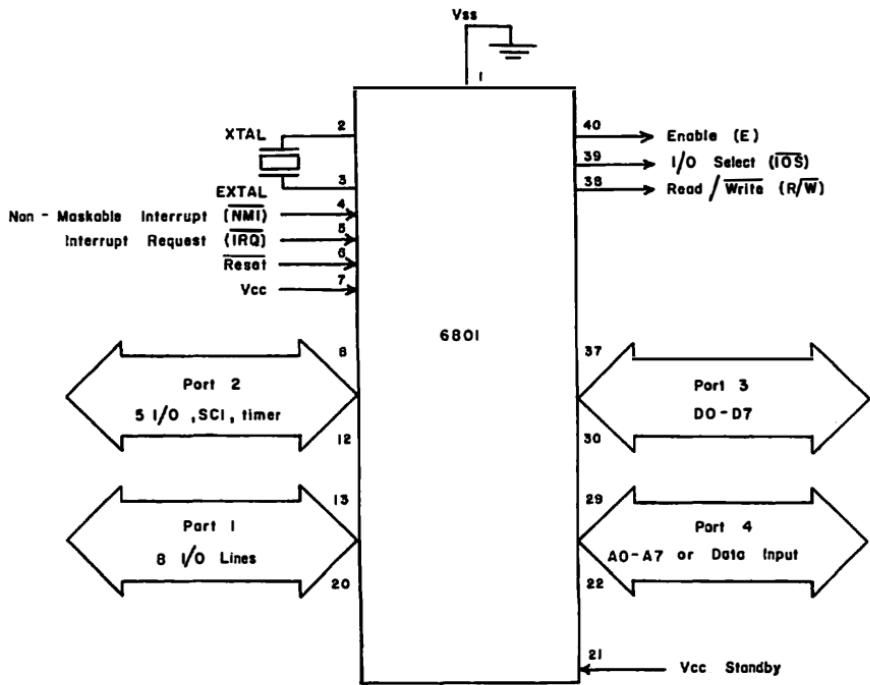


Fig. 3-7. 6801 expanded nonmultiplexed mode.

ports 3 and 4 assuming different functions. Therefore, our discussion will begin at pin 22 (port 4).

Port 4 (Pins 22–29)

In this mode, port 4 provides address bus lines A0–A7. Pin 29 is assigned to address line A0, pin 28 to address line A1, and so on, ending with pin 22 and address line A7. These eight address lines allow for the addressing of external devices using up to 256 unique addresses. In addition, if fewer address lines will satisfy a particular application, any number of the eight lines may be substituted for *input only* data lines, beginning with the most-significant bit (pin 22). This can be accomplished by writing 0's into the respective port 4 data direction register bits. More on this option later.

Port 3 (Pins 30–37)

To facilitate system expansion, port 3 allows access to the 6801's data bus when the chip is used in this mode. The eight lines of port 3 provide the external data bus signals D0 through D7, with pin 37 assigned to D0, pin 36 assigned to D1, and so on, ending

with pin 30 and D7. This is the only function that port 3 assumes in the expanded nonmultiplexed mode.

Read/Write (Pin 38)

When interfacing to external devices such as memories, a control line is needed to indicate when the processor is in a read cycle and when it is performing a write cycle. This is the function of the R/W line. It is used in the same way as the R/W line on the 6800 is utilized. A logic 1 on this line indicates that the 6801 is performing a read operation while a logic 0 indicates a write operation.

I/O Select (Pin 39)

When addressing external devices, you will be using address lines A0 through A7 to provide a 256 external address space. As you will see in Chapter 5, this space is assigned to addresses 0100 through 01FF in the 6801 expanded nonmultiplexed mode address map. The i/o select line (IOS) provides an active low address decode of the external addresses (0100–01FF) and can be used as a device or chip select line for external chip decoding. The i/o select line is only active when any of the addresses 0100 through 01FF are sensed on the internal address bus when the 6801 is configured in the expanded nonmultiplexed mode.

Enable (Pin 40)

This pin is used to provide an external clock or synchronization pulse in the same way as it is used in the single-chip mode.

EXPANDED MULTIPLEXED MODE

The expanded multiplexed mode provides you with external access to the 6801 data bus, D0–D7, and the complete 16-bit address bus, A0–A15. The functional configuration and pin assignments are shown in Fig. 3-8. Note that port 4 provides the upper (high order) eight address lines, A8–A15, while port 3 provides the lower eight address lines, A0–A7. In addition, port 3 supplies the external data bus, D0–D7. In this mode, port 3 is *multiplexed* to provide both the low-order address byte and the data byte. In this way, the 6801 can be expanded to 64K addresses to be completely compatible with the 6800 chip family. Again, the functions of the first 21 pins are identical to that of the single-chip and expanded nonmultiplexed modes. Therefore, our discussion of the expanded multiplexed mode will begin at pin 22 (port 4).

Port 4 (Pins 22–29)

In the expanded multiplexed mode, port 4 supplies the high-order address byte, A8–A15. Pin 29 is assigned to address line A8, pin 28

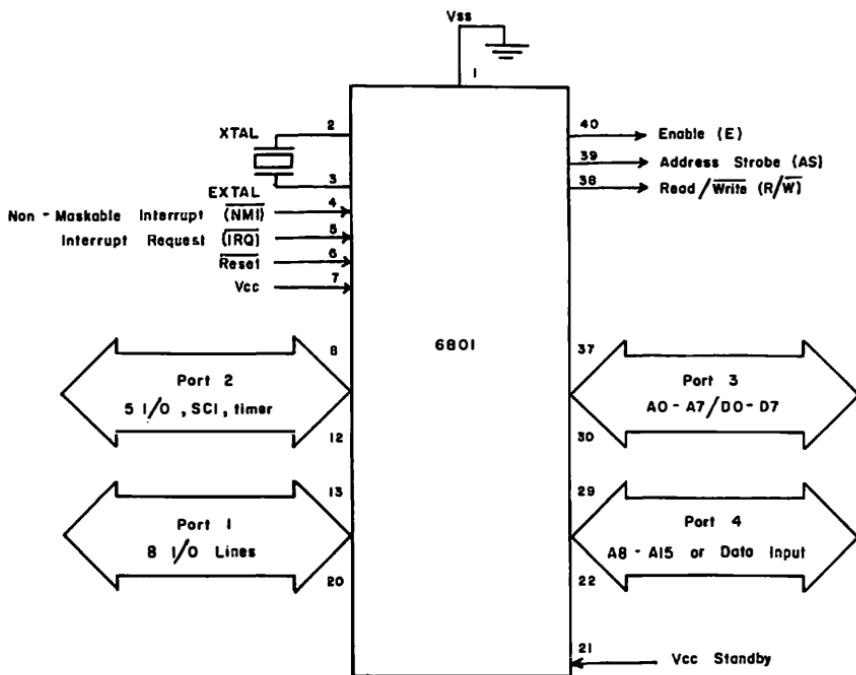


Fig 3-8. 6801 expanded multiplexed mode.

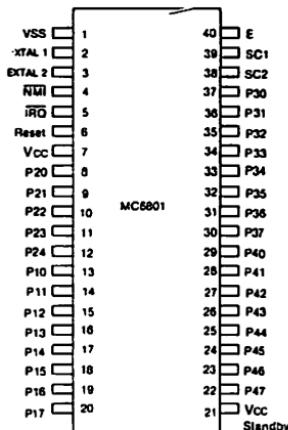
is assigned to address line A9, and so on, ending with pin 22 being assigned to address line A15. As with the expanded nonmultiplexed mode, if fewer address lines will satisfy the application, any number of the eight port 4 lines may be substituted as *input only* data lines, beginning with the most significant bit (pin 22) by writing zeros in the respective port line bit position in the port 4 data direction register.

Port 3 (Pins 30–37)

In this mode, port 3 is used to supply both the lower address byte, A0–A7, and the external data bus, D0–D7. Pin 37 is assigned to A0/D0, pin 36 is assigned to A1/D1, and so on, ending with pin 30 being assigned to A7/D7. These two functions are *multiplexed* and differentiated by an output called the *address strobe* (AS), pin 39. The external multiplexing procedure is discussed in Chapter 4.

Read/Write (Pin 38)

This pin provides the same R/W signal as it did in the expanded nonmultiplexed mode. A logic 1 will indicate that the 6801 is per-



MODE	PORT 1 Eight Lines	PORT 2 Five Lines	PORT 3 Eight Lines	PORT 4 Eight Lines	SC1	SC2
SINGLE CHIP	I/O	I/O	I/O	I/O	IS3(I)	OS3(O)
EXPANDED MUX	I/O	I/O	ADDRESS BUS (A0-A7) DATA BUS (D0-D7)	ADDRESS BUS* (AB-A15)	AS(I)	R/W(I)
EXPANDED NON-MUX	I/O	I/O	DATA BUS (D0-D7)	ADDRESS BUS* (A0-A7)	IOS(I)	R/W(I)

*These lines can be substituted for I/O (Input Only) starting with the most significant address line.

I = Input
 O = Output
 R/W = Read/Write

IS = Input Strobe
 SC = Strobe Control
 OS = Output Strobe
 AS = Address Strobe
 IOS = I/O Select

Fig. 3-9. 6801 mode and port summary.

forming a read operation, while a logic 0 indicates a write operation.

Address Strobe (Pin 39)

The address strobe (AS) is an output control signal that indicates the type of information at port 3. A logic 1 on the AS line indicates that an address is present at port 3, while a logic 0 indicates the presence of data at port 3. As you will see in Chapter 4, this strobe signal will be used by external latching to multiplex the external address and data buses.

Enable (Pin 40)

As in the other two modes, the E, or enable, pin provides an external clock pulse to synchronize data communication between the 6801 and its external devices.

A general-purpose pin assignment diagram is shown in Fig. 3-9. The corresponding table summarizes the various modes and port functions for the 6801.

REVIEW QUESTIONS

1. A 3.58-MHz color burst frequency crystal would provide an operating frequency of _____ MHz for the 6801.
2. How must the crystal pins of the 6801 be connected to drive it with an external clock source?

3. The *retainable R/W* memory is located at hex addresses _____ through _____.
4. Maximum current drain of the retainable R/W memory is _____ mA.
5. The reset interrupt vector is located at addresses _____ and _____.
6. The NMI interrupt vector is located at addresses _____ and _____.
7. List the interrupt priorities beginning with the highest priority.

8. The port 1 data direction register (DDR) is located at address _____.
9. A logic 1 placed in a DDR bit will cause that respective port line to be an _____ line.
10. P25 stands for port _____, bit _____.
11. Timer access is provided through port 2, bits _____ and _____.
12. Serial communications is provided through port 2, bits _____, _____ and _____.
13. Port _____ is always parallel i/o, regardless of the chip's mode.
14. Port _____ has an associated input and output strobe to provide handshaking in the single-chip mode.
15. In all modes, the _____ line is used to provide a synchronization pulse for data communication between the 6801 and its external devices.
16. The first _____ pins take on the same functions for all the operating modes.
17. In the expanded nonmultiplexed mode the external address lines are provided by _____ and the external data bus is provided by _____.
18. In the expanded multiplexed mode, port 4 supplies _____ and port 3 is _____.
19. What is the function of the R/W line in the expanded nonmultiplexed and expanded multiplexed modes?

20. What is the function of the address strobe (AS) in the expanded multiplexed mode?

ANSWERS

1. .895 (3.58/4)
2. The clock signal is applied to pin 3, with pin 2 grounded.
3. 0080 through 00BF
4. 8 mA
5. FFFE and FFFF
6. FFFC and FFFD
7. Reset, nonmaskable interrupt (NMI), software interrupt (SWI), interrupt request (IRQ)
8. 0000
9. output
10. port 2, bit 5
11. bits 0 and 1
12. bits 2, 3, and 4
13. port 1
14. port 3
15. enable (E)
16. 21
17. port 4
port 3
18. the high-order address byte
multiplexed to supply the data bus and low-order address byte
19. The R/ \overline{W} line is an output control line which tells external devices when the processor is in a read or write cycle. A logic 1 on this line indicates that the 6801 is performing a read operation, with a logic 0 indicating a write operation.
20. The address strobe (AS) is an output control signal that indicates the status of port 3 in the expanded multiplexed mode. A logic 1 on this line will indicate that an address is present at port 3, with a logic 0 indicating the presence of data at port 3. This strobe will be used by an external latch to multiplex the external address and data buses.

CHAPTER 4

Mode Selection and Port Usage

INTRODUCTION

You should now have a basic understanding of the three fundamental operating modes and corresponding pin assignments of the 6801. In this chapter you will see that there are really *eight* unique 6801 operating modes. The additional five modes are simply special cases of the three fundamental modes previously discussed. They will provide you with various internal R/W memory and ROM options, and will also allow you to test the internal R/W memory and ROM functions. Each of the eight modes is selected by *external* logic signals applied to port 2 during the **RESET** operation. You *cannot* select an operating mode with internal software. As you will see in this chapter, the operating mode actually controls the configuration of 18 of the 6801's 40 pins. Also, the particular mode of operation defines the memory map, location of interrupt vectors, type of external bus, and the available on-chip resources. The mode selection process is discussed in detail in this chapter. Also, since port 2 is used for other i/o purposes, you must isolate peripheral devices from this port during the mode selection process. In this chapter you will be provided with a circuit that will provide the proper port 2 isolation during mode selection.

You will learn how to configure and access all four of the 6801's i/o ports. Each port, and its associated data direction register, is assigned a unique address in the 6801 memory map. Of special interest will be the uses of port 3 in the single-chip mode. Recall that in this mode, port 3 has two associated control lines which may be used to provide handshaking with an external i/o device. These lines are similar to the control lines associated with the 6820/21

PIA ports, and their use is controlled by a control register similar to that of the PIA control registers. Use of all the 6801's ports to provide parallel data i/o is covered in this chapter, with specific examples given for i/o handshaking via port 3. The use of port 2 to provide timer and serial communication functions is covered in Chapters 6 and 7, respectively.

OBJECTIVES

At the end of this chapter, you will be able to do the following:

- Describe the additional five operating modes and how they relate to the three fundamental modes.
- Explain how to select any one of the eight 6801 operating modes.
- Understand what is meant by peripheral isolation.
- Design a circuit to provide peripheral isolation during the mode selection process.
- Understand how to configure and access each of the four 6801 i/o ports for parallel data i/o.
- Describe how port 3 is used along with its associated control lines and control register to provide complete handshaking for an external i/o device.
- Write a program to provide complete handshaking with an input device via port 3.
- Write a program to provide complete handshaking with an output device via port 3.

6801 MODE SELECTION AND ADDITIONAL OPERATING MODES

The various 6801 operating modes are selected via port 2 during the **RESET** operation. The particular mode of operation will be determined by the logic states present at pins 8, 9, and 10 on the chip during the logic zero **RESET** pulse. These three pins represent the first three bits of port 2; P20, P21, and P22, respectively. You must supply the required logic states on these lines with external hardware. The port 2 data i/o register, located at address 0003, is shown in Fig. 4-1. Recall that port 2 provides five i/o lines and is also

address	0003	7	6	5	4	3	2	1	0
		PC2	PCI	PCO	P24 (I/O)	P23 (I/O)	P22 (I/O)	P21 (I/O)	P20 (I/O)

Fig. 4-1. Port 2 I/o register.

used to gain access to the timer and serial communications interface. For parallel i/o, P20–P24 are configured as either input or output lines by the port 2 data direction register located at address 0001. Timer access is provided via P20 and P21 and will be discussed in Chapter 6. Serial communication discussed in Chapter 7, is provided via P22, P23, and P24. And, as was just stated, mode selection is via P20, P21, and P22. So, as you can see, port 2 is used to provide many different functions. However, regardless of the parallel i/o, timer and serial communication demands on port 2, it *must* be used to select the proper 6801 chip mode prior to the system's operation. As stated earlier, you must hard-wire the required logic to P20, P21, and P22 (pins 8, 9, and 10) to obtain the desired operating mode. When the **RESET** operation occurs, the logic levels present on these lines will be transferred into the *program control* bits PC0, PC1, and PC2 (bits 5, 6, and 7) of the port 2 i/o register (Fig. 4-1). The logic required at port 2 to select each operational mode is shown in Table 4-1. Note that besides the three fundamental modes previously discussed, there are five additional modes that provide the 6801 with a total of eight unique operating

Table 4-1. 6801 Mode Selection

Mode	PC2	PC1	PC0	ROM	RAM	Interrupt Vectors	Bus Mode	Operating Mode
7	H	H	H	I	I	I	I	Single Chip
6	H	H	L	I	I	I	MUX(6)	Multiplexed/Partial Decode
5	H	L	H	I	I	I	NMUX(6)	Nonmultiplexed Partial Decode
4	H	L	L	I(2)	I(1)	I	I	Single Chip Test
3	L	H	H	E	E	E	MUX	Multiplexed/No RAM or ROM
2	L	H	L	E	I	E	MUX	Multiplexed/RAM
1	L	L	H	I	I	E	MUX	Multiplexed/RAM & ROM
0	L	L	L	I	I	I(3)	MUX	Multiplexed Test

LEGEND:

I – Internal
E – External

MUX – Multiplexed
NMUX – Nonmultiplexed

L – Logic "0"
H – Logic "1"

NOTES:

- (1) Internal RAM is addressed at \$XX80
- (2) Internal ROM is disabled
- (3) RESET vector is external for 2 cycles after RESET goes high
- (4) Addresses associated with ports 3 and 4 are considered external in modes 3 and 4
- (5) Addresses associated with port 3 are considered external in modes 5 and 6
- (6) Port 4 default is user data input; address output is optional by writing to port 4 data direction register

modes. The additional five modes are really special cases of the three fundamental modes. Of the eight modes available, you must select the one that best allocates the 6801's resources for your application. Note also from Table 4-1 that the modes are numbered 0 through 7. The three fundamental modes that have been previously discussed are numbered as follows: single-chip-mode 7, expanded multiplexed-mode 6 and expanded nonmultiplexed-mode 5. Now, let us discuss each of the additional five modes in relation to these three fundamental modes.

Modes 1, 2, and 3

These modes are simply variations of the expanded multiplexed mode (mode 6) to provide different interrupt, R/W memory, and ROM options. In mode 1, both the internal R/W memory and ROM are enabled. However, the interrupt vectors are provided externally, versus internally for mode 6. When in mode 1, the 6801 will input data from the external data bus for interrupt vector addresses FFF0 through FFFF. This will allow you to define your own vector table in the 6801 memory map.

In mode 2, the interrupt vectors are again provided by an external source, the internal R/W memory is enabled, and the internal ROM is disabled.

In mode 3, both the internal R/W memory and internal ROM functions are disabled. Again, the interrupt vectors are provided externally. Since the internal R/W memory is not used in this mode, the V_{cc} Standby pin (pin 21) should be tied to ground or V_{cc} .

One other point, recall that in the expanded multiplexed mode, port 4 is used to provide the upper address byte. If all of these lines are not needed for address information, you can use them for *data input* (outputs are not allowed), beginning with the most significant address line. To use these lines as input data lines, you must write 0's into the respective port 4 data direction register bits located at address 0005. Address outputs are provided by writing 1's into the proper port 4 data direction register. This data input option in the expanded multiplexed mode is *only* provided for mode 6 and is not available for modes 1, 2, or 3. However, the same option is available in the expanded nonmultiplexed mode (mode 5). In systems that require some external memory addressing, but not a totally "filled" memory space, this tradeoff allows some of the port 4 lines to be used as input port lines, and the remaining lines to be used for external memory addressing.

Modes 0 and 4

These two remaining modes can be used for testing purposes. Mode 0 is a multiplexed mode which may be used to test the inter-

nal ROM. Mode 4 is a single-chip mode that may be used to test ports 3 and 4 as i/o ports.

The multiplexed test mode (mode 0) is used by the manufacturer to test the on-chip ROM package and is not intended for general use by the customer. This mode will allow the manufacturer to verify that a proper mask-bit pattern has been achieved after the on-chip ROM has been mask-programmed. However, as you will see in Chapter 5, you will use mode 0 to program the 68701 EPROM.

The single-chip test mode (mode 4) will allow you to exercise the interrupts and test the i/o port functions. More information about both of these modes is provided in Chapter 5.

An overview of the eight 6801 operating modes is provided in Table 4-2. A circuit which will provide a simple reset function and mode selection is shown in Fig. 4-2. With this circuit, any of the eight operating modes can be selected by setting the switches for

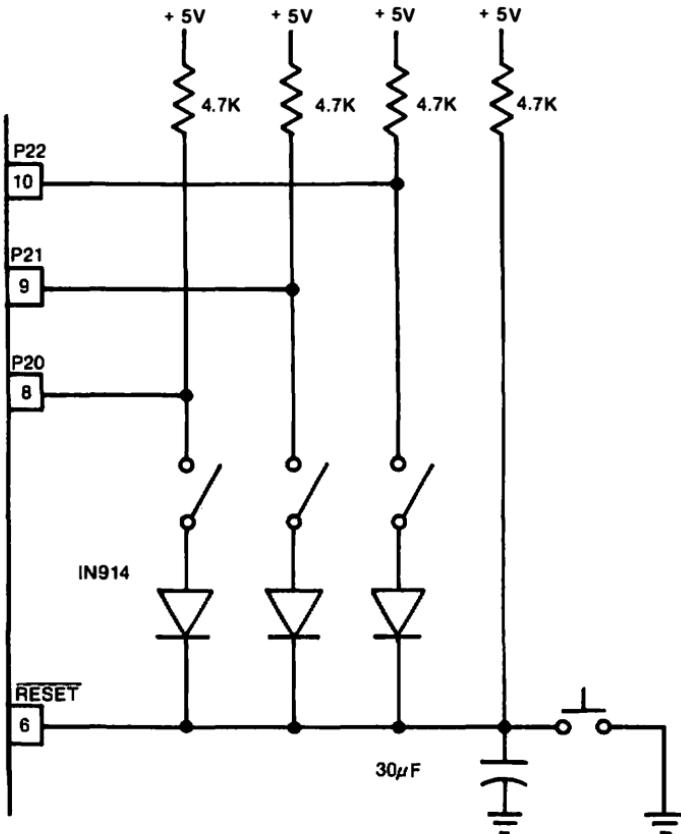


Fig. 4-2. 6801 mode-selection circuit.

the required logic pattern (reference Table 4-1) and depressing the **RESET** pushbutton. The PC0, PC1, and PC2 bits (bits 5, 6, and 7) of the port 2 data i/o register are read-only and cannot be changed with software. Therefore, the modes are *only* hardware-selectable. However, there is one exception to this rule. Mode 4 (single-chip test) may be changed to mode 5 (expanded nonmultiplexed) by writing a logic "1" into bit 5 (PC0) of the port 2 data i/o register located at address 0003.

PERIPHERAL ISOLATION DURING MODE SELECTION

Now, recall from the discussion at the beginning of this chapter that port 2 is also used to provide five lines of parallel i/o and to gain access to the internal timer and serial communications inter-

Table 4-2. An Overview of the 6801 Operating Modes

COMMON TO ALL MODES:

- Reserved register area
- I/O port 1 operation
- I/O port 2 operation
- Timer operation
- Serial I/o operation

SINGLE-CHIP-MODE 7

- 128 bytes of on-chip RAM; 2048 bytes of on-chip ROM
- Port 3 is parallel I/o handshaking port
- Port 4 is parallel I/o port

EXPANDED MEMORY SPACE/NONMULTIPLEXED BUS-MODE 5

- 128 bytes of on-chip RAM; 2048 bytes of on-chip ROM
- Port 3 is 8-bit data bus
- Port 4 is optional 8-bit address bus
- 256 bytes of external memory space
- External memory space select output (**IOS**)

EXPANDED MEMORY SPACE/MULTIPLEXED BUS-MODES 1, 2, 3, 6

- Port 3 is multiplexed address/data bus
- Port 4 is address bus
- 4 memory space options
 - (1) No Internal RAM or ROM (mode 3)
 - (2) Internal RAM (mode 2)
 - (3) Internal RAM and ROM (mode 1)
 - (4) Internal RAM, ROM with complete address bus (mode 6)

TEST-MODES 0 and 4

- Expanded test-mode 0
 - May be used to test Internal RAM and ROM
- Single-chip and nonmultiplexed test-mode 4
 - (1) May be changed to mode 5 without **RESET**
 - (2) May be used to test ports 3 and 4 as I/o ports

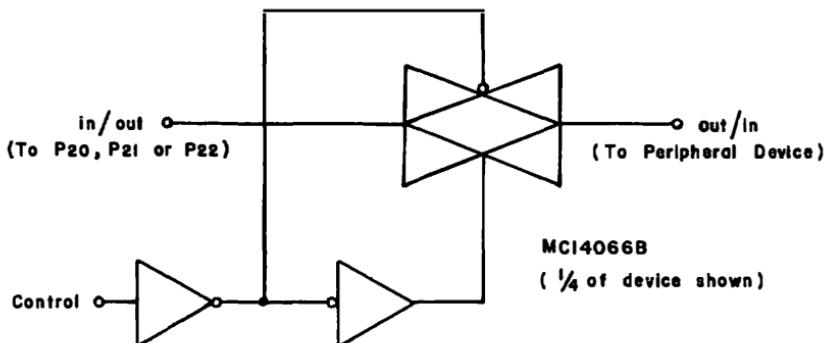


Fig. 4-3. MC14066B quad analog, switch/multiplexer.

face. This creates a problem during the mode selection process since isolation must be provided during **RESET** between the P20, P21, P22 mode selection logic and any peripheral devices which are interfaced to port 2. The required isolation may be achieved using the MC14066B quad analog, switch/multiplexer shown in Fig. 4-3 along with a 7414 inverting Schmitt trigger and D flip-flop. The MC14066B shown in Fig. 4-3 acts as a bidirectional three-state buffer which requires no external logic to determine the direction of information flow. One side of the buffer would connect to P20, P21, or P22 and the other side to the respective peripheral device. When the control line is low, the peripheral device is effectively isolated from port 2. With the control line high, the proper connection would be made between P20, P21, or P22 and the respective peripheral device. You will need to provide isolation during the mode selection process; therefore, the MC14066B control lines should be low during **RESET**. A coupler control circuit, which will assure proper signal coupling, is shown in Fig. 4-4. This circuit utilizes a 7414 inverting Schmitt trigger and a D flip-flop, and will provide the proper control signals for the MC14066B coupler. One of these circuits would be required to interface between the 6801 (**RESET**, Enable)

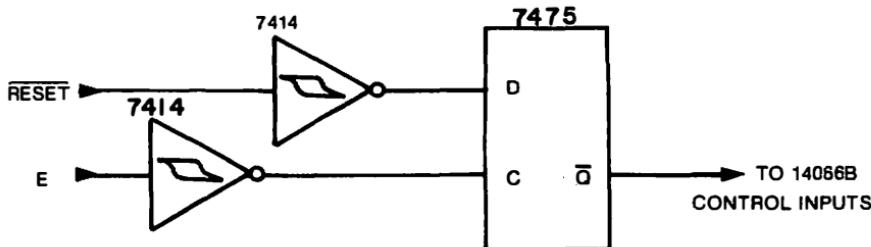


Fig. 4-4. MC14066B coupler control circuit.

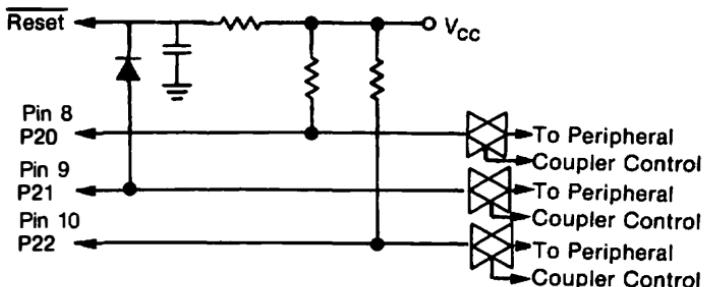


Fig. 4-5. Expanded nonmultiplexed mode (mode 5) selection circuit with MC14066B peripheral isolation.

and the MC14066B control lines. Note that the RESET level will be latched via the D flip-flop to the MC14066B control input lines. The 7414 Schmitt trigger provides a "snap" action to condition the relatively slow and bouncy RESET signal that would result from a hardware pushbutton RESET function.

A circuit which would provide selection of mode 5 (expanded nonmultiplexed) and peripheral isolation using the MC14066B is shown in Fig. 4-5. During RESET a low level is applied to P21 (pin 9) via the diode, and high levels are applied to P20 (pin 8) and P22 (pin 10) via the pull-up resistors. These pins are also connected to their respective peripheral devices via the MC14066B couplers, shown at the right side of the figure. The coupler control lines will connect to the coupler control circuit shown in Fig. 4-4. Also, each MC14066B contains four coupler circuits; therefore, only one MC14066B chip is required.

6801 PORT ACCESS AND CONTROL

The four 6801 i/o ports are accessed via software at the first eight addresses (0000-0007). Actually, the first twenty addresses (0000-0014) are reserved for special purpose registers and may not be assigned to external R/W memory, ROM or i/o. You will eventually become acquainted with all twenty special purpose registers in subsequent chapters, but for now we will concentrate on the registers that are assigned to the four i/o ports.

Each i/o port has an associated data direction register (DDR) and data i/o register. The memory map for these registers is shown in Table 4-3.

Each port can be programmed for input or output via its respective data direction register (DDR). When used for parallel i/o, a particular port line will be programmed for input if its corresponding DDR bit contains a logic 0 and output if its corresponding DDR

Table 4-3. Port and Data Direction Register Memory Map

Hex Address	Register
0000	Port 1 Data Direction
0001	Port 2 Data Direction
0002	Port 1 Data I/O
0003	Port 2 Data I/O
0004	Port 3 Data Direction
0005	Port 4 Data Direction
0006	Port 3 Data I/O
0007	Port 4 Data I/O

bit contains a logic 1. During the **RESET** operation, all lines are configured as input. The following are some general guidelines to be used when using the various 6801 ports for parallel i/o.

1. Port 2 DDR bits 5, 6, and 7 are not used since port 2 has only five i/o lines. Also, P20, P21, and P22 must always be connected high or low, since they provide the operating mode logic during the **RESET** operation. If P23 and P24 are unused, they can remain unconnected (floating).
2. When in the expanded *nonmultiplexed* mode (mode 5), port 3 becomes the external data bus regardless of the port 3 DDR bit values, and port 4 is configured as a data *input* port during the **RESET** operation. By subsequently writing 1's to the port 4 data direction register, you can provide any or all of the low order address byte lines, A0 to A7.
3. In the *expanded multiplexed* mode (modes 0, 1, 2, 3, and 6), port 3 is multiplexed to provide the external data bus and the lower order address lines (A0-A7), regardless of the port 3 DDR bit values. Also, port 4 is configured as the high order address lines (A8-A15) for modes 0, 1, 2, and 3. However, in mode 6 *only*, port 4 is configured as a data *input* port during the **RESET** operation. By subsequently writing 1's to the port 4 data direction register, you can provide any combination of address lines A8-A15.
4. When using port 2 in conjunction with the internal timer and/or the serial communications interface functions, the port 2 DDR bit values may or may not have an effect on the direction of data transfer. The use of the port 2 DDR with the timer and serial functions is discussed in Chapters 6 and 7.
5. To receive and transmit *parallel* data via *any* port, you will simply read and write to the data i/o register of the respective port. However, to receive and transmit *serial* data via port 2 requires that you read and write to other registers than the port 2 data i/o register. This will be discussed in detail in Chapter 7.

Now, let us take a closer look at port 3. Recall that in the single-chip mode, port 3 provides parallel i/o as programmed by its data direction register (DDR). There are also two control lines associated with this port which can be used to provide handshaking in the single-chip mode. They are the input strobe line ($\overline{IS3}$) at pin 39, and the output strobe line ($\overline{OS3}$) at pin 38. These lines are con-

address 000F	7	6	5	4	3	2	1	0
	IS3 Flag	IS3 Enable	X	OSS	Latch Enable	X	X	X

Fig. 4-6. Port 3 control/status register.

trolled by the i/o port 3 control/status register shown in Fig. 4-6. This is a special purpose register located at address 000F. Bits 0, 1, 2, and 5 of this register are not used and are permanently set to 1's. An explanation of bits 3, 4, 6, and 7 follows.

Latch Enable (Bit 3)

This bit provides the means for control of data input latching into port 3, and operates in conjunction with the input strobe ($\overline{IS3}$) at pin 39. When latch enable is *set*, the input data will be latched on the *falling* edge of the input strobe signal ($\overline{IS3}$). The latch enable bit is then cleared when the 6801 CPU *reads* the port 3 data i/o register at address 0006. It must be set once again to re-enable the latching function. This bit is also cleared during a **RESET** operation.

OSS—Output Strobe Select (Bit 4)

This bit is used in conjunction with the output strobe ($\overline{OS3}$) at pin 38. The 6801 will generate an output strobe ($\overline{OS3}$) at pin 38 when *either* a read or write operation to the port 3 data register at address 0006 is performed. Pin 38 is normally high and the strobe will appear as a high-to-low transition, remaining low for one clock cycle. During a handshaking operation, you will want the 6801 to strobe an output peripheral device when data is available for that device *or* to strobe an input peripheral device when data has been accepted from that device. The OSS bit will determine whether the output strobe ($\overline{OS3}$) should be generated by a write-to-port 3, or a read-from-port 3 operation. If this bit is cleared, the strobe will be generated by a *read*-from-port 3 operation. If set, the strobe will be generated by a *write*-to-port 3 operation. This bit is set and cleared by software. **RESET** will also clear this bit.

IS3 Enable—Input Strobe Enable (Bit 6)

This bit is used with the input strobe (IS3) at pin 39. When set, it will allow an interrupt request (IRQ) to be generated during an active input strobe at pin 39. For the interrupt to be acknowledged, the I-flag in the condition code register must be cleared. Once the interrupt has been acknowledged, the 6801 CPU will automatically jump to address FFF8:FFF9 for the IRQ service routine vector. When the IS3 Enable bit is cleared, an interrupt will not be generated during an active input strobe. This bit can be set and cleared by software and is cleared automatically during **RESET**.

IS3 Flag—Input Strobe Flag (Bit 7)

The input strobe flag is a *read-only* status bit which is set by the falling edge of the input strobe (IS3) at pin 39. The flag is used to indicate an active input strobe and an interrupt request (IRQ) may or may not be generated, depending upon the status of the IS3 enable bit (bit 6). In a typical i/o *polling* routine, you would read the port 3 control/status register at address 000F to determine if the input strobe flag bit (bit 7) is set. If set, an active input strobe has occurred and an i/o device is ready to accept data from port 3, or an i/o device has supplied data to port 3. Of course, since you are using this port for a particular application, *you know* whether it is being used as an input, or an output port. Therefore, you will then read from, or write to, the port 3 i/o data register at address 0006. The reading of the port 3 control/status register followed by a read *or* write of the port 3 i/o data register will clear the IS3 flag and thus prepare the 6801 for another active input strobe. The IS3 flag bit is also cleared during **RESET**.

The i/o port 3 control/status register bit functions are summarized in Table 4-4. Let us now see how the port 3 control lines are used in conjunction with the control/status register to perform both input and output handshaking.

Example 4-1: Input Handshaking via Port 3

In this example, you will see how to provide handshaking for an input device via port 3 and its associated control lines. Such a device could be an analog-to-digital (a/d) converter. Recall that the 6801 must be in the single-chip mode to use *port 3 for i/o*. This mode will normally be used for dedicated applications where a/d or d/a conversions are required. An input device such as an a/d converter, is shown interfaced to the 6801 in Fig. 4-7. It is assumed that the 6801 has already been configured for the single-chip mode. When the input device has data available, it will strobe the 6801 at pin 39 (IS3). The high-to-low transition of the input strobe will

Table 4-4. I/O Port 3 Control/Status Register Bit Summary

Register Bit	Set	Cleared
Bit 0	Not Used	Not Used
Bit 1	Not Used	Not Used
Bit 2	Not Used	Not Used
Bit 3 (Latch Enable)	By software: Allows input data to be latched into the port 3 I/O data register with a falling edge of the input strobe (IS3) at pin 39	By reading the port 3 I/O data register at address 0006 or by RESET
Bit 4 (OSS)	By software: Allows an output strobe (OS3) to be generated at pin 38 by a <i>write</i> to port 3 operation	By software: Allows an output strobe (OS3) to be generated at pin 38 by a <i>read</i> to port 3 operation
Bit 5	Not Used	Not Used
Bit 6 (IS3 Enable)	By software: Enables IRQ interrupt to be generated by an active (falling edge) input strobe (IS3) at pin 39	By software or RESET: Masks-out the IRQ interrupt for the input strobe (IS3) function
Bit 7 (IS3 Flag)	By the falling edge of the input strobe (IS3) at pin 39	By reading the I/O port 3 control/status register at address 000F followed by a read or write of the port 3 I/O data register at address 0006 or by RESET

cause an interrupt request (IRQ) to be generated to the 6801 CPU. When the interrupt is acknowledged, the interrupt service routine will read the data available at port 3 and thus cause an output strobe to be generated to the input device. The output strobe tells the input device that the available data has been accepted and the 6801 is ready to accept another data word, thus the handshake is complete.

The required bit configuration of the port 3 control/status register is shown in Fig. 4-8. Note that the latch enable bit (bit 3) is set such that the input data will be latched to the port 3 data register. The output strobe select (OSS) bit is cleared such that an output strobe will be generated when the 6801 performs a *read* operation on port 3. The input strobe enable (IS3 Enable) bit is set such that an interrupt request will be generated when an input strobe is provided and, finally, the input strobe flag (IS3 Flag) bit is cleared. The flag will set when an input strobe is detected.

The following is a program which will initialize the 6801 for this operation and read the input data at port 3.

Main Program

Hex Address	Hex Contents	Mnemonic/Contents	Operation
RESET — Clears IS3-Flag bit			
0080	7F	CLR \$\$	
0081	00	00	
0082	04	04	Clears port 3 DDR for Input only
0083	86	LDAA #	
0084	48	48	
0085	97	STAA \$	
0086	0F	0F	Configures port 3 control/status register per Fig. 4-8
0087	0E	CLI	
0088	3E	WAI	Clear I-Flag Wait for Interrupt
0089	20	BRA	
008A	FD	FD	Branch back to WAI
Interrupt Service Routine			
FA00	96	LDAA \$	
FA01	0F	0F	Read port 3 control/status register
FA02	96	LDAA \$	
FA03	08	08	Read port 3 (generates output strobe)
FA04	97	STAA \$	
FA05	A0	A0	Store Input data at address 00A0
FA06	96	LDAA \$	
FA07	08	08	
FA08	97	STAA \$	
FA09	0F	0F	Set latch enable
FA0A	3B	RTI	Return to main program

Prior to the main program, the **RESET** function is executed. This will clear all the functional bits of the port 3 control/status register.

The main program begins with address 0080 since this is where the on-board R/W memory resides. The program will first clear the port 3 data direction register at address 0004. This will configure port 3 as an input port. Actually, this operation is *not* necessary if the program is to be executed immediately after **RESET**, since the **RESET** operation automatically clears the port 3 data direction register and thus configures port 3 for input. The port 3 control/status register is then configured per Fig. 4-8 by storing 48 hex to

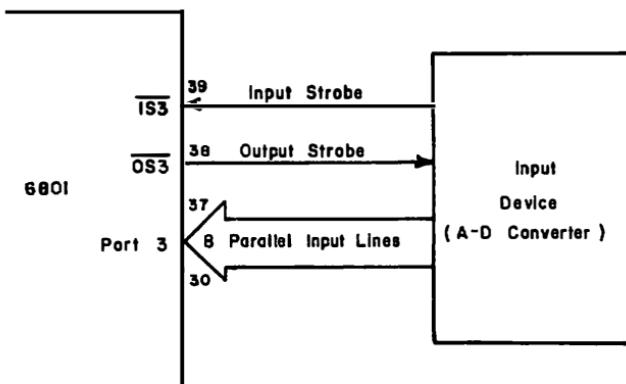


Fig. 4-7. Input device handshaking—single-chip mode.

IS3 Flag	IS3 Enable	OSS	Latch Enable				
0	I	X	0	I	X	X	X

Fig. 4-8. Port 3 control/status register—Input device handshaking.

address 000F. The I flag of the condition code register is cleared to allow acknowledgement of an interrupt request (IRQ).

The 6801 then *waits* for an interrupt to occur. When an input strobe ($\overline{IS3}$) is detected at pin 39, the IS3 flag will set and the CPU will automatically jump to address FFF8:FFF9 for the IRQ interrupt vector. We have chosen address FA00 as the interrupt vector for the interrupt request (IRQ). As you will see in Chapter 5, this is where the on-board ROM (EPROM) is located. Therefore, the interrupt service routine is located in the on-chip ROM (EPROM). The service routine reads the port 3 control/status register at address 000F then reads the port 3 data register at address 0006 to obtain the input data. This will accomplish three things: the IS3 flag is cleared, the latch enable bit is cleared, and the 6801 will generate an output strobe ($\overline{OS3}$) at pin 38. The input data is then stored in R/W memory location 00A0 for future access. Then, the latch enable bit is set to prepare port 3 for the next input data word. The RTI instruction will return program execution to the main program at address 0089. An unconditional branch has been inserted here such that the 6801 will return to the wait for interrupt (WAI) status until another input strobe is detected.

Example 4-2: Output Handshaking via Port 3

In this example, you will see how to provide handshaking for an output device such as a digital-to-analog converter. Fig. 4-9 shows such an output device interfaced to the 6801. When data is written to port 3, the 6801 will generate an output strobe to enable the output device. Once the output device has accepted the data, it will generate an input strobe to tell the 6801 that the available data has been accepted and it is ready to accept another data word,

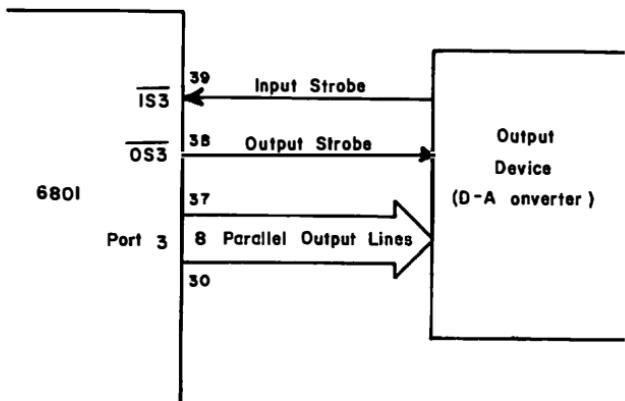


Fig. 4-9. Output device handshaking—single-chip mode.

thus completing the handshake. The input strobe will cause the input strobe flag (**IS3**) to set, which will, in turn, cause a new data word to be written to port 3. The required port 3 control/status register bit configuration is shown in Fig. 4-10. The difference between this and the input device configuration shown in Fig. 4-8 is

IS3 Flag	IS3 Enable	OSS Latch Enable					
0	0	X	I	0	X	X	X

Fig. 4-10. Port 3 control/status register—output device handshaking.

that the output strobe will be generated with a *write* to port 3 operation, and the latch enable bit is cleared to disable the input latching function. Also, an interrupt *will not* be used to detect the input strobe; therefore, the IS3 enable bit is cleared. Instead, you will use a shift routine to detect when the IS3 flag bit is set, indi-

cating an input strobe has occurred. The following output device subroutine will accomplish the output handshaking function.

Output Device Subroutine

Hex Address	Hex Contents	Mnemonic/Contents	Operation
RESET — Clears IS3-Flag bit			
0080	86	LDAA #	
0081	FF	FF	
0082	97	STAA \$	
0083	04	04	
0084	86	LDAA #	
0085	10	10	
0086	97	STAA \$	
0087	0F	0F	
0088	CE	LDX #	
0089	00	00	
008A	00	00	
008B	A6	LDAA X	
008C	B0	B0	
008D	97	STAA \$	
008E	06	06	
008F	96	LDAA \$	
0090	0F	0F	
0091	48	ASLA	
0092	24	BCC	
0093	FB	FB	
0094	08	INX	Increment Index register
0095	8C	CPX #	
0096	00	00	
0097	BA	BA	
0098	26	BNE	
0099	F1	F1	
00A0	3E	WAI	Has last data word been transmitted? If not, branch to address 008B. If so, stop.

This subroutine will transmit the data from ten consecutive memory locations (00B0–00B9) to an output device via port 3. Each time a data word is transmitted, an output strobe will be generated. After the output strobe has been generated, the program will wait until an input strobe is detected (completing the handshake) before transmitting the next data word. Once all ten data words have been transmitted, the program will halt. Note that the subroutine uses indexed addressing to transmit the contents of the ten consecutive memory locations of (00B0–00B9).

When using port 3 for i/o, the 6801 *must* be in the single-chip operating mode since port 3 becomes the external data bus when the 6801 is in the nonmultiplexed mode, and port 3 is *multiplexed* between the external data bus and lower address byte when the 6801 is in one of the multiplexed modes. In Chapter 8, some real a/d, d/a products are interfaced to the 6801 via port 3.

Now, let us take a closer look at the function of port 3 in the multiplexed mode. Recall from previous discussions that port 3 becomes both the external data bus ($D_0 - D_7$) *and* the lower byte of the address bus ($A_0 - A_7$). Therefore, the 6801 must provide an external signal to allow external circuits to distinguish between address bus and data bus signals that are present at port 3. Recall that this signal is the address strobe (AS) available at pin 39. A

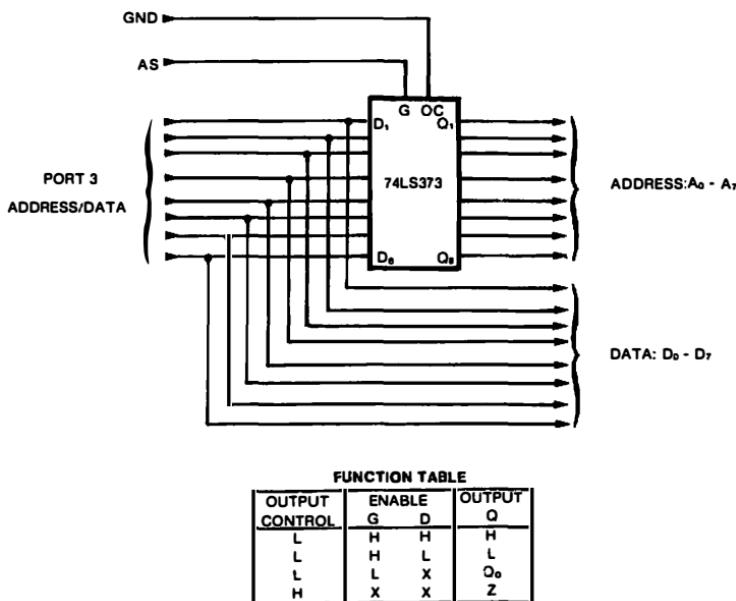


Fig. 4-11. Port 3 address/data latching.

high-level pulse at this pin indicates that an address is present on port 3. The address strobe will allow you to externally separate the address and data functions of the port. You will use this signal to latch the lower address byte ($A_0 - A_7$) as shown in Fig. 4-11. The address strobe (AS) will signal the latch to acquire and hold the address information. In this way, the address information that is latched may be used by memories and external devices, while the port 3 lines are then used as bidirectional data bus lines.

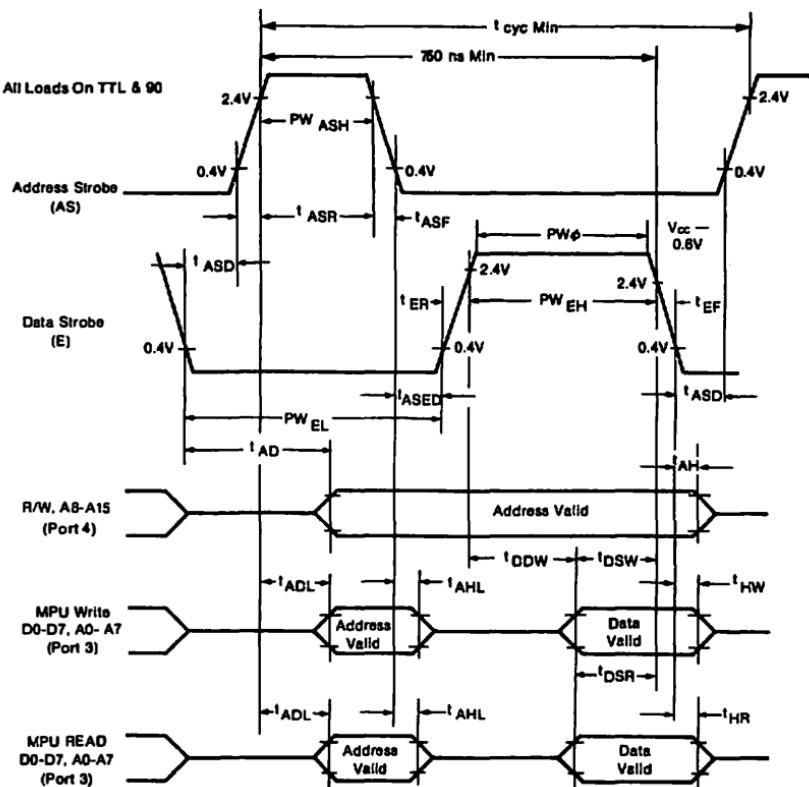


Fig. 4-12. Port 3 multiplexed bus timing.

Data bus functions are active during the E pulse. The signal timing requirements are shown in Fig. 4-12. A 74LS373 transparent octal D-type latch is used to provide address latching in Fig. 4-11. The address strobe (AS) is connected to enable line G, with the 74LS373 output control line (OC) connected to ground. When an address byte is output to port 3 by the 6801, a high-level address strobe (AS) pulse is generated. This enables the 74LS373, which latches the address and transfers it to the external address bus. A particular address byte will remain at the latch output until a new address is output at port 3 by the 6801. When data is present at port 3, the address strobe signal will go low. The 74LS373 does not latch this information, so the data bus information is only present on the external data bus and it does not interfere with the external address bus.

You should now be familiar with the use of the 6801 ports in providing up to 29 parallel i/o lines in the single-chip mode, and up to 64K bytes of external addressing in the expanded multiplexed

mode. The only port related functions to be discussed are the timing and serial communications functions. These items deserve independent discussions and are covered in Chapters 6 and 7.

REVIEW QUESTIONS

1. The eight 6801 operating modes are selected via port _____ with external logic applied to pins _____, _____, and _____.
2. When **RESET** occurs, the logic applied to the above pins is transferred into the _____ bits of the port 2 data i/o register.
3. The mode number designations which represent the three fundamental modes are:

4. The difference between expanded multiplexed mode 6 and expanded multiplexed mode 1 is:

5. The internal ROM/EPROM is disabled in mode _____.
6. The internal R/W memory and ROM functions are disabled in mode _____.

7. The data input option via port 4 is available only in modes _____, _____, and _____.
8. The mode used for testing the internal mask-programmed ROM is mode _____.

9. The mode used to test ports 3 and 4 as i/o ports is mode _____.
10. Why is peripheral isolation required for port 2 during the **RESET** operation?

11. List all the port data direction registers and data i/o register address assignments.

12. To configure a port bit for output, you would write a logic _____ into its corresponding DDR bit.
13. The two control lines associated with port 3 in the single-chip mode are the _____ and _____ control lines.

14. The port 3 control/status register is located at address _____.
15. List the four functional bits of the port 3 control/status register and their bit positions within that register.
16. To generate an output strobe with a write to port 3 operation, the OSS bit must be _____.
17. To generate an interrupt request with an active input strobe, the IS3 E bit must be _____.
18. An active input strobe is defined as a _____ transition at pin _____.
19. The input strobe flag (IS3 F) is cleared by:
20. How would the port 3 control/status register be configured to provide complete input device handshaking via port 3 using interrupts?

ANSWERS

1. port 2 at pins 8, 9, and 10.
2. program control bits PC0, PC1, and PC2 (bits 5, 6, and 7)
3. mode 7—single chip
mode 6—expanded multiplexed
mode 5—expanded nonmultiplexed
4. In mode 6, the interrupt vectors are provided from internal ROM at address FFF0 through FFFF. In mode 1, the interrupt vectors are provided externally via the external data bus for these addresses.
5. mode 2
6. mode 3
7. modes 5, 6, and 7
8. mode 0
9. mode 4
10. Because port 2 is used for mode selection during RESET and, in many applications, is also interfaced to peripheral devices for data i/o.
11. Reference Table 4-3.
12. logic “1”

13. Input strobe ($\overline{IS3}$) at pin 39, and output strobe ($\overline{OS3}$) at pin 38
14. 000F
15. latch enable (LTCH)—bit 3
output strobe select (OSS)—bit 4
input strobe enable (IS3 E)—bit 6
input strobe flag (IS3 F)—bit 7
16. set (logic 1)
17. set (logic 1)
18. high-to-low, at pin 39
19. **RESET** or by reading the port 3 control/status register at address 000F, followed by a read or write of the port 3 i/o data register at address 0006.
20. See Fig. 4-8.

CHAPTER 5

Using the 6801/68701/6803 On-Chip Memory

INTRODUCTION

In Chapter 4 you saw that there are eight unique 6801 operating modes. You became familiar with the procedure required to select any one of these modes to best allocate the 6801's resources to your application. In this chapter, we will discuss the use of the 6801 on-chip ROM and R/W memory. Therefore, we will begin the chapter with a discussion of the 6801 memory map. You will see that there are eight unique memory maps, one for each of the eight 6801 operating modes. In order for you to obtain maximum efficiency in using the 6801, it is extremely important that you understand the differences between the various operating mode memory maps as presented in this chapter. In the discussion of the memory maps, you will see that one thing is common to all the maps. This is the assignment of addresses 0000 through 001F for special purpose register functions. These register functions will be defined in this chapter and discussed in subsequent chapters when appropriate. Following a discussion of the 6801 memory maps, you will learn how to use the on-chip R/W memory and mask-programmed ROM.

The EPROM version of the 6801, the 68701, will be discussed in detail in this chapter. As stated earlier, the 68701 contains 2K bytes of EPROM in place of the 6801's mask-programmed ROM. In this chapter you will learn how to program the 68701 EPROM using a special external ROM monitor called PRObug, which Motorola has developed specifically to facilitate the programming of the 68701 EPROM. You will also learn how to program the 68701 EPROM

without the use of PRObug, if you desire to supply your own external monitor package.

Finally, a detailed discussion of the 6803 and 6803NR will be provided in this chapter. Recall that the 6803 is a no-ROM version of the 6801, while the 6803NR is a no-ROM *and* no-R/W memory version of the 6801.

OBJECTIVES

At the end of this chapter you will be able to do the following:

- Describe the eight different memory maps available to the 6801.
- Define the 6801 special purpose register functions.
- Understand the purpose and function of the R/W Memory/EPROM Control Register.
- Describe the steps necessary for proper information retention in the on-chip R/W memory during a power down (loss of V_{cc}) situation.
- Understand how to use the on-chip R/W memory and mask-programmed ROM.
- Describe the capabilities of the standard on-chip ROM package, LILbug.
- Explain the function of the RESET pin (pin 6) during the programming of the 68701 EPROM.
- Explain how to configure a 68701 EPROM programming system.
- Program your 68701 EPROM using PRObug.
- Program your 68701 EPROM using your own 68701 EPROM programming monitor package.
- Understand the 6803 and 6803NR.
- Describe the differences between the 6801, 68701, 6803, and 6803NR.

6801 MEMORY MAP

Before we begin a discussion of how to use the 6801 on-chip memory, we will examine the 6801 memory maps. As you are now aware, the 6801 has eight different operating modes which are user-selectable to provide the best possible allocation of the 6801's resources for your application. The 6801 will provide up to 64K address bytes for program and/or data storage. Also, since the 6801 uses memory mapped i/o, memory allocation for i/o devices is included in this space. As you will see shortly, each of the eight operating modes has a unique memory map. The only common memory allocations

to all eight modes are the first 32 bytes of memory, located from hex address 0000 through 001F. These addresses are assigned to a group of special purpose registers which control the use of the various 6801 resources. The special purpose register functions are listed in Table 5-1. You have already been acquainted with the first eight registers located at addresses 0000 through 0007. These are associated with the four i/o ports as discussed in Chapter 4. You have also been acquainted with the port 3 control/status register located at address 000F. The remaining registers are for control and use of the on-chip timer and serial communication func-

Table 5-1. Special Purpose Register Functions

Hex Address	Register
00	Data Direction 1
01	Data Direction 2
02	I/O Port 1
03	I/O Port 2
04	Data Direction 3
05	Data Direction 4
06	I/O Port 3
07	I/O Port 4
08	TCSR
09	Counter High Byte
0A	Counter Low Byte
0B	Output Compare High Byte
0C	Output Compare Low Byte
0D	Input Capture High Byte
0E	Input Capture Low Byte
0F	I/O Port 3 C/S Register
10	Serial Rate and Mode Register
11	Serial Control and Status Register
12	Serial Receiver Data Register
13	Serial Transmit Data Register
14	RAM/EPROM Control Register
15-1F Reserved	

tions of the 6801. Each of these registers will be discussed at the appropriate time in subsequent chapters. However, note from Table 5-1 that the usable special purpose registers comprise addresses 0000 through 0014. Addresses 0015 through 001F are reserved and not available to the user at this time.

A complete memory map for each of the 6801 operating modes is shown in Figs. 5-1 and 5-2. Memory maps for modes 0 through 3 are shown in Fig. 5-1, with modes 4 through 7 shown in Fig. 5-2. The following items are worth noting from, and in addition to the figures shown.

Fig. 5-1. 6801/68701 memory map (modes 0-3).

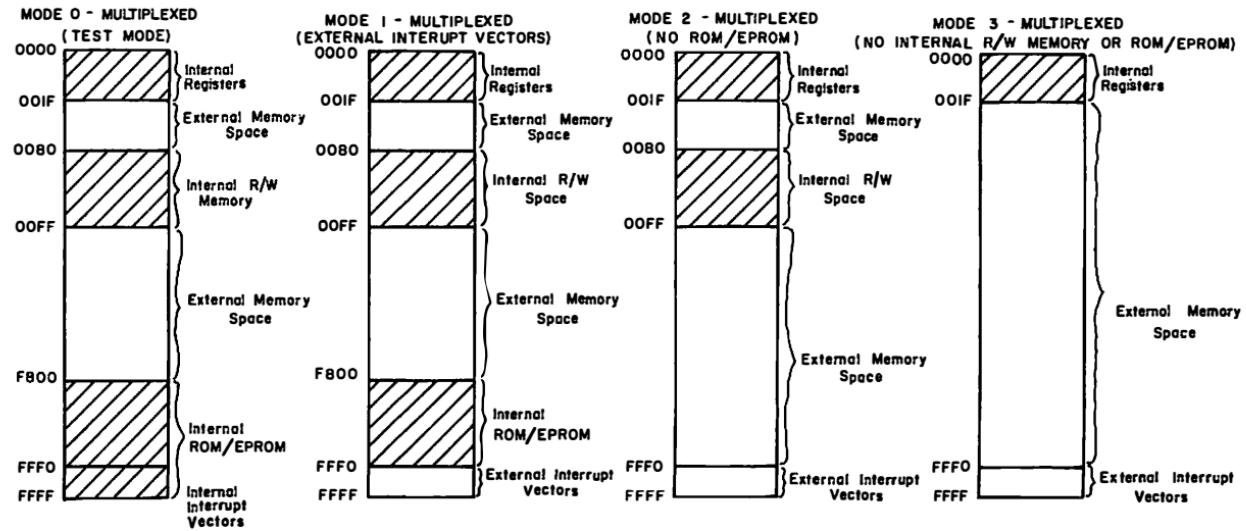


Fig. 5-2. 6801/68701 memory map (modes 4-7).

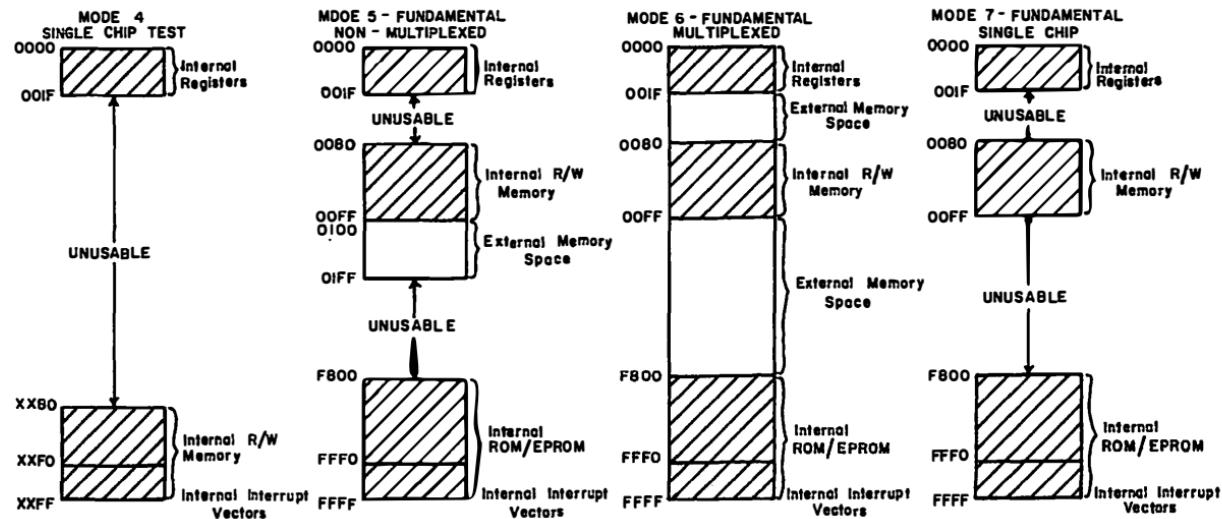


Figure 5-1

With modes 1, 2, and 3, the interrupt vectors are supplied externally. When operating in these modes, the 6801 will input data from the external data bus for interrupt vectors FFF0 through FFFF. Therefore, internal ROM addresses FFF0 through FFFF are not usable. In all other modes, the interrupt vectors are supplied from internal ROM at these addresses.

In mode 0 (multiplexed test), the reset interrupt vector at addresses FF_{FE}:FF_{FF} is considered *external* if accessed within two MPU cycles after a positive edge of RESET. At all other times, the vector is internal. In this way, you may examine the interrupt vectors in internal ROM using an external RESET vector. This will facilitate the manufacturer's testing of the internal mask-programmed ROM.

In the first four modes (mode 0 through mode 3), addresses 0004, 0005, 0006, 0007, and 000F *may* be used externally. The internal special purpose register functions at these addresses are disabled in these four modes.

Figure 5-2

In mode 4 (single-chip test), the internal ROM is disabled and the internal R/W memory will appear at address XX80 through XXF0. In this mode, address lines A8 through A15 are treated as "don't cares" to decode the internal R/W memory. Therefore, the internal R/W memory appears at the upper memory addresses allowing the interrupt vectors to be user defined and under software control. This will allow you to exercise the various interrupts and test the port functions. A test program must first be entered in the internal R/W memory using modes 0, 1, 2, or 6. If the 6801 is reset and subsequently configured into mode 4, execution will begin at XXFE:XXFF. A vector here could point to the beginning of your test program. For example, vectors could be defined to call an interrupt service routine within the on-chip R/W memory which would exercise (test) port 3 by reading and writing information to and from the port. Similar tests can be made on the timer and serial i/o functions of port 2 using the timer and serial communications interrupts which will be presented in Chapters 6 and 7.

Mode 4 (single-chip test) may be changed to mode 5 (expanded nonmultiplexed) without RESET. This can be done by *writing* a "1" into bit 5 (PCO) of the port 2 data i/o register located at address 0003. This is the *only* exception to the rule that all modes be configured by external hardware during RESET.

In modes 5 and 6 (expanded nonmultiplexed and expanded multiplexed), port 3 is used for data and/or address and is not available

for parallel i/o. Therefore, the special purpose registers associated with port 3 i/o are not needed and may be used for external addressing. These registers are located at addresses 0004, 0006, and 000F.

In modes 5 and 6 (expanded nonmultiplexed and expanded multiplexed) port 4 will be configured for input and will not provide address lines until the port 4 data direction register located at address 0005 has been written with 1's in the appropriate bits.

In order to best allocate the 6801's resources to your application, it is important that you understand the differences between the various operating mode memory maps as just discussed. In the next two sections, you will learn how to use the internal R/W memory and different ROM options available to these devices.

USING THE ON-CHIP R/W MEMORY

Now that you are familiar with the 6801 memory maps, you are ready to examine the use of the internal R/W memory. Recall that for all modes except modes 3 and 4, the 128-byte internal R/W memory is located at address 0080 through 00FF, with the first 64 bytes (0080-00BF) retainable using the V_{cc} Standby function discussed previously. Also, as you are now aware, addresses 0000 through 001F are reserved for special purpose status and control registers. These registers include the data direction registers and i/o registers for the four ports, along with a series of registers associated with the timer and serial communications interface. Each of these will be discussed at the appropriate time; however, let us now turn our attention to the R/W Memory/EPROM control register located at address 0014. The main function of this register is to enable and disable the internal 128 bytes of R/W memory and to control the programming of the 68701 EPROM. The register bit functions are shown in Fig. 5-3. Bits two through five are not used. The internal R/W memory enabling function is provided by bit six, RAM (R/W) Enable (RAME). A logic 1 in this bit position will enable the on-chip R/W memory, while a logic 0 will disable the internal R/W memory. This bit is automatically set to a logic 1 by the **RESET** function such that the on-chip R/W memory is enabled. However, if you desire to access external memory or i/o devices

Address 0014	7	6	5	4	3	2	1	0
	STANDBY BIT (SB)	RAME	X	X	X	X	\overline{PGE}	\overline{PLE}

Fig. 5-3. R/W Memory/EPROM control register.

at addresses assigned to the on-board R/W memory (0080-00FF), you may do so by clearing the RAME bit through program control.

A second function of the R/W Memory/EPROM control register is to provide a status indication of the V_{cc} Standby voltage level. This monitoring is provided by the STANDBY PWR bit (SB-bit 7) of the control register. To monitor the standby voltage level, you will normally set the STANDBY PWR bit. Then, when the standby voltage decreases below V_{SBBL} (4.0V), the STANDBY PWR bit will automatically clear, providing an indication of a low-voltage situation. By periodically checking this bit status, you can determine when the standby power source needs recharging. Upon detecting a low-standby power condition, a simple LED output indication could be provided via any of the i/o port lines to alert the operator of the condition. Therefore, the integrity of the data retained in the on-chip R/W memory can always be assured. The STANDBY PWR bit may be set or cleared by *software*, but is *not* affected by RESET. Power is supplied to the above two bits by V_{cc} Standby.

You will also use the R/W Memory/EPROM control register to retain information in the first 64 bytes of on-chip R/W memory during a power down or standby situation. The following procedure is necessary for proper information retention:

1. Write a zero into the RAM (R/W) Enable bit (bit 6) of the R/W Memory/EPROM control register located at address 0014. This will disable the entire on-chip R/W memory for protection purposes.
2. Write a one into the STANDBY PWR bit (bit 7) to provide monitoring of the V_{cc} Standby voltage level.
3. Keep V_{cc} Standby greater than 4.0 V.

These requirements can be met by writing 80 (hex) to address 0014 and maintaining V_{cc} Standby between 4.0 V and 5.25 V dc.

The remaining two usable bits, P_{LE} and P_{GE} (bits 0 and 1) are used for programming the 68701 EPROM. The standard 6801 *does not* use P_{LE} and P_{GE}. The use of these bits in programming the 68701 EPROM is discussed subsequently.

Once the internal R/W memory is enabled, it is used just like any R/W memory by writing and reading data to and from memory locations 0080 through 00FF. For dedicated applications using the single-chip mode, the internal R/W memory will normally be used as a "scratch pad" memory, with the main programs residing in the on-chip ROM/EPROM. In the expanded modes, additional R/W memory may be connected externally in memory space not being used on-chip. Refer to Figs. 5-1 and 5-2 for the external memory areas within each memory map.

USING THE ON-CHIP ROM

The on-chip 2K ROM function is available in modes 0, 1, 5, 6, and 7 at memory locations F800 through FFFF. With the 6801, the internal read-only memory function is masked-programmed, which means it must be programmed by the manufacturer. With the 68701, the read-only memory function is supplied by an EPROM and therefore user programmable. Therefore, we will divide the discussion of the read-only memory function into two separate topics. First, we will discuss the mask-programmed ROM of the 6801 followed by a discussion of the 68701 EPROM.

6801 Masked Programming ROM

With the 6801, you will develop your own ROM routines and define the interrupt vectors located at ROM addresses FFF0-FFFF. Once you are sure that all the "bugs" have been worked out of your routines, you will supply the manufacturer with the program you desire to have implemented into the on-chip 6801 ROM. Usually, your program may be forwarded to the manufacturer in several different formats. Contact the manufacturer prior to ordering your 6801's for the custom programming requirements and required data format. The 1's and 0's that make up the op codes in your program will be *masked* into the on-chip ROM during the last 6801 manufacturing step. This step is referred to as the *metallization* step. The manufacturer will actually produce a mask from your program to be used in the metallization process. One option you have with the mask-programmed ROM is to define the internal ROM memory map to start at an address other than the standard beginning address of F800. The internal ROM decoder may be mask-programmed on address lines A12 and A13 as zero's or one's such that you may elect to have your internal 2K bytes of ROM begin at address C800, D800, E800, or F800. These address lines may also be mask-programmed with "don't cares" such that the internal ROM will respond to any of the above beginning addresses. However, the standard beginning address for the on-chip ROM will be address F800, unless otherwise specified. The interrupt vectors FFF0 through FFFF will always be decoded to provide analogous vectors at the top of the resident ROM. In the expanded modes, additional ROM may be connected externally in memory space not being used on-chip. Refer to Figs. 5-1 and 5-2 for the external areas within each memory map. Of course, the disadvantage of the mask-programmed ROM is that once the ROM is produced, the programs cannot be modified. If modification is required, the existing 6801 must be discarded and replaced or the internal ROM disabled and replaced with external ROM. The latter may be accomplished by config-

ing the 6801 in mode 2. (Reference Fig. 5-1.) The advantage of the mask-programmed 6801 is its lower cost in large quantities versus the 68701 EPROM version. However, the 68701 EPROM can be reprogrammed.

Motorola has a *standard* 6801 ROM package available which contains a resident ROM monitor called *LILbug*. The LILbug version of the 6801 is designated as the 6801LL. The LILbug monitor is designed to be used to evaluate and debug programs under development. The LILbug monitor commands will allow you to do the following:

- Load a program in hexadecimal.
- Examine a loaded program.
- Examine data in any memory location and change data in any R/W memory location, including i/o.
- Examine and change data in the internal 6801 CPU R/W registers.
- Insert, display, and remove breakpoints in the user program.
- Freerun or step (trace) through the user program.
- Calculate the proper relative address offset for branch instructions.
- Dump the program.
- Display blocks of memory.
- Record or punch the program on magnetic or paper tape.

Refer to Appendix C for an explanation of how to use the LILbug monitor and each of the LILbug commands.

THE 68701

The 68701 has all the same features of the 6801 except that the 6801's 2K byte mask-programmed ROM is replaced with a 2K byte EPROM at addresses F800 through FFFF. However, the EPROM may *not* be relocated with a mask option as the 6801 ROM can. Therefore, the address of EPROM is fixed at F800 through FFFF. Also, the on-chip R/W memory is fixed at addresses 0080 through 0OFF. The RESET pin (pin 6) of the 68701 is used to perform two functions. First, it is used in the same way as the 6801 RESET, to reset the microcomputer. Second, it is used to supply the EPROM programming voltage (V_{pp}). This is a 20-volt level which is applied *only* when the EPROM is being programmed. The maximum programming current is 30 milliamperes.

The use of the 68701 EPROM and its associated memory maps for the various operating modes are identical to that of the 6801 mask-programmed ROM previously discussed, except for mode 0, the multiplexed test mode. In this mode, the interrupt vector mem-

ory map is changed from FFFF through FFFF to BFFF through BFFF. This will allow an external monitor program to be executed from the RESET interrupt vector at address BFFE:BFFF and facilitate programming of the 68701 EPROM. As you will see shortly, the 68701 uses mode 0 for programming the on-chip EPROM. All the remaining 6801 features discussed in this text will apply directly to the 68701.

The 68701 can be easily identified from the 6801 by its transparent quartz window above the chip. When this window is exposed to ultraviolet light for several minutes, the on-chip EPROM is erased. After erasing, the EPROM may be reprogrammed. Most EPROMS require the use of a device called a PROM programmer to perform the programming function. However, with the 68701, you will program the on-chip EPROM using only the internal resources of the 68701 EPROM and minimal external hardware/firmware.

68701 EPROM Programming

As previously stated, the 68701 defines mode 0 as the mode used to program the on-chip EPROM. The general procedure for programming the EPROM is as follows. You will first configure the 68701 in mode 0 by applying the required logic (000) to P20, P21, P22 and executing RESET. In this mode, the 68701 locates the RESET interrupt vector at address BFFE and BFFF. The RESET interrupt vector will then direct the 68701 to the beginning of an EPROM programming monitor located in an external ROM. This monitor will facilitate the EPROM programming procedure and will be discussed shortly. After RESET, you will apply a 20-volt level (V_{pp}) to the RESET pin (pin 6) to power the internal EPROM programming-related circuits. Once this is done, you will use the EPROM programming monitor to load the desired EPROM program into R/W memory. The monitor can then transfer this information into the on-chip EPROM memory locations. When the transfer is completed, you should verify that the proper information has been programmed into the EPROM.

Now, let's get into some of the specifics. First, to configure the 68701 for mode 0 and apply the required 20-volt programming voltage, a RESET circuit such as the one shown in Fig. 5-4 can be used. This circuit will apply the required logic to port 2 to configure the 68701 in mode 0 when S1 is depressed. Recall that the chip is configured during RESET. Therefore, depressing S1 will activate RESET and apply the required logic to port 2. If P20, P21, or P22 are also being used for other external purposes, the proper peripheral isolation circuitry must be added (reference Chapter 4). You are now ready to load the desired EPROM program into R/W memory and

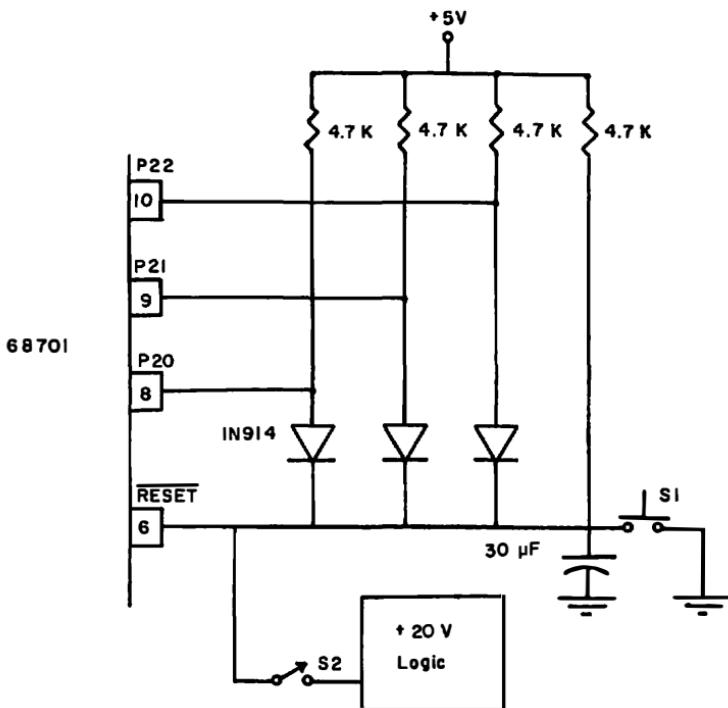


Fig. 5-4. RESET circuit for the 68701 EPROM programming.

transfer this information to the 68701 EPROM. As mentioned earlier, this will require an external ROM monitor package. Fortunately, Motorola has such a package available. It is called PRObug (part #MCM6832PB). Motorola has mask-programmed PRObug into a standard MCM6832 2K×8 ROM. PRObug is very similar to LILbug; however, it contains additional commands to program the 68701 EPROM and verify that the proper programming has been accomplished. The PRObug monitor commands will allow you to do the following:

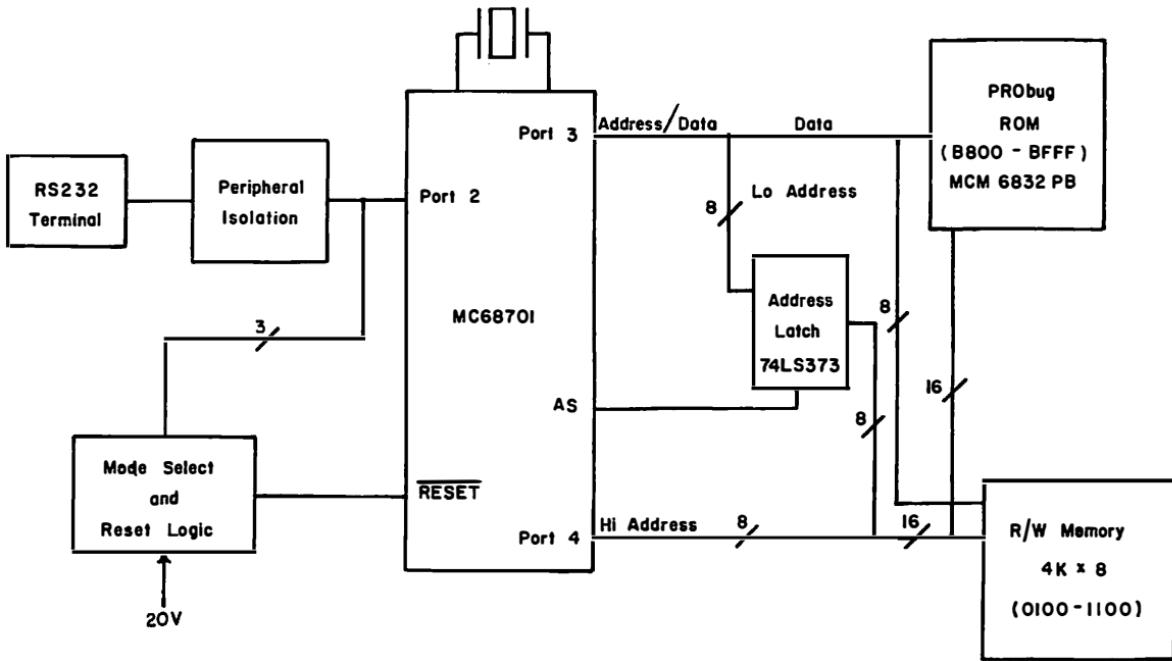
- Load a program in hexadecimal.
- Examine a loaded program.
- Examine data in any memory location and change data in any R/W memory location, including i/o.
- Display blocks of memory in the user's program registers.
- Insert, display and remove breakpoints in the user program.
- Freerun or step (trace) through the user program.
- Calculate the proper relative address offset for branch instructions.

- Dump the program.
- Record or punch the program on magnetic or paper tape.
- Declare the crystal frequency.
- Check for the erased state of the 68701 EPROM.
- Program the 68701 EPROM.
- Compare memory values to the 68701 EPROM contents.

(It is suggested that readers who are interested in the actual use of the PRObug system obtain the latest technical information and data sheets directly from Motorola Semiconductor, 3501 Ed Bluestein Blvd., Austin, TX 78721.)

Of particular importance in programming the 68701 EPROM are the last four functions in the above list. These four functions are provided by the PRObug commands XTAL, CHCK, PROG, and VERF, respectively. For proper programming of the EPROM, power must be supplied to each EPROM byte for approximately 50 milliseconds. PRObug applies power for this time period assuming a 4.91-MHz crystal is being used. If a 2.45-MHz crystal is being used, the XTAL command will allow you to change the PRObug timing such that proper timing will be achieved with the 2.45-MHz crystal. Any other crystal frequencies cannot be accommodated by PRObug. The CHCK command will allow you to check that the EPROM addresses have been properly erased prior to EPROM programming. The erased state is *zero* for the 68701 EPROM; therefore, the addresses which will be programmed must contain all zeros before the EPROM is programmed. The PROG command allows you to transfer the R/W memory information to the 68701 EPROM. All you need to do with this command is to specify the R/W memory block addresses which frame the program *and* the beginning address in EPROM where the program is to transfer. Once the program has been loaded into R/W memory, executing the PROG command *once* will transfer the entire program to the 68701 EPROM. You do not need to program the entire EPROM, since with the PROG command you may program the EPROM a section at a time. The PROG command will first *verify* that the EPROM section to be programmed has been properly erased in the same way that the CHCK command verifies proper erasure. Any address not containing all zeros will be displayed with the values at those addresses. If the EPROM has been properly erased, the PROG command will then prompt the user with "PWR?" At this time you will connect 20 volts (S2-Fig. 5-4) to the RESET pin (pin 6) and respond with "Y." The EPROM will then be programmed from the R/W memory locations specified in the PROG command. Once this takes place, the user will again be prompted with "PWR?" At this point you will *disconnect* the 20 volts from pin 6 and respond with

Fig. 5-5. 68701 EPROM programming system.



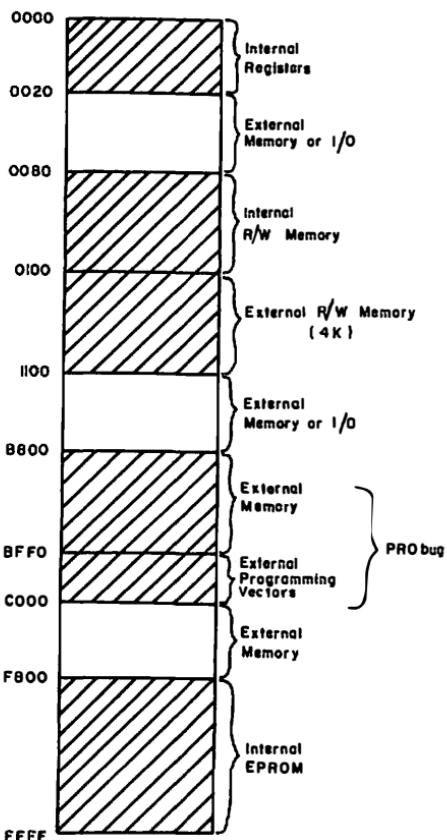
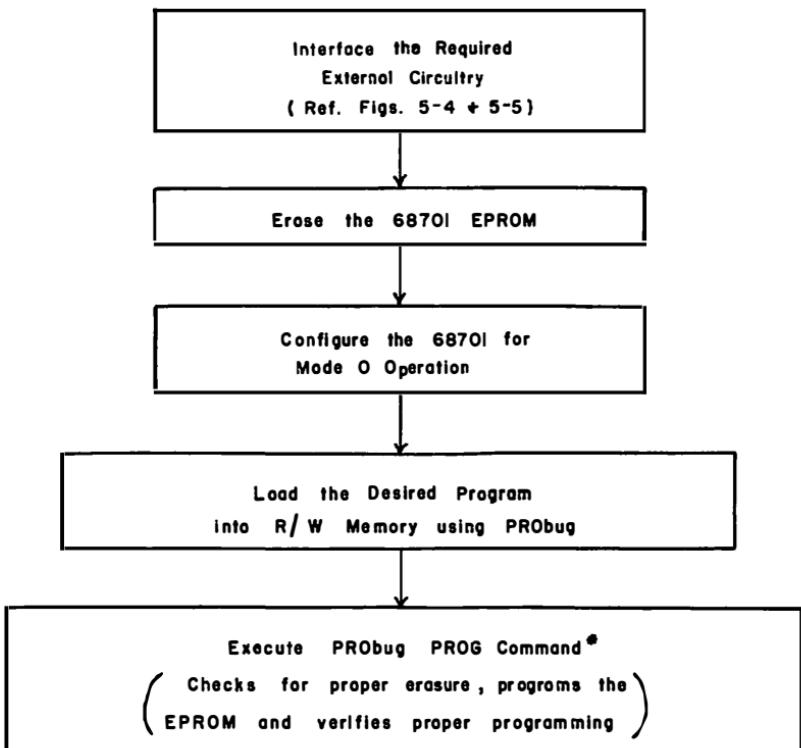


Fig. 5-6. 68701 programming circuit memory map.

“N.” The EPROM contents are then automatically compared to the R/W memory locations to assure proper EPROM programming has been achieved. If an error is found, the EPROM address is displayed with the value at that address along with the R/W memory value that failed to compare. The VERF command will also allow you to compare the programmed EPROM in the same way, to the R/W memory locations to assure proper programming has occurred. However, note that the PROG command performs both the CHCK and VERF commands automatically.



* Note: +20 volts must be applied to pin 6 after the first "PWR?" user prompt is received then removed after the second "PWR?" user prompt is received.

Fig. 5-7. 68701 EPROM programming flowchart using PRObug.

Besides providing the external PRObug monitor, you will need to supply some external R/W memory for temporary program storage. The required system configuration is shown in Fig. 5-5. Since the 68701 must be programmed in mode 0 (multiplexed test mode), port 3 is multiplexed to provide both the external data bus and low address byte. The low address byte is demultiplexed with the 74LS373 latch (Reference Chapter 4). The external 2K PRObug ROM is located at addresses B800-BFFF and we chose to locate 4K bytes of external R/W memory at addresses 0100 through 1100. The PRObug ROM monitor *must* be located at these addresses because of the RESET interrupt vectoring. However, the external

R/W memory can be located within any unused memory address block. Communication is provided with an RS232 terminal via port 2 (Reference Chapters 7 and 9). The proper peripheral isolation must be provided since port 2 is being used for mode configuration and serial communication. The associated memory map is shown in Fig. 5-6. With this system, you should have no trouble programming your 68701 EPROM. After programming, you may select any operating mode desired by changing the mode select logic. PRObug can still be utilized for communication in any of the expanded multiplexed modes. The flowchart in Fig. 5-7 summarizes the 68701 EPROM programming process using PRObug.

Without PRObug, you must supply your own monitor program to facilitate the EPROM programming. The following discussion will provide you with some general guidelines for writing such a monitor. First, the 68701 adds two control bits to the R/W Memory/EPROM control register located at address 0014. These are the program latch enable (PLE-bit 0) and program enable (PGE-bit 1) bits. Refer to Fig. 5-3. These bits are both set to a logic 1 state by RESET and they can *only* be changed when operating in mode 0. When PLE is set (logic 1) data can only be read from the EPROM. When cleared (logic 0), the program latch enable bit (PLE) allows information to be written to the EPROM. However, the EPROM data byte and address are only *latched* and *not gated* into the EPROM until PGE is cleared. The program enable bit (PGE) allows the EPROM to be programmed. When cleared, this bit gates the 20-volt programming voltage level to the EPROM. This bit can

SIGNAL STATE \	<u>PGE</u>	<u>PLE</u>	<u>RESET</u> / <u>V_{pp}</u>
READ	1	1	+5 ± .25V
LATCH ADDRESS AND DATA	1	0	+5 ± .25V or +20 ± IV
UNDEFINED	0	1	
PROGRAM EPROM	0	0	+20 ± IV

Fig. 5-8. PGE/PLE truth table.

only be cleared *after* PLE is cleared. A truth table for these two control bits is shown in Fig. 5-8. After erasure, all bits of the 68701 EPROM are cleared (logic 0). Data is entered by programming "1's" into the desired bit locations. Once a "1" is programmed into a bit location, it can *only* be changed to a "0" by ultraviolet light erasure. As stated earlier, EPROM programming can only occur in mode 0. Therefore, as the 68701 is released from RESET, the reset interrupt vector is fetched from address BFFE:BFFF. This vector will be part of your external ROM monitor package and will "point" to the beginning of your EPROM programming monitor. To place the EPROM in the programming state, your monitor must first clear the PLE control bit. After PLE is cleared, the 25-volt programming voltage must be applied to pin 6 and can remain there for the entire programming operation. Data can now be written to the location in EPROM to be programmed. The cleared PLE control bit and write (store) operation will latch the data and address information. Data will be entered into the EPROM one byte at a time. To enter the data byte into EPROM, the PGE control bit must be cleared for 50 milliseconds, then returned to a logic 1 state. A programming process flowchart is shown in Fig. 5-9. The EPROM locations may be programmed individually, sequentially (indexed addressing), or at random. After the programming operation is completed, the 20-volt programming voltage must be removed from pin 6 since data may *not* be read from the EPROM with this voltage connected. Your monitor should then verify, by a compare operation, that the EPROM has been properly programmed. Of course, PRObug accomplishes all of the preceding functions with the PLE and PGE bits being transparent to the user.

THE 6803

As mentioned earlier, the 6803 is simply a no-ROM version of the 6801. Since no internal ROM is provided, the 6803 can only operate in the multiplexed no-ROM mode (mode 2). Refer to Fig. 5-10 for a functional block diagram and pin assignments of the 6803. Note that the pin assignments are identical to those of the 6801. However with the 6803, port 3 is *always* multiplexed to provide the low order address byte (A0-A7) and data byte (D0-D7) with port 4 *always* providing the high order address byte, A8-A15. External demultiplexing of port 3 is provided in the same way as for the 6801 in the expanded multiplexed mode (Reference Chapter 4). Parallel i/o is provided via port 1 and/or port 2. Eight parallel i/o lines are provided by port 1, with up to five parallel lines provided by port 2. As with the 6801, port 2 is also used for mode configuration, serial communications and timer applications. Since the

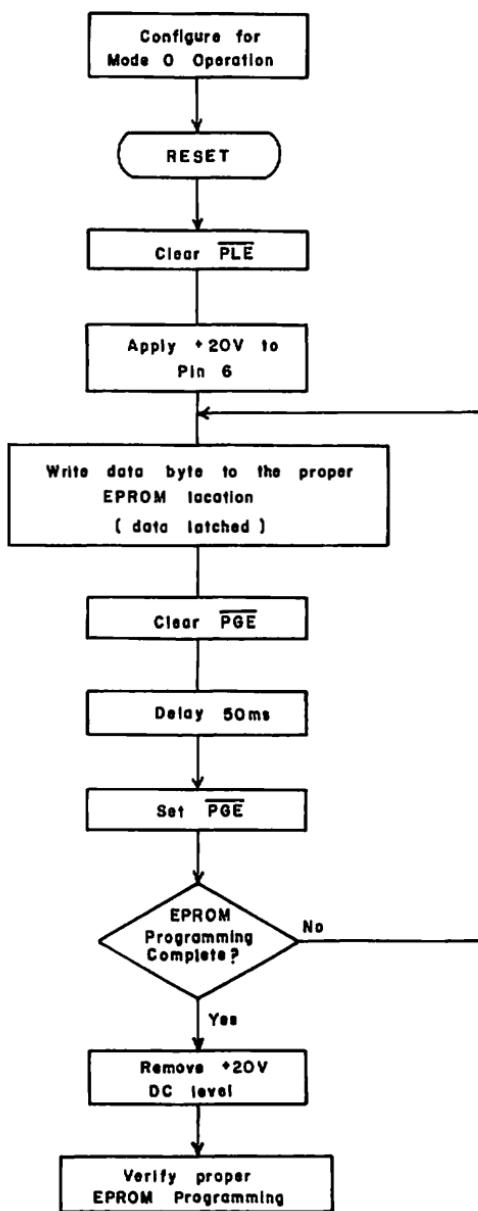
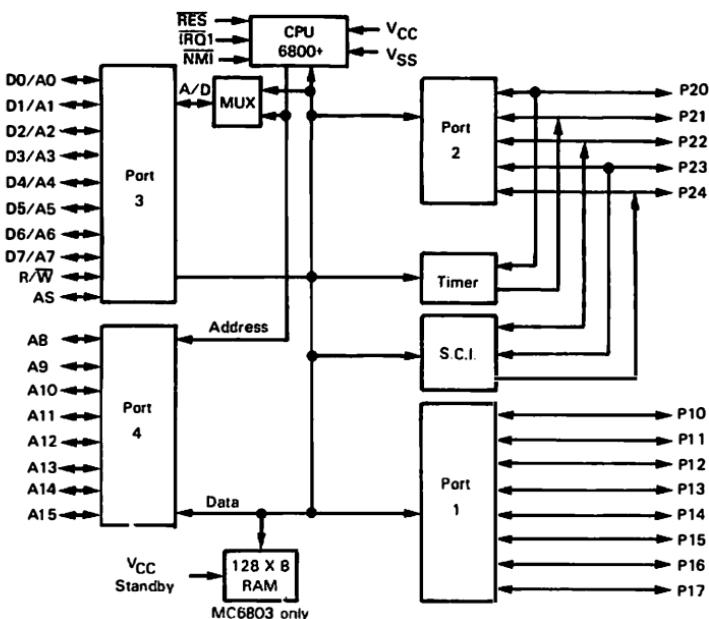


Fig. 5-9. 68701 EPROM programming flowchart.



V _{SS}	1	40	E
XTAL1	2	39	AS
EXTAL2	3	38	R/W
NMI	4	37	D0/A0
IRQ1	5	36	D1/A1
RESET	6	35	D2/A2
V _{CC}	7	34	D3/A3
P20	8	33	D4/A4
P21	9	MC6803	D5/A5
P22	10	32	D6/A6
P23	11	31	D7/A7
P24	12	29	AB
P10	13	28	A9
P11	14	27	A10
P12	15	26	A11
P13	16	25	A12
P14	17	24	A13
P15	18	23	A14
P16	19	22	A15
P17	20	21	V _{CC} Standby

Fig. 5-10. 6803 functional block diagram and pin assignments.

6803 only operates in mode 2, the logic present at P20, P21, and P22 during RESET must be low, high, low respectively to properly configure the chip. If any of these lines are also being used for other i/o purposes, the proper peripheral isolation must be provided (Reference Chapter 4). Port 2 can also be used as a timer and to provide serial communications. These functions are identical to those of the 6801 and are discussed in detail in Chapters 6 and 7. The 6803 is also available in a no-R/W memory version as the 6803NR. The 6803NR is essentially the same as the 6801-mode 3 configuration. Therefore, with the 6803NR, the logic present at P20, P21, and P22 during RESET must be high, high, and low, respectively, to properly configure the 6803NR. Of course, *both* external ROM and R/W memory must be supplied with the 6803NR. A minimum 6803 system would consist of *two* chips: the 6803 and external ROM, while the minimum 6803NR system would be a *three-chip* system: the 6803NR, external ROM and external R/W memory. The 6803 and 6803NR memory maps are shown in Fig. 5-11. You might note that

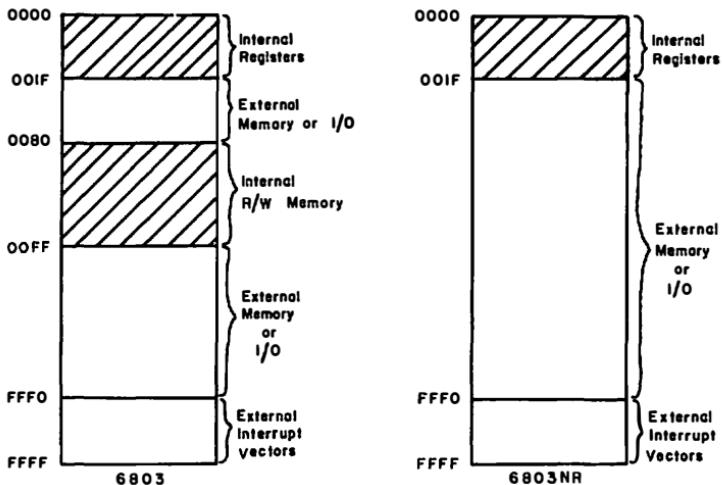


Fig. 5-11. 6803/6803NR memory maps.

these are identical to the 6801 mode 2 and mode 3 memory maps, respectively, presented earlier. The special purpose internal register functions of the 6803 are shown in Table 5-2. Since ports 3 and 4 are not used for i/o, their associated special purpose registers located at addresses 04, 05, 06, 07, and 0F are not used internally and are available for external use. All of the functional 6803 special purpose registers are identical to those of the 6801 and are used in the same way. Reference the appropriate 6801 discussions within the text.

Table 5-2. 6803/6803NR Special Purpose Register Functions

Hex Address	Register Function
00	Data Direction 1
01	Data Direction 2
02	I/O Port 1
03	I/O Port 2
04	Not Used
05	Not Used
08	Not Used
07	Not Used
08	TCSR
09	Counter High Byte
0A	Counter Low Byte
0B	Output Compare High Byte
0C	Output Compare Low Byte
0D	Input Capture High Byte
0E	Input Capture Low Byte
0F	Not Used
10	Serial Rate and Mode Register
11	Serial Control and Status Register
12	Serial Receiver Data Register
13	Serial Transmit Data Register
14	R/W Memory/EPROM Control Register
15-1F	Reserved

The 6803 and 6803NR instruction sets are identical to that of the 6801. (Reference Chapter 2 and Appendix A.) Interrupt vector assignments are also identical to those of the 6801. However, the interrupt vectors *must* be provided externally, since no internal ROM is provided.

REVIEW QUESTIONS

1. The only common memory allocations to all eight operating modes are:
2. Memory locations _____ through _____ are not available to the user.
3. Can any of the internal special purpose register addresses be used externally? If so, when?
4. What is the only exception to the rule that all modes must be configured by external hardware during RESET?
5. The three functions of the R/W Memory/EPROM Control Register located at address 0014 are:

6. What steps are necessary for proper information retention in the on-chip R/W memory during a power down (loss of V_{cc}) situation?
7. The 6801 on-chip ROM may be mask-programmed to begin at any of the following addresses:
8. The standard 6801 on-chip ROM package is called _____ and is designated as part #_____.
9. The 68701 EPROM is fixed at addresses _____ through _____.
10. With the 68701, the two functions of the RESET pin (pin 6) are:
11. The 68701 is *always* programmed in mode _____.
12. For 68701 EPROM programming, the interrupt vector map is located at addresses _____ through _____.
13. The "erased state" of the 68701 EPROM is when all the EPROM bit locations contain _____.
14. The external ROM package which Motorola has developed to facilitate the 68701 EPROM programming is called _____ and is designated as part #_____.
15. The four main PRObug commands directly associated with the 68701 EPROM programming are: _____, _____, _____, and _____.
16. Describe the general procedure for programming the 68701 EPROM using PRObug.
17. Describe the general procedure for programming the 68701 EPROM without PRObug.
18. Without PRObug, the user must configure the PLE and PCE bits of the R/W Memory /EPROM control register located at address 0014. Describe the function of these control bits.

19. Each time a 68701 EPROM location is programmed, how long must the 20-volt programming level be gated to the EPROM?
20. The only mode of operation possible with this 6803 is _____.
21. With the 6803, the maximum number of parallel i/o lines available is _____.
22. Are the serial i/o and timer functions of port 2 available in the 6803?
23. A minimum 6803 system would consist of how many chips?
24. What is the difference between the 6803 and 6803NR?
25. What is the difference between the 6801 and 6803 instruction sets?

ANSWERS

1. the special purpose register memory allocations from hex address 0000 through 001F
2. 0015 through 001F
3. Yes, when ports 3 and/or 4 are being used to provide the external data, or address bus 0 in the expanded modes. When this is the case, the internal special purpose register functions for these ports are disabled and the associated addresses may be used externally.
4. When changing from mode 4 (single-chip test) to mode 5 (expanded non-multiplexed). This can be done by writing a "1" into bit 5 (PC0) of the port 2 data i/o register located at address 0003.
5. to enable and disable the on-chip R/W memory.
to provide monitoring of the V_{cc} Standby voltage level.
to allow programming of the 68701 EPROM.
6. The following procedure must be followed to retain information in the first 64 bytes of on-chip R/W memory during a loss of V_{cc}:
 1. Write a zero into the RAM (R/W) enable bit (bit 6) of the R/W Memory/EPROM control register.
 2. Write a one into the STANDBY PWR bit (bit 7) of the R/W Memory/EPROM control register.
 3. Keep V_{cc} Standby greater than 4.0 volts.
7. C800, D800, E800, or F800
8. LILbug, 6801L1
9. F800 through FFFF

10. to reset the microcomputer.
to supply the 20-volt EPROM programming voltage (V_{pp}) during the EPROM programming operation.
11. mode 0
12. BFF0 through BFFF
13. logic 0's
14. PRObug, MCM6832PB
15. XTAL, CHCK, PROG, VERF
16. Refer to Fig. 5-7
17. Refer to Fig. 5-9
18. PLE, program latch enable, allows information to be latched from R/W memory prior to being gated into the 68701 EPROM.
PGE, program enable, allows the EPROM to be programmed by gating the 20-volt programming voltage level to the EPROM.
19. 50 milliseconds.
20. mode 2
21. 13, provided by ports 1 and 2
22. Yes.
23. two-6803 and external ROM.
24. The 6803NR is a no-internal-R/W memory version of the 6803.
25. There is no difference between the 6801 and 6803 instruction sets.

CHAPTER 6

How To Use the 6801 On-Chip Timer

INTRODUCTION

One of the most important and frequent tasks which a microcomputer must perform for many dedicated applications is a timing function. Many applications require the microcomputer to provide a time-of-day clock to control functions which require periodic servicing; that is, hourly, daily, monthly, and so on. Yet other applications require the microcomputer to count events, generate a pulse after an elapsed time interval (one-shot), measure pulse widths, measure the times between pulses, or perform frequency measurements. The automobile will require the microcomputer to perform many timing functions. For example, the revolutions per minute and dwell of an automobile engine can be easily determined by measuring the point-closure duration and point-closure period. The engine timing angle can be determined by measuring the point-closure period and the time duration between plug #1 firing and when piston #1 is at the top of its stroke (0° top dead center). If your requirement is to interface a microcomputer to the "real-world" it must be capable of performing various timing functions since almost all so-called real-world events involve time intervals.

With the first generation microprocessors such as the 6800, a separate chip would be required to perform the timing function. In the Motorola 6800 family, this chip is the MC6840 programmable timer module (PTM). Its purpose is to allow the MPU and its associated R/W memory to be free of the timing function task. With

the 6801, this function has been integrated onto the CPU chip, thus reducing the required chip count. The timer section of the 6801 is accessed through port 2 and, as you will see shortly, can be used to perform a variety of timing tasks such as single-shot or continuous pulse generation, event counting, event timing, interval measurement, and frequency measurement.

OBJECTIVES

At the end of this chapter you will be able to do the following:

- Describe the four user-addressable parts of the 6801 timer.
- Understand the use of the timer counter, output compare register, input capture register, and control register.
- Explain how to generate a one-shot or continuous waveform of variable pulse width using the 6801 timer.
- Explain how to count external events with the 6801 timer.
- Explain how to measure external pulse width using the 6801 timer.
- Explain how to measure time between external events with the 6801 timer.
- Explain how to measure frequency with the 6801 timer.

TIMER STRUCTURE

In Chapter 1, we briefly discussed the on-board timer function of the 6801. We stated that it was essentially a 16-bit free-running counter integrated onto the 6801 chip, and that it was accessed via bits 0 and 1 of port 2 (P20 and P21). You are now ready to learn about the 6801 timer function in more detail. Incidentally, the same timer function is provided in the 68701 and 6803 chips. The operation and use of the timer on these chips is identical to that of the 6801, so the following discussion is applicable to all three chips. The timer consists of the following four functional registers:

- 16-bit free-running counter
- 16-bit output compare register
- 16-bit input capture register
- 8-bit control and status register (TCSR)

These registers are shown in Fig. 6-1. Now, refer back to Table 5-1 and note that each of the above registers has a unique address as a special-purpose register. The timer control and status register (TCSR) is located at address 0008, the counter register at addresses 0009 and 000A, the output compare register at addresses 000B and 000C, and the input capture register at addresses 000D

TIMER CONTROL/STATUS REGISTER

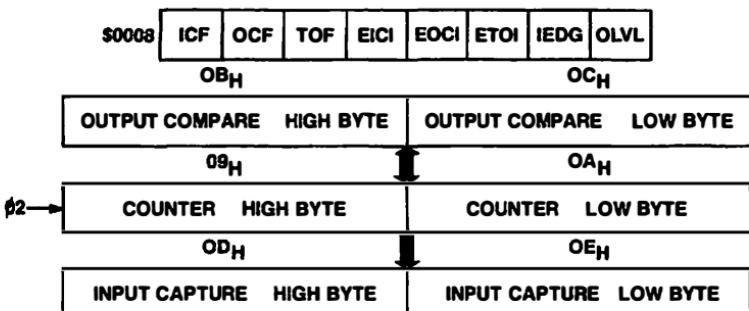


Fig. 6-1. Timer registers.

and 000E. The latter three registers require two addresses each, since they are 16-bit registers. Using these addresses, you will be able to read or write to any of the timer registers. First, each register will be discussed independently. Then, it will be shown how they can be used in conjunction with each other to perform various timing functions such as waveform generation, frequency measurements, event counting, interval measurement, event timing, etc.

Counter (0009:000A)

This is a 16-bit free-running counter which is incremented by the internal 6801 clock. Therefore, the fastest count possible is at a 1.25-MHz rate using a 5-MHz crystal. For all practical purposes, you should consider the counter as a *read-only* register. It can be read by software at any time with a load 0009 instruction. However, a write (store) operation to address 0009 will always result in FFF8 being placed in the counter, regardless of the write value. The write operation is used by the manufacturer for timer testing purposes and this operation can also adversely affect the operation of the serial communications interface (SCI). Therefore, you must avoid writing to the counter address (0009:000A). The counter is cleared by a hardware RESET at pin 6. You can begin a timing interval by clearing the counter with RESET or by reading the counter contents to determine its starting value. In either case you or the 6801 know the counter value at the beginning of the timed interval.

Output Compare Register (000B:000C)

The compare register is a 16-bit read/write register. Its contents are used to control the *length of a timed interval* or the *period of an output waveform*. The value in the compare register is continually compared to the 16-bit counter value. When a match occurs,

two events take place: a flag is set in the timer control and status register which may or may not generate an interrupt request to the 6801, *and* an output level is transferred to P21 (port 2, bit 1).

To preset the compare register to any desired 16-bit value, you can use a store accumulator D (STD) to address 000B instruction. A compare is not made for *one* machine cycle following the store operation to ensure that a valid 16-bit value is present in the output compare register prior to the first comparison. The compare register is automatically preset to FFFF by a hardware RESET at pin 6. Its value can be read with software at any time.

As was mentioned earlier, the timer is accessed through bits 0 and 1 of port 2 (P20 and P21). You will use P21 at pin 9 as a timer-output pin to provide a one-shot timing pulse, or a continuous output waveform whose width (period) may be varied from a few microseconds to many seconds as a result of the compare register contents and associated software. The use of the compare register to provide an output waveform is demonstrated in Example 6-1.

Input Capture Register (000D:000E)

The input capture register is a 16-bit *read-only* register located at addresses 000D:000E. It can be read into accumulator D with a load accumulator D (LD_D) instruction. The input capture register is used to store the present value of the counter when an active transition is provided to P20 (pin 8) from an external source. The proper active transition (1 to 0, or 0 to 1) is defined by software and will be discussed shortly. When the proper transition is detected on P20, two events occur: an interrupt request may or may not be generated to the 6801, *and* the present counter contents are stored in the input capture register. By reading the input capture register each time an active transition occurs at P20, you can measure frequency or the time duration between external events.

Timer Control and Status Register (0008)

The timer control and status register (TCSR) is an 8-bit register located at address 0008. The various bit designations are shown in Fig. 6-2. All eight bits are readable; however, only the first five bits may be written into. These first five bits are used for timer control functions with the upper three bits used as flags for the timer's status information. A description of each TCSR bit, beginning with bit 7, follows.

ICF, the input capture flag (bit 7), is a *read-only* flag which is set by an active input transition on the P20 line (pin 8). Recall that this active transition will also cause the counter's contents to be transferred to the input capture register. Therefore, the input capture flag indicates that this transfer has taken place, and that an

MC6801

TIMER CONTROL/STATUS REGISTER

\$0008	ICF	OCF	TOF	EICI	EOCI	ETOI	IEDG	OLVL
--------	-----	-----	-----	------	------	------	------	------

ICF — INPUT CAPTURE FLAG

OCF — OUTPUT COMPARE FLAG

TOF — TIMER OVERFLOW FLAG

EICI — ENABLE INPUT CAPTURE INTERRUPT

EOCI — ENABLE OUTPUT COMPARE INTERRUPT

ETOI — ENABLE TIMER OVERFLOW

IEDG — INPUT EDGE

OLVL — OUTPUT LEVEL

Fig. 6-2. Timer control and status register (TCSR).

active transition at P20 has occurred. The flag is cleared by *reading* the TCSR at address 0008, followed by *reading* the input capture register at address 000D:000E.

OCF, the *output compare flag* (bit 6), indicates that a match has occurred between the contents of the counter and the output compare register. It is a read-only flag which is set when the match takes place. To clear the flag, the program must *read* the TCSR at address 0008, followed by a *write* operation to the output compare register at address 000B:000C. The write operation can be used to load the compare register with a new value or reload the compare register with the same value for no real change.

TOF, the *timer overflow flag* (bit 5), is also a *read-only* flag which is set when the counter goes through a 0000 to FFFF transition. Therefore, the timer overflow flag will set each time the counter “rolls over.” With a 1-MHz clock, this will happen every 65.536 milliseconds. The flag is cleared by *reading* the TCSR at address 0008, followed by consecutively *reading* the counter contents at address 0009:000A.

EICI, the *enable input capture interrupt* (bit 4), is a read/write register bit which is associated with the input capture register and timer input line, P20. We previously stated that an active transition on P20 may, or may not, cause an interrupt request to be generated to the 6801. If the EICI bit is set, an interrupt request *will* be generated to the 6801 CPU upon an active transition on P20. If this bit is cleared, the interrupt will be masked-out. *However, the interrupt request generated is not the same as the hardware interrupt request (IRQ) with which you are familiar.*

The 6801 includes a second level interrupt request specifically for interrupts generated by the timer and serial communications interface (SCI). We will designate the original 6800 type interrupt

	Vector		Description
	MS	LS	
Highest Priority	FFFF,	FFFF	Restart
	FFFC,	FFFD	Non-Maskable Interrupt
	FFFA,	FFFB	Software Interrupt
	FFF8,	FFF9	IRQ1
	FFF6,	FFF7	IRQ2/Timer Input Capture
	FFF4,	FFF5	IRQ2/Timer Output Compare
	FFF2,	FFF3	IRQ2/Timer Overflow
	FFFO,	FFF1	IRQ2/Serial I/O Interrupt
Lowest Priority			

Fig. 6-3. Interrupt vectors memory map.

request as IRQ1. As you are aware, this interrupt is the old IRQ and is generated via pin 5, with its interrupt vector located at addresses FFF8 and FFF9. We will designate an interrupt request generated by the timer or SCI as IRQ2. The 6801 includes *four* separate interrupt vectors associated with IRQ2. The memory map for these vectors is shown in Fig. 6-3. Note that there are three vectors associated with the timer and one with the SCI. Fig. 6-3 also indicates the relative priority of each vector.

Now, if the EICI bit of the TCSR is set, an IRQ2/timer input capture interrupt request will be generated for the 6801 CPU. If the I-flag of the 6801 condition code register (CCR) is cleared, the interrupt will be acknowledged and the 6801 will automatically branch to address FFF6:FFF7 for the interrupt vector. Note, this interrupt can be masked-out by either clearing the EICI bit of the TCSR, or setting the I-flag of the CCR.

EOCI, the *enable output compare interrupt* (bit 3), is similar to the EICI bit function just discussed. However, it is associated with the output compare register function. If EOCI is set, a match between the contents of the counter and output compare register will cause the IRQ2/timer output compare interrupt request to be generated for the 6801 CPU. If the I-flag of the CCR is cleared, the interrupt will be acknowledged and the 6801 will automatically branch to address FFF4:FFF5 for the interrupt vector. This interrupt can also be masked-out by clearing the EOCI bit, or setting the I-flag.

ETOI, the *enable timer overflow interrupt* (bit 2), is similar to the two-bit functions just discussed. However, this bit is associated with the counter register. If the counter rolls over, an interrupt request can be generated for the 6801. If the ETOI bit is set, the interrupt will be generated and acknowledged, provided the I-flag is cleared. This interrupt request is called IRQ2/timer overflow and its interrupt vector is located at address FFF2:FFF3. The interrupt can be masked-out by clearing the ETOI bit, or setting the I-flag.

IEDG, the input edge bit (bit 1), is used in conjunction with the input capture register. It defines the active state of the input pulse which will cause a transfer of the counter information to the input capture register. If the IEDG bit is set (1), the transfer will take place with a low-to-high transition on the P20 line (pin 8). If the IEDG bit is cleared (0), a high-to-low transition at pin 8 will cause the count to be transferred to the input capture register. This function will allow you to measure the width of a pulse, or the time between consecutive pulses.

Since P20 (pin 8) is being used as an input line for the timer function, you must remember to include the necessary program steps to clear the corresponding bit in the port 2 data direction register (DDR) located at address 0001, so that it is configured as an input line.

OLVL, the output level bit (bit 0), is used in conjunction with the output compare register. When a match takes place between the contents of the output compare register and counter, the value of the OLVL bit is transferred to the P21 line (pin 9), provided that the P21 data register bit (DDR-P21) is set. Recall that each port has an associated data direction register (DDR). For a particular port line to be output, its corresponding DDR bit must be set. Conversely, for a port line to be input, its corresponding DDR bit must be cleared. Therefore, the output level bit (OLVL) can only be transferred *out* to P21 if the P21 data direction register bit has been set through the use of the proper software instruction. The port 2 data direction register is located at address 0001.

Once the output level bit has been transferred to P21, it can be changed with a write (store) to address 0008 instruction. Then, the next time a match is made, the new level will appear at P21. Therefore, with the use of this bit, you can set the state of the logic level that is to be output at P21 when a counter/compare register match is made. By alternating this output level, you can output a waveform of continuous or variable pulse widths via P21 (pin 9).

The timer control and status register (TCSR) bit functions are summarized in Table 6-1. You might note from the table that the entire TCSR is cleared with a RESET operation.

As mentioned in the chapter introduction, the timer can be used to perform a variety of tasks, such as single-shot or continuous pulse generation, event counting, event timing, interval measurement, and frequency measurement. These tasks can be reduced to the four basic timer operating modes shown in Fig. 6-4. The *single-shot* mode produces only one logic 1 pulse for a definite period. In this mode the timer can operate as a programmable, retriggerable one-shot. In the *continuous mode*, the output is low for the first time period, and changes state after each subsequent time period. You

Table 6-1. Timer Control and Status Register (TCSR) Bit Summary

TCSR	Set	Cleared
Bit 0 (OLVL)	By software: Clocked to P21 when a compare register match occurs	By software: (or RESET) Clocked to P21 when a compare register match occurs
Bit 1 (IEDG)	By software: Counter transfer to Input capture register will take place with a low-to-high transition on P20	By software: (or RESET) Counter transfer to Input capture register will take place with a high-to-low transition on P20
Bit 2 (ETOI)	By software: Enables IRQ2 interrupt to be generated when the counter contains 0000 (roll over)	By software: (or RESET) Masks-out the IRQ2 interrupt for the roll-over function
Bit 3 (EOCI)	By software: Enables IRQ2 interrupt to be generated when a compare register match occurs	By software: (or RESET) Masks-out the IRQ2 interrupt for the compare function
Bit 4 (EICI)	By software: Enables IRQ2 interrupt to be generated when the count is transferred to the input capture register	By software: (or RESET) Masks-out the IRQ2 interrupt for the input capture function
Bit 5 (TOF)	When a counter roll-over occurs	By reading the timer control and status register (TCSR) then the counter contents. (or RESET)
Bit 6 (OCF)	When a match occurs between the counter and output compare register	By reading the timer control and status register (TCSR) followed by writing to the output compare register. (or RESET)
Bit 7 (ICF)	By an active transition on P20, indicating the counter contents have been transferred to the input capture register	By reading the timer control and status register (TCSR) then the input capture register. (or RESET)

will use the output compare register to control these two functional modes.

The input capture register will be used to control the *period measurement* and *pulse width measurement* modes. You can measure frequency (reciprocal of period) using the period measurement mode. In this mode, you will simply determine the time between

two successive high-to-low (or low-to-high) transitions. The width of a pulse may be measured in the pulse measurement mode. In this mode, the timed interval will begin when the input pulse goes low

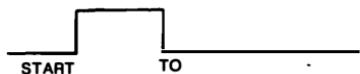
Continuous Mode



The output is low for the first Time Out period, and changes state for each subsequent Time Out period. This produces a 50% duty cycle square twice the duration of the Time Out.

Single-Shot Mode

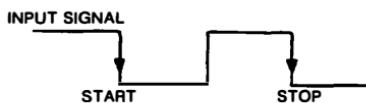
The Single-Shot Mode is similar to the continuous mode except that the output produces only one high pulse for the Time Out period.



In this mode a timer operates as a programmable, retriggerable one shot.

Period Measurement Mode

One operation performed in this mode is to measure the period (reciprocal of the frequency) of a given digital signal:



When the pulse goes low the first time, the timer starts counting down from its programmed number. When the signal goes low again, the timer stops counting down and generates an interrupt. The MPU can now determine the frequency easily ($1/p = f$).

Pulse Width Measurement Mode

The width of a given pulse may be determined by using this mode. The "down-time" of a digital signal can be measured in the same manner as in the Period Measurement Mode:

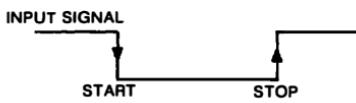


Fig 6-4. Basic timer modes of operation.

(high) and stops when the input pulse returns high (low). You will now see how the 6801 timer can be used to provide some of these timing functions.

TIMER EXAMPLES

Now that you are familiar with the timer registers and their functions, let us see how these various registers can be used in conjunction with each other to provide some meaningful functions. The first application that you will look at is the continuous generation of pulses. You can generate a continuous pulse output on the P21 line (pin 9) using the compare register and OLVL bit of the timer control and status register (TCSR). Recall that when a match is made between the contents of the counter and the compare register, the OLVL bit's status (in the TCSR) is transferred to output pin P21 (pin 9). By changing the OLVL bit's status after each match occurs through the use of appropriate instructions, you can generate a series of pulses. The pulse width and time between pulses are simply a function of the compare register's contents.

Generating an Output Waveform

Assuming the internal clock frequency is 1 MHz, the maximum time between a match of the contents of the compare register and the counter would be: $\text{FFFF} \times 1 \text{ microsecond}$ or 65.636 milliseconds. Therefore, the pulse width or time between pulses could be a *maximum* of 65.536 milliseconds for a compare register value of FFFF. Of course, this may be extended by using additional instructions to count the matching operations, taking action only after X-matches have taken place. In this example, we have assured only one period, from 0000 → FFFF is used. The *minimum* value would depend on the number of MPU cycles that it would take to execute the associated instructions in a program. Typically, this would be about 12-18 microseconds for a 1-MHz clock frequency. The program shown in Example 6-1 will provide a continuous waveform generation at P21 (pin 9).

Example 6-1: Continuous Pulse Generation

This program can provide a continuous waveform generation.

Hex Address	Hex Contents	Mnemonic/Contents	Operation
0080	86	LDAA #	Set DDR-P21
0081	02	02	
0082	97	STAA \$	
0083	01	01	
0084	0F	SEI	Set I-flag
0085	7F	CLR \$\$	Clear TSCR (Clears OLVL)
0086	00	00	
0087	08	08	

0088	96	LDAA \$		
0089	08	08		Clear output compare flag (OCF)
008A	97	STAA \$		
008B	0B	0B		
008C	DC	LDD \$		
008D	09	09		Load count into ACCD
008E	C3	ADDD #		
Pulse width HI Byte				
008F	-	-		Add desired pulse width to count
0090	-	-		
0091	DD	STD \$		
0092	0B	0B		Store sum to compare register
0093	96	LDAA \$		
0094	08	08		
0095	48	ASLA		Shift output compare flag (OCF) Into C-register
0096	48	ASLA		
0097	24	BCC		
0098	FA	FA		Branch if carry clear (to address 0093)
0099	73	COM \$\$		
009A	00	00		
009B	08	08		Complement TSCR (Sets OLVL)
009C	96	LDAA \$		
009D	08	08		
009E	97	STAA \$		Clear output compare flag (OCF)
009F	0B	0B		
00A0	DC	LDD \$		
00A1	09	09		Load count into ACCD
00A2	C3	ADDD #		
time between pulses				
00A3	-	-		Add desired time between pulses to count
00A4	-	-		
00A5	DD	STD \$		
00A6	0B	0B		Store sum to compare register
00A7	96	LDAA \$		
00A8	08	08		
00A9	48	ASLA		Shift output compare flag (OCF) Into C-register
00AA	48	ASLA		
00AB	24	BCC		
00AC	FA	FA		Branch if carry clear (to address 00A7)
00AD	20	BRA		
00AE	D6	D6		Branch always (to address 0085)

The program in Example 6-1 begins by configuring P21 as an output line. It then sets the I-flag of the condition code register to mask-out any interrupt generated by a compare register match. The TSCR is then cleared which results in the OLVL bit (bit 0) being cleared. However, this operation will not clear the output compare flag (OCF) since this is a read-only flag bit. To clear this flag, you must read the TSCR and then write to the compare register. This must be done because the OCF will be used to detect a counter/compare-register match. The OCF is cleared by the load-and-store instructions at addresses 0088-008B. Next, the present contents of the counter are loaded into accumulator D and the number of counts ($1 \mu\text{sec}/\text{count}$) are added to the present counter value to obtain the value to be compared. This is the desired *pulse width*. This sum is then stored in the compare register.

The program will then loop between addresses 0093 and 0098 until the output compare flag is detected. When a match occurs, the output compare flag (OCF) will be set and the logic zero level of the OLVL bit will be transferred to the P21 output. The program will then break out of this loop and complement the TSCR register, which in turn sets the OLVL bit. The output compare flag (OCF) is then cleared and the desired *time between pulses* (in counts) is added to the present count, with the sum being stored in the compare register. When the count reaches this value, the logic 1 status of the OLVL bit will be transferred to the P21 output. This new level will remain at the P21 output until OLVL is changed and another match is made. The cycle will then repeat itself, providing alternate high-low states at the P21 output. You can change the pulse width or time between pulses by changing the compare register values at memory locations 008F:0090 and 00A3:00A4, respectively. A time line diagram of this process is shown in Fig. 6-5. Here, for example purposes, the desired pulse width is five cycles and the time between pulses is five cycles. Therefore, to obtain the compare register value, five cycles (+5) are added to the existing counter contents each time a match is made. This sum is then stored in the compare register. When the counter reaches the sum value, another match will be made and the process repeated. The OLVL bit status is alternately changed with each match to provide a continuous output pulse waveform. A word of caution: you must be careful when you wish to generate outputs with very small time durations since the program's instruction cycle times could interfere with the detection of the count. Actually, the five cycle time periods in Fig. 6-5 would probably interfere with the count detection. Therefore, when working with small time periods, you will have to take the MPU cycles required for each instruction into consideration when determining the total elapsed time. Also, if the pro-

gram in Example 6-1 were executed immediately after RESET, it would not be necessary to initially clear the OLVL and OCF bits since the ~~RESET~~ operation clears the TCSR.

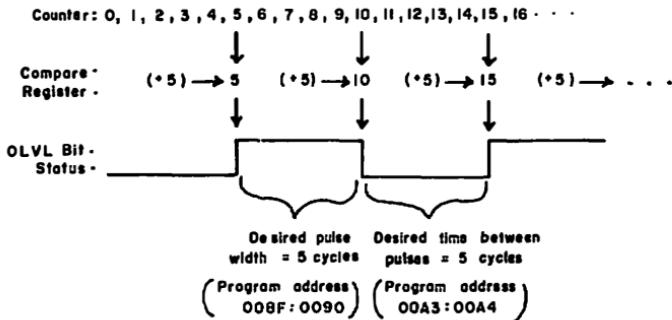


Fig. 6-5. Time line diagram of the continuous output pulse.

How might you increase the pulse width or time between pulses beyond 65.536 milliseconds into many seconds? You would have to set up a counter, possibly using the index register, to increment or decrement a count each time a match is made. The matches would be detected by reading the output compare flag (OCF). Depending upon the time interval desired, you would not change the OLVL bit status until a particular *index register count* was obtained. In this way, you could achieve time intervals of up to $\text{FFFF} \times \text{FFFF}$ (hex) microseconds, or 4295 seconds (almost 72 minutes).

There is another important point to be made here. The program in Example 6-1 could be modified so that it could be used as an interrupt service routine, by using the IRQ2/timer-output-compare interrupt function of the 6801. The interrupt function would be used in place of the branching loops to detect a counter/compare register match. Also, to use this function, you would have to make sure the I-flag is cleared and the EOCl bit (bit 3) of the timer control register is set such that an interrupt would be generated and acknowledged when the match occurs.

Performing Measurements on an Input Waveform

Now, let us see how you can use the 6801's input timer function to count external events and to measure input timing intervals and frequency.

Example 6-2: Event Counting

Many times it is desirable to determine the number of times that an external event occurs. The 6801 can accomplish this through the use of the input capture flag ICF. The input signal will enter the 6801 at the P20 pin (pin 8). Of course, this must be properly configured for an input function. Using the input edge (IEDG) bit (bit 1) of the TCSR, you can program the 6801 to recognize an event by either a high-to-low transition or a low-to-high transition at pin 8. Each time the "event" takes place, as indicated by the transition, the input capture flag (ICF) will cause an interrupt to be generated, which in turn will call an interrupt service routine to increment a counter. The count will be stored at some convenient memory location which can be read at any time. The following program will accomplish this task.

Main Program			
Hex Address	Hex Contents	Mnemonic/Contents	Operation
0080	86	LDAA #	
0081	00	00	
0082	97	STAA \$	
0083	01	01	
0084	96	LDAA \$	
0085	08	08	
0086	DC	LDD \$	
0087	0D	0D	
0088	86	LDAA #	
0089	12	12	
008A	97	STAA \$	
008B	08	08	
008C	0E	CLI	Clear I-flag
008D	7F	CLR \$\$	
008E	00	00	
008F	F0	F0	Clear event count storage location (00F0)
0090	3E	WAI	Wait for interrupt
0091	20	BRA	
0092	FD	FD	Branch back to WAI when return from interrupt

Interrupt Service Routine			
FA00	7C	INC \$\$	
FA01	00	00	
FA02	F0	F0	
FA03	96	LDAA \$	Increment event count

FA04	08	08			Clear ICF (TCSR, bit 7)
FA05	DC	LDD \$			
FA06	0D	0D			
FA07	3B	RTI			Return from interrupt

The main program portion is used to configure the 6801 for the event counting task. Port 2, bit zero (P20) is configured as an input line by clearing bit zero of the port 2 DDR, located at address 0001. Then, the input capture flag is cleared with a load TCSR instruction, followed by a load input capture register instruction. Next, the program sets the EICI (bit 4) and IEDG (bit 1) bits of the TCSR. Setting the EICI bit will allow an interrupt to be generated and sent to the 6801 CPU when an active transition is detected at P20 (pin 8). Setting the IEDG bit defines the active state on pin 8 as a low-to-high transition. Finally, the I-flag is cleared so that the 6801 will acknowledge the interrupt, and the event-count storage location is cleared. Memory location 00F0 has been chosen as the event-count storage location. The WAI instruction will halt the 6801 execution until the interrupt occurs. When an active transition is seen at pin 8, ICF will set and an IRQ2/timer input capture interrupt will be generated to the 6801 CPU. The 6801 will then automatically jump to address FFF6:FFF7 for the interrupt vector. We have chosen address FA00 as the interrupt vector for the IRQ2/timer input capture interrupt. Therefore, the interrupt service routine resides in the on-board ROM (EPROM). The interrupt service routine simply increments the event counter and clears the ICF flag with a load TCSR instruction followed by a load input capture register instruction. The 6801 will then return to the main program at address 0091. Since the program will return to address 0091, an unconditional branch has been inserted here such that the 6801 will branch back to the wait for interrupt (WAI) status and wait for the next event to occur.

Example 6-3: Event Timing (Interval and Frequency Measurement)

Besides counting external events, you can also use the 6801's timer to measure the time between external events. Again, you will use the IRQ/timer input capture interrupt to call an interrupt service routine. However, rather than simply providing a counting function, the interrupt service routine will read the present counter value and use it, along with the counter's value when the last "event" was detected, to calculate an interval between the two events. The time interval could represent pulse width, time between successive pulses or the period of a waveform. In the latter case, the waveform's frequency can then be easily calculated. Suppose you wish to measure the time interval between successive external events (pulses). Each

event will be indicated by a low-to-high transition on the P20 i/o line (pin 8). Each time an event takes place, the input capture flag (ICF) will cause an interrupt to be generated which in turn will call an interrupt service routine. Recall that this transition will also cause the present internal counter contents to be transferred to the input capture register. Therefore, the service routine will read this present count and calculate the difference between the present count and the previous count to determine the elapsed time interval. The maximum interval this program will accommodate is FFFF microseconds (65.536 milliseconds). The following program will perform this task.

Main Program			
Hex Address	Hex Contents	Mnemonic/Contents	Operation
0080	88	LDAA #	
0081	00	00	
0082	97	STAA \$	
0083	01	01	
0084	96	LDAA \$	
0085	08	08	
0086	DC	LDI \$	
0087	0D	0D	
0088	86	LDAA #	
0089	12	12	
008A	97	STAA \$	
008B	08	08	
008C	0E	CLI	
008D	3E	WAI	Clear I-flag Wait for Interrupt
008E	20	BRA	
008F	FD	FD	Branch back to WAI when return from interrupt
Interrupt Service Routine			
FA00	96	LDAA \$	
FA01	08	08	
FA02	48	ASLA	
FA03	48	ASLA	
FA04	48	ASLA	
FA05	25	BCS	
FA06	0C	0C	
FA07	DC	LDI \$	
FA08	0D	0D	
FA09	93	SUBD \$	
FA0A	F0	F0	Subtract previous count from present count

FA0B	DD	STD \$		
FA0C	F4	F4		Store result (elapsed time)
FA0D	DC	LDD \$		
FA0E	0D	0D		
FA0F	DD	STD \$		Save present count at memory location 00F0 for next calculation
FA10	F0	F0		
FA11	3B	RTI		Return from Interrupt
FA12	CC	LDD #		
FA13	FF	FF		
FA14	FF	FF		Load ACCD Immediate with FFFF
FA15	93	SUBD \$		
FA16	F0	F0		Subtract previous count from FFFF
FA17	D3	ADDD \$		
FA18	0D	0D		Add present count to above result
FA19	C3	ADDD #		
FA1A	01	01		Add one (1) to above result
FA1B	DD	STD \$		
FA1C	F4	F4		Store result (elapsed time)
FA1D	DC	LDD \$		
FA1E	0D	0D		
FA1F	DD	STD \$		Store present count at memory location C0F0 for next calculation
FA20	F0	F0		
FA21	96	LDAA \$		
FA22	08	08		
FA23	DC	LDD \$		Clear TOF
FA24	09	09		
FA25	3B	RTI		Return from Interrupt

The main program portion configures the 6801 so that it is used in the same way as it was in Example 6-2. Port 2, bit zero (P20) is configured as an input line, the input capture flag (ICF) is cleared and the EICI and IEDG bits of the TCSR are set. The I-flag is also cleared to allow the 6801 CPU to acknowledge an IRQ2 interrupt. The 6801 then *waits* for the interrupt to occur.

When the interrupt occurs, the counter's present contents are transferred automatically to the input capture register located at address 000D:000E. The interrupt service routine is then called. As before, we chose to locate the IRQ2/timer input capture interrupt service routine beginning at ROM address FA00. We have chosen the following addresses in the on-board R/W memory for temporary storage of data, as follows:

00F0:00F1—previous count
00F2:00F3—present count
00F4:00F5—elapsed time interval

The interrupt service routine must first make a decision based on the status of the time overflow flag (TOF), bit five of the TCSR. If the counter has rolled over during the elapsed time interval you wish to measure, the second reading (present count) will be smaller than the first reading (previous count). When this occurs, you are interested in the *absolute value* time difference and must calculate the elapsed time interval according to the following equation:

$$\begin{aligned}\text{elapsed time interval} = & (\text{FFFF} - \text{previous count}) \\ & + (\text{present count}) + 1\end{aligned}$$

Therefore, the service routine first determines if the TOF flag is set. If so, the program will branch to memory location FA12, where the above calculation will be performed. Once the elapsed time interval is calculated, it will be stored at address 00F4:00F5 where it can be read at any time or used in further calculations, such as a frequency calculation.

If the counter has not rolled over (TOF clear), the program will simply subtract the previous count from the present count to determine the elapsed time interval (in counts). Again, the result is stored at address 00F4:00F5. Also, with each calculation, once the result is obtained the present count becomes the so-called previous count for the *next* calculation. Therefore, the present count is saved at address 00F0:00F1 at the end of the interrupt service routine.

As previously stated, this program will measure a maximum time interval of 65.536 milliseconds between two consecutive pulses. If the time interval is longer, the program must be modified to count the number of times that the counter rolls over.

This can be done using the index register as previously discussed. An interrupt could be generated each time the TOF flag sets, indicating a roll-over has occurred. This interrupt service routine would increment the index register to keep track of the number of times the timer counter passes its beginning position. Each increment would represent a 65.536-millisecond interval with a 1-MHz clock. Then, the following equation can be used to determine the time interval:

$$\begin{aligned}\text{elapsed time interval} = & (\text{number of roll-overs}) * \text{FFFF} \\ & - (\text{previous count}) + (\text{final count}) + 1\end{aligned}$$

The program would also need to be modified to measure pulse width. For pulse widths of under 65.536 milliseconds, you would

set input edge bit (IEDG) bit of the TCSR such that the 6801 would first recognize a low-to-high transition (leading pulse edge) on P20, then clear the IEDG bit such that the 6801 would recognize a high-to-low (falling pulse edge) on P20. For pulse widths of more than 65.536 milliseconds, the computer also needs to be programmed to *recognize the number of times the counter rolls over, as previously discussed*.

Example 6-4: Input to Output Echo

In the following program, the input capture register is used to detect an input signal edge at P20 and the output compare register is then used to repeat or “echo” the input signal on the timer output pin (P21). The input level will be echoed on the output pin with a small delay. The program serves no practical function except to exercise the timer input and output functions of the 6801.

Reset			
Hex Address	Hex Contents	Mnemonic/Contents	Operation
0080	86	LDAA #	
0081	02	02	
0082	97	STAA \$	
0083	01	01	Set DDR-P21 to configure as output
0084	7C	INC \$\$	
0085	00	00	
0086	08	08	Increment TCSR (Sets OLVL bit)
0087	96	LDAA \$	
0088	08	08	
0089	DC	LDD \$	
008A	0D	0D	Clear Input capture flag (ICF)
008B	96	LDAA \$	
008C	08	08	
008D	2A	BPL	
008E	FC	FC	Check Input capture flag (ICF) until set
008F	88	EORA #	
0090	03	03	
0091	97	STAA \$	
0092	08	08	Toggle Input edge (IEDG) and output level (OLVL) bits
0093	DC	LDD \$	
0094	0D	0D	Clear Input capture flag (ICF)
0095	DC	LDD \$	
0096	09	09	Load count into ACCD

0097	03	ADDD #		Add offset
0098	00	00		
0099	0A	0A		
009A	DD	STD \$		Store sum to
009B	0B	0B		compare register
009C	20	BRA		Repeat
009D	ED	ED		(Branch to address 008B)

The program begins by setting bit one of the port 2 data direction register (DDR) such that P21 is configured as an output pin for timer output. There is no need to configure P20 for input since the RESET will automatically clear the P20 bit of the port 2 DDR. The RESET will also clear the OLVL bit in the TCSR; therefore, the next instruction sets that bit such that we begin with a high level output. To clear the input capture flag, the program reads the TCSR followed by a read of the input capture register. The program will then check the input capture flag status and loop until the input capture flag is set, indicating an input signal edge has been detected. Note that since the input capture flag is bit 7 of the TCSR, you can check its status by using a branch if plus (BPL) instruction. Once an input signal has been detected, the IEDG and OLVL bits of the TCSR are toggled and the ICF is cleared. The IEDG bit is toggled such that the program will detect the next

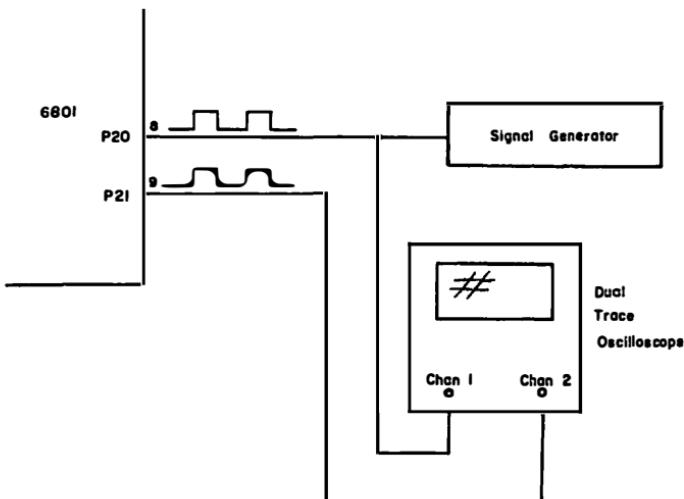


Fig. 6-6. Echo monitor circuit.

signal edge and the OLVL bit is toggled such that the output signal will "echo" the input signal. The ICF bit is cleared to recognize the next input signal edge. The present count is then read and an offset (OA) is added such that the output will follow the input after a short delay. The sum is then stored to the output compare register. When the counter reaches this sum value, a match will be made and the OLVL bit status will be transferred out to P21. The program will then repeat itself with the output at P21 echoing the input provided at P20.

The circuit shown in Fig. 6-6 may be used to provide an input signal and monitor the output signal for the above program. The input square wave from the signal generator must have a period (T) of at least 64 MPU cycles for the program to respond properly.

REVIEW QUESTIONS

1. The four user-addressable parts of the 6801 timer and their addresses are:

2. The counter is a _____-bit free-running counter which counts at a rate equal to _____.

3. The counter should be considered as a _____ register and can be cleared by _____.

4. The output compare register is what type of a 16-bit register?

5. When a match occurs between the counter and output compare register, two events take place. They are:

6. A _____ instruction can be used to preset the output compare register to any desired value.

7. The input capture register is what type of a 16-bit register?

8. The input capture register is used in conjunction with port _____, bit _____.

9. When an active transition is detected at the above port bit, two events take place. They are:

10. The three *read-only* status bits of the timer control and status register are:
11. The two TCSR *control* bits associated with the input capture register are:
12. The two TCSR *control* bits associated with the output compare register are:
13. The TCSR *control* bit associated with the counter is:
14. State four tasks that the 6801 timer can perform.
15. The four basic timer operating modes are _____, _____, _____, and _____.
16. The output compare register is used to provide the _____ and _____ modes.
17. The input capture register is used to provide the _____ and _____ modes.
18. A continuous or one-shot output waveform is obtained via port _____, bit _____ (pin _____).
19. A waveform to be measured is input via port _____, bit _____ (pin _____).
20. Explain how you would use the 6801 timer to measure frequency.

ANSWERS

1. counter (address 0009:000A)
output compare register (address 000B:000C)
input capture register (address 000D:000E)
timer control and status register (address 0008)
2. 16, the internal clock frequency

3. read-only, RESET
4. A read/write register.
5. TCSR bit 6—output compare flag (OCF) is set.
If TCSR bit 3 (EOCI) is set, an IRQ2/timer output compare interrupt will be generated.
6. Store accumulator D to address 000B (STAD → 000B)
7. a read-only register.
8. port 2, bit 0 (P20)
9. The present counter contents are stored in the input capture register.
If TCSR bit 4 (EICI) is set, an IRQ2/timer input capture interrupt will be generated.
10. bit 7—input capture flag (ICF)
bit 6—output compare flag (OCF)
bit 5—timer overflow flag (TOF)
11. Bit 4—enable input capture interrupt (EICI)
bit 1—input edge (IEDG)
12. bit 3—enable output compare interrupt (EOCI)
bit 0—output level (OLVL)
13. bit 2—enable timer overflow interrupt (ETOI)
14. Continuous pulse generation, event counting, event timing, interval measurement.
15. single-shot, continuous, period measurement, pulse-width measurement
16. single-shot and continuous
17. period measurement and pulse-width measurement
18. port 2, bit 1 (pin 9)
19. port 2, bit 0 (pin 8)
20. See Example 6-3. In addition, a routine must be added to take the inverse of the timed interval (period).

CHAPTER 7

How To Use the On-Chip Serial Communications Interface (SCI)

INTRODUCTION

In the previous chapters, you saw how the 6801 could be configured to provide up to 29 *parallel* i/o lines. Given one of the 6801 8-bit i/o ports, data can be sent to, or received from, the 6801 with eight bits in parallel. When interfacing this port to the outside world, eight lines are needed between the 6801 and a peripheral device. When interfacing over long distances, eight separate lines would have to be used over the whole distance. However, if the data could be sent in *serial* form (one bit at a time), only one or two lines would be required. Since the 6801 is a parallel system, the parallel data byte must be converted to serial form for serial data transmission, and the serial data must be converted back to parallel when it is received. This can be accomplished in two ways: by software or hardware. The software method is relatively slow and it generally means that the microcomputer is being used to "serialize" the data when it could be better used elsewhere. Therefore, most microcomputer systems use the hardware method of conversion in the form of a universal asynchronous receiver-transmitter (UART) or universal synchronous receiver-transmitter (USRT). The *asynchronous* device (UART) is generally used for a low- to medium-speed operation, and the synchronous device (USRT) is used for high-speed operation.

In first generation microprocessors such as the 6800, you needed to provide the external hardware required for the serial/parallel

translations. In the 6800 family, this hardware would be composed of either the MC6850 *asynchronous* interface adapter (ACIA) or the MC6852 *synchronous* serial data adapter (SSDA). However, in the 6801, the serial/parallel conversion function has been integrated onto the same chip. The serial data section of the 6801 is called the *serial communications interface* (SCI). The SCI is essentially a UART and it operates via port 2. It has three user-addressable sections: the receiver, the transmitter, and the control section. These sections are accessed with software at addresses 0010 through 0013.

In this chapter you will learn how to use the 6801's SCI to provide the serial data communication required by such peripheral devices as teletypewriters (tty's), terminals (crt's), cassette tape recorders, magnetic disks, and modems.

OBJECTIVES

At the end of this chapter you will be able to do the following:

- Understand the difference between synchronous and asynchronous serial data transmission.
- Describe the format of asynchronous serial data.
- Describe the timing of asynchronous serial data.
- Describe the use of the start bit, stop bit, and parity bit in an asynchronous system.
- Describe the 6801 SCI operating modes.
- Describe the available 6801 SCI data formats.
- Understand how to select the desired data transmission format and baud rate.
- Describe the internal functional parts of the SCI.
- Understand how to access the user-addressable parts of the SCI for system configuration and serial data communication.
- Explain the use of port 2 for serial data communication.
- Explain the 6801 serial data transmission operation.
- Explain the 6801 serial data reception operation.

SERIAL COMMUNICATIONS

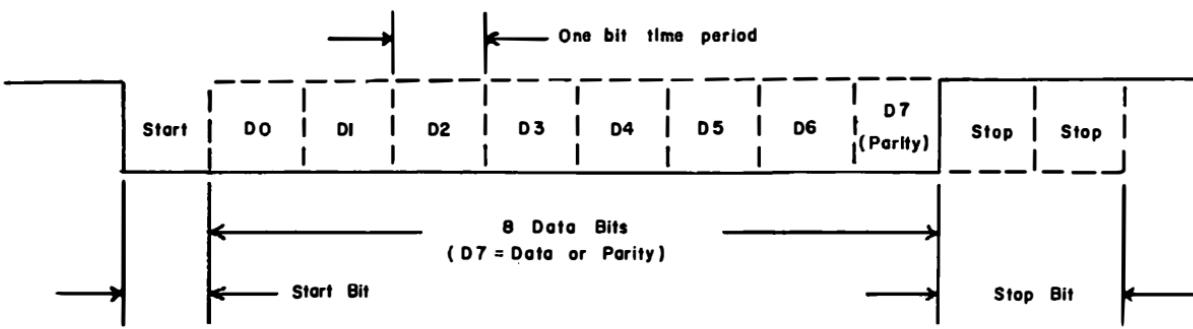
Before we begin a discussion of the 6801 serial communications interface (SCI), we will examine serial communications in general. In any digital computer system, two types of serial data transmission are possible. They are *synchronous transmission* and *asynchronous transmission*. The word synchronous means "clocked" and, therefore, with synchronous data transmission the data transmission is controlled by the system clock. The clock *synchronizes* the receiver and transmitter in much the same way that the "sync"

pulse in a tv signal synchronizes the tv camera and tv receiver scanning. In a digital system using synchronous transmission, the receiver will normally request data from the transmitter, then wait a fixed clocked period before reading the data. During the waiting period, the transmitter will place the data on the bus. In this way, the transmission is coordinated between both the receiver and transmitter. A hardware device which will provide parallel-to-serial and serial-to-parallel data conversions and transmit using synchronous transmission is the USRT (universal synchronous receiver-transmitter). In the 6800 family, the MC6852 synchronous serial data adapter (SSDA) is used to provide the USRT function.

With *asynchronous* transmission, the transmission is not directly controlled by the system clock. In this type of transmission, "start" and "stop" bits are added to the data word by the transmitter such that the receiver knows where each word begins and ends. A diagram of the asynchronous data format is shown in Fig. 7-1. For standard ASCII (American Standard Code for Information Interchange) transmission, eight data bits are "*framed*" with a start bit and a stop bit. The start and stop bits are added by the transmitter and detected by the receiver. The start bit is detected as a high-to-low transition, sometimes referred to as a *mark-to-space* transition. This bit signifies the beginning of the data word. The last bit of the serial data word is the stop bit which signals the end of that word. The stop bit is a high (logic 1) signal and lasts for two "bit time" periods. The bit time period is determined by the data transmission rate expressed as *baud rate* or *bits per second*. Baud rate is the measure of the number of signal elements transmitted or received per second. For example, if the specified baud rate is 300, then each signal element will have a bit time duration of 1/300 or 3.33 milliseconds. If each signal element carries one data bit, then the baud rate is numerically equal to the number of data bits transmitted or received per second. However, the format shown in Fig. 7-1 includes the start and stop bits; therefore, the baud and data bit rates are not equivalent. We will use baud rate in this chapter since the 6801 SCI specifications are given in baud rates. However, keep in mind that this rate is not equivalent to the data bit rate since the 6801 uses the format shown in Fig. 7-1.

In many cases, the 8-bit data word will include a *parity bit* in the eighth bit position (D7) of the data word. The transmitter will add the parity bit. Two types of parity are used: odd and even parity. If odd parity is used, the sum of the 1's transmitted, including the parity bit, will be odd. If even parity is used, the sum will be even. Sometimes during data transmission, a bit or bits get changed which results in an incorrect data word being received. However, with a parity system, the receiver checks the parity of the trans-

Fig. 7-1. Asynchronous serial data transmission format.



mitted word in an effort to catch such bit-change errors. This technique will detect a 1-bit error but 2-bit errors will go undetected.

A hardware device which will provide asynchronous serial data transmission per the format shown in Fig. 7-1 is the UART (universal asynchronous receiver-transmitter). The 6800 family equivalent of the UART is the MC6850 asynchronous communications interface adapter (ACIA). However, as you will see shortly, the 6801 serial communications interface (SCI) provides for asynchronous serial data transmission without use of any external UART or ACIA.

SCI MODES AND DATA FORMATS

The SCI is simply an on-board universal asynchronous receiver-transmitter (UART). It can perform asynchronous serial data communication in essentially two different modes, full and half duplex as shown in Fig. 7-2. Recall that half duplex is two-way serial data communication using one line (one direction at a time) and full duplex provides the transmit/receive functions on two separate lines. Naturally, with half duplex the transmit/receive communication cannot be simultaneous as with full duplex. Simplex is "simply" a special case of half duplex where only one-way communication is required.

MC6801 SERIAL COMMUNICATION INTERFACE DATA CHANNELS

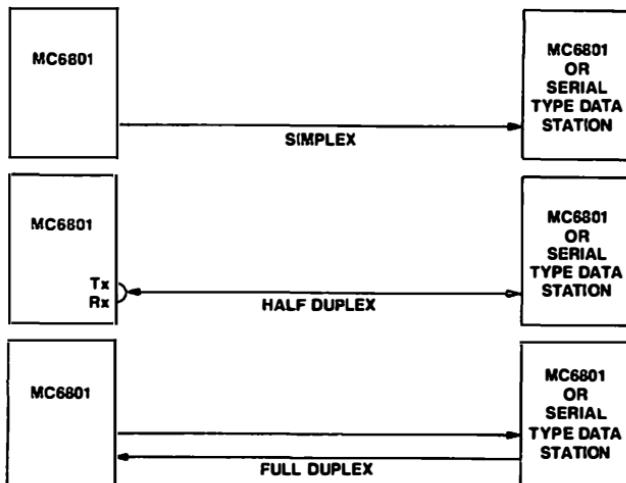


Fig. 7-2. 6801 serial communications interfacing modes.

In each mode, communication can take place in one of two data formats: standard mark/space for standard serial interfaces, or biphase for interfacing between 6801 processors. The mark/space or NRZ (non-return-to-zero) format uses amplitude changes to represent binary information. (Typically, 0-V and 5-V levels are used to represent 0's and 1's, respectively. The biphase or fm (frequency modulation) format uses amplitude changes coupled with a change in frequency to represent 0's and 1's. As with fm radio broadcast, the advantage here is that biphase is less sensitive to extraneous amplitude variations (noise), and provides greater line-noise immunity in the data transmission link. As you will soon see, special precautions must be taken when using the mark/space format to reduce the noise problem. Both of the SCI data formats are shown in Fig. 7-3. Note that each data word will consist of ten bits, the eight data bits (D0-D7) which are preceded by a start bit and followed by a stop bit. The start and stop bits define or *frame* the data word. The start bit is detected as a high-to-low (mark-to-space) transition, and the stop bit is represented with a high (logic 1) level. The data word 01001101 (4D) is shown in both the mark/space and the biphase format in Fig. 7-3. Note that in the biphase format, there is a logic level transition at the start of each bit interval, and a transition at the center of each bit which should contain a logic 1. In effect, this scheme doubles the data frequency for logic 1 levels and thus is referred to as an fm data transmission.

The serial data may be transmitted at one of four rates by using

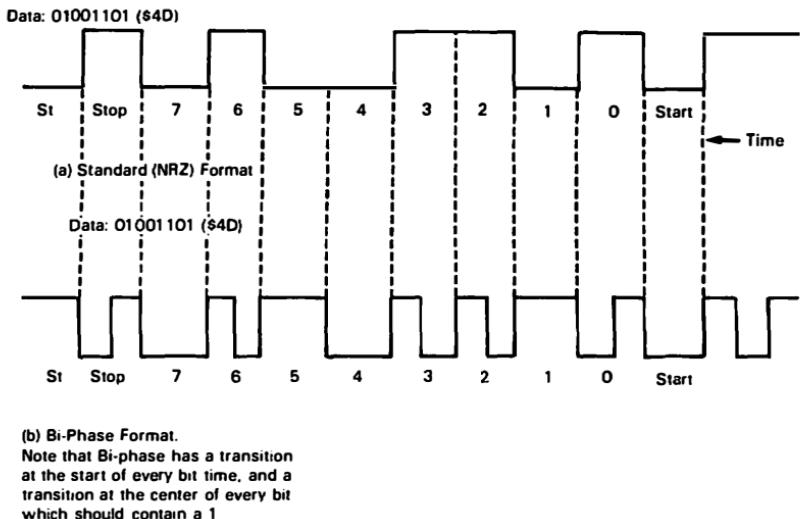


Fig. 7-3. Serial communications interface (SCI) data formats

the internal 6801 clock, or at any desired rate using an external clock signal, with some limitations. The SCI bit time duration and baud rates available for three different internal clock frequencies are shown in Fig. 7-4. Note that there are four rates available for each frequency. You will select one of the four available rates by software configuration of the rate and mode control register, to be discussed shortly. You may also control the baud rate by the appli-

Rate		XTAL -	4.0 MHz	4.9152 MHz*	2.4576 MHz
S1, S0		$\phi 2$	1.0 μs /1.0 MHz	814 ns/1.2288 MHz	1.63 μs /614.4 KHz
0 0	$\div 16$	16 μs /62,500 Baud	13.0 μs /76,800 Baud	26 μs /38,400 Baud	208 μs /4,800 Baud
0 1	$\div 128$	128 μs /7812.5 Baud	104.2 μs /9,600 Baud	1.67ms/600 Baud	6.67ms/150 Baud
1 0	$\div 1024$	1.024ms/976.6 Baud	833.3 μs /1,200 Baud		
1 1	$\div 4096$	4.096ms/244.1 Baud	3.33ms/300 Baud		

*the 68A03 should be used if this frequency is desired

Fig 7-4 Serial communications Interface (SCI) bit times and baud rates

cation of an external clock signal at P22 (pin 10). The clock frequency applied must be eight times ($\times 8$) the desired baud rate. However, this signal must *not* exceed the 6801's internal clock frequency.

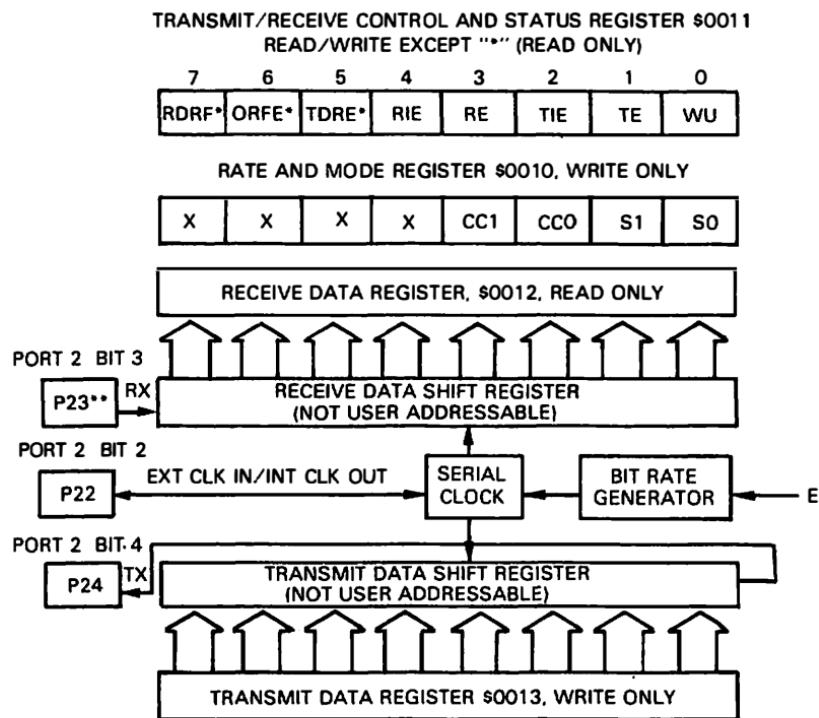
Serial data is transmitted at P24 (pin 12), and serial data is received via P23 (pin 11). The transmit and receive functions operate independently of each other but always in the same data format and baud rate for a single 6801. For full-duplex operation they would each operate from a separate data line, and for half duplex operation they would share a common data line. Simplex operation would involve either the transmit or receive function (not both) at P24 or P23, respectively.

SCI INTERNAL STRUCTURE

Now, you will examine the structure of the 6801 serial communications interface (SCI). The SCI consists of four user-addressable registers. They are:

- an 8-bit *write-only* transmit data register,
located at address 0013
- an 8-bit *read-only* receive data register,
located at address 0012
- an 8-bit control and status register,
located at address 0011
- a 4-bit *write-only* rate and mode control
register, located at address 0010

These registers are shown in Fig. 7-5 along with the serial data shift registers which are not directly accessible to the user. Note that serial data bits are input to the receive data shift register via P23, where they are then *shifted* into the proper bits of the receive data register, one bit at a time, until all 10 have been received. You may read the data using a load operation for address 0012. To trans-



**P23 refers to Port 2, bit 3.

Fig. 7-5. Serial communications interface (SCI) registers.

mit data, you write (store) the data word to be transmitted to the transmit data register at address 0013. The data will then be shifted by the transmit data shift register to provide a serial output at P24. The start and stop bits are automatically added during data transmission and are automatically detected and removed when the data word is received. Also, note from Fig. 7-5 that the *bit rate generator* is controlled by the enable (E) internal clock pulse. The bit rate generator, in turn, controls the *serial clock signal* to establish the baud rate, and this serial clock's frequency *can* be output at P22 (pin 10) if desired. Also, as previously mentioned, P22 may be used

as an input, to supply an externally generated clock signal that will determine the data transmission and reception rates. The external clock frequency must be eight times the desired bit rate. In this case, the externally supplied frequency signal will drive the serial clock.

Now, as we did with the 6801 timer, each of the SCI user-addressable registers will be discussed, then you will see how they can be used in conjunction with each other to provide serial data communication.

Transmit Data Register

This is an 8-bit write-only register located at address 0013. You will store a data word at this address. If the transmit function is enabled the data word will be transferred to the output shift register and serial transmission of the data word will begin on P24 (pin 12). The necessary start bit and stop bit are automatically added by the output shift register circuits on the chip.

Receive Data Register

This is an 8-bit read-only register located at address 0012. Properly formatted serial data coming in to the 6801 via P23 (pin 11) will be shifted into the proper bit positions of this register. You may then perform a read (load) operation from address 0012 to obtain the data.

Transmit/Receive Control and Status (TRCS) Register

The TRCS register is an 8-bit register of which all eight bits may be read; however, only the five least significant bits (LSBs) may be written. This register is similar to the timer control and status register (TCSR) in that the five LSBs (0-4) provide enable/disable functions for the SCI, with the three most significant bits (MSBs) (5-7) being used as status flags for the SCI. A description of each TRCS bit, beginning with bit 7 (MSB) follows.

RDRF, the receive data register full (bit 7) (read-only bit) acts as a status flag to indicate when a transfer from the receiver's input shift register to the receiver's data register has taken place. You will normally read the TRCS to check the status of this flag. If set (logic 1), you will then read the receive data register to obtain the data. This process of reading the TRCS register and then the receive data register will also *clear* the flag so that another data word may be received. If the flag is a logic zero when tested, no new information has been received since the last read operation took place. The flag is also cleared by RESET.

ORFE, the overrun framing error (bit 6) (read-only bit) acts as a status flag which is set when an *overrun* or *framing* error occurs

while information is being received. An overrun condition will occur when a new data byte is received before the last data byte has been read. The new byte will *not* be transferred to the receive data register as long as RDRF is set; therefore, the previously received byte is not lost. A framing error will occur when the data word is not properly "framed" to the required data format; that is, no stop bit, improper data rate, etc., or when the byte boundaries in the bit stream are not properly synchronized to the bit counter. In either case, the ORFE flag will be set to indicate an error. Of course, you will not be able to determine *which* error took place, just that an error was detected. To determine exactly which error has occurred, you must also check the RDRF flag status. If RDRF=ORFE=1, then an overrun error has occurred. If RDRF=0 and ORFE=1, a framing error has been detected. The flag is cleared the same way you clear the RDRF (bit 7) flag: *by reading the TRCS register, then the receive data register.* It is also cleared by **RESET**.

TDRE, the transmit data register empty (bit 5) (*read-only bit*), acts as a status flag for the SCI transmit function. The flag will set when a transfer is made from the transmitter's data register to the output shift register, indicating that a data word has been transmitted. You will read the TRCS register to check the status of this flag. If set (logic 1), then the last data word has been transmitted and you may write a new word to the transmitter's data register for subsequent transmission. Reading the TRCS register followed by writing to the transmitter's data register will also clear the TDRE flag (logic 0) to indicate that the SCI is *busy* transmitting the new information. At the end of the transmission, the flag is set back to a logic 1, indicating that the transmitter section is ready for the next data byte that is to be transmitted. Performing a **RESET** operation will *set* the TDRE flag.

RIE, the receiver interrupt enable (bit 4) (*R/W bit*), is a read/write bit that is used to enable the IRQ2/serial i/o interrupt. When set (logic 1), it will permit an IRQ2/serial i/o interrupt to be generated when either one of the following two events occurs:

RDRF (bit 7) is set, indicating a data word has been received.

ORFE (bit 6) is set, indicating an overrun or framing error.

In either case, the I-flag of the condition code register (CCR) must be cleared for the interrupt to be acknowledged.

You may set and clear the RIE flag through program control. When cleared, the interrupt will be masked-out and a software routine would be required to check the receiver flag status. When detecting a flag's status without the use of an interrupt, the processor must periodically break from what it is doing and read the status register. Recall that the flag status can then be detected by rotating

the status register bits into the carry (C) bit of the condition code register. This wastes valuable processor time, since the status check is made whether or not the flag is active. However, if the status flag has an interrupt capability, as do the SCI status flags, processor time can be saved since the processor only needs to service the interrupt when the flag is active.

RE, the receiver enable (bit 3) (R/W bit), configures P23 (pin 11) for serial data input. When set, serial data entering P23 will be gated into the input shift register *regardless* of the port 2 data direction register value for P23. Setting the RE bit will automatically clear the P23 data direction register bit. You may set and clear the RE bit through program control. When clear, the serial receiver function is disabled and P23 can be used for parallel data i/o via its i/o register at address 0003.

TIE, the transmit interrupt enable (bit 2) (R/W bit), is similar to the RIE bit in that it enables the IRQ2/serial i/o interrupt. When set, it will permit an interrupt to be generated when the TDRE bit (bit 5) is set, indicating that data word has been transmitted. The interrupt will only be acknowledged if the I-flag of the CCR is clear. Clearing the TIE bit will also mask-out the interrupt.

There is only one IRQ2 interrupt for the SCI. Therefore, the interrupt service routine must determine if the interrupt has been generated as a result of a data word being received, transmitted, or received in error. You will make this determination through software steps in the service routine, by reading the contents of the TRCS register and checking the status of the TDRE, ORFE, and RDRF status flags.

TE, the transmit enable (bit 1) (R/W bit) configures P24 (pin 12) for serial data output. When set (logic 1), serial data will be gated from the output shift register of P24 regardless of the port 2 data direction register's value for P24. Setting the TE bit will automatically set the P24 data direction register bit. Following a RESET operation, you will normally have to configure the TRCS register for serial i/o operations. If the TE bit is set during this configuration, the serial output will be initiated by first transmitting a *10-bit preamble of 1's*. That is, a logic 1 state is transmitted for 10-bit time periods. Therefore, information cannot be sent immediately after the TE bit is set. You must wait at least 10-bit time periods before sending the first word. It should be noted here that an *idle line* is represented by a constant logic 1 state being transmitted. In NRZ format, this is represented by a constant mark (logic 1) on the line. In biphase format an idle line will be toggling every half-bit time. Thus, the 10-bit preamble of 1's represents an idle line for 10-bit time periods. After this preamble has been transmitted, internal synchronization is established and the transmitter section of the SCI

is ready for operation. You may set and clear the TE register under program control. When clear, the serial transmit function is disabled and P24 can be used for parallel data i/o via its i/o register at address 0003.

WU, the wake up on next message (bit 0) (R/W bit), is also a read/write bit that enables and disables the wake-up feature of the 6801. This feature is used in multiprocessor applications. When messages are sent and received on a common multiprocessor transmission line, the initial byte(s) of the message will usually include a specific processor destination address. However, nonselected processors should ignore the message. To accomplish this in the 6801, you can set the WU bit in the nonselected processors. Once the message is completed, an idle line that is a logic one for 10-bit time periods will indicate an idle transmission line and will "wake-up" the "sleeping" processors by automatically clearing the WU bit. The wake-up bit (WU) simply enables and disables the RDRF status flag. When the wake-up bit (WU) is set, the receiver section will continue to process the message, but the RDRF status flag is disabled and will not set. Upon receiving an idle line which is a logic 1 for 10-bit time periods, the receiver section will clear the wake-up (WU) bit and thus enable the RDRF flag for normal operation. It should be noted that the ORFE status flag is also disabled when the wake-up bit (WU) is set. Also, you cannot set the wake-up bit if the line is idle.

That completes our discussion of the transmit/receive control and status (TRCS) register. The preceding discussion is summarized in Table 7-1.

Rate and Mode Control Register

The last user-addressable register we need to discuss is the rate and mode control register. This is an 8-bit *write-only* register located at address 0010. Of the eight bits, only the four least significant bits are used. As you have probably guessed from the name, the register is used to select the baud rate and data format. Besides these two functions, this register is also used to specify an internal or external serial clocking source and to properly configure port 2, bit 2 (P22). Bit 2 of port 2 is used *only* if the internal clock-out or external clock-in options are selected. The register bit functions are summarized in Fig. 7-6 and Tables 7-2 and 7-3. The four functional bits can be broken down into a pair of 2-bit fields. The two lower order bits (S0, S1) control the SCI baud rate and the two higher order bits (CC0, CC1) control the serial data format and clock select logic.

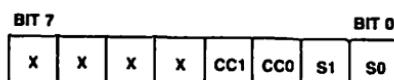
The available baud rates for three separate internal clock frequencies are shown in Table 7-2. Note that you can select one of four

Table 7-1. Transmit/Receive Control and Status (TRCS) Bit Summary

TRCS Register	Set	Cleared
Bit 0 (WU)	By software: Allows processor to ignore multiprocessor system messages	By an Idle line which is a logic 1 for 10-bit time periods or by software or by <u>RESET</u> —“wakes-up” processor
Bit 1 (TE)	By software: Enables transmit section via P24	By software or <u>RESET</u> —disables transmit function
Bit 2 (TIE)	By software: Enables IRQ2 Interrupt to be generated when TDRE sets	By software or <u>RESET</u> : Masks-out IRQ2 Interrupt for the transmit function
Bit 3 (RE)	By software: Enables receive section via P23	By software or <u>RESET</u> : Disables receive function
Bit 4 (RIE)	By software: Enables IRQ2 Interrupt to be generated when either RDRF or ORFE sets	By software or <u>RESET</u> : Masks-out IRQ2 Interrupt for the receive function
Bit 5 (TDRE)	When data has been transmitted or by <u>RESET</u>	By reading TRCS followed by writing to the transmit data register
Bit 6 (ORFE)	When overrun or framing error occurs	By reading TRCS, then the receive data register or <u>RESET</u>
Bit 7 (RDRF)	When data has been received	By reading TRCS, then the receive data register or <u>RESET</u>

MC6801 MCU

SERIAL COMMUNICATIONS INTERFACE RATE AND MODE CONTROL REGISTER



S1, S0 — SPEED SELECTS FOR SERIAL BIT STREAM
CC1, CC0 — CLOCK CONTROL AND DATA FORMAT SELECT

Fig. 7-6. Rate and mode control register bit functions.

Table 7-2. SCI Internal Baud Rates

S1,S0	XTAL	4.0 MHz	4.9152 MHz	2.5476 MHz
00	$\phi 2$	1.0 MHz	1.2288 MHz	0.6144 MHz
	$\phi 2 \div 16$	62.5k Bits/s	76.8k Bits/s	38.4k Bits/s
01	$\phi 2 \div 128$	7,812.5 Bits/s	9,600 Bits/s	4,800 Bits/s
10	$\phi 2 \div 1024$	976.6 Bits/s	1,200 Bits/s	600 Bits/s
11	$\phi 2 \div 4096$	244.1 Bits/s	300 Bits/s	150 Bits/s

baud rates for each frequency by using the proper S1, S0 combination. For frequencies not shown here, the four available baud rates would be the internal frequency is divided by 16, 128, 1024, and 4096, respectively. Note that the frequency of the crystal should be 4.9152 MHz or 2.4576 MHz to give the *standard* baud rates. If external clocking is selected, the S1, S0 bits are ignored.

Table 7-3. SCI Format and Clock Control

CC1, CC0	Format	Clock Source	Port 2 Bit 2	Port 2 Bit 3	Port 2 Bit 4
00	Bi-Phase	Internal	Not Used	†	†
01	NRZ	Internal	Not Used	†	†
10	NRZ	Internal	Output*	Serial Input	Serial Output
11	NRZ	External	Input	Serial Input	Serial Output

*Clock output is available regardless of values for bits RE and TE.

†Bit 3 is used for serial input if RE = "1" In TRCS; bit 4 is used for serial output if TE = "1" In TRCS.

The format and clock control options are shown in Table 7-3. First, you can select which one of the two basic serial data formats (biphase or NRZ) is to be used. Second, you can select the source of the serial data clock which can be either internal (based on the 6801's clock frequency) or external. If the internal clock is selected, the baud rates are then selected according to Table 7-2. You can also choose to have the serial clock signal output via P22 (CC1, CC0 = 10). However, with this option, the *maximum* available output clock rate is $\phi 2 \div 16$. The output signal at P22 will be at the same frequency as the bit rate and it will have a rising edge at approximately the mid-bit point of each data bit. If you desire to provide an external clock signal for serial data synchronization, the CC1, CC0 bit field in the rate and mode control register must be set to 11. With this option, the only data format available is mark/space (NRZ). The external clock signal must be input on P22 (pin 10). The external clock frequency must be eight times ($\times 8$) the desired baud rate and it *cannot* exceed the 6801's *internal* clock frequency. Finally, all the functional rate and control register bits are cleared on RESET.

SCI OPERATION

You will now see how the various SCI registers just discussed are used together to perform the serial transmit and receive functions. Once you understand the operations required for both the transmit and receive modes, your serial data communication programs will be relatively easy to write. A flowchart will be used to explain the SCI operation.

The transmit operation is shown in Fig. 7-7. A brief explanation of each operational step will now be provided. Assuming the se-

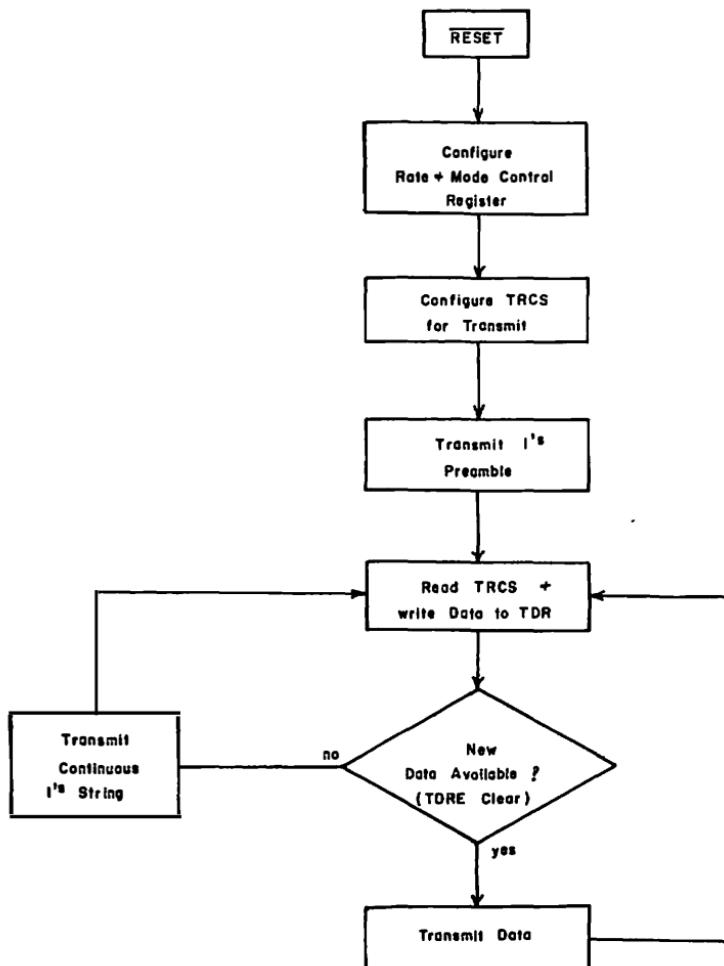


Fig. 7-7. Transmit operation flowchart.

quence begins with a ~~RESET~~, the first thing you must do is to configure the rate and mode control *and* the transmit/receive control and status (TRCS) registers. You will configure the rate and mode control register at address 0010 to select the desired data format, clock options, and baud rate per Tables 7-2 and 7-3. The TRCS register is then configured at address 0011. Recall that you can only write to the first five bits (0-4) in this register. Bits 1 and 2, transmit enable (TE) and transmit interrupt enable (TIE) are the two bits of interest here. You must set the TE bit (bit 1) to enable the transmitter portion of the SCI. You will set the TIE bit (bit 2) if you desire an IRQ2/serial i/o interrupt to be generated each time a data word is transmitted. Setting the TE bit during the TRCS configuration will cause a logic 1 to be transmitted for 10-bit time periods. This will establish internal synchronization for data transmission and indicate that the transmitter section of the SCI is ready for operation. You will now read the TRCS register with a load operation at address 0011 and store the first data byte into the transmit data register (TDR) with a write operation to address 0013. This accomplishes two things: first, it prepares the first data byte for transmission and second, the read TRCS followed by a write to the TDR clears the TDRE flag (bit 5) of the TRCS register. The data byte will *not* be transmitted unless the TDRE status flag is clear. Recall from Table 7-1 that the ~~RESET~~ operation had set the TDRE flag. In the next operation, the SCI checks to see if the TDRE status flag is clear. If it is not clear, the transmitter goes into the idle mode, and its output is always a logic 1. If the flag is clear, the data word, including the start and stop bits, will be transmitted. The start bit is transmitted first, followed by the eight data bits beginning with bit zero. Then, the stop bit is transmitted. Once the first word is transmitted, you will read the TRCS. The TDRE status flag (bit 5) should be set, indicating that the transmission has taken place and the transmit data register (TDR) is empty. If so, you will then write the new data byte to the TDR. The read TRCS followed by a write to the TDR will clear the TDRE status flag and allow transmission of the new data. Note: If the TDRE bit is still set when the next data byte transfer *should* occur, an idle transmission line (constant logic 1) will be sent until more data is provided.

You will then repeat the above process, until the entire message has been transmitted. If you had set the transmit interrupt enable bit (bit 2) during the TRCS configuration, an IRQ2/serial i/o interrupt would be generated each time a word was transmitted. The interrupt service routine would then read the TRCS and write the data bytes to the transmit data register.

The receive operation is shown in Fig. 7-8. Again, you will con-

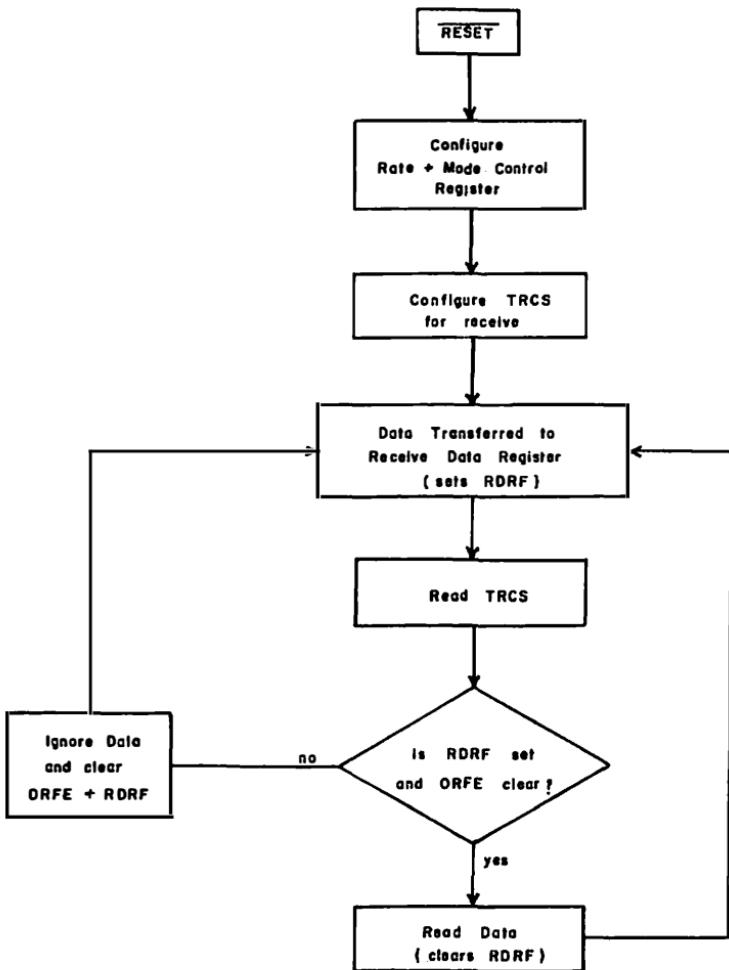


Fig. 7-8. Receive operation flowchart.

figure the rate and mode control register for the required data format, clock options and baud rate per Tables 7-2 and 7-3. Then, you will configure the TRCS for the receive operation. The RE bit (bit 3) must be set to enable the receiver portion of the SCI. You can set the RIE bit (bit 4) if you desire an IRQ2/serial i/o interrupt to be generated each time that a data word is received. Once the RE bit is set, data will be gated from P23 into the receive data register (RDR). In standard NRZ format, the receiver is then immediately ready to accept data; however, in biphase format, an idle line must be presented to the receiver for at least one-bit time to permit synchronization to occur. When data is received, the RDRF status flag

in the TRCS register will set. You will then read the TRCS register and verify that the RDRF flag is set, indicating the receive data register is full and verify that the ORFE status flag is clear. If ORFE is set, an overrun or framing error exists and the data should be ignored. If RDRF is set and ORFE is clear, you will then read the valid data word from the receive data register at address 0012. This operation, coupled with the previous read TRCS operation, will clear the RDRF status flag to allow the reception of a new data word. You will continue this process until the entire message is received.

REVIEW QUESTIONS

1. Describe the difference between simplex, half-duplex, and full-duplex serial data transmission.
2. Describe the difference between NRZ and biphase serial data transmission.
3. The ten bits which comprise the serial data word are:
4. The start bit is represented with a _____ transition.
5. The stop bit is represented by a _____ level.
6. Serial data transmission speed is measured in _____.
7. Given an internal clock frequency, there are _____ user selectable transmission speeds available. (how many?)
8. When controlling the transmission speed by an external clock source, the external signal is applied at pin _____ and must be _____ times the desired baud rate but must not exceed _____.
9. Serial data is transmitted via _____ and received via _____ on the 6801.
10. The four SCI user addressable registers and their addresses are:
11. The *transmitter* section of the SCI has three associated *status* and *control* bits in the TRCS register. They are:

12. The *receiver* section of the SCI has four associated *status* and *control bits* in the TRCS. They are:
13. When is the "wake-up" feature of the SCI normally utilized?
14. The IRQ2/serial i/o interrupt service routine vector is located at address _____.
15. The desired baud rate is selected with bits _____ and _____ of the rate and mode control register.
16. The desired serial data transmission format is selected with bits _____ and _____ of the rate and mode control register.
17. The four baud rates available can be calculated by dividing the internal clock frequency by _____, _____, _____, and _____, respectively.
18. If an external synchronization signal is applied at P22, the only serial data format available is _____.
19. Describe the serial data transmit operation.
20. Describe the serial data receive operation.

ANSWERS

1. Simplex is one-line, one-way serial data communication.
Half-duplex is one-line, two-way serial data communication.
Full-duplex is two-line, two-way serial data communication.
2. NRZ (non-return-to-zero) or mark/space format uses amplitude changes to represent binary information. Biphasic or fm format uses amplitude changes coupled with a change in frequency to represent binary information.
3. start bit
eight data bits (D0-D7)
stop bit
4. high-to-low
5. high (logic 1)
6. baud rate
7. four
8. pin 10 (P22), 8X, the internal clock frequency
9. P24 (pin 12), P23 (pin 11)

10. transmit data register (address 0013)
receive data register (address 0012)
transmit/receive control and status register (address 0011)
rate and mode control register (address 0010)
11. bit 1—transmit enable (TE)
bit 2—transmit interrupt enable (TIE)
bit 5—transmit data register empty (TDRE)
12. bit 3—receive enable (RE)
bit 4—receive interrupt enable (RIE)
bit 6—overrun/framing error (ORFE)
bit 7—receive data register full (RDRF)
13. For multiprocessor applications.
14. FFF0:FFF1
15. 0 and 1 (S0, S1)
16. 2 and 3 (CC0, CC1)
17. 16, 128, 1024, 4096
18. NRZ
19. Refer to Fig. 7-7.
20. Refer to Fig. 7-8.

CHAPTER 8

General Interfacing

INTRODUCTION

Now that you are familiar with the 6801 and its various operating modes, you will see how this powerful chip can be interfaced to the "outside world." We will begin our discussion of 6801 interfacing with switch interfacing. Many microcomputer input devices are simply groups of switches. Even a complicated keyboard can be broken down to a set of single switches. Therefore, it is important that you understand the basics of interfacing to a single switch. Once this is accomplished, you will see how a group of switches arranged in both a switch column and switch matrix can be interfaced to your 6801. This will lead you to a discussion of unencoded and fully encoded keyboard interfacing.

A frequently used output device in the microcomputer industry is the 7-segment LED display. In this chapter we will discuss 7-segment LEDs and you will see how they can be interfaced to your 6801. A single display will be interfaced first, then you will see how a group of displays can be interfaced to display intelligible messages. You will be given programs throughout the discussion to show you how the 6801 is configured and the display characters are generated.

Finally, many electrical and mechanical input and output devices produce or require a continuous range of voltage or current values rather than two-state binary logic. For example, a thermocouple will produce continuous voltage values as a function of temperature. These continuous values are referred to as *analog* signals. In order for the 6801 to recognize these signals and to operate on them, they must be *converted* to *digital* information. Other devices, such

as motors, might require that the digital information produced by a microcomputer system be converted to analog signals for control purposes. Fortunately, there are single chips that will perform these conversions. They are referred to as digital-to-analog (d/a) and analog-to-digital (a/d) converters. In this chapter, you will see how the following converters can be interfaced to your 6801.

Signetics NE5018 d/a converter.

Motorola MC1408-8/MC1508-8 d/a converter.

Intersil ICL7109 a/d converter.

Teledyne 8703 a/d converter.

OBJECTIVES

At the end of this chapter you will be able to do the following:

- Interface your 6801 to a single push-button switch.
- Interface your 6801 to a switch column.
- Interface your 6801 to a switch matrix.
- Interface your 6801 to an unencoded or fully encoded keyboard.
- Interface your 6801 to a single 7-segment LED display.
- Multiplex several 7-segment LEDs to provide for the display of messages.
- Interface your 6801 to the Signetics NE5018 and Motorola MC1408-8/MC1508-8 d/a converters.
- Interface your 6801 to the Intersil ICL7109 and Teledyne 8703 a/d converters.

INTERFACING WITH SWITCHES AND KEYBOARDS

Switches

You will begin by interfacing your 6801 to a single pushbutton switch. Then, you will interface to a switch column and matrix. When interfacing to a switch or switches, there are four basic interfacing requirements. They are *switch addressing*, *detecting switch closure*, *switch debouncing*, and *switch decoding*.

The first requirement, switch addressing, is simple for the 6801 since addressing is accomplished by addressing the respective 6801 port. No external address decoding is required. You can connect a single push-button switch to any one of the 6801 port lines, as shown in Fig. 8-1, as long as that line is not needed for another function. Port 1 has been chosen here since this port is always used for i/o, regardless of the chip mode.

Once you have addressed the switch or switches, you must detect a switch closure, if any. Note that in Fig. 8-1, switch S1 is con-

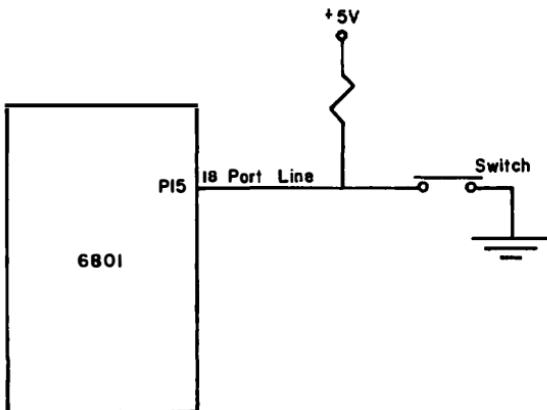


Fig. 8-1. Interfacing to a single push-button switch.

nected to the 6801 through a pull-up resistor. This will cause a logic 1 to appear on that line until the switch is closed. You can use a logical AND operation to determine a device condition. The procedure involves AND'ing a *status* byte with a *mask* byte to determine if a device is on or off. In this case, your device is a switch. A logic 1 indicates that the switch is open, while a logic 0 indicates a closed switch. The status byte will be the input data byte from the respective port. The mask byte will mask-out all other bits of that port with zeros and contain a "1" in the bit position of the incoming switch signal.

For example, if the switch were connected to P15, the mask byte would be 0010 0000 (20_{16}). Now, if the switch at P15 is open, the status byte would be 00100000. If you AND this with a mask byte of 0010 0000 the result is 0010 0000. However, if the switch is closed, the status byte would be 0000 0000. ANDing this with the mask byte would provide a result of 0000 0000. If this operation were performed by the 6801, the Z-flag of the condition code register would be set upon switch closure. Therefore, the Z-flag would act as a flag to indicate switch closure. Once the closure is detected, you can store the binary input information in a memory location. The program in the following example will simply store 01 in memory location 00A0 to indicate that switch closure has been detected. If memory location 00A0 is cleared, no closure has been detected. The program in Example 8-1 will address the switch by addressing port 1 and then detect switch closure.

You first configure port 1 as an input port. Then, the mask byte (0010 000) is AND'ed with the port 1 data until contact closure is detected. When closure is detected, memory location 00A0 is in-

Example 8-1: Detecting a Single Switch Closure

<i>Hex Address</i>	<i>Hex Contents</i>	<i>Mnemonic/Contents</i>	<i>Operation</i>
0080	7F	CLR \$\$	
0081	00	00	
0082	00	00	Clear port 1 data direction register (port 1 = Input)
0083	7F	CLR \$\$	
0084	00	00	
0085	A0	A0	Clear memory location 00A0
0086	96	LDAA \$	
0087	02	02	Read port 1
0088	84	ANDA #	
0089	20	20	AND status and mask bytes
008A	26	BNE	
008B	FA	FA	Branch If Z-flag clear (to address 0086)
008C	7C	INC \$\$	
008D	00	00	
008E	A0	A0	Increment memory location 00A0
008F	3E	WAI	Stop

cremented. This program was written to reside in the 6801's on-board R/W memory. Also, you would not need to configure port 1 for *input* if the program is executed immediately after *RESET*, since the *RESET* operation automatically clears the port 1 data direction register.

The next requirement for switch interfacing is debouncing. Each single closure of a mechanical switch does not produce a single voltage transition because the mechanical contacts "bounce," open-closed, open-closed, etc., for a few milliseconds before settling to a "firm" closure. This bouncing action looks like a series of 1's and 0's to a microcomputer. Therefore, you must wait until the bounce period ends before you read the data from the switch. This can be accomplished in one of two ways. You can use a cross-coupled *NAND* gate or inverter circuit, as shown in Fig. 8-2, that will immediately latch in one state and ignore all switch bouncing, or you can provide a software delay in your program during the bounce period. In the latter case, the 6801 is performing the debouncing action. A typical delay period is 10 milliseconds; however, this will vary from switch to switch. To provide software debouncing, you will provide a 10-millisecond delay after you detect contact closure. Then, after the delay, your program will read the data from the switch (switches) again. If closure is still detected, you can be

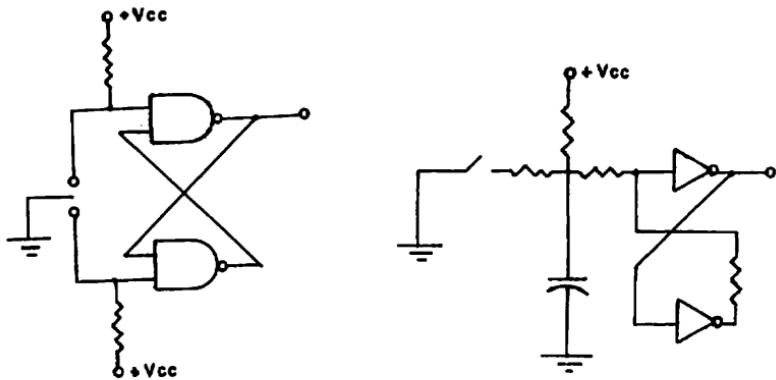
certain the switch is closed. The program in Example 8-2 will address the switch, detect closure, and provide software debouncing.

Example 8-2: Detecting Switch Closure and Providing Software Debouncing

Hex Address	Hex Contents	Mnemonic/Contents	Operation
0080	7F	CLR \$\$	
0081	00	00	
0082	00	00	Clear port 1 DDR (port 1 = Input)
0083	7F	CLR \$\$	
0084	00	00	
0085	A0	A0	Clear memory location 00A0
0086	96	LDAA \$	
0087	02	02	Read port 1
0088	84	ANDA #	
0089	20	20	AND status and mask bytes
008A	26	BNE	
008B	FA	FA	Branch If Z-flag clear (to address 0086)
008C	BD	JSR \$\$	
008D	00	00	Jump to subroutine at address 00F0
008E	F0	F0	(debouncing delay)
008F	96	LDAA \$	
0090	02	02	Read port 1
0091	84	ANDA #	
0092	20	20	AND status and mask bytes
0093	26	BNE	
0094	F1	F1	Branch If Z-flag cleared (to address 0086)
0095	7C	INC \$\$	
0096	00	00	Increment memory location 00A0
0097	A0	A0	
0098	3E	WAI	Stop
(DEBOUNCING DELAY: 10 ms)			
00F0	CE	LDX #	
00F1	05	05	05 → X _H
00F2	00	00	00 → X _L
00F3	09	DEX	Decrement the Index register
00F4	8C	CPX #	
00F5	00	00	Compare 0000 to the
00F6	00	00	Index register

00F7	26	BNE FA	Branch if Z-flag clear (to address 00F3)
00F8	FA		
00F9	39	RTS	Return to main program (address 008F)

You first configure port 1 for input by clearing its data direction register. (*Note:* this operation is not necessary if the program is executed after RESET.) Then, you clear memory location 00A0 since you will increment this location when you are sure of switch closure. Now, read the data at port 1 and perform a logic AND operation to determine switch closure. If no closure is detected (Z-flag cleared) you will branch back to read the port until closure is de-



(A) NAND debouncing circuit
spdt switch.

(B) INVERTER debouncing circuit
spst switch.

Fig. 8-2. Cross-coupled NAND gate debouncing circuit.

tected (Z-flag set). When closure is detected, the 6801 will call the subroutine located at address 00F0. This subroutine will provide a 10-millisecond delay for switch debouncing using the index register. Once the delay has been provided, the 6801 returns to the main program to read port 1 and again verify contact closure. If closure is detected again, memory location 00A0 is incremented. If no closure is detected the program will branch back to the first port 1 read operation and the cycle will repeat itself until a "firm" closure is detected.

The last requirement for switch interfacing is switch decoding. After the 6801 detects a switch closure, it must decide which switch is closed. Naturally, this is not a problem with a single switch. However, with multiple switches, a decoding procedure must be provided. Fig. 8-3 shows how four switches might be connected to your 6801 via port 1. Note that switches S1 through S4 are con-

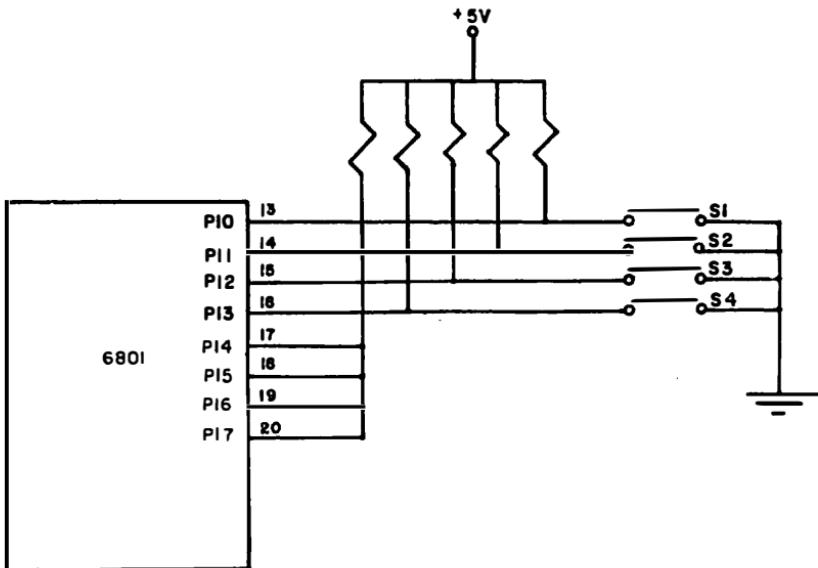


Fig. 8-3. Interfacing to a switch column.

nected to P10 through P13, respectively, and port 1 lines P14 through P17 are held high. When one of the switches is closed, its corresponding port line goes low; therefore, when you load port 1 data into the accumulator, the corresponding bit is also "0." All the other accumulator bits will be "1" and the 0 bit can be *decoded* by rotating the accumulator contents into the carry flag of the condition code register until that flag is cleared. The 6801 can then determine which switch is closed by counting the number of rotations that it takes to detect the cleared, or logic 0 bit position. Assuming that switch closure has been detected and debounced, the program in Example 8-3 will *decode* the switches. The program will load port 1 data into the accumulator, then rotate that data until the carry flag is cleared. After each rotation, the index register is incremented to count the number of rotations necessary to clear the C-flag. The proper switch number is then stored in memory location 00A0 from the incremented index register.

This program assumes that switch closure has been detected. When more than one switch is connected in a *column* to the 6801 as in Fig. 8-3, you will use a compare instruction rather than a logic AND to detect closure. In this case, comparing the input data immediately to FF would detect closure. If none of the switches were closed, the Z-flag would set as a result of the compare instruction. However, if a switch is closed, the Z-flag would clear as a

Example 8-3: Decoding a Switch Column

Hex Address	Hex Contents	Mnemonic/Contents	Operation
0080	96	LDAA \$	
0081	02	02	Read port 1
0082	CE	LDX #	
0083	00	00	
0084	00	00	Clear the Index register
0085	46	RORA	
0086	08	INX	
0087	25	BCS	
0088	FC	FC	Rotate right - ACCA Increment the index register
0089	DF	STX \$	
008A	9F	9F	Branch if carry set (to address 0085)
008B	3E	WAI	Store Index register $X_H \rightarrow M_{8F}, X_L \rightarrow M_{A0}$
—	—	—	Stop

result of the compare instruction and the debouncing routine would then be initiated, followed by the decoding scheme. (Reference Examples 8-2 and 8-3.)

The same general procedure can be used to interface up to 16 switches in a column using two of the 6801's 8-bit ports. In the single-chip mode, you could interface 16 switches to ports 3 and 4. Since the i/o data registers for these two ports are located in consecutive memory locations (0006:0007), you can use the 16-bit accumulator D to read and rotate the switch data.

Keyboards

Keyboards are simply a collection of switches and, therefore, they can be connected to the 6801 in a switch column as just discussed. In this case, the keyboard becomes a set of switches in which each key is connected to a separate input port line. When interfacing a keyboard in this manner, the procedures for detecting key closure, debouncing, and decoding are the same as discussed earlier. This is fine for a small number of keys, but when more than eight keys are involved, you must use multibyte operations since more than one input port will be required. The number of port lines required may be reduced by connecting the keys in a switch *matrix*. The matrix will be an n-by-m, or $n \times m$, matrix where n is the number of rows and m is the number of columns in the matrix. Each key will represent the intersection of a row and column. A typical keyboard matrix is shown in Fig. 8-4. This is a 4×4 matrix which only requires eight port lines. In this scheme, an $m \times n$ key-

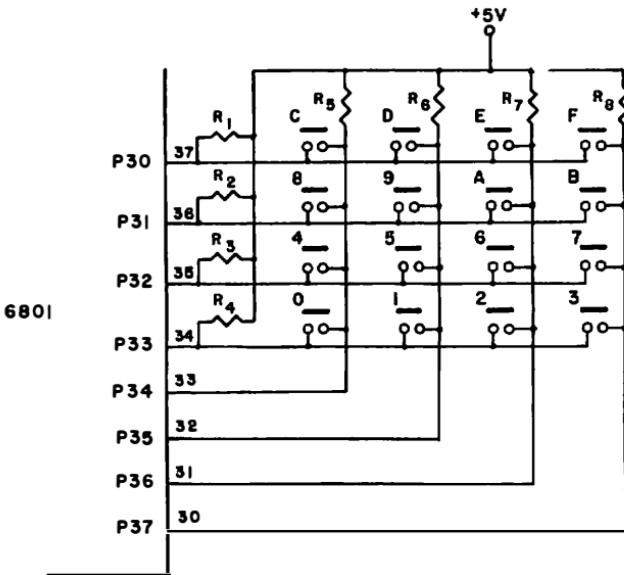


Fig. 8-4. Interfacing to a switch matrix.

board matrix will always require $m + n$ port lines while if you were to connect the keys in a switch column configuration you would need $m \times n$ port lines. Here, we are connecting the matrix rows to P30 through P33 and the columns to P34 through P37. We have chosen to interface the switch matrix to port 3. Therefore, the 6801 must be operating in the single-chip mode.

However, if operating in one of the expanded modes, port 3 cannot be used, since in these modes it is used for data and/or external addressing. The keys are labeled to correspond to a hexadecimal keyboard. Each matrix row contains four keys as does each column. You will configure P30-P33 as input lines and P34-P37 as output lines. The procedure for detecting key closure will be to *scan* each column by successively applying a logic 0 to each output line, P34-P37. For example, suppose a logic 0 is applied to column one, P34. (This can be done by storing EF in port 3's data i/o register at address 0006.) If none of the switches along the P34 line (0, 4, 8, C) are closed, the input lines (P30-P33) will all be held at a logic 1 by pull-up resistors R1 through R4. However, if one of the keys is depressed, its corresponding input line will go low since the low state of P34 will be applied directly to the input line through the switch. If no closure is detected in the first column, the logic 0 will be removed from P34 and applied to P35 by storing DF in the port 3 data i/o register. The program can then check to see if any of the

keys connected to the P35 line (1, 5, 9 or D) are closed. You would continue to scan the keyboard matrix in this manner, one line at a time, until a closure is detected. Once detected, you must provide debouncing and decoding program steps as previously discussed. A rotate-right procedure will be used to decode key closure within a column.

This procedure could be used to interface a keyboard containing as many as 64 keys using only two 8-bit ports. The keys would have to be configured in an 8×8 matrix. The technique just described was for an *unencoded keyboard*. Another type of keyboard which makes the job of interfacing much easier is the *encoded keyboard*. This keyboard contains the key switches along with internal logic that will perform all the scanning and decoding which *you* had to provide for the unencoded keyboard. The encoded keyboard will provide you with a unique code for each key. This code is usually an ASCII (American Standard Code for Information Interchange) value, but can be any code that would identify each key uniquely or separately. Most encoded keyboards will also provide you with circuitry for switch debouncing, thus further simplifying the interfacing task. The tradeoff between the two types is the simpler software required by the encoded keyboard versus the lower cost of the unencoded keyboard.

Fig. 8-5 shows how you would interface an encoded keyboard to your 6801 system. In this figure, the keyboard data is supplied to port 3. Port 3 is used since most encoded keyboards will supply an input strobe pulse. The strobe will indicate a new key closure

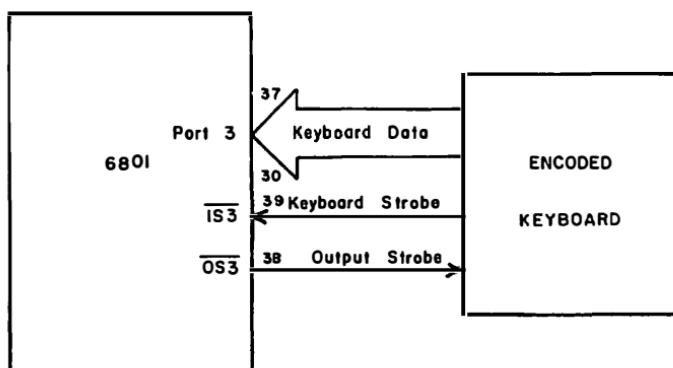


Fig. 8-5. Interfacing to an encoded keyboard.

and will be connected to the input strobe line (IS3) of port 3. Recall that the input strobe line (IS3) is an input-only line used to set the input strobe flag (IS3 flag, bit 7) of the i/o port 3 control/status register. If the IS3 enable bit (bit 6) is set, setting the strobe flag will cause an interrupt request to be generated to the 6801 CPU. The IS3 flag will be cleared by a read of the control/status register followed by a read of port 3. After the flag has been cleared, the 6801 will be ready to accept another key closure through port 3 by another strobe pulse on the IS3 line. Recall that the latch enable bit (bit 3) of the control/status register must also be set such that the input data is latched with the falling edge of the input strobe (IS3). After reading the port 3 data, you must again set this bit since it is cleared by the read operation. If you desire to provide complete handshaking with the keyboard, you can use the output strobe function of port 3. Recall that the port 3 output strobe (OS3) is provided via pin 38. To provide complete handshaking, the output strobe would be generated to the keyboard each time the keyboard data is read from port 3. The strobe tells the keyboard that the 6801 has accepted the data and is ready to accept more. To provide the output strobe with a read port 3 operation, the output strobe select bit (OSS-bit 4) of the control/register must be cleared. You can make use of the interrupt capabilities of the 6801 via port 3 as just discussed or you might want to use a polling routine to enter keyboard data. The program in Example 8-4 uses a polling routine.

Example 8-4: Polling Routine for Encoded Keyboards

<i>Hex Address</i>	<i>Hex Contents</i>	<i>Mnemonic/Contents</i>	<i>Operation</i>
0080	7F	CLR \$\$	
0081	00	00	
0082	04	04	
			Clear port 3 DDR (port 3 = Input)
0083	86	LDAA #	
0084	08	08	
0085	97	STAA \$	
0086	0F	0F	
			Clear IS3 enable and output select strobe Set latch enable
0087	96	LDAA \$	
0088	0F	0F	
			Read control/status register
0089	2A	BPL	
008A	FC	FC	
			Branch If plus (to address 0087)
008B	96	LDAA \$	
008C	06	06	
			Read port 3
008D	3E	WAI	Stop

Note how the port 3 control/status register is configured. The input strobe enable bit (IS3-bit 6) of the control/status register is cleared, thereby preventing the interrupt flag from generating an interrupt request to the 6801. Instead, you will *poll* the contents of the control/status register. The output strobe select (OSS-bit 4) is also cleared such that an output strobe will be generated when a read operation is performed on port 3. Finally, the latch enable bit (bit 3) is set to allow incoming data to be latched into the port 3 data i/o register.

If no strobe has occurred, bit 7 of the control/status register will remain cleared. The 6801 will therefore recognize the control/status register data as "positive" and branch until a "negative" (bit 7=0) condition is recognized. When a keyboard strobe has caused bit 7 of the control/status register to set, the 6801 will recognize the control/status register data as "negative" and will read the keyboard data from the port 3 data i/o register. When the read operation occurs, bit 7 will be cleared to allow for another keyboard strobe. The data values are not "signed," but we have used the sign flag for the keyboard detecting function.

INTERFACING WITH DISPLAYS

To provide a simple light pattern output from the 6801, you can interface single LED units to any of the 29 6801 i/o port lines. All of these lines have TTL drive capability. You must configure the port lines as output by writing ones into the respective bit locations of the port data direction registers. Then, to cause a particular light pattern to be displayed, you would store the proper bit pattern in the respective port data i/o register. This type of display is fine for indicating status, conditions, and codes; however, it is not very meaningful unless specifically defined for the user. A more meaningful display would be one that could give you an alphanumeric display such that decimal and hexadecimal numbers could be represented as well as a limited alphabet so that messages could be displayed using groups of displays. Such a display is the 7-segment LED display. A typical 7-segment display is shown in Fig. 8-6. The display consists of seven separate LED "bar" displays labeled "a" through "g" and a decimal point display labeled DP.

There are two general categories of 7-segment LED displays. They are *common cathode* and *common anode*. The common cathode has the LED cathodes tied together and connected to ground. Therefore, a logic 1 (with sufficient current driving capability) must be applied to supply the current to illuminate a particular LED segment. This is referred to as *positive logic*. The common anode variety has the LED anodes tied together and connected to

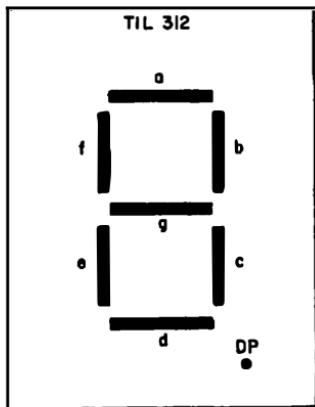


Fig. 8-6. Typical 7-segment LED display.

the +5-V dc supply. This type uses *negative logic*, meaning that a logic 0 must be applied to sink the current necessary to illuminate a particular LED segment. Regardless of whether the display is common anode or common cathode, you will assign each LED segment to a specific data i/o line of one of the 8-bit ports as shown in Fig. 8-7. We have chosen to use port 1. However, ports 3 and 4 may also be used. In this manner, you can control the LED illumination by the data that is written to the respective port data i/o register. Since there are eight total LED segments including the decimal point, there are 256 possible unique displays available. However, many of them are meaningless and you will actually use less than 50 different combinations to form the numbers, letters, and characters shown in Table 8-1.

You can connect the TIL-312 common anode 7-segment display to port 1 of the 6801 as shown in Fig. 8-7. Port 1 will provide you the necessary drive to properly illuminate the LED segments. You will configure port 1 as an output port, then to display a particular character you will store the character representation in the port 1 data i/o register. The program in Example 8-5 will accomplish this task. The character designation will be inserted at address 0085. If you desire to display a hexadecimal number directly from its binary equivalent, you must provide a conversion since the character designations are different from the actual hex numbers. Such a conversion can be provided by means of a *look-up table*. A look-up table is simply a set of consecutive memory locations that contain the proper character designations. You can access the table and provide the proper conversion with the use of indexed addressing as shown in Example 8-6.

Example 8-5: The Output of 7-Segment LED Character Designations via Port 1

Hex Address	Hex Contents	Mnemonic/Contents	Operation
0080	86	LDAA #	
0081	FF	FF	
0082	97	STAA \$	
0083	00	00	Set all port 1 DDR bits (port 1 = output)
0084	86	LDAA #	
0085	--	--	Load accumulator A with character designation
0086	97	STAA \$	
0087	02	02	Output character designation to port 1
0088	3E	WAI	Stop

The program in Example 8-6 assumes that the 7-segment *common anode* LED such as the TIL-312 is connected to port 1 and this port has been configured as an output port. Once the binary number is in accumulator A, the program will convert it to its proper character designation, then output that designation to port 1 such that the proper display is achieved. Note that the key to the whole procedure is in using the accumulator contents as the indexed offset to access the look-up table. The first instruction stores the accumulator contents at the offset address.

To display intelligible words and messages, several 7-segment displays must be *multiplexed* together from one common display bus. Fig. 8-8 shows how the 6801 can be utilized to multiplex eight separate 7-segment LED displays. Note that port 3 of the 6801 is supplying the character representation code to a common display bus. Therefore, port 3 must be configured as an output port. The displays are of the common cathode variety requiring positive logic and are sequentially enabled by port 4. Therefore, port 4 provides for the display multiplexing and will also be configured as an output port. A logic 1 state on a particular port 4 output line will forward bias the corresponding transistor base-emitter junction causing the LED display that is connected to it to be enabled. Since the port 3 and port 4 data i/o registers are located at consecutive memory locations, you can use accumulator D to provide the character designations to port 3 and multiplexing output to port 4 simultaneously. To display a message, you will simply enable each display sequentially by rotating a logic 1 through port 4. As you do this, the proper character representation codes will be simultaneously stored to port 3. Usually, you will want to display a particular message as the result of some condition. The program in Example 8-7 is a subroutine that could be used to display a given message.

Example 8-6: 7-Segment LED Character Designation Look-Up Table Routine

Hex Address	Hex Contents	Mnemonic/Contents	Operation
0080	97	STAA \$	
0081	86	86	Store accumulator A at address 0086
0082	CE	LDX #	
0083	00	00	
0084	B0	B0	Load the Index register with the first address of the table
0085	A6	LDAA X	
0086	--	--	Load accumulator A with the proper character designation
0087	97	STAA \$	
0088	02	02	ACCA → Port 1 (Output character designation to port 1)
:	:	:	:

Look-Up Table

Hex Address	Hex Contents	7-Segment Representation
00B0	C0	hex designation for 0
00B1	F9	hex designation for 1
00B2	A4	hex designation for 2
00B3	B0	hex designation for 3
00B4	99	hex designation for 4
00B5	92	hex designation for 5
00B6	82	hex designation for 6
00B7	F8	hex designation for 7
00B8	80	hex designation for 8
00B9	98	hex designation for 9
00BA	88	hex designation for A
00BB	83	hex designation for B
00BC	C6	hex designation for C
00BD	A1	hex designation for D
00BE	86	hex designation for E
00BF	8E	hex designation for F

Your main program would call this subroutine using a jump-to-subroutine (JSR) instruction. The first part of the subroutine configures ports 3 and 4 as output ports. You will then load the index register with the address immediately before the beginning address of your eight consecutive memory locations that contain the character codes. In this example, addresses 00B0 through 00B7 contain the eight character codes for the message; therefore, you have loaded the index register with 00AF. You will then clear accumulator B and set the carry flag. The carry flag is then rotated into

Table 8-1. Hexadecimal Display Representations

Display	Common Cathode	Common Anode
0	3F	C0
1	06	F9
2	5B	A4
3	4F	B0
4	66	99
5	6D	92
6	7D	82
7	07	F8
8	7F	80
9	67	98
A	77	88
b	7C	83
C	39	C6
d	5E	A1
E	79	86
F	71	8E
G	BD	42
H	76	89
I	19	E6
J	1E	E1
K	70	8F
L	38	C7
M	37	C8
n	54	AB
o	63	9A
P	73	8C
Q	6B	94
r	50	AF
S	2D	D2
t	78	87
u	1C	E3
v	72	8D
W	3E	C1
x	64	9B
y	6E	91
z	1B	E4

(Courtesy, Mr. Alex Funk, 110 E. Lynch St., Durham, NC 27701)

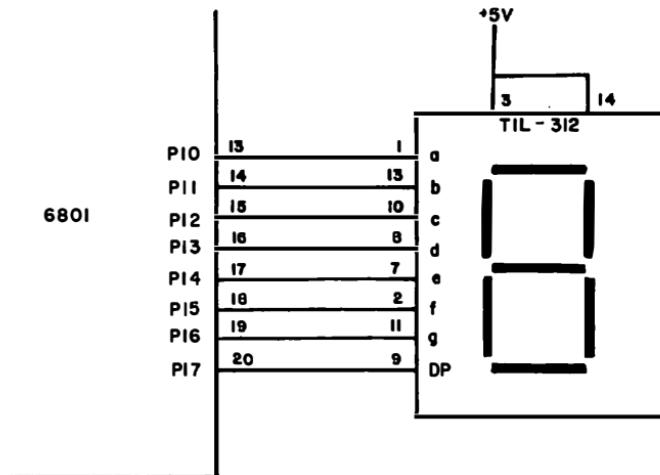


Fig. 8-7. Interfacing to a single 7-segment display.

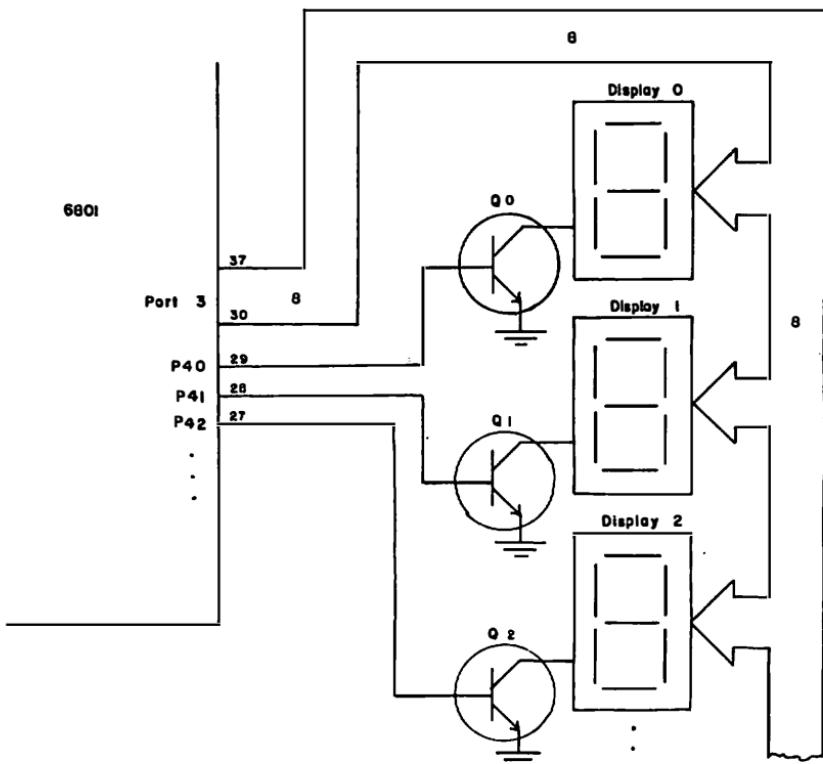


Fig. 8-8. Using the 8801 to multiplex displays.

Example 8-7: 7-Segment LED Multiplexing Subroutine

Hex Address	Hex Contents	Mnemonic/Contents	Operation
0080	4F	CLRA	
0081	43	COMA	
0082	5F	CLRB	
0083	53	COMB	
0084	DD	STD \$	Set all port 3 and port 4 DDR bits (port 3 = Output)
0085	04	04	(port 4 = Output)
0088	FE	LDX \$\$	
0087	00	00	
0088	AF	AF	Load Index register with beginning address of code table
0089	5F	CLRB	
008A	OD	SEC	
008B	59	ROLB	Clear ACCB. Set C-flag Point to next display
008C	24	BCC	
008D	01	01	Branch If carry clear (to address 008F)
008E	39	RTS	
008F	08	INX	Return If message complete Point to next character code
0090	A6	LDAA X	
0091	00	00	Load next character code in ACCA
0092	DD	STD \$	
0093	06	08	Display next character code
0094	4F	CLRA	
0095	4C	INCA	
0098	26	BNE	
0097	FD	FD	Delay approximately 15 ms
0098	20	BRA	
0099	F1	F1	Branch always (to address 008B)

bit 0 of accumulator B. This will enable display #0 when accumulator B is stored to port 4. A branch if carry clear (BCC) instruction is inserted such that the program will return to the main program (RTS) if the carry is set, meaning the C-flag has been completely rotated through accumulator B and the entire message has been displayed. However, at this point, the C-flag is cleared as a result of the rotate instruction. Therefore, the index register is incremented and accumulator A is loaded with the first character code. The character code and display enable bit are then stored to ports 3 and 4 simultaneously using a store accumulator D (STD) instruction. This will cause display #0 to display the proper character. You will then clear and increment accumulator A, through FF to 00 to

provide a delay which will allow the display to illuminate for approximately 15 milliseconds before going to the next display. This delay loop assumes a 1-MHz clock frequency.

The procedure is then repeated by rotating accumulator B to enable display #1. The character code and enabling bit for display #1 are then stored to ports 3 and 4 to provide its proper display. The cycle is repeated until all eight displays have been illuminated and the entire message has been displayed. In practice, you would want to call this subroutine many times a second to give the impression of a constant display. Since these types of displays require constant refreshing, an interrupt is sometimes used to interrupt the main program to refresh the display. Refresh rates of 50-60 Hz are quite common, with lower rates resulting in display flicker. The 6801 timer section could be utilized to generate the refresh interrupt.

INTERFACING WITH DIGITAL-TO-ANALOG CONVERTERS (DACs)—SOME REAL PRODUCTS

Digital-to-analog converters are needed in many practical systems to translate the system digital code to a continuous voltage level (analog signal) required by motors, ovens, relays, and other electromechanical devices. These converters can be made from standard resistor and op-amp circuitry, or can be purchased as a single-chip device. Since the cost of these single-chip devices is very reasonable (and decreasing), you will find it more economical and less time consuming to use a manufactured d/a chip rather than designing your own circuit. In this section we will discuss two d/a converter chips, the Signetics NE5018 and Motorola MC-1408-8/MC1508-8.

The Signetics NE5018 will convert an 8-bit digital signal to a continuous output voltage level. Interfacing this chip to your system is relatively simple. You will configure port 3 of the 6801 as an output port and use $\overline{OS3}$ as an output strobe to enable the d/a converter. The interfacing scheme is shown in Fig. 8-9. The NE5018 contains an input latch that will store the digital input data when a high-to-low transition is seen on \overline{LE} . This transition is supplied by $\overline{OS3}$ of the 6801. Recall from Chapter 4 that $\overline{OS3}$ (pin 38) can be configured as an output strobe in the single-chip mode. $\overline{OS3}$ will normally be high then go low after a *write* port B operation is executed if the OSS bit of the port 3 control/status register is set. It will remain low for one enable signal (ϕ_2) cycle. This is long enough to allow the NE5018 converter to latch the digital input data. The program in Example 8-8 could be used to configure port 3 and supply the digital information to the NE5018.

Example 8-8: NE5018 Data Transfer Program

Hex Address	Hex Contents	Mnemonic/Contents	Operation
0080	86	LDAA #	
0081	FF	FF	
0082	97	STAA \$	
0083	04	04	Set all port 3 DDR bits (port 3 = output)
0084	86	LDAA #	
0085	10	10	
0086	97	STAA \$	
0087	0F	OF	Set OSS bit of port 3 control/status register
0088	96	LDAA \$	
0089	A0	A0	Get digital data at memory location 00A0
008A	97	STAA \$	
008B	DE	DE	
008C	3E	WAI	Store digital data to port 3

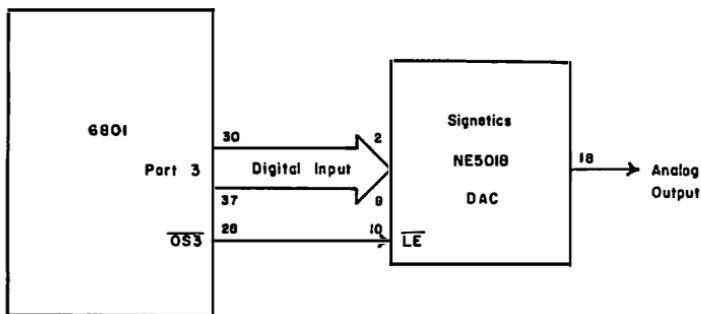


Fig. 8-9. Interfacing to the Signetics NE5018 d/a converter.

Once port 3 is configured, the program assumes the digital data are located at memory location 00A0. The data are obtained from that location and then written to port 3. Immediately after the write operation occurs, $\overline{OSS3}$ will automatically go low for one clock pulse, allowing the data to be transferred to the NE5018. The converter will then produce a corresponding analog output within a few microseconds. You could ground \overline{LE} pin if latching is not required.

The Motorola MC1408-8/MC1508-8 interfacing is even more straightforward. This d/a chip does not contain a latch and therefore does not require a strobe or enabling pulse as did the NE5018. Therefore, port 1 or 4 may be used. The latching capabilities of the 6801 ports provide data latching to the MC1408-8/MC1508-8. The interfacing scheme is shown in Fig. 8-10. We have chosen to

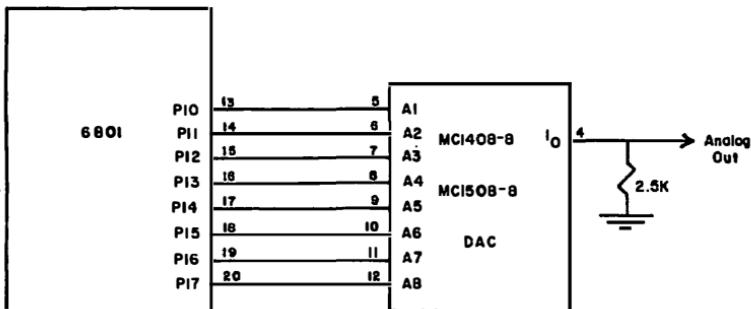


Fig. 8-10. Interfacing the Motorola MC1408-8/MC1508-8 d/a converter.

use port 1. You will configure port 1 for output. Then, to provide a conversion, you will simply store an 8-bit data word to port 1. The MC1408/MC1508 will then provide a continuous analog output voltage level which corresponds to the digital input. This chip will yield 256 different output voltage levels from 0 to -4.980 volts corresponding to the digital input data. Consult Appendix B for more information on both the Signetics NE5018 and Motorola MC-1408/MC1508 d/a converters.

INTERFACING WITH ANALOG-TO-DIGITAL CONVERTERS—SOME REAL PRODUCTS

Many input devices such as various types of sensors and transducers generate analog signals. Before the 6801 can process the signal, it must be converted to a digital data word. This conversion process is not a simple task if *you* must provide the external analog and digital circuitry as well as the software required for an accurate conversion. Fortunately, complete a/d converter chips are becoming increasingly available at low cost. Such a chip is the Intersil 7109. This device is a complete a/d converter that will convert an analog signal to a 12-bit binary word, plus a 2-bit indication for signal polarity and overrange. It will provide up to 30 conversions per second, and can be operated in a partial or complete hand-shaking mode. Fig. 8-11 shows one way in which the 7109 can be interfaced to your 6801 system. Given an analog input signal, the 7109 will provide 14 bits of output data to ports 3 and 4 of the 6801. Twelve data bits (B1-B12) plus a polarity (POL) bit and overrange (OR) bit. The 12 data bits are divided into a low-order data byte (B1-B8) and high-order data byte (B9-B12). The low-order byte is connected to port lines P30-P37 and the high-order byte connected to P40-P43. Therefore, both ports must be configured as input ports. The control register for port 3 can be set up such that

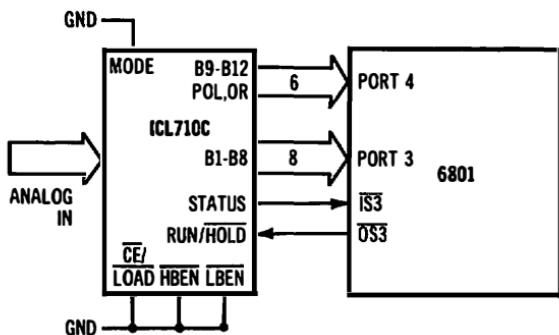


Fig. 8-11. Interfacing to the Intersil ICL7109 a/d converter.

$\overline{IS3}$ will be activated and generate an interrupt request on a high-to-low transition of the 7109 STATUS pin. With RUN/HOLD held high, the 7109 will be allowed to RUN and provide continuous conversions by interrupting the 6801 via the $\overline{IS3}$ line when each conversion becomes available. Complete handshaking can be provided by connecting the RUN/HOLD line to OS3 of the 6801 as shown in Fig. 8-11. The 7109 also contains an output latch such that the data will not be lost before the 6801 acknowledges the interrupt. Consult the 7109 specification in Appendix B for other methods of interfacing and more chip detail.

Another common a/d converter is the Teledyne 8703. This is an 8-bit converter which also has an output latch. The interfacing scheme is essentially the same as for the 7109 previously discussed. Fig. 8-12 shows how the 8703 might be connected to your system. The eight digital data bits (B0-B7) are connected to port 3 lines P30-P37. Output strobe ($\overline{OS3}$) is used to initiate a conversion. This provides the initiate conversion pulse needed to start the conversion

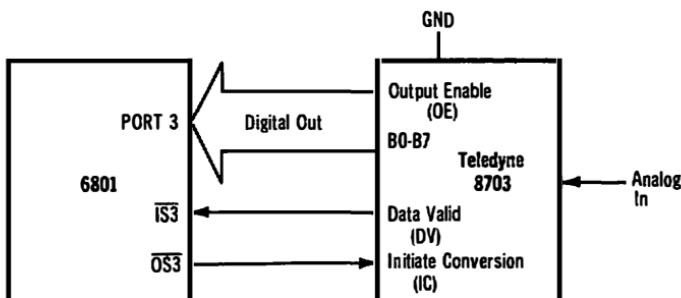


Fig. 8-12. Interfacing to the Teledyne 8703 a/d converter.

process. Once the conversion is complete, the 8703 data valid line will go from low to high indicating data is available. This line should be inverted to provide the proper $\overline{IS3}$ signal to the 6801. This will cause an interrupt to be generated via $IS3$. A read operation can then be performed on port 3 to enter the converted data into the system. After the read operation, an output strobe will be generated which will initiate another conversion and the cycle is repeated. This a/d converter is also available in a 10-bit (8704) and 12-bit (8705) version. Naturally, with more than eight bits, multi-byte data operations will be involved; however, a better digital resolution of the analog signal can be achieved with more data bits.

You have now had a brief introduction to show you how some d/a and a/d devices can be connected to the 6801. For more information on these devices and interfacing techniques, consult the following:

1. *Microcomputer-Analog Converter Software and Hardware Interfacing*, Howard W. Sams & Co., Inc., 1978.
2. *IC Converter Cookbook*, Howard W. Sams & Co., Inc., 1978.
3. *Analog-Digital Conversion Notes*, Analog Devices, Inc., Norwood, MA. 02062, 1977.

REVIEW QUESTIONS

1. What are the four basic interfacing requirements when interfacing to switches? _____,
2. What mode must the 6801 be in to use port 3 for switch input?
3. How is switch closure detected when interfacing to a single switch?
4. A typical software delay that would provide switch debouncing is _____ seconds.
5. The debouncing delay is usually provided by decrementing the _____.
6. How is switch closure detected in a switch column?
7. How is a switch column decoded?

8. A 2×2 switch matrix would require that _____ 6801 port lines be used.
9. How is a switch closure detected and decoded in a keyboard matrix?
10. Two types of keyboards available are the _____ and _____.
11. A common anode LED display would use _____ logic.
12. The LED character designations are generated from a _____ located in memory.
13. To display words and messages, several displays must be _____ together.
14. A message should be refreshed at a _____ rate to prevent flickering.
15. What is the difference between the Signetics NE5018 and Motorola MC-1408-8?
16. The Intersil 7109 is a _____ bit a/d converter.
17. The Teledyne 8703 is a _____ bit a/d converter.
18. What is the advantage of more data bits in an a/d converter?
19. An analog signal can be defined as _____.
20. The Intersil 7109 a/d converter provides 12 data bits plus a 2-bit indication of _____ and _____.

ANSWERS

1. Switch addressing, detecting switch closure, switch debouncing, switch decoding.
2. Single-chip.
3. By ANDing a mask byte to the switch status byte, then checking the Z-flag status with a branch (BNE) instruction.
4. 10 milliseconds
5. index register
6. By comparing the switch status byte to FF.

7. By rotating the switch status byte through the C-flag, checking the C-flag status after each rotation, and counting the number of rotations required to clear the flag.
8. four
9. With the matrix connected as in Fig. 8-4, switch closure is detected by scanning each column with P34-P37 and simultaneously reading P30-P33 of port 3. A compare operation is then used to detect the closure and a rotate right operation used to decode the closure within a particular column.
10. unencoded and fully encoded
11. negative
12. look-up table
13. multiplexed
14. 50-60 Hz
15. The NE5018 has an internal 8-bit latch, requiring an output strobe or enabling pulse from the 6801.
16. 12
17. 8
18. Greater precision.
19. continuous voltage or current values
20. signal polarity and overrange

CHAPTER 9

6801 Applications

INTRODUCTION

You should now have a complete understanding of the 6801 microcomputer. Given an application problem, and knowing the capabilities of the 6801, you can now make a decision on the proper 6801 system configuration. Once you decide on the configuration and operating mode, you now have the basic skills needed to configure, program, and interface the 6801 to meet the application requirements. In an effort to tie everything together, we will present some "real-world" applications of the 6801 in this chapter. The first discussion will show you how to connect the 6801 serial i/o port to an RS-232C interface. Since many common i/o devices use this serial data transmission standard, the material presented here should enhance your basic understanding of 6801 interfacing.

We will then zero-in on some very specific applications. The first will show you how the 6801 can be used to measure temperature using an MC1555 (555) timer circuit and the internal timing function of the 6801. Many dedicated applications require temperature to be monitored for control purposes and the simple technique used here should prove valuable whenever a temperature monitoring problem is encountered. Next, you will be given a complete discussion on how to use the 6801 to measure automobile engine rpm, dwell angle, and timing angle. The automobile is predicted to be the single largest application for the microcomputer. Microcomputers will become standard equipment on most automobiles through the 1980's to perform such tasks as exhaust emission control, fuel efficiency, and various engine monitoring functions. Because of its extreme flexibility and i/o capability, the 6801 is ideally suited for

automobile applications. At this printing, Motorola is currently working on a special version of the 6801 for General Motors and Ford Motor Company to be used in their future automobile production. The ideas presented here should give you a basic understanding of the problems encountered when interfacing a microcomputer to an automobile engine and how to solve those problems. You should also gain insight on how to use the 6801 to perform other automobile interfacing tasks. Finally, you will see how the 6801 can be dedicated to function as a remote data acquisition controller.

OBJECTIVES

At the end of this chapter you will be able to do the following:

- Understand the requirements of the EIA RS-232C Standard.
- Explain how to interface the 6801 to an RS-232C terminal connection.
- Describe how to use the MC1555 (555) timer and 6801 to monitor temperature in °C or °F.
- Understand the signal conditioning requirements for an automobile engine spark plug, points, and 0° top dead center (TDC) signals.
- Explain how the above signals can be multiplexed into the 6801 and used to calculate engine rpm, dwell angle, and timing angle.
- Describe the overall 6801 process flow required to monitor and display engine rpm, dwell angle and timing angle.
- Explain how the 6801 can be used as a remote data acquisition controller.

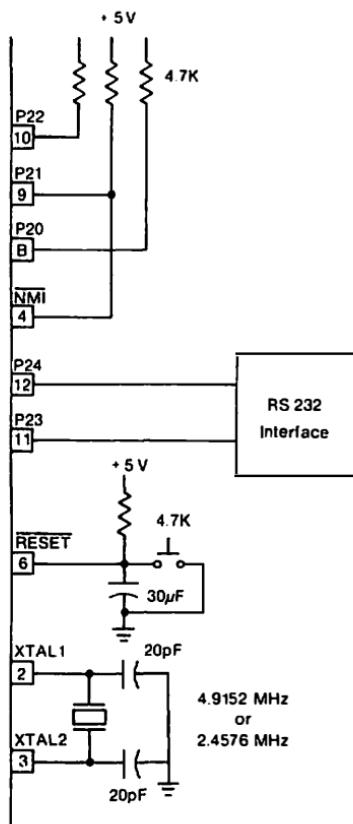
SERIAL COMMUNICATION VIA RS-232C

As you saw in Chapter 7, the 6801 can send and receive data in serial format, the advantage being that only one or two wires are needed to carry all the necessary signals. Peripheral devices which commonly use serial data communication are teletypewriters (tty's), modems, printers, crt's, cassette tapes and floppy disks. In an effort to standardize the serial transmission format and electrical signal connections, the Electronics Industry Association (EIA) developed a standard called RS-232C which covers the electrical specifications for serial transmission as well as the physical connection specifications. The standard uses plus and minus 12-volt pulses to represent data and control signals. The RS-232C standard also specifies a 25-pin connector with signals that are shown in Table 9-1. Many pe-

Table 9-1. RS-232C Signal Definitions

Pin No.	Signal Name
1	Frame Ground
2	Receive Data
3	Transmit Data
4	Request To Send
5	Clear To Send
6	Data Set Ready
7	Signal Ground
8	Carrier Detect
9	Current Loop Input +
10	Current Loop-Rx
11	Secondary Request To Send
13	Current Loop Supply +
14	Current Loop Supply -
18	Current Loop Output +
20	Data Terminal Ready
22	Ring Indicator
25	Current Loop Output-Tx

Fig. 9-1. Serial communication with RS-232C.



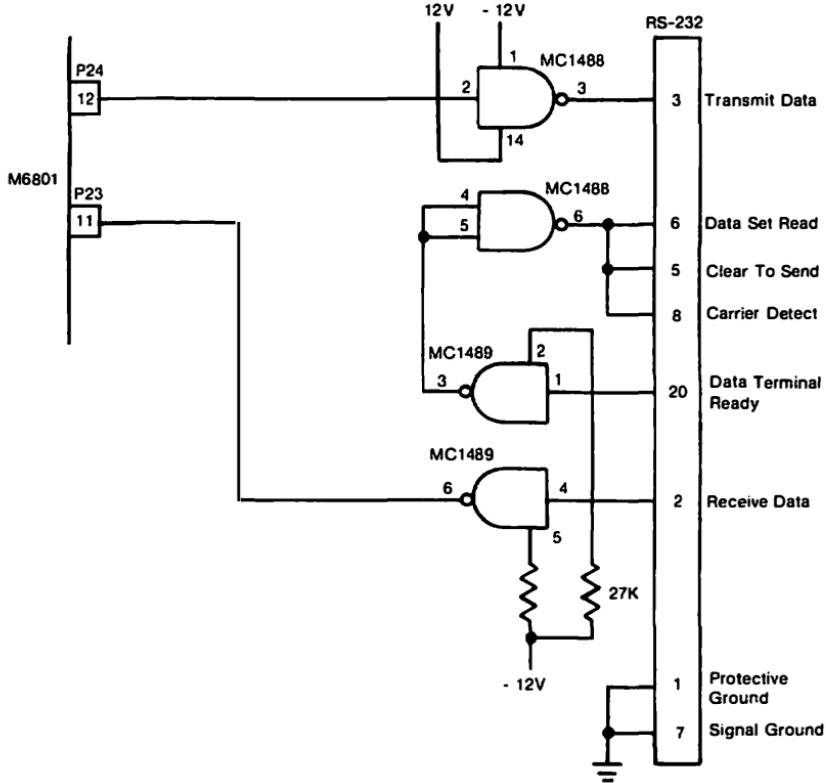


Fig. 9-2. RS-232C Interface circuit.

ipheral devices are “wired” for an RS-232C interface; therefore, a 6801 application might require that you provide your system with RS-232C capability as shown in Fig. 9-1. The 6801 shown here is wired for the single-chip mode, with the RS-232C interface circuit connected to serial data receive and transmit lines P23 and P24 (pins 11 and 12). The required RS-232C interface circuitry is shown in Fig. 9-2. This circuit will interface the serial input/output of the 6801 directly to an RS-232C terminal *without* any additional hardware. The circuit uses the MC1488 and MC1489A devices to provide the interface capability. Note that plus and minus 12-volt supplies are required to develop the signal levels required by the standard. Also, only eight of the 25 RS-232C pin connections are needed.

A standard similar to RS-232C is 20-mA current loop. This standard is normally used for mechanical type teletypewriters and is also available on many crt's and other serial i/o devices. The 20-mA current loop standard uses 20-mA current pulses to represent data and control signals, and it is advantageous where long transmission

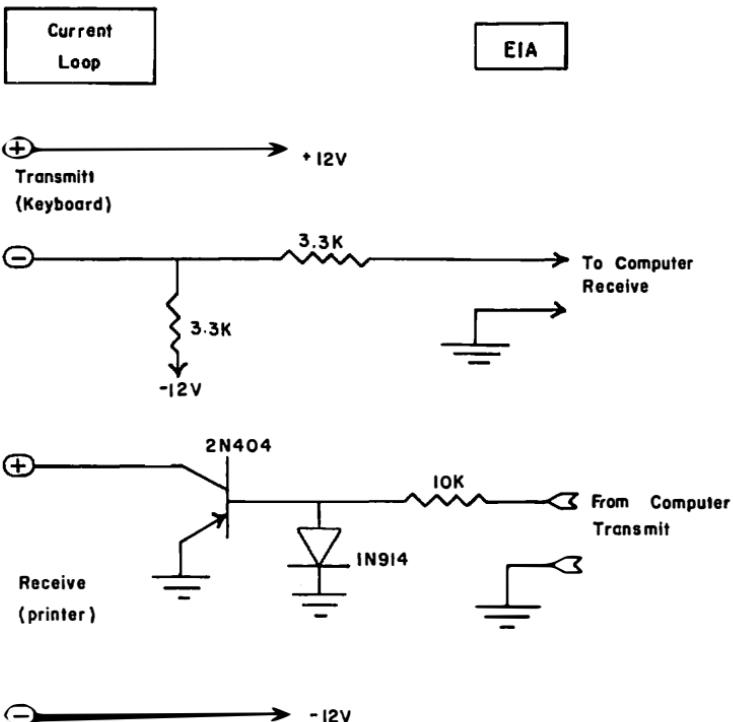


Fig. 9-3. Loop-to-EIA converter circuit.

distances (up to one mile) are required. To interface this standard to your 6801 system, you can convert the 20-mA loop devices to the RS-232C standard with a *loop-to-EIA* converter and use the RS-232C interface circuit previously discussed. A *loop-to-EIA* converter circuit is shown in Fig. 9-3.

TEMPERATURE MONITORING DEVICE

In this section, you will see how the 6801 timer can be used to monitor temperature, using an MC1555 (555) timer to generate a pulse train that has a frequency that is proportional to temperature. The required circuit is shown in Fig. 9-4. Note that the MC1555 is being used to generate a train of pulses for the 6801. The MC1555 will provide a pulse width that is linearly related to the temperature sensed by the thermistor on its input. The pulse train will be fed into the 6801 at pin 8, P20. The 6801 timer will be used in the pulse measurement mode to obtain the temperature reading.

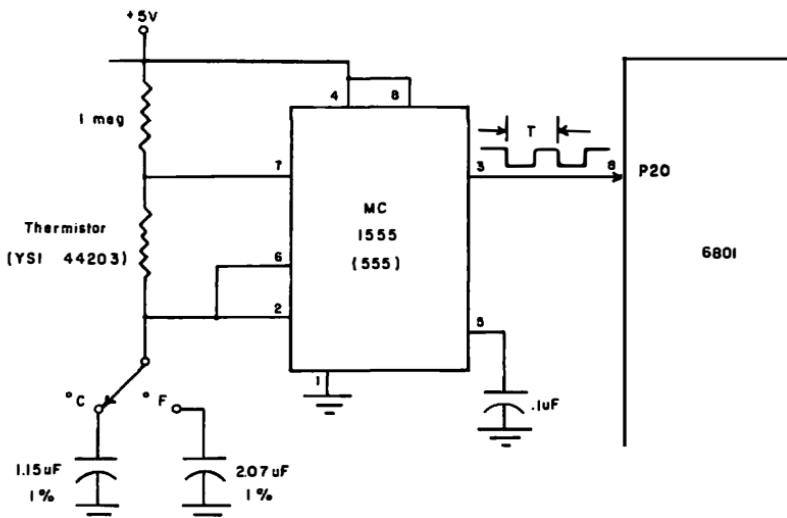


Fig. 9-4. Temperature-monitoring circuit.

The MC1555 (555) timer is connected in the astable mode. Therefore, it will retrigger itself and cause the capacitor voltage to oscillate between $1/3 V_{cc}$ and $2/3 V_{cc}$. When the MC1555 output is high, the capacitor ($1.15 \mu F$ or $2.07 \mu F$) begins charging through the 1-megohm resistor and thermistor series combination. When the voltage across the capacitor reaches $2/3 V_{cc}$, the MC1555 output will go low. The capacitor now discharges through the thermistor until its voltage drops to $1/3 V_{cc}$. When this value is reached, the MC1555 output will go back to its original state (high). The cycle repeats itself to generate a rectangular output waveform. By varying the ratio of the two input resistors, the output duty cycle (T) can be varied. The 1-megohm input resistor is sufficiently large to limit the discharge current at pin 7 to be within the maximum rating of the internal MC1555 circuitry. A Yellow Springs^o 44203 thermistor is used as the other input resistance. It has a resistance of 12,175 ohms at $0^{\circ}C$ and decreases linearly at a rate of 127.096 ohms per degree. The change in output pulse's logic 0 period (T) will vary directly with the thermistor resistance value. Two separate charging capacitors are used to provide temperature readings in $^{\circ}C$ or $^{\circ}F$. Note that the ratio of the capacitor values is 5/9 (.5555).

As stated earlier, the 6801 timer will be used in the pulse measurement mode for this application. You must measure the pulse

^o Yellow Springs Instrument Co., Yellow Springs, Ohio 45387.

train period (T), as shown in Fig. 9-4. Therefore, each time a high-to-low (or low-to-high) transition is detected at P20, you will want the existing internal counter contents to be transferred to the timer input capture register (see Chapter 6). When this happens, the input capture flag will be set and will generate an interrupt to the 6801 CPU. The interrupt service routine will then read the existing count and use the difference between this and the previous count to calculate the temperature. The program in Example 6-3 may be used to accomplish this task. The *maximum* measureable time duration (T) will be 65.536 milliseconds using a 1-MHz clock. Ten microseconds is the *minimum* recommended time duration (T). The program could be modified to measure only the pulse's logic 0 or logic 1 period. To do this, you would clear the IEDG bit of the TCRS such that the 6801 would first recognize a high-to-low (falling pulse edge) transition at P20, then set the IEDG bit such that the 6801 would recognize a low-to-high (rising pulse edge) transition at P20. This would extend your temperature measurement range. In either of the above cases, the system must be calibrated to determine the correct temperature-vs-count relationship.

AUTOMOBILE APPLICATIONS

You will now see how the power of the 6801 can be harnessed to monitor three automobile engine functions: revolutions per minute (rpm), point dwell, and timing angle. The variable engine signals required to monitor these three functions are point closure, plug No. 1 firing, and 0° top dead center (TDC). You will need to monitor point closure duration and point closure period to calculate engine rpm and point dwell. However, to measure timing angle, you must monitor the point signal along with determining when plug No. 1 fires, and when piston No. 1 is at the top of its stroke (0° TDC). If plug No. 1 fires before piston No. 1 reaches 0° TDC, the "spark" or timing angle is said to be advanced (+); however, if plug No. 1 fires after piston No. 1 reaches 0° TDC, the "spark" or timing angle is said to be retarded (-). Constant values needed to make the required calculations are the number of engine cylinders and cycles. Once these values have been obtained, the 6801 can determine engine rpm, point dwell, and timing as follows.

RPM =

$$\frac{30 \times (\text{Number of engine cycles})}{(\text{Number of engine cylinders}) \times (\text{point closure duration})}$$

Dwell Angle ($^\circ$) =

$$\frac{360 \times (\text{point closure duration})}{(\text{Number of engine cylinders}) \times (\text{point closure period})}$$

$$\text{Timing Angle } (\circ) = \frac{180 \times (\text{timing duration}) \times (\text{Number of engine cycles})}{(\text{Number of engine cylinders}) \times (\text{point closure period})}$$

NOTE: The timing angle will indicate retarded spark (-) if the result is less than 100°. If the result is greater than or equal to 100°, you must subtract 360° from this result to get an advanced spark (+) indication.

The engine signal requirements, measurements, and calculations are summarized in Figs. 9-5 and 9-6. Note that the point signal is used to calculate both the rpm and dwell angle. This signal can be applied to the 6801 timer input at P20 (pin 8). The internal 6801 timer can then be used to measure point closure duration (A) and period (B) for the rpm and point dwell calculations. However, to measure the timing angle three signals must be monitored simultaneously; the point signal, plug No. 1 firing, and 0° TDC signal. The point signal is needed to determine the point closure period (B), with the plug No. 1 firing and 0° TDC signals needed to determine the timing duration (C). Both values (B and C) are required by the timing angle equation. Since the 6801 has only one timer and thus one timer input, the signals required for the timing angle measurement must be multiplexed into the timer. We will therefore discuss the rpm and dwell angle measurements first, and then discuss the timing angle measurement separately.

The engine monitor process flow is shown in Fig. 9-7. These five major steps must be accomplished for all the above engine monitoring functions. A general description will now be given for each of the steps shown. Specific circuit configuration will depend on the particular engine involved; however, these same general procedures will apply.

Signal Detection and Conditioning

As indicated in Fig. 9-5, each engine signal must be properly converted to a TTL compatible level before it can be used by the 6801. Therefore, each signal will be coupled to the 6801 via a *signal conditioner*. The output of the signal conditioner must be a clean TTL compatible signal containing no spikes or "glitches" that could cause false triggering. Care must also be taken to ensure that the engine's electrical system is not loaded, resulting in reduced engine performance. Therefore, the signal conditioners must represent a very high impedance load to the engine electrical system. Detection and conditioning of each of the required signals is shown in Fig. 9-8. The point closure signal may be dc coupled from the point/coil connection into a high impedance signal conditioner, since a low impedance input on the signal conditioner could degrade the per-

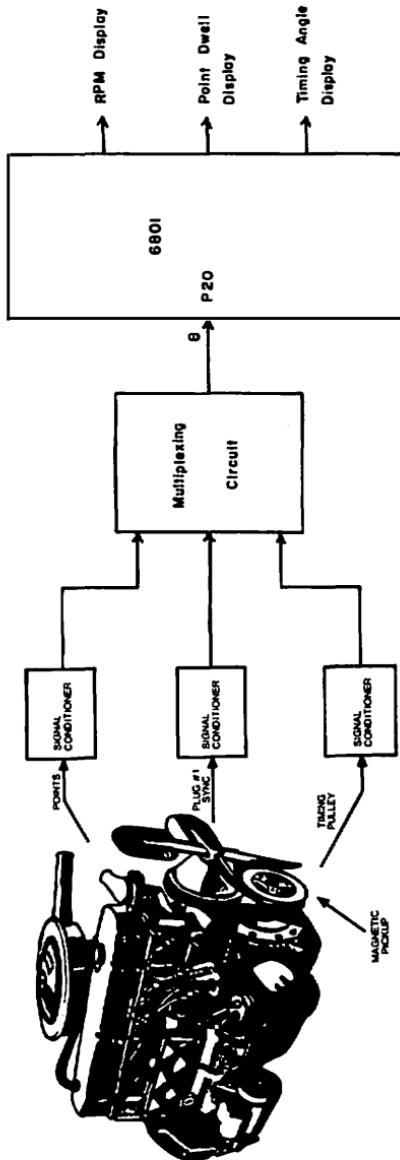


Fig. 9-5. Automobile engine signal requirements.

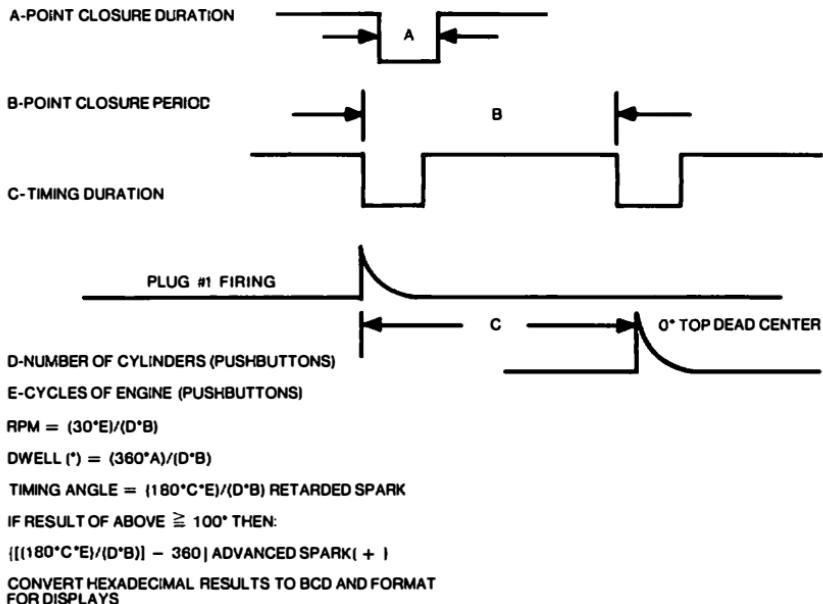


Fig. 9-6. Automobile engine signal measurements and calculations.

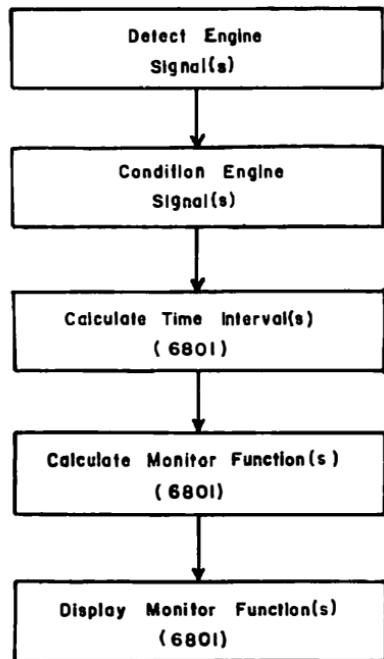


Fig. 9-7. Automobile engine monitor process flow.

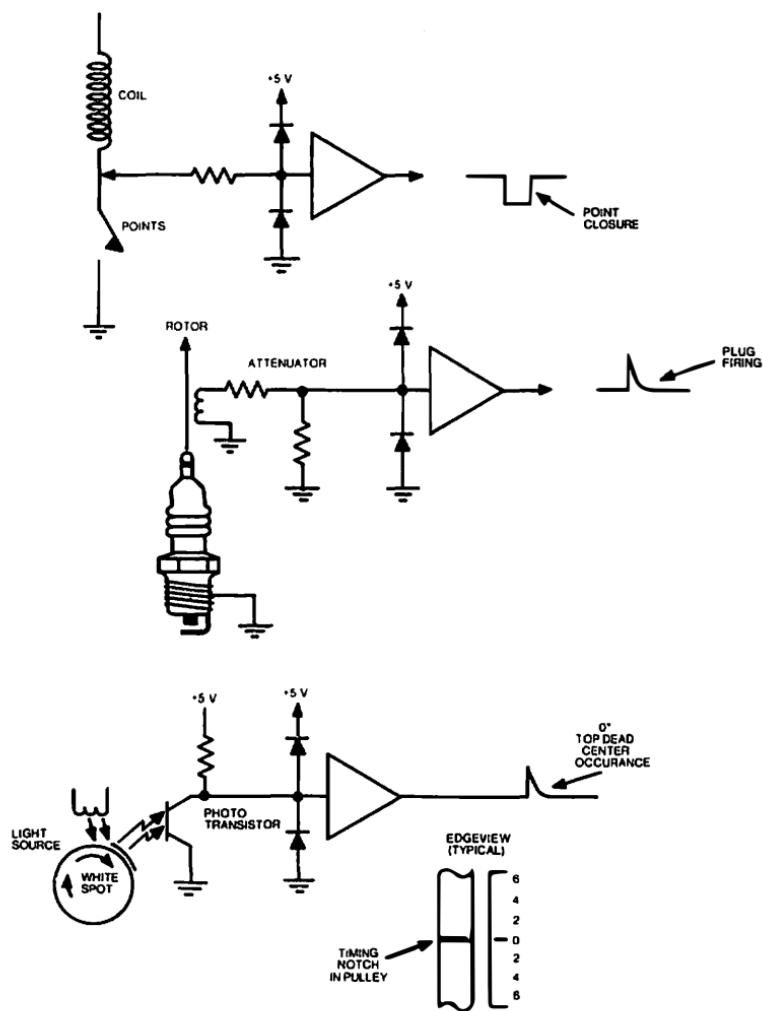


Fig. 9-8. Automobile engine signal detection and conditioning.

formance of the coil. The signal conditioner must provide a clean rectangular wave signal, free of any glitches. A CMOS Schmitt trigger circuit such as the MC14584B hex Schmitt trigger could be used here to provide both high noise immunity and high input impedance characteristics. However, when using a CMOS device such as this, care must be taken to protect the inputs against damage due to high static voltages or electric fields which might be present in an automobile engine electrical system. You must avoid application of any voltage higher than the maximum rated input voltage of this

circuit. For more information on the MC14584B, refer to its specification sheets in Appendix B. Extra circuitry might also be needed to provide the proper "window" for the Schmitt trigger, such that the required noise rejection is achieved.

The plug No. 1 firing signal *cannot* be dc coupled into the signal conditioner because of the several thousands of volts present during the plug firing. Instead, the plug firing can be detected with a small coil. The output of the coil can then be attenuated to the proper level and used to drive a standard TTL Schmitt trigger circuit such as the 7414. The coil will provide the required high impedance isolation with the 7414 providing the required noise immunity.

Finally, the 0° top dead center (TDC) signal can be detected magnetically or optically. The optical method is shown in Fig. 9-8. The timing pulley notch can be marked with a white reflective paint to indicate when piston No. 1 is at the 0° TDC position. A light source shining on the reflective white paint at 0° TDC can then be detected by a photo transistor as the paint spot passes under the light source. The photo transistor can drive a 7414 Schmitt trigger to provide the proper TTL signal to the 6801.

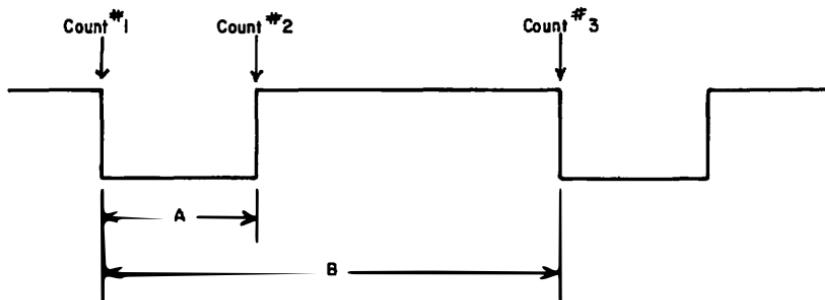
The required 6801 calculations can be divided into two general categories as follows:

- those required to measure engine rpm and point dwell,
- those required to measure timing angle.

Recall that the point signal is the only engine signal required to measure engine rpm and point dwell. However, the point signal, plug No. 1 firing signal, and 0° TDC signal are required to measure the timing angle.

RPM and Point Dwell

To measure rpm and point dwell, the conditioned point signal is entered into the 6801 via P20 at pin 8. The conditioned point signal should look like the one shown in Fig. 9-9. To calculate engine rpm, you must measure the point closure period (B). To calculate the point dwell, you must measure the point closure duration (A) and period (B). To accomplish this, the 6801 timer is configured to generate an interrupt when a high-to-low transition is detected at P20 (pin 8), indicating the points have closed. An interrupt service routine will then store the present counter contents (count No. 1) and reconfigure the timer to generate an interrupt when a subsequent low-to-high transition is detected, indicating the points have opened. When this transition has been detected, the service routine must take the difference between the counter's contents at this time (count No. 2) and the previously stored value (count No. 1). The



$$A = \text{Point Closure Duration} = \text{Count } \#2 - \text{Count } \#1$$

$$B = \text{Point Closure Period} = \text{Count } \#3 - \text{Count } \#1$$

Fig. 9-9. Point signal measurements.

difference will be the point closure duration (A). The service routine must then reconfigure the timer to generate an interrupt on the next high-to-low transition at pin 8, indicating that the points have closed again. The count present at this transition is count No. 3 (Fig. 9-9). To calculate the point closure period (B), the service routine must take the difference between this and count No. 1. One word of caution: typical engine idle speeds are 500-600 r/min. This translates to a point closure period of 120-100 milliseconds, respectively. Therefore, for low rpm's, the 6801 counter will roll over during the point closure period, and the number of roll-overs must be counted, using the timer overflow flag (OCF) and output compare register, to obtain the proper reading. Once the point closure duration (A) and period (B) values are obtained, the program can calculate the engine rpm and dwell angle using the given equations. The techniques and programs discussed in Chapter 6 should provide enough information for you to write the required main program and interrupt service routine.

Timing Angle

The timing angle signals can be coupled to the 6801 at P20 (pin 8) via the circuit shown in Fig. 9-10. Since three engine signals are required to measure timing angle and the 6801 has only one timer input, the signals must be multiplexed. The 7476 JK flip/flop will provide a pulse whose width is the difference in time between plug trigger circuit such as the MC14584B hex Schmitt trigger could be No. 1 firing and when piston No. 1 is at 0° TDC (timing duration). This pulse is then multiplexed with the points signal into the timer input at P20 (pin 8). Multiplexing is provided with a 74LS126 3-state buffer. The buffer enable lines are connected to P22 and P23 of

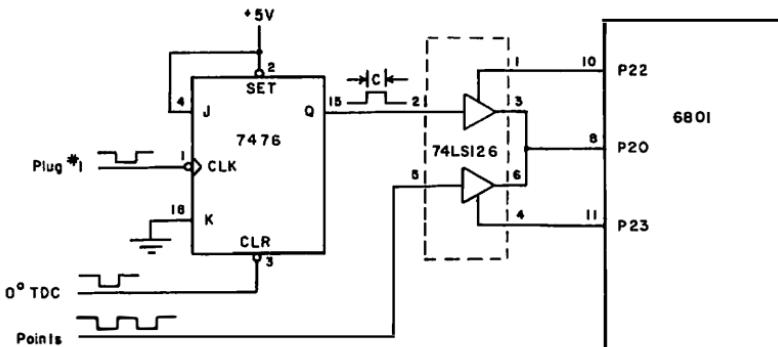


Fig. 9-10. Timing angle measurement—multiplexing circuit.

the 6801. You will configure these two port lines for output and alternately write a "1" to each port line to enable the respective 3-state buffer. In this way, you can measure the timing duration (C), then the point closure period (B) to calculate the timing angle.

The 7476 JK flip/flop output at Q will provide a positive going pulse whose width (C) is the timing duration (Figs. 9-6 and 9-10). Recall that the timing duration is simply the difference in time between the plug No. 1 firing and 0° TDC signals. When plug No. 1 fires, the flip-flop is clocked and its normally low output goes high until the 0° TDC signal clears the flip/flop and causes the output at Q to go low. This circuit assumes negative going plug No. 1 and 0° TDC signals are obtained from the respective signal conditioning circuits. To enter the timing duration signal (C) into the 6801 timer at P20 (pin 8), you will first enable its 3-state buffer by writing a "1" to P22 at address 0003. At the same time, the P23 bit at address 0003 must contain a "0" such that the points signal 3-state buffer is not simultaneously enabled. It is assumed that P22 and P23 have already been configured as output data lines via the port 2 data direction register at address 0001. To measure the timing duration (C), you will use the pulse width measurement mode of the 6801 timer. The timer will be configured to generate an interrupt when a low-to-high transition is detected on P20 (pin 8), indicating that plug No. 1 has fired. The interrupt service routine will store the present counter contents, then reconfigure the timer to recognize a subsequent high-to-low transition at P20 (pin 8), indicating piston No. 1 is at 0° TDC. When this transition is detected, the timing duration (C) is determined by taking the difference between the counter contents at this time and the previously stored value. The timing duration value (C) is then stored for the timing angle calculation and its 3-state buffer is disabled by writing a "0" to P22. Next, you will write a "1" to P23 to enable the points signal three-state buffer. The point closure period (B) can then be measured as

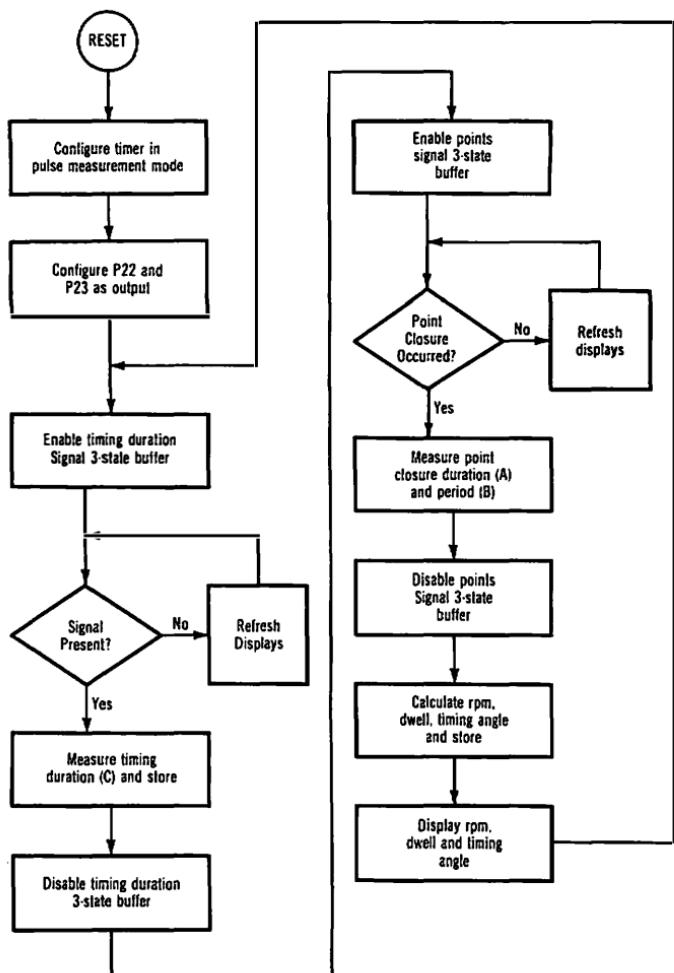
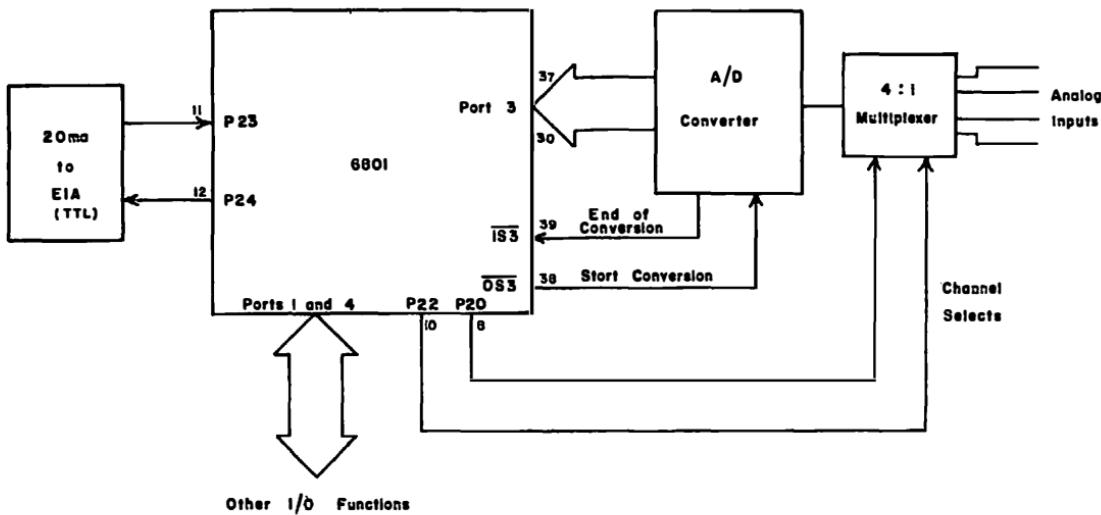


Fig. 9-11. Automobile engine rpm, dwell, and timing angle flowchart.

previously discussed. Once this value has been obtained, the timing angle can be calculated using the timing angle formula.

You can also measure the point closure duration (A) from the points signal at this time. In this way, you can simultaneously calculate the rpm, dwell, and timing angle. These values can then be stored and displayed on 7-segment LEDs as discussed in Chapter 8. The above process is then repeated to continually up-date the displays with new engine monitoring information. An engine rpm, dwell, and timing angle measurement flowchart is shown in Fig. 9-11. This chart summarizes the discussion just completed.

Fig. 9-12. Remote data-acquisition controller.



REMOTE DATA ACQUISITION

An application which demonstrates the utility of the 6801 in a stand-alone (single-chip) application is that of a remote data acquisition controller. Such a controller is shown in Fig. 9-12. Here, you will observe that we are using many of the 6801's capabilities. Serial communication is provided via port 2 and an RS-232C connection. The RS-232C interface can be subsequently converted to a 20-milliampere current format for longer distance communication with a loop-to-EIA converter circuit. Four analog inputs are multiplexed with a 4:1 multiplexer circuit. Channel selection is accomplished by using P20 and P22 as output lines. Any one of the four analog inputs may be selected by writing the proper 1's and 0's to the P20 and P22 data output register bits. The a/d converter will convert the selected analog signal to digital. The digital information is then fed to port 3 of the 6801. Port 3 is configured for input hand-shaking (reference Example 4-1). Each time a conversion is completed, an interrupt will be generated via the port 3 input strobe (\overline{IS}_3). Upon receiving the interrupt, the 6801 will read the digital data at port 3. When the read port 3 operation is performed, an output strobe will be generated to signal the a/d converter to start another conversion. The output strobe is provided via the port 3 output strobe line (\overline{OS}_3). In this application, ports 1 and 4 are still available for additional parallel i/o functions.

REVIEW QUESTIONS

1. The RS-232C standard uses _____ volt levels to represent data.
2. The _____ and _____ devices are the only extra hardware required to interface the 6801 SCI to the RS-232C standard.
3. To convert 20-mA current loop devices to RS-232C, you would use a _____.
4. A _____ chip circuit is used to generate a pulse train to the 6801 for temperature measurement.
5. What determines the output duty cycle of the MC1555 in Fig. 9-4?
6. The 6801 timer is used in the _____ mode to monitor temperature.
7. What engine signal(s) is required to measure rpm and dwell angle?

8. What engine signal(s) is required to measure timing angle?
9. The five steps required for monitoring engine RPM, dwell angle, and timing angle are:
10. A _____ circuit is used to provide the timing duration signal required to measure timing angle.
11. A _____ circuit can be used to provide both high noise immunity and high input impedance when interfacing to an automobile engine.
12. The only engine signal discussed which may be dc coupled into the 6801 is the _____ signal.
13. The timing duration signal is determined from the _____ and _____ engine signals.
14. Explain how the timing duration signal and points signal can be multiplexed into the 6801 timer input to provide monitoring of all three engine functions discussed in this chapter.
15. Describe the overall 6801 process flow required to monitor and display engine rpm, dwell angle, and timing angle.

ANSWERS

1. ± 12
2. MC1488 and MC1489A
3. loop-to-EIA converter (refer to Fig. 9-3)
4. MC1555 (555) timer
5. The input resistance ratio.
6. pulse measurement
7. Points signal.
8. Points signal, plug No. 1 firing signal, and 0° TDC signal
9. Detect the engine signal(s)
Condition the engine signal(s)
Calculate the required time interval(s)
Calculate the monitor function(s)
Display the monitor function(s)
10. 7476 J-K flip/flop

11. CMOS Schmitt trigger (MC14584B)
12. points
13. plug No. 1 firing and 0° TDC
14. The timing duration signal and points signal can be multiplexed into the 6801 timer input using two 3-state buffers. The buffer enable lines are connected to P22 and P23 of the 6801. Multiplexing is provided by configuring P22 and P23 for output and alternately writing 1's to the P22 and P23 bits of the port 2 data i/o register.
15. See Fig. 9-11.

APPENDIX A

6801/68701/6803 Instruction Set

The following pages contain detailed definitions of the 83 executable instructions. These pages are provided through the courtesy of Motorola Semiconductor Products, Inc., Austin, TX.

A.1 Nomenclature

The following nomenclature is used in the subsequent definitions.

(a) *Operators*

()	= contents of
\leftarrow	= is transferred to
\uparrow	= "is pulled from stack"
\downarrow	= "is pushed into stack"
\wedge	= Boolean AND
\odot	= Boolean (Inclusive) OR
\oplus	= Exclusive OR
\approx	= Boolean NOT

(b) *Registers in the MPU*

ACCA	= Accumulator A
ACCB	= Accumulator B
ACCX	= Accumulator ACCA or ACCB
ACCD	= Accumulator D
CC	= Condition codes register
IX	= Index register, 16 bits
IXH	= Index register, higher order 8 bits
IXL	= Index register, lower order 8 bits
PC	= Program counter, 16 bits
PCH	= Program counter, higher order 8 bits
PCL	= Program counter, lower order 8 bits
SP	= Stack pointer
SPH	= Stack pointer high
SPL	= Stack pointer low

(c) *Memory and Addressing*

M	= A memory location (one byte)
M + 1	= The byte of memory at 0001 plus the address of the memory location indicated by "M."
Rel	= Relative address (i.e. the two's complement number stored in the second byte of machine code corresponding to a branch instruction).

(d) *Bits 0 thru 5 of the Condition Codes Register*

C	= Carry — borrow	bit — 0
V	= Two's complement overflow indicator	bit — 1
Z	= Zero indicator	bit — 2
N	= Negative indicator	bit — 3
I	= Interrupt mask	bit — 4
H	= Half carry	bit — 5

(e) *Status of Individual Bits BEFORE Execution of an Instruction*

An	= Bit n of ACCA (n=7,6,5,...,0)
Bn	= Bit n of ACCB (n=7,6,5,...,0)
IXHn	= Bit n of IXH (n=7,6,5,...,0)
IXLn	= Bit n of IXL (n=7,6,5,...,0)
Mn	= Bit n of M (n=7,6,5,...,0)
SPHn	= Bit n of SPH (n=7,6,5,...,0)
SPLn	= Bit n of SPL (n=7,6,5,...,0)
Xn	= Bit n of ACCX (n=7,6,5,...,0)

(f) *Status of Individual Bits of the RESULT of Execution of an Instruction*

(i) *For 8-bit Results*

Rn = Bit n of the result (n = 7,6,5,...,0)

This applies to instructions which provide a result contained in a single byte of memory or in an 8-bit register.

(ii) For 16-bit Results

R_{Hn} = Bit n of the more significant byte of the result

(n = 7, 6, 5, ..., 0)

R_{Ln} = Bit n of the less significant byte of the result

(n = 7, 6, 5, ..., 0)

This applies to instructions which provide a result contained in two consecutive bytes of memory or in a 16-bit register.

A.2 Executable Instructions (definition of)

Detailed definitions of the 72 executable instructions of the source language are provided on the following pages.

ABA

Operation: ACCA \leftarrow (ACCA) + (ACCB)

Description: Adds the contents of ACCB to the contents of ACCA and places the result in ACCA.

Condition Codes: H: Set if there was a carry from bit 3; cleared otherwise.

I: Not affected.

N: Set if most significant bit of the result is set; cleared otherwise.

Z: Set if all bits of the result are cleared; cleared otherwise.

V: Set if there was two's complement overflow as a result of the operation; cleared otherwise.

C: Set if there was a carry from the most significant bit of the result; cleared otherwise.

Boolean Formulae for Condition Codes:

$$H = A_3 \cdot B_3 + B_3 \cdot \bar{R}_3 + \bar{R}_3 \cdot A_3$$

$$N = R_7$$

$$Z = \bar{R}_7 \cdot \bar{R}_6 \cdot \bar{R}_5 \cdot \bar{R}_4 \cdot \bar{R}_3 \cdot \bar{R}_2 \cdot \bar{R}_1 \cdot \bar{R}_0$$

$$V = A_7 \cdot B_7 \cdot \bar{R}_7 + \bar{A}_7 \cdot \bar{B}_7 \cdot R_7$$

$$C = A_7 \cdot B_7 + B_7 \cdot \bar{R}_7 + \bar{R}_7 \cdot A_7$$

Addressing Modes, Execution Time, and Machine Code (hexadecimal/ octal/ decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
Inherent	2	1	1B	033	027

Add Accumulator B to Index Register

ABX

Operation: X \leftarrow B + X

Description: Adds the contents of ACCB to the contents of the Index register and places the result in the Index register.

Condition Codes: Not affected.

Addressing Modes, Execution Time, and Machine Code (Hexadecimal/octal/decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
Inherent	3	1	3A	072	058

Add with Carry

ADC

Operation: $ACCX \leftarrow (ACCX) + (M) + (C)$

Description: Adds the contents of the C bit to the sum of the contents of ACCX and M, and places the result in ACCX.

Condition Codes: H Set if there was a carry from bit 3; cleared otherwise.

I: Not affected.

N: Set if most significant bit of the result is set; cleared otherwise.

Z: Set if all bits of the result are cleared; cleared otherwise.

V: Set if there was two's complement overflow as a result of the operation; cleared otherwise.

C: Set if there was a carry from the most significant bit of the result; cleared otherwise.

Boolean Formulae for Condition Codes:

$$H = X_3 \cdot M_3 + M_3 \cdot \bar{R}_3 + \bar{R}_3 \cdot X_3$$

$$N = R_7$$

$$Z = \bar{R}_7 \cdot \bar{R}_6 \cdot \bar{R}_5 \cdot \bar{R}_4 \cdot \bar{R}_3 \cdot \bar{R}_2 \cdot \bar{R}_1 \cdot \bar{R}_0$$

$$V = X_7 \cdot M_7 \cdot \bar{R}_7 + \bar{X}_7 \cdot \bar{M}_7 \cdot R_7$$

$$C = X_7 \cdot M_7 + M_7 \cdot \bar{R}_7 + \bar{R}_7 \cdot X_7$$

Addressing Formats:

Addressing Modes, Execution Time, and Machine Code (hexadecimal/ octal/ decimal):

(DUAL OPERAND)

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
A IMM	2	2	89	211	137
A DIR	3	2	99	231	153
A EXT	4	3	B9	271	185
A IND	4	2	A9	251	169
B IMM	2	2	C9	311	201
B DIR	3	2	D9	331	217
B EXT	4	3	F9	371	249
B IND	4	2	E9	351	233

Add Double Accumulator**ADDD**Operation: $D \leftarrow D + M:M + 1$

Description: Adds ACCD to M:M + 1 and places the result in ACCD

Condition Codes: H: Not affected.

I: Not affected.

N: Set if most significant bit of the result is set; cleared otherwise.

Z: Set if all bits of the result are cleared; cleared otherwise.

V: Set If there was two's complement overflow as a result of the operation; cleared otherwise.

C: Set if there was a carry from the most significant bit of the result; cleared otherwise.

Addressing Formats:**Addressing Modes, Execution Time, and Machine Code (hexadecimal/octal/decimal):**

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	DEC.	OCT.
D IMM	4	3	C3	303	195
D DIR	5	2	D3	323	211
D EXT	6	3	F3	363	243
D IND	6	2	E3	343	227

Add Without Carry**ADD**Operation: $ACCX \leftarrow (ACCX) + (M)$

Description: Adds the contents of ACCX and the contents of M and places the result in ACCX.

Condition Codes: H: Set if there was a carry from bit 3; cleared otherwise.

I: Not affected.

N: Set if most significant bit of the result is set; cleared otherwise.

Z: Set if all bits of the result are cleared; cleared otherwise.

V: Set If there was two's complement overflow as a result of the operation; cleared otherwise.

C: Set if there was a carry from the most significant bit of the result; cleared otherwise.

Boolean Formulae for Condition Codes:

$$H = X_3 \cdot M_3 + M_3 \cdot \bar{R}_3 + \bar{R}_3 \cdot X_3$$

$$N = R_7$$

$$Z = \bar{R}_7 \cdot \bar{R}_6 \cdot \bar{R}_5 \cdot \bar{R}_4 \cdot \bar{R}_3 \cdot \bar{R}_2 \cdot \bar{R}_1 \cdot \bar{R}_0$$

$$V = X_7 \cdot M_7 \cdot \bar{R}_7 + \bar{X}_7 \cdot \bar{M}_7 \cdot R_7$$

$$C = X_7 \cdot M_7 + M_7 \cdot \bar{R}_7 + \bar{R}_7 \cdot X_7$$

Addressing Formats:

Addressing Modes, Execution Time, and Machine Code (hexadecimal/octal/decimal):
 (DUAL OPERAND)

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
A IMM	2	2	8B	213	139
A DIR	3	2	9B	233	155
A EXT	4	3	BB	273	187
A IND	4	2	AB	253	171
B IMM	2	2	CB	313	203
B DIR	3	2	DB	333	219
B EXT	4	3	FB	373	251
B IND	4	2	EB	353	235

Logical AND

AND

Operation: $ACCX \leftarrow (ACCX) \cdot (M)$

Description: Performs logical "AND" between the contents of ACCX and the contents of M and places the result in ACCX. (Each bit of ACCX after the operation will be the logical "AND" of the corresponding bits of M and of ACCX before the operation.)

Condition Codes: H: Not affected.
 I: Not affected.
 N: Set if most significant bit of the result is set; cleared otherwise.
 Z: Set if all bits of the result are cleared; cleared otherwise.
 V: Cleared.
 C: Not affected.

Boolean Formulae for Condition Codes:

$$\begin{aligned}N &= R_7 \\Z &= \bar{R}_7 \cdot \bar{R}_6 \cdot \bar{R}_5 \cdot \bar{R}_4 \cdot \bar{R}_3 \cdot \bar{R}_2 \cdot \bar{R}_1 \cdot \bar{R}_0 \\V &= 0\end{aligned}$$

Addressing Formats:

Addressing Modes, Execution Time, and Machine Code (hexadecimal/octal/decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
A IMM	2	2	84	204	132
A DIR	3	2	94	224	148
A EXT	4	3	B4	264	180
A IND	4	2	A4	244	164
B IMM	2	2	C4	304	196
B DIR	3	2	D4	324	212
B EXT	4	3	F4	364	244
B IND	4	2	E4	344	228

Shift Left Double Accumulator**ASLD**

Operation: $C \leftarrow \boxed{} \dots \boxed{} \leftarrow 0$
 b₁₅ b₀

Description: Shifts all bits of ACCD one place to the left. Bit 0 is loaded with a zero. The C bit is loaded from the most significant bit of ACCD.

Condition Codes: H: Not affected.
 I: Not affected.

N: Set if most significant bit of the result is set; cleared otherwise.

Z: Set if all bits of the result are cleared; cleared otherwise.

V: Set if, after completion of the shift operation, EITHER (N is set and C is cleared) OR (N is cleared and C is set); cleared otherwise.

C: Set if, before the operation, the most significant bit of ACCD was set; cleared otherwise.

Addressing Formats:**Addressing Modes, Execution Time, and Machine Code (Hexadecimal/octal/decimal):**

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
D Inherent	3	1	05	005	005

Arithmetic Shift Left**ASL**

Operation:

Description: Shifts all bits of the ACCX or M one place to the left. Bit 0 is loaded with a zero. The C bit is loaded from the most significant bit of ACCX or M.

Condition Codes: H: Not affected.
 I: Not affected.

N: Set if most significant bit of the result is set; cleared otherwise.

Z: Set if all bits of the result are cleared; cleared otherwise.

V: Set if, after the completion of the shift operation, EITHER (N is set and C is cleared) OR (N is cleared and C is set); cleared otherwise.

C: Set if, before the operation, the most significant bit of the ACCX or M was set; cleared otherwise.

Boolean Formulae for Condition Codes:

$$N = R_7$$

$$Z = \bar{R}_7 \cdot \bar{R}_6 \cdot \bar{R}_5 \cdot \bar{R}_4 \cdot \bar{R}_3 \cdot \bar{R}_2 \cdot \bar{R}_1 \cdot \bar{R}_0$$

$$V = N \oplus C = [N \cdot \bar{C}] \odot [\bar{N} \cdot C]$$

(the foregoing formula assumes values of N and C after the shift operation)

$$C = M_7$$

Addressing Formats:

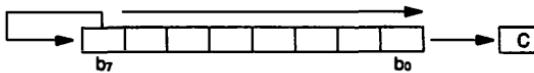
Addressing Modes, Execution Time, and Machine Code (hexadecimal/octal/decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
A	2	1	48	110	072
B	2	1	58	130	088
EXT	6	3	78	170	120
IND	6	2	68	150	104

Arithmetic Shift Right

ASR

Operation:



Description: Shifts all bits of ACCX or M one place to the right. Bit 7 is held constant. Bit 0 is loaded into the C bit.

Condition Codes: H: Not affected.

I: Not affected.

N: Set if the most significant bit of the result is set; cleared otherwise.

Z: Set if all bits of the result are cleared; cleared otherwise.

V: Set if, after the completion of the shift operation, EITHER (N is set and C is cleared) OR (N is cleared and C is set); cleared otherwise.

C: Set if, before the operation, the least significant bit of the ACCX or M was set; cleared otherwise.

Boolean Formulae for Condition Codes:

$$N = R_7$$

$$Z = \bar{R}_7 \cdot \bar{R}_6 \cdot \bar{R}_5 \cdot \bar{R}_4 \cdot \bar{R}_3 \cdot \bar{R}_2 \cdot \bar{R}_1 \cdot \bar{R}_0$$

$$V = N \oplus C = [N \cdot \bar{C}] \odot [\bar{N} \cdot C]$$

(the foregoing formula assumes values of N and C after the shift operation)

$$C = M_0$$

Addressing Formats:

Addressing Modes, Execution Time, and Machine Code (hexadecimal/octal/decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
A	2	1	47	107	071
B	2	1	57	127	087
EXT	6	3	77	167	119
IND	6	2	67	147	103

Branch If Carry Clear**BCC**Operation: $PC \leftarrow (PC) + 0002 + Rel\ if\ (C)=0$

Description: Tests the state of the C bit and causes a branch if C is clear.

See BRA instruction for further details of the execution of the branch.

Condition Codes: Not affected.

Addressing Formats:

Addressing Modes, Execution Time, and Machine Code (hexadecimal/ octal/ decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
REL	3	2	24	044	036

BCS**Branch If Carry Set**Operation: $PC \leftarrow (PC) + 0002 + Rel\ if\ (C)=1$

Description: Tests the state of the C bit and causes a branch if C is set.

See BRA instruction for further details of the execution of the branch.

Condition Codes: Not affected.

Addressing Formats:

Addressing Modes, Execution Time, and Machine Code (hexadecimal/ octal/ decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
REL	3	2	25	045	037

Branch If Equal**BEQ**Operation: $PC \leftarrow (PC) + 0002 + Rel\ if\ (Z)=1$

Description: Tests the state of the Z bit and causes a branch if the Z bit is set.

See BRA instruction for further details of the execution of the branch.

Condition Codes: Not affected.

Addressing Formats:

Addressing Modes, Execution Time, and Machine Code (hexadecimal/ octal/ decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
REL	3	2	27	047	039

Branch If Greater than or Equal to Zero

Operation: $PC \leftarrow (PC) + 0002 + \text{Rel if } (N) \oplus (V) = 0$
 i.e. if $(ACCX) \geq (M)$
 (Two's complement numbers)

Description: Causes a branch if (N is set and V is set) OR (N is clear and V is clear).

If the BGE instruction is executed immediately after execution of any of the instructions CBA, CMP, SBA, or SUB, the branch will occur if and only if the two's complement number represented by the minuend (i.e. ACCX) was greater than or equal to the two's complement number represented by the subtrahend (i.e. M).

See BRA instruction for details of the branch.

Condition Codes: Not affected.

Addressing Formats:

Addressing Modes, Execution Time, and Machine Code (hexadecimal/ octal/ decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
REL	3	2	2C	054	044

Branch If Greater than Zero

Operation: $PC \leftarrow (PC) + 0002 + \text{Rel if } (Z) \odot [(N) \oplus (V)] = 0$
 i.e. if $(ACCX) > (M)$
 (two's complement numbers)

Description: Causes a branch if [Z is clear] AND [(N is set and V is set) OR (N is clear and V is clear)].

If the BGT instruction is executed immediately after execution of any of the instructions CBA, CMP, SBA, or SUB, the branch will occur if and only if the two's complement number represented by the minuend (i.e. ACCX) was greater than the two's complement number represented by the subtrahend (i.e. M).

See BRA instruction for details of the branch.

Condition Codes: Not affected.

Addressing Formats:

Addressing Modes, Execution Time, and Machine Code (hexadecimal/ octal/ decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
REL	3	2	2E	056	046

Branch If Higher

Operation: $PC \leftarrow (PC) + 0002 + \text{Rel if } (C) \cdot (Z) = 0$
 i.e. if $(ACCX) > (M)$
 (unsigned binary numbers)

Description: Causes a branch if (C is clear) AND (Z is clear).
 If the BHI instruction is executed immediately after execution of any of the instructions CBA, CMP, SBA, or SUB, the branch will occur if and only if the unsigned binary number represented by the minuend (i.e. ACCX) was greater than the unsigned binary number represented by the subtrahend (i.e. M).
 See BRA instruction for details of the execution of the branch.

Condition Codes: Not affected.

Addressing Formats:

Addressing Modes, Execution Time, and Machine Code (hexadecimal/octal/decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
REL	3	2	22	042	034

BIT Test

BIT

Operation: (ACCX) · (M)
Description: Performs the logical "AND" comparison of the contents of ACCX and the contents of M and modifies condition codes accordingly. Neither the contents of ACCX or M operands are affected. (Each bit of the result of the "AND" would be the logical "AND" of the corresponding bits of M and ACCX.)
Condition Codes: H: Not affected.
 I: Not affected.
 N: Set if the most significant bit of the result of the "AND" would be set; cleared otherwise.
 Z: Set if all bits of the result of the "AND" would be cleared; cleared otherwise.
 V: Cleared.
 C: Not affected.

Boolean Formulae for Condition Codes:

$$\begin{aligned}N &= R_7 \\Z &= \bar{R}_7 \cdot \bar{R}_6 \cdot \bar{R}_5 \cdot \bar{R}_4 \cdot \bar{R}_3 \cdot \bar{R}_2 \cdot \bar{R}_1 \cdot \bar{R}_0 \\V &= 0\end{aligned}$$

Addressing Formats:

Addressing Modes, Execution Time, and Machine Code (hexadecimal/octal/decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
A IMM	2	2	85	205	133
A DIR	3	2	95	225	149
A EXT	4	3	B5	265	181
A IND	4	2	A5	245	165
B IMM	2	2	C5	305	197
B DIR	3	2	D5	325	213
B EXT	4	3	F5	385	245
B IND	4	2	E5	345	229

BLE**Branch If Less than or Equal to Zero**

Operation: $PC \leftarrow (PC) + 0002 + Rel \text{ if } (Z) \odot ((N) \oplus (V)) = 1$
 i.e. if $(ACCX) \leq (M)$
 (two's complement numbers)

Description: Causes a branch if [Z is set] OR [(N is set and V is clear) OR (N is clear and V is set)].

If the BLE instruction is executed immediately after execution of any of the instructions CBA, CMP, SBA, or SUB, the branch will occur if and only if the two's complement number represented by the minuend (i.e. ACCX) was less than or equal to the two's complement number represented by the subtrahend (i.e. M).

See BRA instruction for details of the branch.

Condition Codes: Not affected.

Addressing Formats:

Addressing Modes, Execution Time, and Machine Code (hexadecimal/octal/decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
REL	.3	2	2F	057	047

BLS**Branch If Lower or Same**

Operation: $PC \leftarrow (PC) + 0002 + Rel \text{ if } (C) \odot (Z) = 1$
 i.e. if $(ACCX) \leq (M)$
 (unsigned binary numbers)

Description: Causes a branch if (C is set) OR (Z is set).

If the BLS instruction is executed immediately after execution of any of the instructions CBA, CMP, SBA, or SUB, the branch will occur if and only if the unsigned binary number represented by the minuend (i.e. ACCX) was less than or equal to the unsigned binary number represented by the subtrahend (i.e. M).

See BRA instruction for details of the execution of the branch.

Condition Codes: Not affected.

Addressing Formats:

Addressing Modes, Execution Time, and Machine Code (hexadecimal/octal/decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
REL	3	2	23	043	035

BLT**Branch If Less than Zero**Operation: $PC \leftarrow (PC) + 0002 + Rel \text{ if } (N) \oplus (V) = 1$ i.e. if $(ACCX) < (M)$
(two's complement numbers)

Description: Causes a branch if (N is set and V is clear) OR (N is clear and V is set).
 If the BLT instruction is executed immediately after execution of any of the instructions CBA, CMP, SBA, or SUB, the branch will occur if and only if the two's complement number represented by the minuend (i.e. ACCX) was less than the two's complement number represented by the subtrahend (i.e. M).
 See BRA instruction for details of the branch.

Condition Codes: Not affected.

Addressing Formats:

Addressing Modes, Execution Time, and Machine Code (hexadecimal/ octal/ decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
REL	3	2	2D	055	045

BMI**Branch If Minus**Operation: $PC \leftarrow (PC) + 0002 + Rel \text{ if } (N) = 1$

Description: Tests the state of the N bit and causes a branch if N is set.
 See BRA instruction for details of the execution of the branch.

Condition Codes: Not affected.

Addressing Formats:

Addressing Modes, Execution Time, and Machine Code (hexadecimal/ octal/ decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
REL	3	2	2B	053	043

BNE**Branch If Not Equal**Operation: $PC \leftarrow (PC) + 0002 + Rel \text{ if } (Z) = 0$

Description: Tests the state of the Z bit and causes a branch if the Z bit is clear.
 See BRA instruction for details of the execution of the branch.

Condition Codes: Not affected.

Addressing Formats:**Addressing Modes, Execution Time, and Machine Code (hexadecimal/ octal/ decimal):**

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
REL	3	2	26	046	038

Branch If Plus**BPL**Operation: $PC \leftarrow (PC) + 0002 + Rel \text{ if } (N) = 0$

Description: Tests the state of the N bit and causes a branch if N is clear.

See BRA instruction for details of the execution of the branch.

Condition Codes: Not affected.

Addressing Formats:**Addressing Modes, Execution Time, and Machine Code (hexadecimal/ octal/ decimal):**

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
REL	3	2	2A	052	042

Branch Always**BRA**Operation: $PC \leftarrow (PC) + 0002 + Rel$

Description: Unconditional branch to the address given by the foregoing formula, in which R is the relative address stored as a two's complement number in the second byte of machine code corresponding to the branch instruction.

Note: The source program specifies the destination of any branch instruction by its absolute address, either as a numerical value or as a symbol or expression which can be numerically evaluated by the assembler. The assembler obtains the relative address R from the absolute address and the current value of the program counter PC.

Condition Codes: Not affected.

Addressing Formats:**Addressing Modes, Execution Time, and Machine Code (hexadecimal/ octal/ decimal):**

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
REL	3	2	20	040	032

Branch Never**BRN**Operation: $PC \leftarrow PC + 2$

Description: This is a two-byte instruction which causes only the program counter to be incremented twice. The relative address byte will be ignored. No other registers are affected.

Condition Codes: Not affected.

Addressing Formats:

Addressing Modes, Execution Time, and Machine Code (hexadecimal/octal/decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
Relative	3	2	21	041	033

Branch to Subroutine**BSR**Operation: $PC \leftarrow (PC) + 0002$ $\downarrow (PCL)$ $SP \leftarrow (SP) - 0001$ $\downarrow (PCH)$ $SP \leftarrow (SP) - 0001$ $PC \leftarrow (PC) + Rel$

Description: The program counter is incremented by 2. The less significant byte of the contents of the program counter is pushed into the stack. The stack pointer is then decremented (by 1). The more significant byte of the contents of the program counter is then pushed into the stack. The stack pointer is again decremented (by 1). A branch then occurs to the location specified by the program.

See BRA instruction for details of the execution of the branch.

Condition Codes: Not affected.

Addressing Formats:

Addressing Modes, Execution Time, and Machine Code (hexadecimal/octal/decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
REL	6	2	8D	215	141

BRANCH TO SUBROUTINE EXAMPLE

		Memory Location	Machine Code (Hex)	Label	Assembler Language Operator	Operand
A.	Before					
	PC	←	\$1000 8D \$1001 50		BSR	CHARLI
	SP	←	\$EFFF			
B.	After					
	PC	←	\$1052 ..	CHARLI
	SP	←	\$EFFD \$EFFE 10 \$EFFF 02			

Branch If Overflow Clear

BVC

Operation: $PC \leftarrow (PC) + 0002 + \text{Rel if } (V) = 0$

Description: Tests the state of the V bit and causes a branch if the V bit is clear.

See BRA instruction for details of the execution of the branch.

Condition Codes: Not affected.

Addressing Formats:

Addressing Modes, Execution Time, and Machine Code (hexadecimal/ octal/ decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
REL	3	2	28	050	040

Branch If Overflow Set

BVS

Operation: $PC \leftarrow (PC) + 0002 + \text{Rel if } (V) = 1$

Description: Tests the state of the V bit and causes a branch if the V bit is set.

See BRA instruction for details of the execution of the branch.

Condition Codes: Not affected.

Addressing Formats:

Addressing Modes, Execution Time, and Machine Code (hexadecimal/ octal/ decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
REL	3	2	29	051	041

CBA**Compare Accumulators**

Operation: (ACCA) – (ACCB)

Description: Compares the contents of ACCA and the contents of ACCB and sets the condition codes, which may be used for arithmetic and logical conditional branches. Both operands are unaffected.

Condition Codes: H: Not affected.

I: Not affected.

N: Set if the most significant bit of the result of the subtraction would be set; cleared otherwise.

Z: Set if all bits of the result of the subtraction would be cleared; cleared otherwise.

V: Set if the subtraction would cause two's complement overflow; cleared otherwise.

C: Set if the subtraction would require a borrow into the most significant bit of the result; clear otherwise.

Boolean Formulae for Condition Codes:

$$N = R_7$$

$$Z = \bar{R}_7 \cdot \bar{R}_6 \cdot \bar{R}_5 \cdot \bar{R}_4 \cdot \bar{R}_3 \cdot \bar{R}_2 \cdot \bar{R}_1 \cdot \bar{R}_0$$

$$V = A_7 \cdot \bar{B}_7 \cdot \bar{R}_7 + \bar{A}_7 \cdot B_7 \cdot R_7$$

$$C = \bar{A}_7 \cdot B_7 + B_7 \cdot R_7 + R_7 \cdot \bar{A}_7$$

Addressing Modes, Execution Time, and Machine Code (hexadecimal/octal/decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
INHERENT	2	1	11	021	017

CLC**Clear Carry**Operation: C bit $\leftarrow 0$

Description: Clears the carry bit in the processor condition codes register.

Condition Codes: H: Not affected.

I: Not affected.

N: Not affected.

Z: Not affected.

V: Not affected.

C: Cleared

Boolean Formulae for Condition Codes:

$$C = 0$$

Addressing Modes, Execution Time, and Machine Code (hexadecimal/octal/decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
INHERENT	2	1	0C	014	012

Clear Interrupt Mask**CLI**Operation: I bit \leftarrow 0

Description: Clears the interrupt mask bit in the processor condition codes register. This enables the microprocessor to service an interrupt from a peripheral device if signalled by a high state of the "Interrupt Request" control input.

Condition Codes: H: Not affected.
 I: Cleared.
 N: Not affected.
 Z: Not affected.
 V: Not affected.
 C: Not affected.

Boolean Formulae for Condition Codes:

$$I = 0$$

Addressing Modes, Execution Time, and Machine Code (hexadecimal/octal/decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
INHERENT	2	1	0E	016	014

Clear**CLR**Operation: ACCX \leftarrow 00or: M \leftarrow 00

Description: The contents of ACCX or M are replaced with zeros.

Condition Codes: H: Not affected.
 I: Not affected.
 N: Cleared
 Z: Set
 V: Cleared
 C: Cleared

Boolean Formulae for Condition Codes:

$$N = 0$$

$$Z = 1$$

$$V = 0$$

$$C = 0$$

Addressing Formats:

Addressing Modes, Execution Time, and Machine Code (hexadecimal/octal/decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
A	2	1	4F	117	079
B	2	1	5F	137	095
EXT	6	3	7F	177	127
IND	6	2	6F	157	111

Clear Two's Complement Overflow BitOperation: V bit \leftarrow 0

Description: Clears the two's complement overflow bit in the processor condition codes register.

Condition Codes: H: Not affected.
 I: Not affected.
 N: Not affected.
 Z: Not affected.
 V: Cleared.
 C: Not affected.

Boolean Formulae for Condition Codes:

$$V = 0$$

Addressing Modes, Execution Time, and Machine Code (hexadecimal/octal/decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
INHERENT	2	1	0A	012	010

Compare

Operation: (ACCX) – (M)

Description: Compares the contents of ACCX and the contents of M and determines the condition codes, which may be used subsequently for controlling conditional branching. Both operands are unaffected.

Condition Codes: H: Not affected.
 I: Not affected.
 N: Set if the most significant bit of the result of the subtraction would be set; cleared otherwise.
 Z: Set if all bits of the result of the subtraction would be cleared; cleared otherwise.
 V: Set if the subtraction would cause two's complement overflow; cleared otherwise.
 C: Carry is set if the absolute value of the contents of memory is larger than the absolute value of the accumulator; reset otherwise.

Boolean Formulae for Condition Codes:

$$N = R_7$$

$$Z = \bar{R}_7 \cdot \bar{R}_6 \cdot \bar{R}_5 \cdot \bar{R}_4 \cdot \bar{R}_3 \cdot \bar{R}_2 \cdot \bar{R}_1 \cdot \bar{R}_0$$

$$V = X_7 \cdot \bar{M}_7 \cdot \bar{R}_7 + \bar{X}_7 \cdot M_7 \cdot R_7$$

$$C = \bar{X}_7 \cdot M_7 + M_7 \cdot R_7 + R_7 \cdot \bar{X}_7$$

Addressing Formats:

Addressing Modes, Execution Time, and Machine Code (hexadecimal/ octal/ decimal):
 (DUAL OPERAND)

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
A IMM	2	2	81	201	129
A DIR	3	2	91	221	145
A EXT	4	3	B1	261	177
A IND	4	2	A1	241	161
B IMM	2	2	C1	301	193
B DIR	3	2	D1	321	209
B EXT	4	3	F1	361	241
B IND	4	2	E1	341	225

Complement

COM

Operation: $ACCX \leftarrow \sim(ACCX) = FF - (ACCX)$

or: $M \leftarrow \sim(M) = FF - (M)$

Description: Replaces the contents of ACCX or M with its one's complement. (Each bit of the contents of ACCX or M is replaced with the complement of that bit.)

Condition Codes: H: Not affected.

I: Not affected.

N: Set if most significant bit of the result is set; cleared otherwise.

Z: Set if all bits of the result are cleared; cleared otherwise.

V: Cleared.

C: Set.

Boolean Formulae for Condition Codes:

$$N = R_7$$

$$Z = \bar{R}_7 \cdot \bar{R}_6 \cdot \bar{R}_5 \cdot \bar{R}_4 \cdot \bar{R}_3 \cdot \bar{R}_2 \cdot \bar{R}_1 \cdot \bar{R}_0$$

$$V = 0$$

$$C = 1$$

Addressing Formats:

Addressing Modes, Execution Time, and Machine Code (hexadecimal/ octal/ decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
A	2	1	43	103	067
B	2	1	53	123	083
EXT	6	3	73	163	115
IND	6	2	63	143	099

Compare Index Register**CPX**

Operation: $(IXL) - (M+1)$
 $(IXH) - (M)$

Description: The more significant byte of the contents of the index register is compared with the contents of the byte of memory at the address specified by the program. The less significant byte of the contents of the index register is compared with the contents of the next byte of memory, at one plus the address specified by the program. The Z bit is set or reset according to the results of these comparisons, and may be used subsequently for conditional branching.

The N, V and C bits are also affected as indicated below and may be used subsequently for conditional branching.

- Condition Codes:**
- H: Not affected.
 - I: Not affected.
 - N: Set if the most significant bit of the result of the subtraction from the more significant byte of the index register would be set; cleared otherwise.
 - Z: Set if all bits of the results of both subtractions would be cleared; cleared otherwise.
 - V: Set if the subtraction from the more significant byte of the index register would cause two's complement overflow; cleared otherwise.
 - C: Set if the absolute value of the contents of memory are larger than the absolute value of the Index register; reset otherwise.

Boolean Formulae for Condition Codes:

$$\begin{aligned}N &= RH_7 \\Z &= (\overline{RH_7} \cdot \overline{RH_6} \cdot \overline{RH_5} \cdot \overline{RH_4} \cdot \overline{RH_3} \cdot \overline{RH_2} \cdot \overline{RH_1} \cdot \overline{RH_0}) \cdot \\&\quad (\overline{RL_7} \cdot \overline{RL_6} \cdot \overline{RL_5} \cdot \overline{RL_4} \cdot \overline{RL_3} \cdot \overline{RL_2} \cdot \overline{RL_1} \cdot \overline{RL_0}) \\V &= IXH_7 \cdot M_7 \cdot RH_7 + IXH_7 \cdot M_7 \cdot RH_7\end{aligned}$$

Addressing Formats:**Addressing Modes, Execution Time, and Machine Code (hexadecimal/octal/decimal):**

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
IMM	4	3	8C	214	140
DIR	5	2	9C	234	156
EXT	6	3	BC	274	188
IND	6	2	AC	254	172

Decimal Adjust ACCA**DAA**

Operation: Adds hexadecimal numbers 00, 06, 60, or 66 to ACCA, and may also set the carry bit, as indicated in the following table:

State of C-bit before DAA (Col. 1)	Upper Half-byte (bits 4-7) (Col. 2)	Initial Half-carry H-bit (Col.3)	Lower to ACCA (bits 0-3) (Col. 4)	Number Added after by DAA (Col. 5)	State of C-bit DAA (Col. 6)
0	0-9	0	0-9	00	0
0	0-8	0	A-F	06	0
0	0-9	1	0-3	06	0
0	A-F	0	0-9	60	1
0	9-F	0	A-F	66	1
0	A-F	1	0-3	66	1
1	0-2	0	0-9	60	1
1	0-2	0	A-F	66	1
1	0-3	1	0-3	66	1

Note: Columns (1) through (4) of the above table represent all possible cases which can result from any of the operations ABA, ADD, or ADC, with initial carry either set or clear, applied to two binary-coded-decimal operands. The table shows hexadecimal values.

Description: If the contents of ACCA and the state of the carry-borrow bit C and the half-carry bit H are all the result of applying any of the operations ABA, ADD, or ADC to binary-coded-decimal operands, with or without an initial carry, the DAA operation will function as follows.

Subject to the above condition, the DAA operation will adjust the contents of ACCA and the C bit to represent the correct binary-coded-decimal sum and the correct state of the carry.

Condition Codes:

- H: Not affected.
- I: Not affected.
- N: Set if most significant bit of the result is set; cleared otherwise.
- Z: Set if all bits of the result are cleared; cleared otherwise.
- V: Not defined.
- C: Set or reset according to the same rule as if the DAA and an immediately preceding ABA, ADD, or ADC were replaced by a hypothetical binary-coded-decimal addition.

Boolean Formulae for Condition Codes:

$$N = R_7$$

$$Z = \bar{R}_7 \cdot \bar{R}_6 \cdot \bar{R}_5 \cdot \bar{R}_4 \cdot \bar{R}_3 \cdot \bar{R}_2 \cdot \bar{R}_1 \cdot \bar{R}_0$$

C = See table above.

Addressing Modes, Execution Time, and Machine Code (hexadecimal/octal/decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
INHERENT	2	1	19	031	025

Decrement**DEC**

Operation: $ACCX \leftarrow (ACCX) - 01$
or: $M \leftarrow (M) - 01$

Description: Subtract one from the contents of ACCX or M.

The N, Z, and V condition codes are set or reset according to the results of this operation.

The C bit is not affected by the operation.

Condition Codes:

- H: Not affected.
- I: Not affected.
- N: Set if most significant bit of the result is set; cleared otherwise.
- Z: Set if all bits of the result are cleared; cleared otherwise.
- V: Set if there was two's complement overflow as a result of the operation; cleared otherwise. Two's complement overflow occurs if and only if $(ACCX)$ or (M) was 80 before the operation.
- C: Not affected.

Boolean Formulae for Condition Codes:

$$N = R_7$$

$$Z = \bar{R}_7 \cdot \bar{R}_6 \cdot \bar{R}_5 \cdot \bar{R}_4 \cdot \bar{R}_3 \cdot \bar{R}_2 \cdot \bar{R}_1 \cdot \bar{R}_0$$

$$V = X_7 \cdot \bar{X}_6 \cdot \bar{X}_5 \cdot \bar{X}_4 \cdot \bar{X}_3 \cdot \bar{X}_2 \cdot \bar{X}_0 = \bar{R}_7 \cdot R_6 \cdot R_5 \cdot R_4 \cdot R_3 \cdot R_2 \cdot R_1 \cdot R_0$$

Addressing Formats:**Addressing Modes, Execution Time, and Machine Code (hexadecimal/octal/decimal):**

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
A	2	1	4A	112	074
B	2	1	5A	132	090
EXT	6	3	7A	172	122
IND	6	2	6A	152	106

Decrement Stack Pointer**DES**

Operation: $SP \leftarrow (SP) - 0001$

Description: Subtract one from the stack pointer.

Condition Codes: Not affected.

Addressing Modes, Execution Time, and Machine Code (hexadecimal/octal/decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
INHERENT	3	1	34	064	052

Decrement Index RegisterOperation: $IX \leftarrow (IX) - 0001$

Description: Subtract one from the index register.

Only the Z bit is set or reset according to the result of this operation.

Condition Codes: H: Not affected.

I: Not affected.

N: Not affected.

Z: Set if all bits of the result are cleared; cleared otherwise.

V: Not affected.

C: Not affected.

Boolean Formulae for Condition Codes:

$$Z = (\overline{RH_7} \cdot \overline{RH_6} \cdot \overline{RH_5} \cdot \overline{RH_4} \cdot \overline{RH_3} \cdot \overline{RH_2} \cdot \overline{RH_1} \cdot \overline{RH_0}) \cdot \\ (\overline{RL_7} \cdot \overline{RL_6} \cdot \overline{RL_5} \cdot \overline{RL_4} \cdot \overline{RL_3} \cdot \overline{RL_2} \cdot \overline{RL_1} \cdot \overline{RL_0})$$

Addressing Modes, Execution Time, and Machine Code (hexadecimal/octal/decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
INHERENT	3	1	09	011	009

Exclusive OROperation: $ACCX \leftarrow (ACCX) \oplus (M)$

Description: Perform logical "EXCLUSIVE OR" between the contents of ACCX and the contents of M, and place the result in ACCX. (Each bit of ACCX after the operation will be the logical "EXCLUSIVE OR" of the corresponding bit of M and ACCX before the operation.)

Condition Codes: H: Not affected.

I: Not affected.

N: Set if most significant bit of the result is set; cleared otherwise.

Z: Set if all bits of the result are cleared; cleared otherwise.

V: Cleared

C: Not affected.

Boolean Formulae for Condition Codes:

$N = R_7$

$Z = \bar{R}_7 \cdot \bar{R}_6 \cdot \bar{R}_5 \cdot \bar{R}_4 \cdot \bar{R}_3 \cdot \bar{R}_2 \cdot \bar{R}_1 \cdot \bar{R}_0$

$V = 0$

Addressing Formats:

Addressing Modes, Execution Time, and Machine Code (hexadecimal/ octal/ decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
A IMM	2	2	88	210	136
A DIR	3	2	98	230	152
A EXT	4	3	B8	270	184
A IND	4	2	A8	250	168
B IMM	2	2	C8	310	200
B DIR	3	2	D8	330	216
B EXT	4	3	F8	370	248
B IND	4	2	E8	350	232

INC

Increment

Operation: $ACCX \leftarrow (ACCX) + 01$
or: $M \leftarrow (M) + 01$

Description: Add one to the contents of ACCX or M.

The N, Z, and V condition codes are set or reset according to the results of this operation.

The C bit is not affected by the operation.

Condition Codes: H: Not affected.
I: Not affected.
N: Set if most significant bit of the result is set; cleared otherwise.
Z: Set if all bits of the result are cleared; cleared otherwise.
V: Set if there was two's complement overflow as a result of the operation; cleared otherwise. Two's complement overflow will occur if and only if $(ACCX)$ or (M) was 7F before the operation.
C: Not affected.

Boolean Formulae for Condition Codes:

$$N = R_7$$

$$Z = \bar{R}_7 \cdot \bar{R}_6 \cdot \bar{R}_5 \cdot \bar{R}_4 \cdot \bar{R}_3 \cdot \bar{R}_2 \cdot \bar{R}_1 \cdot \bar{R}_0$$

$$V = \bar{X}_7 \cdot X_6 \cdot X_5 \cdot X_4 \cdot X_3 \cdot X_2 \cdot X_1 \cdot X_0$$

$$C = \bar{R}_7 \cdot \bar{R}_6 \cdot \bar{R}_5 \cdot \bar{R}_4 \cdot \bar{R}_3 \cdot \bar{R}_2 \cdot \bar{R}_1 \cdot \bar{R}_0$$

Addressing Formats:

Addressing Modes, Execution Time, and Machine Code (hexadecimal/ octal/ decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
A	2	1	4C	114	076
B	2	1	5C	134	092
EXT	6	3	7C	174	124
IND	6	2	6C	154	108

Increment Stack Pointer**INS**Operation: $SP \leftarrow (SP) + 0001$

Description: Add one to the stack pointer.

Condition Codes: Not affected.

Addressing Modes, Execution Time, and Machine Code (hexadecimal/ octal/ decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
INHERENT	3	1	31	061	049

Increment Index Register**INX**Operation: $IX \leftarrow (IX) + 0001$

Description: Add one to the index register.

Only the Z bit is set or reset according to the result of this operation.

Condition Codes: H: Not affected.

I: Not affected.

N: Not affected.

Z: Set if all 16 bits of the result are cleared; cleared otherwise.

V: Not affected.

C: Not affected.

Boolean Formulae for Condition Codes:

$$Z = (\overline{RH_7} \cdot \overline{RH_6} \cdot \overline{RH_5} \cdot \overline{RH_4} \cdot \overline{RH_3} \cdot \overline{RH_2} \cdot \overline{RH_1} \cdot \overline{RH_0}) \cdot \\ (\overline{RL_7} \cdot \overline{RL_6} \cdot \overline{RL_5} \cdot \overline{RL_4} \cdot \overline{RL_3} \cdot \overline{RL_2} \cdot \overline{RL_1} \cdot \overline{RL_0})$$

Addressing Modes, Execution Time, and Machine Code (hexadecimal/ octal/ decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
INHERENT	3	1	08	010	008

Jump**JMP**Operation: $PC \leftarrow \text{numerical address}$

Description: A jump occurs to the instruction stored at the numerical address. The numerical address is obtained according to the rules for EXTended or INDexed addressing.

Condition Codes: Not affected.

Addressing Formats:

Addressing Modes, Execution Time, and Machine Code (hexadecimal/ octal/ decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
EXT	3	3	7E	176	126
IND	3	2	6E	156	110

Jump to Subroutine

JSR

Operation:

Either: $PC \leftarrow (PC) + 0003$ (for EXTended addressing)
 or: $PC \leftarrow (PC) + 0002$ (for INDexed addressing)
 Then:
 $\downarrow (PCL)$
 $SP \leftarrow (SP) - 0001$
 $\downarrow (PCH)$
 $SP \leftarrow (SP) - 0001$
 $PC \leftarrow$ numerical address

Description: The program counter is incremented by 3 or by 2, depending on the addressing mode, and is then pushed onto the stack, eight bits at a time. The stack pointer points to the next empty location in the stack. A jump occurs to the instruction stored at the numerical address. The numerical address is obtained according to the rules for EXTended or INDexed addressing.

Condition Codes: Not affected.

Addressing Formats:

Addressing Modes, Execution Time, and Machine Code (hexadecimal/ octal/ decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
EXT	6	3	BD	275	189
IND	6	2	AD	255	173
DIR	5	2	9D	235	157

JUMP TO SUBROUTINE EXAMPLE (extended mode)

	Memory Location	Machine Code (Hex)	Label	Assembler Language Operator	Operand
A. Before:					
PC	→ \$0FFF	BD		JSR	CHARLI
	\$1000	20			
	\$1001	77			
SP	← \$EFFF				
B. After:					
PC	→ \$2077	..	CHARLI
SP	→ \$EFFD				
	\$EFFE	10			
	\$EFFF	02			

Load Accumulator D**LDD**Operation: $ACCD \leftarrow M:M+1$

Description: Loads the contents of the two consecutive memory locations $M:M+1$ into ACCD. The condition codes are set according to the data.

Condition Codes

- H: Not affected.
- I: Not affected.
- N: Set if most significant bit of the result is set; cleared otherwise.
- Z: Set if all bits of the result are cleared; cleared otherwise.
- V: Cleared.
- C: Not affected.

Addressing Formats:

Addressing Modes, Execution Time, and Machine Code (hexadecimal/octal/decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
D IMM	3	3	CC	314	204
D DIR	4	2	DC	334	220
D EXT	5	3	FC	374	252
D IND	5	2	EC	254	236

Load Accumulator**LDA**Operation: $ACCX \leftarrow (M)$

Description: Loads the contents of memory into the accumulator. The condition codes are set according to the data.

Condition Codes:

- H: Not affected.
- I: Not affected.
- N: Set if most significant bit of the result is set; cleared otherwise.
- Z: Set if all bits of the result are cleared; cleared otherwise.
- V: Cleared.
- C: Not affected.

Boolean Formulae for Condition Codes:

$$N = R_7$$

$$Z = \bar{R}_7 \cdot \bar{R}_6 \cdot \bar{R}_5 \cdot \bar{R}_4 \cdot \bar{R}_3 \cdot \bar{R}_2 \cdot \bar{R}_1 \cdot R_0$$

$$V = 0$$

Addressing Formats:

**Addressing Modes, Execution Time, and Machine Code (hexadecimal/ octal/ decimal):
(DUAL OPERAND)**

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
A IMM	2	2	86	206	134
A DIR	3	2	96	226	150
A EXT	4	3	B6	266	182
A IND	4	2	A6	246	166
B IMM	2	2	C6	306	198
B DIR	3	2	D6	326	214
B EXT	4	3	F6	366	246
B IND		2	E6	346	230

Load Stack Pointer

LDS

Operation: SPH \leftarrow (M)
SPL \leftarrow (M+1)

Description: Loads the more significant byte of the stack pointer from the byte of memory at the address specified by the program, and loads the less significant byte of the stack pointer from the next byte of memory, at one plus the address specified by the program.

Condition Codes: H: Not affected.
I: Not affected.
N: Set if the most significant bit of the stack pointer is set by the operation; cleared otherwise.
Z: Set if all bits of the stack pointer are cleared by the operation; cleared otherwise.
V: Cleared.
C: Not affected.

Boolean Formulae for Condition Codes:

$$\begin{aligned} N &= RH_7 \\ Z &= (\overline{RH}_7 \cdot \overline{RH}_6 \cdot \overline{RH}_5 \cdot \overline{RH}_4 \cdot \overline{RH}_3 \cdot \overline{RH}_2 \cdot \overline{RH}_1 \cdot \overline{RH}_0) \cdot \\ &\quad (\overline{AL}_7 \cdot \overline{AL}_6 \cdot \overline{AL}_5 \cdot \overline{AL}_4 \cdot \overline{AL}_3 \cdot \overline{AL}_2 \cdot \overline{AL}_1 \cdot \overline{AL}_0) \\ V &= 0 \end{aligned}$$

Addressing Formats:

Addressing Modes, Execution Time, and Machine Code (hexadecimal/ octal/ decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
IMM	3	3	8E	216	142
DIR	4	2	9E	236	158
EXT	5	3	BE	276	190
IND	5	2	AE	256	174

Operation: $0 \rightarrow \boxed{} \dots \boxed{} \rightarrow \boxed{C}$

Description: Shifts all bits of ACCD one place to the right.
Bit 15 is loaded with zero. The C bit is loaded from the least significant bit (b_0) of ACCD.

Condition Codes:

- H: Not affected.
- I: Not affected.
- N: Cleared.
- Z: Set if all bits of the result are cleared; cleared otherwise.
- V: Set if, after the completion of the shift operation, EITHER (N is set and C is cleared) OR (N is cleared and C is set); cleared otherwise.
- C: Set if, before the operation, the least significant bit of ACCD was set; cleared otherwise.

Addressing Modes, Execution Time, and Machine Code (hexadecimal/octal/decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
D Inherent	3	1	04	04	04

Load Index Register

LDX

Operation: $IXH \leftarrow (M)$
 $IXL \leftarrow (M+1)$

Description: Loads the more significant byte of the index register from the byte of memory at the address specified by the program, and loads the less significant byte of the index register from the next byte of memory, at one plus the address specified by the program.

Condition Codes:

- H: Not affected.
- I: Not affected.
- N: Set if the most significant bit of the index register is set by the operation; cleared otherwise.
- Z: Set if all bits of the index register are cleared by the operation; cleared otherwise.
- V: Cleared.
- C: Not affected.

Boolean Formulae for Condition Codes:

$$\begin{aligned} N &= RH_7 \\ Z &= (\overline{RH}_7 \cdot \overline{RH}_6 \cdot \overline{RH}_5 \cdot \overline{RH}_4 \cdot \overline{RH}_3 \cdot \overline{RH}_2 \cdot \overline{RH}_1 \cdot \overline{RH}_0) \cdot \\ &\quad (\overline{RL}_7 \cdot \overline{RL}_6 \cdot \overline{RL}_5 \cdot \overline{RL}_4 \cdot \overline{RL}_3 \cdot \overline{RL}_2 \cdot \overline{RL}_1 \cdot \overline{RL}_0) \\ V &= 0 \end{aligned}$$

Addressing Formats:

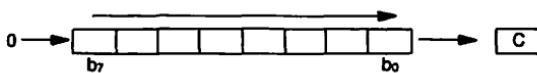
Addressing Modes, Execution Time, and Machine Code (hexadecimal/ octal/ decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
IMM	3	3	CE	316	206
DIR	4	2	DE	336	222
EXT	5	3	FE	376	254
IND	5	2	EE	356	238

Logical Shift Right:

LSR

Operation:



Description: Shifts all bits of ACCX or M one place to the right. Bit 7 is loaded with a zero. The C bit is loaded from the least significant bit of ACCX or M.

Condition Codes: H: Not affected.

I: Not affected.

N: Cleared.

Z: Set if all bits of the result are cleared; cleared otherwise.

V: Set if, after the completion of the shift operation, EITHER (N is set and C is cleared) OR (N is cleared and C is set); cleared otherwise.

C: Set if, before the operation, the least significant bit of the ACCX or M was set; cleared otherwise.

Boolean Formulae for Condition Codes:

$$N = 0$$

$$Z = \bar{R}_7 \cdot \bar{R}_6 \cdot \bar{R}_5 \cdot \bar{R}_4 \cdot \bar{R}_3 \cdot \bar{R}_2 \cdot \bar{R}_1 \cdot \bar{R}_0$$

$$V = N \oplus C = (N \cdot \bar{C}) \odot (\bar{N} \cdot C)$$

(the foregoing formula assumes values of N and C after the shift operation).

$$C = M_0$$

Addressing Formats:

Addressing Modes, Execution Time, and Machine Code (hexadecimal/ octal/ decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
A	2	1	44	104	068
B	2	1	54	124	084
EXT	6	3	74	164	116
IND	6	2	64	144	100

Multiply**MUL**Operation: $D \leftarrow A * B$

Description: Multiplies the eight bits of ACCA with the eight bits of ACCB to obtain a 16-bit unsigned number in ACCD.

Condition Codes:

- H: Not affected.
- I: Not affected.
- N: Not affected.
- Z: Not affected.
- V: Not affected.
- C: Set if bit 7 (b_7) of the result is set; cleared otherwise.

Addressing Modes, Execution Time, and Machine Code (hexadecimal/octal/decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
D Inherent	10	1	3D	075	61

Negate**NEG**Operation: $ACCX \leftarrow - (ACCX) = 00 - (ACCX)$
or: $M \leftarrow - (M) = 00 - (M)$

Description: Replaces the contents of ACCX or M with its two's complement. Note that 80 is left unchanged.

Condition Codes:

- H: Not affected.
- I: Not affected.
- N: Set if most significant bit of the result is set; cleared otherwise.
- Z: Set if all bits of the result are cleared; cleared otherwise.
- V: Set if there would be two's complement overflow as a result of the implied subtraction from zero; this will occur if and only if the contents of ACCX or M is 80.
- C: Set if there would be a borrow in the implied subtraction from zero; the C bit will be set in all cases except when the contents of ACCX or M is 00.

Boolean Formulae for Condition Codes:

$$\begin{aligned}N &= R_7 \\Z &= \bar{R}_7 \cdot \bar{R}_6 \cdot \bar{R}_5 \cdot \bar{R}_4 \cdot \bar{R}_3 \cdot \bar{R}_2 \cdot \bar{R}_1 \cdot \bar{R}_0 \\V &= R_7 \cdot \bar{R}_6 \cdot \bar{R}_5 \cdot \bar{R}_4 \cdot \bar{R}_3 \cdot \bar{R}_2 \cdot \bar{R}_1 \cdot \bar{R}_0 \\C &= R_7 + R_6 + R_5 + R_4 + R_3 + R_2 + R_1 + R_0\end{aligned}$$

Addressing Formats:**Addressing Modes, Execution Time, and Machine Code (hexadecimal/octal/ decimal):**

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
A	2	1	40	100	064
B	2	1	50	120	080
EXT	7	3	70	160	112
IND	6	2	60	140	096

No Operation**NOP**

Description: This is a single-word instruction which causes only the program counter to be incremented. No other registers are affected.

Condition Codes: Not affected.

Addressing Modes, Execution Time, and Machine Code (hexadecimal/octal/ decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
INHERENT	2	1	01	001	001

Inclusive OR**ORA**

Operation: $ACCX \leftarrow (ACCX) \odot (M)$

Description: Perform logical "OR" between the contents of ACCX and the contents of M and places the result in ACCX. (Each bit of ACCX after the operation will be the logical "OR" of the corresponding bits of M and of ACCX before the operation).

Condition Codes: H: Not affected.
I: Not affected.
N: Set if most significant bit of the result is set; cleared otherwise.
Z: Set if all bits of the result are cleared; cleared otherwise.
V: Cleared.
C: Not affected.

Boolean Formulae for Condition Codes:

$$N = R_7$$

$$Z = \bar{R}_7 \cdot \bar{R}_6 \cdot \bar{R}_5 \cdot \bar{R}_4 \cdot \bar{R}_3 \cdot \bar{R}_2 \cdot \bar{R}_1 \cdot \bar{R}_0$$

$$V = 0$$

Addressing Formats:

See Table A-1.

**Addressing Modes, Execution Time, and Machine Code (hexadecimal/octal/decimal):
(DUAL OPERAND)**

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
A IMM	2	2	8A	212	138
A DIR	3	2	9A	232	154
A EXT	4	3	BA	272	186
A IND	4	2	AA	252	170
B IMM	2	2	CA	312	202
B DIR	3	2	DA	332	218
B EXT	4	3	FA	372	250
B IND	4	2	EA	352	234

Push Data Onto Stack

PSH

Operation: $\downarrow (ACCX)$
 $SP \leftarrow (SP) - 0001$

Description: The contents of ACCX is stored in the stack at the address contained in the stack pointer. The stack pointer is then decremented.

Condition Codes: Not affected.

Addressing Formats:

Addressing Modes, Execution Time, and Machine Code (hexadecimal/octal/decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
A	3	1	36	066	054
B	3	1	37	067	055

Push Index Register to Stack

PSHX

Operation: $\downarrow X$
 $SP \leftarrow SP - 2$

Description: The contents of the index register are pushed onto the stack at the address contained in the stack pointer. The stack pointer is decremented by two.

Condition Codes: Not affected.

Addressing Modes, Execution Time, and Machine Code (hexadecimal/octal/decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
Inherent	4	1	3C	074	060

Pull Index Register from Stack

PULX

Operation: $SP \leftarrow SP + 2$
 $\uparrow X$

Description: The Index register is pulled from the stack beginning at the current address contained in the stack pointer + 1. The stack pointer is incremented by two in total.

Condition Codes: Not affected.

Addressing Modes, Execution Time, and Machine Code (hexadecimal/octal/decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
Inherent	5	1	38	070	056

Pull Data from Stack

PUL

Operation: $SP \leftarrow (SP) + 0001$
 $\uparrow ACCX$

Description: The stack pointer is incremented. The ACCX is then loaded from the stack, from the address which is contained in the stack pointer.

Condition Codes: Not affected.

Addressing Formats:

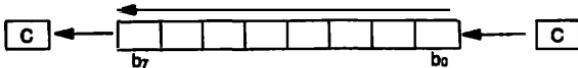
Addressing Modes, Execution Time, and Machine Code (hexadecimal/octal/decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
A	4	1	32	062	050
B	4	1	33	063	051

Rotate Left

ROL

Operation:



Description: Shifts all bits of ACCX or M one place to the left. Bit 0 is loaded from the C bit. The C bit is loaded from the most significant bit of ACCX or M.

Condition Codes: H: Not affected.

I: Not affected.

N: Set if most significant bit of the result is set; cleared otherwise.

Z: Set if all bits of the result are cleared; cleared otherwise.

V: Set if, after the completion of the operation, EITHER (N is set and C is cleared) OR (N is cleared and C is set); cleared otherwise.

C: Set if, before the operation, the most significant bit of the ACCX or M was set; cleared otherwise.

Boolean Formulae for Condition Codes:

$$N = R_7$$

$$Z = \bar{R}_7 \cdot \bar{R}_8 \cdot \bar{R}_9 \cdot \bar{R}_{10} \cdot \bar{R}_3 \cdot \bar{R}_2 \cdot \bar{R}_1 \cdot \bar{R}_0$$

$$V = N \oplus C = [N \cdot \bar{C}] \odot [\bar{N} \cdot C]$$

(the foregoing formula assumes values of N and C after the rotation)

$$C = M_7$$

Addressing Formats:

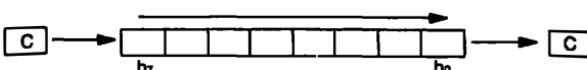
Addressing Modes, Execution Time, and Machine Code (hexadecimal/ octal/ decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
A	2	1	49	111	073
B	2	1	59	131	089
EXT 6	6	3	79	171	121
IND	6	2	69	151	105

Rotate Right

ROR

Operation:



Description: Shifts all bits of ACCX or M one place to the right. Bit 7 is loaded from the C bit. The C bit is loaded from the least significant bit of ACCX or M.

Condition Codes: H: Not affected.

I: Not affected.

N: Set if most significant bit of the result is set; cleared otherwise.

Z: Set if all bits of the result are cleared; cleared otherwise.

V: Set if, after the completion of the operation, EITHER (N is set and C is cleared) OR (N is cleared and C is set); cleared otherwise.

C: Set if, before the operation, the least significant bit of the ACCX or M was set; cleared otherwise.

Boolean Formulae for Condition Codes:

$$N = R_7$$

$$Z = \bar{R}_7 \cdot \bar{R}_6 \cdot \bar{R}_5 \cdot \bar{R}_4 \cdot \bar{R}_3 \cdot \bar{R}_2 \cdot \bar{R}_1 \cdot \bar{R}_0$$

$$V = N \oplus C = (N \cdot \bar{C}) \odot (\bar{N} \cdot C)$$

(the foregoing formula assumes values of N and C after the rotation)

$$C = M_0$$

Addressing Formats:

Addressing Modes, Execution Time, and Machine Code (hexadecimal/octal/decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
A	2	1	46	106	070
B	2	1	56	126	086
EXT	6	3	76	166	118
IND		2	66	146	102

Return from Interrupt

RTI

Operation:

- SP \leftarrow (SP) + 0001 , \uparrow CC
- SP \leftarrow (SP) + 0001 , \uparrow ACCB
- SP \leftarrow (SP) + 0001 , \uparrow ACCA
- SP \leftarrow (SP) + 0001 , \uparrow IHX
- SP \leftarrow (SP) + 0001 , \uparrow IXL
- SP \leftarrow (SP) + 0001 , \uparrow PCH
- SP \leftarrow (SP) + 0001 , \uparrow PCL

Description: The condition codes, accumulators B and A, the index register, and the program counter, will be restored to a state pulled from the stack. Note that the interrupt mask bit will be reset if and only if the corresponding bit stored in the stack is zero.

Condition Codes: Restored to the states pulled from the stack.

Addressing Modes, Execution Time, and Machine Code (hexadecimal/octal/decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
INHERENT	10	1	3B	073	059

Return from Interrupt

Example

		Memory Location	Machine Code (Hex)	Label	Assembler Language Operator	Operand
A.	Before					
	PC	→ \$D066	3B		RTI	
	SP	→ \$EFF8 \$EFF9 \$EFFA \$EFFB \$EFFC \$EFFD \$EFFE \$EFFF	11HINZVC 12 34 56 78 55 67	(binary)		
B.	After				...	****
	PC	→ \$5567	****
		\$EFF8 \$EFF9 \$EFFA \$EFFB \$EFFC \$EFFD \$EFFE	11HINZVC 12 34 56 78 55	(binary)		
	SP	→ \$EFFF	67			
CC = HINZVC (binary)						
ACCB = 12 (Hex)		IXH = 56 (Hex)				
ACCA = 34 (Hex)		IXL = 78 (Hex)				

Return from Subroutine

RTS

Operation: $SP \leftarrow (SP) + 0001$
 $\uparrow PCH$
 $SP \leftarrow (SP) + 0001$
 $\uparrow PCL$

Description: The stack pointer is incremented (by 1). The contents of the byte of memory, at the address now contained in the stack pointer, are loaded into the 8 bits of highest significance in the program counter. The stack pointer is again incremented (by 1). The contents of the byte of memory, at the address now contained in the stack pointer, are loaded into the 8 bits of lowest significance in the program counter.

Condition Codes: Not affected.

Addressing Modes, Execution Time, and Machine Code (hexadecimal/octal/decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
INHERENT	5	1	39	071	057

Return from Subroutine

EXAMPLE

	Memory Location	Machine Code (Hex)	Label	Assembler Language Operator	Operand
A. Before					
PC	\$30A2	39		RTS	
SP	\$EFFF				
	\$EFFD				
	\$EFFE	10			
	\$EFFF	02			
B. After					
PC	\$1002
	\$EFFD				
	\$EFFE	10			
SP	\$EFFF	02			

Subtract Accumulators

SBA

Operation: $ACCA \leftarrow (ACCA) - (ACCB)$

Description: Subtracts the contents of ACCB from the contents of ACCA and places the result in ACCA. The contents of ACCB are not affected.

Condition Codes: H: Not affected.

t: Not affected.

N: Set if most significant bit of the result is set; cleared otherwise.

Z: Set if all bits of the result are cleared; cleared otherwise.

V: Set if there was two's complement overflow as a result of the operation.

C: Carry is set if the absolute value of accumulator B plus previous carry is larger than the absolute value of accumulator A; reset otherwise.

Boolean Formulae for Condition Codes:

$$N = R_7$$

$$Z = \bar{R}_7 \cdot \bar{R}_6 \cdot \bar{R}_5 \cdot \bar{R}_4 \cdot \bar{R}_3 \cdot \bar{R}_2 \cdot \bar{R}_1 \cdot \bar{R}_0$$

$$V = A_7 \cdot \bar{B}_7 \cdot \bar{R}_7 + \bar{A}_7 \cdot B_7 \cdot R_7$$

$$C = \bar{A}_7 \cdot B_7 + B_7 \cdot R_7 + R_7 \cdot \bar{A}_7$$

Addressing Modes, Execution Time, and Machine Code (hexadecimal/octal/decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
INHERENT	2	1	10	020	016

Subtract with Carry

SBC

Operation: $ACCX \leftarrow (ACCX) - (M) - (C)$

Description: Subtracts the contents of M and C from the contents of ACCX and places the result in ACCX.

- Condition Codes:**
- H: Not affected.
 - I: Not affected.
 - N: Set if most significant bit of the result is set; cleared otherwise.
 - Z: Set if all bits of the result are cleared; cleared otherwise.
 - V: Set if there was two's complement overflow as a result of the operation; cleared otherwise.
 - C: Carry is set if the absolute value of the contents of memory plus previous carry is larger than the absolute value of the accumulator; reset otherwise.

Boolean Formulae for Condition Codes:

$$N = R_7$$

$$Z = \bar{R}_7 \cdot \bar{R}_6 \cdot \bar{R}_5 \cdot \bar{R}_4 \cdot \bar{R}_3 \cdot \bar{R}_2 \cdot \bar{R}_1 \cdot \bar{R}_0$$

$$V = X_7 \cdot \bar{M}_7 \cdot \bar{R}_7 + \bar{X}_7 \cdot M_7 \cdot R_7$$

$$C = \bar{X}_7 \cdot M_7 + M_7 \cdot R_7 + R_7 \cdot \bar{X}_7$$

Addressing Formats:

Addressing Modes, Execution Time, and Machine Code (hexadecimal/octal/decimal):

(DUAL OPERAND)

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
A IMM	2	2	B2	202	130
A DIR	3	2	92	222	146
A EXT	4	3	B2	262	178
A IND	4	2	A2	242	162
B IMM	2	2	C2	302	194
B DIR	3	2	D2	322	210
B EXT	4	3	F2	362	242
B IND	4	2	E2	342	226

Set Carry

SEC

Operation: C bit $\leftarrow 1$

Description: Sets the carry bit in the processor condition codes register.

- Condition Codes:**
- H: Not affected.
 - I: Not affected.
 - N: Not affected.
 - Z: Not affected.
 - V: Not affected.
 - C: Set.

Boolean Formulae for Condition Codes:

$$C = 1$$

Addressing Modes, Execution Time, and Machine Code (hexadecimal/octal/decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
INHERENT	2	1	0D	015	13

Set Interrupt Mask**SEI**Operation: I bit \leftarrow 1

Description: Sets the interrupt mask bit in the processor condition codes register. The microprocessor is inhibited from servicing an interrupt from a peripheral device, and will continue with execution of the instructions of the program, until the interrupt mask bit has been cleared.

Condition Codes: H: Not affected.

I: Set.

N: Not affected.

Z: Not affected.

V: Not affected.

C: Not affected.

Boolean Formulae for Condition Codes:

$$I = 1$$

Addressing Modes, Execution Time, and Machine Code (hexadecimal/octal/decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
INHERENT	2	1	0F	017	015

Set Two's Complement Overflow Bit**SEV**Operation: V bit \leftarrow 1

Description: Sets the two's complement overflow bit in the processor condition codes register.

Condition Codes: H: Not affected.

I: Not affected.

N: Not affected.

Z: Not affected.

V: Set.

C: Not affected.

Boolean Formulae for Condition Codes:

$$V = 1$$

Addressing Modes, Execution Time, and Machine Code (hexadecimal/octal/decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
INHERENT	2	1	0B	013	011

Store Accumulator**STA**Operation: M \leftarrow (ACCX)

Description: Stores the contents of ACCX in memory. The contents of ACCX remains unchanged.

- Condition Codes:**
- H: Not affected.
 - I: Not affected.
 - N: Set if the most significant bit of the contents of ACCX is set; cleared otherwise.
 - Z: Set if all bits of the contents of ACCX are cleared; cleared otherwise.
 - V: Cleared.
 - C: Not affected.

Boolean Formulae for Condition Codes:

$$\begin{aligned}N &= X_7 \\Z &= \bar{X}_7 \cdot \bar{X}_6 \cdot \bar{X}_5 \cdot \bar{X}_4 \cdot \bar{X}_3 \cdot \bar{X}_2 \cdot \bar{X}_1 \cdot \bar{X}_0 \\V &= 0\end{aligned}$$

Addressing Formats:

Addressing Modes, Execution Time, and Machine Code (hexadecimal/ octal/ decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
A DIR	3	2	97	227	151
A EXT	4	3	B7	267	183
A IND	4	2	A7	247	167
B DIR	3	2	D7	327	215
B EXT	4	3	F7	367	247
B IND	4	2	E7	347	231

Store Stack Pointer

STS

Operation: $M \leftarrow (\text{SPH})$
 $M + 1 \leftarrow (\text{SPL})$

Description: Stores the more significant byte of the stack pointer in memory at the address specified by the program, and stores the less significant byte of the stack pointer at the next location in memory, at one plus the address specified by the program.

- Condition Codes:**
- H: Not affected.
 - I: Not affected.
 - N: Set if the most significant bit of the stack pointer is set; cleared otherwise.
 - Z: Set if all bits of the stack pointer are cleared; cleared otherwise.
 - V: Cleared.
 - C: Not affected.

Boolean Formulae for Condition Codes:

$$\begin{aligned}N &= \overline{\text{SPH}_7} \\Z &= (\overline{\text{SPH}_7} \cdot \overline{\text{SPH}_6} \cdot \overline{\text{SPH}_5} \cdot \overline{\text{SPH}_4} \cdot \overline{\text{SPH}_3} \cdot \overline{\text{SPH}_2} \cdot \overline{\text{SPH}_1} \cdot \overline{\text{SPH}_0}) \\&\quad (\overline{\text{SPL}_7} \cdot \overline{\text{SPL}_6} \cdot \overline{\text{SPL}_5} \cdot \overline{\text{SPL}_4} \cdot \overline{\text{SPL}_3} \cdot \overline{\text{SPL}_2} \cdot \overline{\text{SPL}_1} \cdot \overline{\text{SPL}_0}) \\V &= 0\end{aligned}$$

Addressing Formats:**Addressing Modes, Execution Time, and Machine Code (hexadecimal/octal/decimal):**

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
DIR	4	2	9F	237	159
EXT	5	3	BF	277	191
IND	5	2	AF	257	175

Store Accumulator D**STD****Operation:** M:M + 1 ← (ACCD)**Description:** Store the contents of ACCD in memory. The contents of ACCD remain unchanged.

Condition Codes:

- H: Not affected.
- I: Not affected.
- N: Set if the most significant bit of ACCD is set; cleared otherwise.
- Z: Set if all bits of ACCD are cleared; cleared otherwise.
- V: Cleared.
- C: Not affected.

Addressing Formats:**Addressing Modes, Execution Time, and Machine Code (hexadecimal/octal/decimal):**

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
D DIR	4	2	DD	335	221
D EXT	5	3	FD	375	253
D IND	5	2	ED	355	237

Store Index Register**STX****Operation:** M ← (IXH)
M + 1 ← (IXL)**Description:** Stores the more significant byte of the index register in memory at the address specified by the program, and stores the less significant byte of the index register at the next location in memory, at one plus the address specified by the program.

- Condition Codes:**
- H: Not affected.
 - I: Not affected.
 - N: Set if the most significant bit of the index register is set; cleared otherwise.
 - Z: Set if all bits of the index register are cleared; cleared otherwise.
 - V: Cleared.
 - C: Not affected.

Boolean Formulae for Condition Codes:

$$\begin{aligned} N &= IXH_7 \\ Z &= (IXH_7 \cdot \overline{IXH}_6 \cdot \overline{IXH}_5 \cdot \overline{IXH}_4 \cdot \overline{IXH}_3 \cdot \overline{IXH}_2 \cdot \overline{IXH}_1 \cdot \overline{IXH}_0) \\ &\quad (\overline{IXL}_7 \cdot IXL_6 \cdot \overline{IXL}_5 \cdot IXL_4 \cdot \overline{IXL}_3 \cdot IXL_2 \cdot \overline{IXL}_1 \cdot IXL_0) \\ V &= 0 \end{aligned}$$

Addressing Formats:

Addressing Modes, Execution Time, and Machine Code (hexadecimal/ octal/ decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
DIR	4	2	DF	337	223
EXT	5	3	FF	377	255
IND	5	2	EF	357	239

Subtract

SUB

Operation: $ACCX \leftarrow (ACCX) - (M)$

Description: Subtracts the contents of M from the contents of ACCX and places the result in ACCX.

- Condition Codes:**
- H: Not affected.
 - I: Not affected.
 - N: Set if most significant bit of the result is set; cleared otherwise.
 - Z: Set if all bits of the result are cleared; cleared otherwise.
 - V: Set if there was two's complement overflow as a result of the operation; cleared otherwise.
 - C: Set if the absolute value of the contents of memory are larger than the absolute value of the accumulator; reset otherwise.

Boolean Formulae for Condition Codes:

$$\begin{aligned} N &= R_7 \\ Z &= \overline{R}_7 \cdot \overline{R}_6 \cdot \overline{R}_5 \cdot \overline{R}_4 \cdot \overline{R}_3 \cdot \overline{R}_2 \cdot \overline{R}_1 \cdot \overline{R}_0 \\ V &= X_7 \cdot \overline{M}_7 \cdot \overline{R}_7 \cdot \overline{X}_7 \cdot M_7 \cdot R_7 \\ C &= \overline{X}_7 \cdot M_7 + M_7 \cdot R_7 + R_7 \cdot \overline{X}_7 \end{aligned}$$

Addressing Formats:

**Addressing Modes, Execution Time, and Machine Code (hexadecimal/octal/decimal):
(DUAL OPERAND)**

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
A IMM	2	2	80	200	128
A DIR	3	2	90	220	144
A EXT	4	3	B0	260	176
A IND	4	2	A0	240	160
B IMM	2	2	C0	300	192
B DIR	3	2	D0	320	208
B EXT	4	3	F0	360	240
B IND	4	2	E0	340	224

Subtract Accumulator D**SUBD**

Operation: $(ACCD) \leftarrow (ACCD) - (M:M + 1)$

Description: Subtracts the contents of M:M + 1 from the contents of ACCD and places the result in ACCD.

Condition Codes:

- H: Not affected.
- I: Not affected.
- N: Set if most significant bit of the result is set; cleared otherwise.
- Z: Set if all bits of the result are cleared; cleared otherwise.
- V: Set if there was two's complement overflow as a result of the operation; cleared otherwise.
- C: Set if the absolute value of the contents of memory are larger than the absolute value of the accumulator; reset otherwise.

Addressing Formats:

Addressing Modes, Execution Time, and Machine Code (hexadecimal/octal/decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
D IMM	4	3	83	203	131
D DIR	5	2	93	223	147
D EXT	6	3	B3	263	179
D IND	6	2	A3	243	163

Software Interrupt

Operation:

```

PC ← (PC) + 0001
↓ (PCL), SP ← (SP)-0001
↓ (PCH), SP ← (SP)-0001
↓ (IXL), SP ← (SP)-0001
↓ (IXH), SP ← (SP)-0001
↓ (ACCA), SP ← (SP)-0001
↓ (ACCB), SP ← (SP)-0001
↓ (CC), SP ← (SP)-0001
I ← 1
PCH ← (n-0005)
PCL ← (n-0004)

```

Description: The program counter is incremented (by 1). The program counter, index register, and accumulator A and B, are pushed into the stack. The condition codes register is then pushed into the stack, with condition codes H, I, N, Z, V, C going respectively into bit positions 5 thru 0, and the top two bits (in bit positions 7 and 6) are set (to the 1 state). The stack pointer is decremented (by 1) after each byte of data is stored in the stack.

The interrupt mask bit is then set. The program counter is then loaded with the address stored in the software interrupt pointer at memory locations (n-5) and (n-4), where n is the address corresponding to a high state on all lines of the address bus.

Condition Codes:

- H: Not affected.
- I: Set.
- N: Not affected.
- Z: Not affected.
- V: Not affected.
- C: Not affected.

Boolean Formula for Condition Codes:

$$I = 1$$

Addressing Modes, Execution Time, and Machine Code (hexadecimal/octal/decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
INHERENT	12	1	3F	077	63

Software Interrupt

EXAMPLE

A. Before:

CC = HINZVC (binary)
 ACCB = 12 (Hex) IXH = 56 (Hex)
 ACCA = 34 (Hex) IXL = 78 (Hex)

	Memory Location	Machine Code (Hex)	Label	Assembler Language Operator	Operand
PC	→ \$5566	3F			SWI
SP	→ \$EFFF				
	\$FFFA	D0			
	\$FFFFB	55			

B. After:

PC	→ \$D055				
SP	→ \$EFF8				
	\$EFF9	11HINZVC	(binary)		
	\$EFFA	12			
	\$EFFB	34			
	\$EFFC	56			
	\$EFFD	78			
	\$EFFE	55			
	\$EFFF	67			

Note: This example assumes that FFFF is the memory location addressed when all lines of the address bus go to the high state.

TAB

Transfer from Accumulator A to Accumulator B

Operation: ACCB ← (ACCA)

Description: Moves the contents of ACCA to ACCB. The former contents of ACCB are lost.
 The contents of ACCA are not affected.

Condition Codes: H: Not affected.
 I: Not affected.
 N: Set if the most significant bit of the contents of the accumulator is set; cleared otherwise.
 Z: Set if all bits of the contents of the accumulator are cleared; cleared otherwise.
 V: Cleared.
 C: Not affected.

Boolean Formulae for Condition Codes:

$$\begin{aligned} N &= R_7 \\ Z &= \bar{R}_7 \cdot \bar{R}_6 \cdot \bar{R}_5 \cdot \bar{R}_4 \cdot \bar{R}_3 \cdot \bar{R}_2 \cdot \bar{R}_1 \cdot \bar{R}_0 \\ V &= 0 \end{aligned}$$

Addressing Modes, Execution Time, and Machine Code (hexadecimal/octal/decimal):

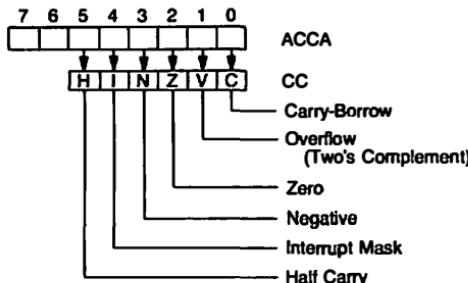
Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
INHERENT	2	1	16	026	022

**Transfer from Accumulator A
to Processor Condition Codes Register**

TAP

Operation: $CC \leftarrow (ACCA)$

Bit Positions



Description: Transfers the contents of bit positions 0 thru 5 of accumulator A to the corresponding bit positions of the processor condition codes register. The contents of accumulator A remain unchanged.

Condition Codes: Set or reset according to the contents of the respective bits 0 thru 5 of accumulator A.

Addressing Modes, Execution Time, and Machine Code (hexadecimal/octal/decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
INHERENT	2	1	06	006	006

Transfer from Accumulator B to Accumulator A

TBA

Operation: $ACCA \leftarrow (ACCB)$

Description: Moves the contents of ACCB to ACCA. The former contents of ACCA are lost. The contents of ACCB are not affected.

Condition Codes: H: Not affected.
I: Not affected.
N: Set if the most significant accumulator bit is set; cleared otherwise.
Z: Set if all accumulator bits are cleared; cleared otherwise.
V: Cleared.
C: Not affected.

Boolean Formulae for Condition Codes:

$$\begin{aligned} N &= R_7 \\ Z &= \bar{R}_7 \cdot \bar{R}_6 \cdot \bar{R}_5 \cdot \bar{R}_4 \cdot \bar{R}_3 \cdot \bar{R}_2 \cdot \bar{R}_1 \cdot \bar{R}_0 \\ V &= 0 \end{aligned}$$

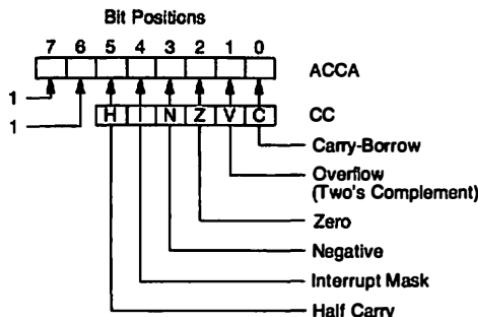
Addressing Modes, Execution Time, and Machine Code (hexadecimal/octal/decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
INHERENT	2	1	17	027	023

**Transfer from Processor Condition Codes Register to
Accumulator A**

TPA

Operation: $\text{ACCA} \leftarrow (\text{CC})$



Description: Transfers the contents of the processor condition codes register to corresponding bit positions 0 thru 5 of accumulator A. Bit positions 6 and 7 of accumulator A are set (i.e. go to the "1" state). The processor condition codes register remains unchanged.

Condition Codes: Not affected.

Addressing Modes, Execution Time, and Machine Code (hexadecimal/ octal/ decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
INHERENT	2	1	07	007	007

Test

TST

Operation: $(\text{ACCX}) - 00$
 $(\text{M}) - 00$

Description: Set condition codes N and Z according to the contents of ACCX or M.

Condition Codes:

- H: Not affected.
- I: Not affected.
- N: Set if most significant bit of the contents of ACCX or M is set; cleared otherwise.
- Z: Set if all bits of the contents of ACCX or M are cleared; cleared otherwise.
- V: Cleared.
- C: Cleared.

Boolean Formulae for Condition Codes:

$$\begin{aligned} N &= M_7 \\ Z &= \bar{M}_7 \cdot \bar{M}_6 \cdot \bar{M}_5 \cdot \bar{M}_4 \cdot \bar{M}_3 \cdot \bar{M}_2 \cdot \bar{M}_1 \cdot \bar{M}_0 \\ V &= 0 \\ C &= 0 \end{aligned}$$

Addressing Formats:

Addressing Modes, Execution Time, and Machine Code (hexadecimal/ octal/ decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
A	2	1	4D	115	077
B	2	1	5D	135	093
EXT	6	3	7D	175	125
IND	6	2	6D	155	109

Transfer from Stack Pointer to Index Register

TSX

Operation: IX \leftarrow (SP) + 0001

Description: Loads the index register with one plus the contents of the stack pointer. The contents of the stack pointer remain unchanged.

Condition Codes: Not affected.

Addressing Modes, Execution Time, and Machine Code (hexadecimal/ octal/ decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
INHERENT	3	1	30	060	048

Transfer From Index Register to Stack Pointer

TXS

Operation: SP \leftarrow (IX) - 0001

Description: Loads the stack pointer with the contents of the index register, minus one. The contents of the index register remain unchanged.

Condition Codes: Not affected.

Addressing Modes, Execution Time, and Machine Code (hexadecimal/ octal/ decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
INHERENT	3	1	35	065	053

Wait for Interrupt**WAI**

Operation: $PC \leftarrow (PC) + 0001$
 $\downarrow (PCL), SP \leftarrow (SP)-0001$
 $\downarrow (PCH), SP \leftarrow (SP)-0001$
 $\downarrow (IXL), SP \leftarrow (SP)-0001$
 $\downarrow (IXH), SP \leftarrow (SP)-0001$
 $\downarrow (ACCA), SP \leftarrow (SP)-0001$
 $\downarrow (ACCB), SP \leftarrow (SP)-0001$
 $\downarrow (CC), SP \leftarrow (SP)-0001$

Condition Codes: Not affected.

Description: The program counter is incremented (by 1). The program counter, index register, and accumulators A and B, are pushed into the stack. The condition codes register is then pushed into the stack, with condition codes H, I, N, Z, V, C going respectively into bit positions 5 thru 0, and the top two bits (in bit positions 7 and 6) are set (to the 1 state). The stack pointer is decremented (by 1) after each byte of data is stored in the stack.

Execution of the program is then suspended until an interrupt from a peripheral device is signalled, by the interrupt request control input going to a low state.

When an interrupt is signalled on the interrupt request line, and provided the I bit is clear, execution proceeds as follows. The interrupt mask bit is set. The program counter is then loaded with the address stored in the internal interrupt pointer at memory locations (n-7) and (n-6), where n is the address corresponding to a high state on all lines of the address bus.

Condition Codes: H: Not affected.
I: Not affected until an interrupt request signal is detected on the interrupt request control line. When the interrupt request is received the I bit is set and further execution takes place, provided the I bit was initially clear.
N: Not affected.
Z: Not affected.
V: Not affected.
C: Not affected.

Addressing Modes, Execution Time, and Machine Code (hexadecimal/ octal/ decimal):

Addressing Modes	Execution Time (No. of cycles)	Number of bytes of machine code	Coding of First (or only) byte of machine code		
			HEX.	OCT.	DEC.
INHERENT	9	1	3E	076	62

APPENDIX B

Specification Sheets

MC6801

ICL7109

MC6801L1/MC6801P1

NE5018

MC6803/MC6803NR

MC14066B

MC1508L-8

MC14584B

MC6801/03 Port Expansion Application Note



MOTOROLA

SEMICONDUCTORS

3501 E. 6TH BLUETEEN BLVD., AUSTIN, TEXAS 78721

Advance Information

MICROCOMPUTER UNIT (MCU)

The MC6801 MCU is an 8-bit microcomputer system which is compatible with the MC6800 family of parts. The MC6801 MCU is object code compatible with the MC6800 with improved execution times of key instructions plus several new 16-bit and 8-bit instructions including an 8x8 unsigned multiply with 16-bit result. The MC6801 MCU can operate as a single chip microcomputer or be expanded to 65K words. The MC6801 MCU is TTL compatible and requires one +5.0 volt power supply. The MC6801 MCU has 2K bytes of ROM and 128 bytes of RAM onchip, Serial Communications Interface (S.C.I.), and parallel I/O as well as a three function 16-bit timer. Block diagram is shown in Figure 1. Features of the MC6801 include the following:

- Expanded MC6800 Instruction Set
- 8 X 8 Multiply
- On-Chip Serial Communications Interface (S.C.I.)
- Object Code Compatible With The MC6800 MPU
- 16-Bit Timer
- Single Chip Or Expandable To 65K Words
- 2K Bytes Of ROM
- 128 Bytes Of RAM (64 Bytes Retainable On Power Down)
- 31 Parallel I/O Lines
- Internal Clock/Divided-By-Four
- TTL Compatible Inputs And Outputs
- Interrupt Capability
- External Clock/Divide-By-One Mask Option (MC6801E) And EPROM Versions MC68701 And MC68701E Available Soon.

MC6801

MOS

(IN-CHANNEL, SILICON-GATE DEPLETION LOAD)

MICROCOMPUTER

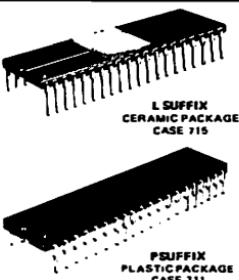


FIGURE 1 - SINGLE-CHIP MICROCOMPUTER BLOCK DIAGRAM

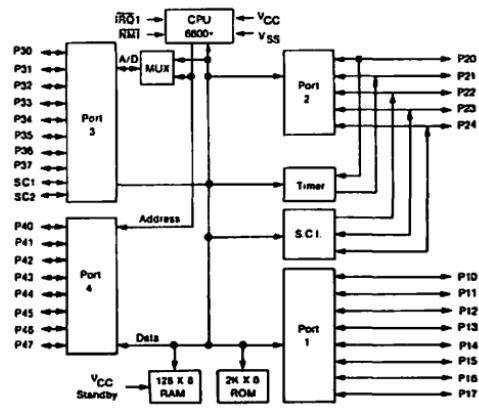


FIGURE 2 -- PIN ASSIGNMENT

VSS	1	40	E
KTAL 1	2	39	SC1
EXTAL 2	3	38	SC2
REG1	4	37	P20
REG2	5	36	P21
Reset	6	35	P22
Vcc	7	34	P23
P20	8	33	P24
P21	9	32	P25
P22	10	31	P26
P23	11	30	P27
P24	12	29	P28
P10	13	28	P41
P11	14	27	P42
P12	15	26	P43
P13	16	25	P44
P14	17	24	P45
P15	18	23	P46
P16	19	22	P47
P17	20	21	VCC Standby

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

©Motorola Inc., 1978

AD-603R1

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

Characteristic	Symbol	Min	Typ	Max	Unit
Input High Voltage Reset	V _{HI}	V _{SS} + 2.0 V _{SS} + 4.0	-	V _{CC} V _{SS}	Vdc
Input Low Voltage	V _{LO}	V _{SS} - 0.3	-	V _{SS} - 0.8	Vdc
Three-State (Off State) Input Current P10-P17 (V _{IN} = 0.4 to 2.4 Vdc) P20-P24, P30-P37	I _{IN}	-	2.0	10	μA/dc
Output High Voltage All Outputs Except XTAL 1 and EXTAL 2 (I _{LOAD} = 200 μA/dc)	V _{OH}	V _{SS} + 2.4	-	-	Vdc
Output Low Voltage All Outputs Except XTAL 1 and EXTAL 2 (I _{LOAD} = 1.6 mA/dc)	V _{OL}	-	-	V _{SS} - 0.4	Vdc
Power Dissipation	P _O	-	-	1200	mW
Capacitance (V _{IN} = 0, T _A = 25°C, f = 1.0 MHz) P10-P17, P20-P24, P40-P47 P30-P37 Reset S1.C1,SC2.IRQ	C _O	-	-	12.5 10 7.5	pF
Peripheral Data Setup Time (Figure 5)	T _{PDSU}	200	-	-	ns
Peripheral Data Hold Time (Figure 5)	T _{PDH}	0	-	-	ns
Delay Time, Enable negative transition to OS3 negative transition	T _{OSD1}	-	-	1.0	μs
Delay Time, Enable negative transition to OS3 positive transition	T _{OSD2}	-	-	1.0	μs
Delay Time, Enable negative transition to Peripheral Data Valid (Figure 5)	T _{PWD}	-	-	350	ns
Delay Time, Enable negative transition to Peripheral CMOS Data Valid (V _{CC} - 30% V _{CC} , P20-P24 (Figure 5))	T _{CMOS}	-	-	2.0	μs
Darlington Drive Current V _O = 1.5 Vdc P10-P17	I _{OD}	-1.0	-2.5	-10	mA/dc
Standby Voltage (Not Operating) (Operating)	V _{SSB} V _{SS}	4.00 4.75	-	5.25 5.25	Vdc

NOTE: The above electrics satisfy Ports 1 and 2 always, and Ports 3 and 4 in the single chip mode only.

BUS TIMING (Figure 9)

Characteristic	Symbol	Min	Typ	Max	Unit
Cycle Time	T _{CC}	1000	-	-	ns
Address Strobe Pulse Width High	PW _{ADH}	220	-	-	ns
Address Strobe Rise Time	t _{AR}	-	-	50	ns
Address Strobe Fall Time	t _{AF}	-	-	50	ns
Address Strobe Delay Time	t _{ASD}	60	-	-	ns
Enable Rise Time	t _{ENR}	-	-	50	ns
Enable Fall Time	t _{EF}	-	-	50	ns
Enable Pulse Width High Time	PW _{EH}	450	-	-	ns
Enable Pulse Width Low Time	PW _{EL}	450	-	-	ns
Address Strobe to Enable Delay Time	t _{ASD}	60	-	-	ns
Address Delay Time	t _{AD}	-	-	270	ns
Data Delay Write Time	t _{DDW}	-	-	225	ns
Data Set-up Time	t _{SD}	100	-	-	ns
Hold Time Read	t _{HR}	20	-	100	ns
Write	t _{HW}	20	-	-	ns
Address Delay Time for Latch	t _{ADL}	-	-	200	ns
Address Hold Time for Latch	t _{HL}	20	-	-	ns
Pulse Width	PW _O	370	370	-	ns
Address Hold Time	t _{AH}	20	-	-	ns
Total Up Time	t _{UT}	750	-	-	ns



MOTOROLA Semiconductor Products Inc.

MAXIMUM RATINGS

Rating	Symbol	Value	Unit
Supply Voltage	VCC	-0.3 to +7.0	Vdc
Input Voltage	Vin	-0.3 to +7.0	Vdc
Operating Temperature Range	TA	0 to 70	°C
Storage Temperature Range	Tstg	-55 to +150	°C
Thermal Resistance	θ_{JA}	100 50	°C/W
Plastic Package			
Ceramic Package			

This device contains circuitry to protect the inputs against damage due to high static voltage or electric fields; however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum rated voltage to this high impedance circuit. For proper operation it is recommended that V_{DD} and V_{SS} be constrained to the range $V_{DD} \leq (V_{DD} + V_{SS}) \leq V_{DD}$.

TABLE 1 — MODE AND PORT SUMMARY

MCU SIGNAL DESCRIPTION

This section gives a description of the MCU signals for the various modes. Figure 2 shows the general pin assignments for the signals. SC1 and SC2 are signals which vary with the mode that the chip is in. Table 1 gives a summary of their function.

MODE	PORT 1 Eight Lines	PORT 2 Five Lines	PORT 3 Eight Lines	PORT 4 Eight Lines	SC1	SC2
SINGLE CHIP	I/O	I/O		I/O	IS(1) OS(1)	
EXPANDED MUX	I/O	I/O	ADDRESS BUS (A0-A7) DATA BUS (D0-D7)	ADDRESS BUS* (A8-A15)	AS(0)	R/W(0)
EXPANDED NON-MUX	I/O	I/O	DATA BUS (D0-D7)	ADDRESS BUS* (A0-A7)	IS(0) OS(0)	R/W(0)

*These lines can be substituted for I/O (Input Only) starting with the most significant address line.

I = Input

IS = Input Strobe

SC = Strobe Control

O = Output

OS = Output Strobe

AS = Address Strobe

R/W = Read/Write

IOS = I/O Select

READ/WRITE TIMING FOR PORTS 3 AND 4 (Figures 3-4)

Characteristic	Symbol	Min	Typ	Max	Unit
Address Delay	tAD	-	-	270	ns
Peripheral Read Access Time	tacc	-	-	530	ns
(tacc = tAD + tDSR)					
Data Setup Time (Read)	tDSR	100	-	-	ns
Input Data Hold Time	tHR	10	-	-	ns
Output Data Hold Time	tHW	20	-	-	ns
Address Hold Time (Address, R/W)	tAH	20	-	-	ns
Data Delay Time (Write)	tDDW	-	165	225	ns
Processor Controls					
Processor Control Setup Time	tPCS	200	-	-	ns
Processor Control Rise and Fall Time (Measured between 0.5V and 2.0V)	tPCr, tPCh	-	-	100	ns

PORT 3 STROBE TIMING (Figures 7-8)

Characteristic	Symbol	Min	Typ	Max	Unit
Output Strobe Delay 1	tODS1	-	-	1.0	μs
Output Strobe Delay 2	tODS2	-	-	1.0	μs
Input Strobe Pulse Width	tPWs	200	-	-	ns
Input Data Hold Time	tIH	20	-	-	ns
Input Data Setup Time	tIS	100	-	-	ns



MOTOROLA Semiconductor Products Inc.

FIGURE 3 — READ DATA FROM MEMORY OR PERIPHERALS EXPANDED NON-MULTIPLEXED

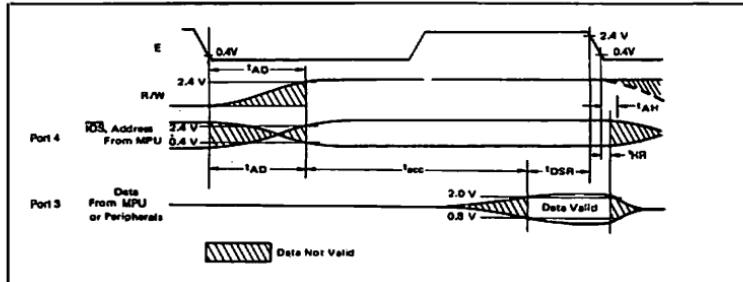
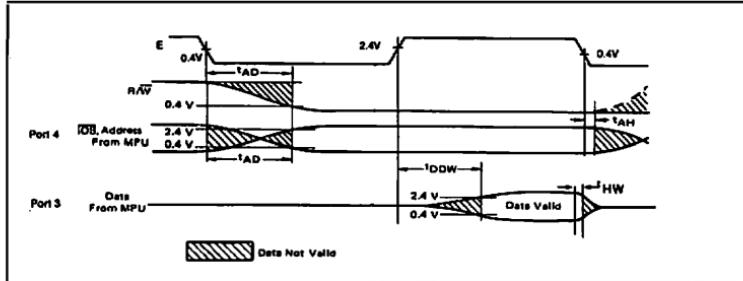
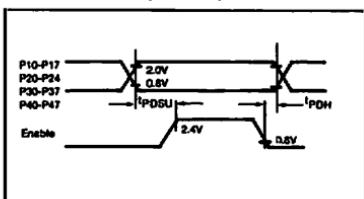
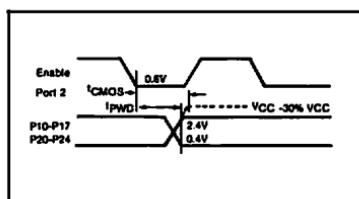


FIGURE 4 — WRITE DATA IN MEMORY OR PERIPHERALS EXPANDED NON-MULTIPLEXED



POR TS 1 AND 2, AND PORTS 3 AND 4 IN THE SINGLE CHIP MODE

FIGURE 5 — PERIPHERAL DATA SETUP AND HOLD TIMES
(Read Mode)FIGURE 6 — PERIPHERAL CMOS DATA DELAY TIMES
(Write Mode)

MOTOROLA Semiconductor Products Inc. —

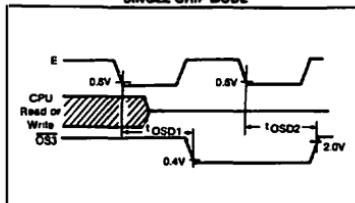
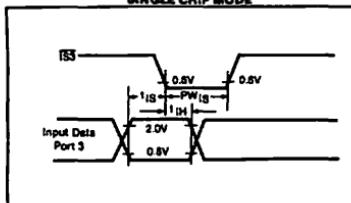
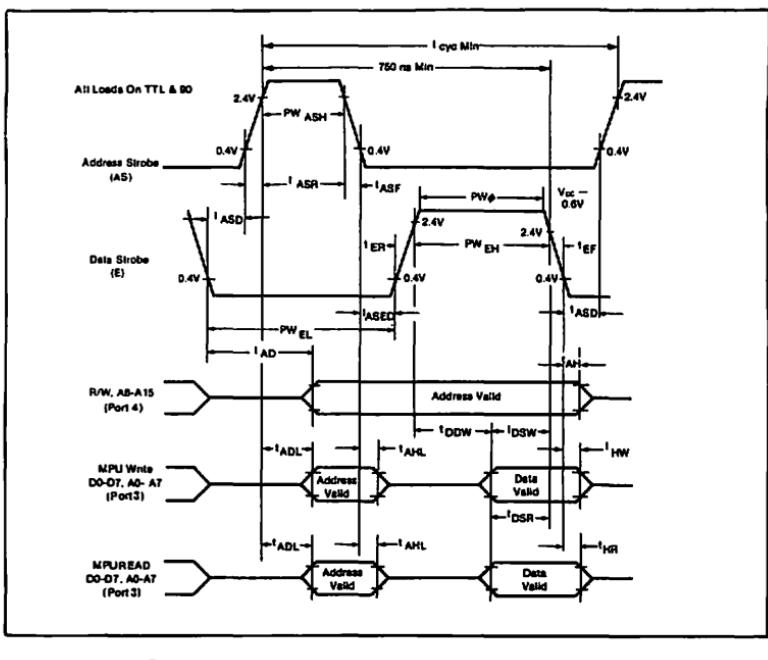
FIGURE 7 — OUTPUT STROBE TIMING —
SINGLE CHIP MODEFIGURE 8 — INPUT STROBE TIMING —
SINGLE CHIP MODE

FIGURE 9 — MULTIPLEXED BUS TIMING



MOTOROLA Semiconductor Products Inc.

FIGURE 10—CMOS LOAD

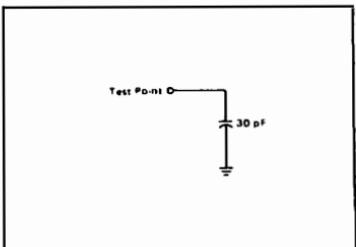
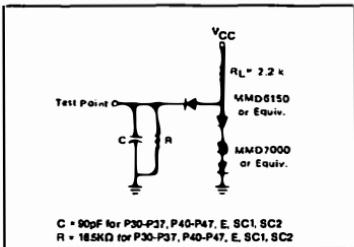
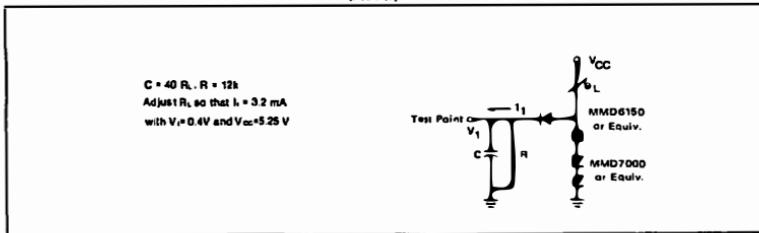
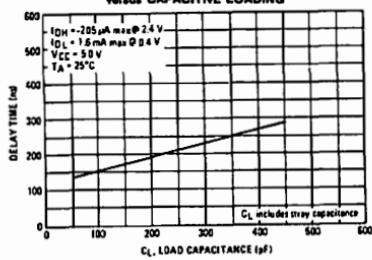
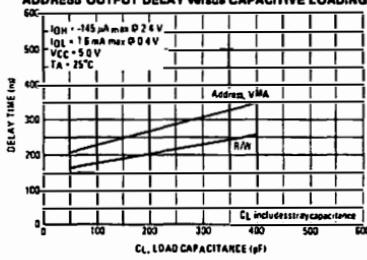


FIGURE 11 — BUS TIMING TEST LOAD AND PORTS 1, 3 AND 4 FOR SINGLE CHIP MODE

FIGURE 12 — TEST LOADS FOR PORT 1
Darlington Load
(P10-P17)FIGURE 13 — TYPICAL DATA BUS OUTPUT DELAY
versus CAPACITIVE LOADINGFIGURE 14 — TYPICAL READ/WRITE, VMA AND
ADDRESS88 OUTPUT DELAY versus CAPACITIVE LOADING

MOTOROLA Semiconductor Products Inc.

SIGNAL DESCRIPTIONS

Vcc and Vss

These two pins are used to supply power and ground to the chip. The voltage supplied will be +5 volts $\pm 5\%$.

XTAL 1 and XTAL 2

These connections are for a parallel resonant fundamental crystal. AT cut. Divide by 4 circuitry is included with the internal clock, so a 4 MHz crystal may be used to run the system at 1 MHz. The divide by 4 circuitry allows for use of the inexpensive 3.58 MHz Color TV crystal for non-time critical applications. Two 27 pF capacitors are needed from the two crystal pins to ground to insure reliable operation. XTAL2 may be driven by an external clock source at a 4 MHz rate to run at 1 MHz with a 40/60% duty cycle. It is not restricted to 4 MHz, as it will divide by 4 any frequency less than or equal to 4 MHz. XTAL1 must be grounded if an external clock is used. The following are the recommended crystal parameters:

```
AT - Cut Parallel Resonance Crystal
C1 = 7 pF MAX
FREQ = 4.0 MHz @ C1 = 24 pF
R1 = 50 ohms MAX.
Frequency Tolerance =  $\pm 5\%$  to  $\pm 0.02\%$ 
The best E output "Worst Case Design"
tolerance is  $\pm 0.05\%$  (500 ppm) using
A  $\pm 0.02\%$  crystal
```

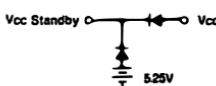
Vcc Standby

This pin will supply +5 volts $\pm 5\%$ to the standby RAM on the chip. The first 64 bytes of RAM will be maintained in the power down mode with 8 mA current max in the ROM version. The circuit of figure 15 can be utilized to assure that Vcc Standby does not go below Vss during power down.

To retain information in the RAM during power down the following procedure is necessary:

- 1) Write '0' into the RAM enable bit, RAM E. RAM E is bit 8 of the RAM Control Register at location \$0014. This disables the standby RAM, thereby protecting it at power down.
- 2) Keep Vcc Standby greater than Vss.

FIGURE 15—BATTERY BACKUP FOR Vcc STANDBY



Reset

This input is used to reset and start the MPU from a power down condition, resulting from a power failure or an initial start-up of the processor. On power up, the reset must be held low for at least 20 ms. During复位, Reset, when brought low, must be held low at least 3 clock cycles.

When a high level is detected, the MPU does the following:

- All the higher order address lines will be forced high.
- I/O Port 2 bits, 2, 1, and 0 are latched into programmed control bits PC2, PC1 and PC0.
- The last two (FFFE, FFFF) locations in memory will be used to load the program addressed by the program counter.
- The interrupt mask bit is set, must be cleared before the MPU can recognize maskable interrupts.

Enable (E)

This supplies the external clock for the rest of the system when the internal oscillator is used. It is a single phase, TTL

compatible clock, and will be the divide by 4 result of the crystal frequency. It will drive one TTL load and 90 pF.

Non-Maskable Interrupt (NMI)

A low-going edge on this input requests that an non-maskable interrupt sequence be generated within the processor. As with the interrupt request signal, the processor will complete the current instruction that is being executed before it recognizes the NMI signal. The interrupt mask bit in the Condition Code Register has no effect on NMI.

In response to an NMI interrupt, the Index Register, Program Counter, Accumulators, and Condition Code Register are stored on the stack. At the end of the sequence, a 16-bit address will be loaded that points to a vectoring address located in memory locations FFFC and FFFD. An address loaded at these locations causes the MPU to branch to a non-maskable interrupt service routine in memory.

A 3.3 k Ω external resistor to Vcc should be used for wire-OR and optimum control of interrupts.

Inputs IRQ and RM₁ are hardware interrupt lines that are sampled during E and will start the interrupt routine on the clock bus following the completion of an instruction.

Interrupt Request (IRQ)

This level sensitive input requests that an interrupt sequence be generated within the machine. The processor will wait until it completes the current instruction that is being executed before it recognizes the request. At that time, if the interrupt mask bit in the Condition Code Register is not set, the machine will begin an interrupt sequence. The Index Register, Program Counter, Accumulators, and Condition Code Register are stored on the stack. Next the MPU will respond to the interrupt request by setting the interrupt mask bit high so that no further maskable interrupts may occur. At the end of the cycle, a 16-bit address will be loaded that points to a vectoring address which is located in memory locations FFF8 and FFF9. An address loaded at these locations causes the MPU to branch to an interrupt routine in memory.

The IRQ requires a 3.3 k Ω external resistor to Vcc which should be used for wire-OR and optimum control of interrupts. Internal interrupts will use an internal interrupt line (IRQ2). This interrupt will operate the same as IRQ except that it will use the vector address of FFF0 and FFF7. IRQ1 will have priority over IRQ2; both occur at the same time. The interrupt mask bit in the condition code register masks both interrupts. (See Figure 25).

The following pins are available in the Single Chip Mode, and are associated with Port 3 only.

Input Strobe (IS3) (SC3)

This sets an interrupt for the processor when the IS3 Enable bit is set. As shown in Figure 8 Input Strobe Timing, IS3 will fall to a minimum after data is valid on Port 3. If IS3 Enable is set in the I/O Port Control/Status Register, an interrupt will occur. If the latch enable bit in the I/O Control/Status Register is set, this strobe will latch the input data from another device when that device has indicated that it has valid data.

Output Strobe (OS3) (SC2)

This signal is used by the processor to strobe an external device, indicating valid data is on the I/O pins. The timing for the Output Strobe is shown in Figure 7. I/O Port Control/Status Register is discussed in the following section.



MOTOROLA Semiconductor Products Inc.

The following pins are available in the Expanded Modes.

Read/Write (RW) (SC2)

This TTL compatible output signal controls the peripherals and memory devices whether the MPU is in a Read (high) or a Write (low) state. The normal standby state of this signal is Read (high). This output is capable of driving one TTL load and 90 pF.

I/O Strobe (IOS) (SC1)

In the expanded non-multiplexed mode of operation, IOS internally decodes AD through A15 as zero's and A8 as a one. This allows external access of the 256 locations from \$0100 to \$01FF. The timing diagrams are shown as Figures 3 and 4.

Address Strobe (AS) (SC1)

In the expanded multiplexed mode of operation address strobe is output on this pin. This signal is used to latch the 8 LSB's of address which are multiplexed with data on Port 3. An 8-bit latch is utilized in conjunction with Address Strobe, as shown in figure 29, Expanded Multiplexed Mode. Address Strobe signals the latch when it is latched to catch the address lines so the lines can become data bus lines during the E pulse. The timing for this signal is shown in the MC6801 Bus Timing Figure 9. This signal is also used to disable the address from the multiplexed bus allowing a deselect time, T_{AS} , before the data is enabled to the bus.

MC6801 PORTS

There are four I/O ports on the MC6801 MCU: three 8-bit ports and one 5-bit port. There are two control lines associated with one of the 8-bit ports. Each port has an associated write only Data Direction Register which allows each I/O line to be programmed to act as an input or an output.* A "1" in the corresponding Data Direction Register bit will cause that I/O line to be an output. A "0" in the corresponding Data Direction Register bit will cause that I/O line to be an input. There are four ports: Port 1, Port 2, Port 3, and Port 4. Their addresses and the addresses of their Data Direction registers are given in Table 2.

*The only exception is bit 1 of Port 2, which can either be data input or Timer output.

TABLE 2 — PORT AND DATA DIRECTION REGISTER ADDRESSES

Ports	Port Address	Data Direction Register Address
I/O Port 1	\$0002	\$0000
I/O Port 2	\$0003	\$0001
I/O Port 3	\$0006	\$0004
I/O Port 4	\$0007	\$0005

UD Port 1

This is an 8-bit port whose individual bits may be defined as inputs or outputs by the corresponding bit in its data direction register. The 8 output buffers have three-state capability, allowing them to enter a high impedance state when the peripheral data lines are used as inputs. In order to read properly, the voltage on the input lines must be greater than 2.0 volt for a logic "1" and less than 0.8 volt for a logic "0". As outputs, these lines are TTL compatible and may also be used as a source of up to 1 mA at 1.5 volts to directly drive a Darlington base. After Reset, the I/O lines are configured as inputs. In all three modes, Port 1 is always parallel I/O.

I/O Port 2

This port has 16 lines that may be defined as inputs or outputs by its data direction register. The 5 output buffers have three-state capability, allowing them to enter a high impedance state when used as an input. In order to be read properly, the voltage on the input lines must be greater than 2.0 volt for a logic "1" and less than 0.8 volt for a logic "0". As outputs, this port has no internal pullup resistors but will drive TTL inputs directly. For driving CMOS inputs, external pullup resistors are required. After Reset, the I/O lines are configured as inputs. Three pins on Port 2 (pins 10, 9 and 8 of the chip) are used to program the mode of operation during reset. The values of these pins at reset are latched into the three MSB's (bits 7, 6 and 5) of Port 2 which are read only. This is explained in the Mode Selection Section.

In all three modes, Port 2 can be configured as I/O and provides access to the Serial Communications Interface and the Timer. Bit 1 is the only pin restricted to data input or Timer output.

I/O Port 3

This is an 8-bit port that can be configured as I/O, a data bus, or an address bus multiplexed with the data bus—depending on the mode of operation hardware programmed by the user at reset. As a data bus, Port 3 is bi-directional. As an input for peripherals, it must be supplied regular TTL levels, that is, greater than 2.0 volts for a logic "1" and less than 0.8 volt for a logic "0".

Its TTL compatible three-state output buffers are capable of driving one TTL load and 90 pF. In the Expanded Modes, after reset, the data direction register is inhibited and data flow depends on the state of the R/W line. The input strobe (IS3) and the output strobe (OS3) used for handshaking are explained later.

In the three modes Port 3 assumes the following characteristics:

Single Chip Mode: Parallel Inputs/Outputs as programmed by its associated Data Direction Register. There are two control lines associated with this port in this mode, an input strobe and an output strobe, that can be used for handshaking. They are controlled by the I/O Port Control/Status Register explained at the end of this section.

Expanded Non-Multiplexed Mode: In this mode Port 3 becomes the data bus (D7-D0).

Expanded Multiplexed Mode: In this mode Port 3 becomes both the data bus (D7-D0) and lower bits of the address bus (A7-A0). An address strobe output is true when the address is on the port.

I/O PORT 3 CONTROL/STATUS REGISTER

7	6	5	4	3	2	1	0
IS3 FLAG	IS3 ENABLE	X	OS3 ENABLE	LATCH	X	X	X

Bit 0 Not used.

Bit 1 Not used.

Bit 2 Not used.

Bit 3 Latch Enable. This controls the input latch for I/O Port 3. If this bit is set high the input data will be latched with the falling edge of the Input Strobe, IS3. This bit is cleared by reset, or CPU Read Port 3.



MOTOROLA Semiconductor Products Inc.

Bit 4 (DS3) Output Strobe Select. This bit will select if the Output Strobe should be generated by a write to I/O Port 3 or a read of I/O Port 3. When this bit is cleared the strobe is generated by a read Port 3. When this bit is set the strobe is generated by a write Port 3.

Bit 5 Not used.

Bit 6 (SI3) ENABLE. This bit will be the interrupt caused by IS3. When set to a low level the IS3 Flag will be set by input strobe but the interrupt will not be generated. This bit is cleared by reset.

Bit 7 (SI3 FLAG). This is a read only status bit that is set by the falling edge of the input strobe, IS3. It is cleared by a read of the Control Status Register followed by a read or write of I/O Port 3. Reset will clear this bit.

I/O Port 4

This is an 8-bit port that can be configured as I/O or as address lines depending on the mode of operation. In order to be read properly, the voltage on the input lines must be greater than 2.0 volts for a logic "1" and less than 0.8 volt for a logic "0".

As outputs, each line is TTL compatible and can drive 1 TTL load and 90 pF. After reset, the lines are configured as inputs. To use the pins as addresses, therefore, they should be programmed as outputs. In the three modes, Port 4 assumes the following characteristics:

Single Chip Mode: Parallel Inputs/Outputs as programmed by its associated Data Direction Register.

Expanded Non-Multiplexed Mode: In this mode Port 4 is configured as the lower order address lines (A7-A0) by writing one's to the data direction register. When all eight address lines are not needed, the remaining lines, starting with the most significant bit, may be used as I/O (inputs only).

Expanded Multiplexed Mode: In this mode Port 4 is configured as the high order address lines (A15-A8) by writing one's to the data direction register. When all eight address lines are not needed, the remaining lines, starting with the most significant bit, may be used as I/O (inputs only).

MODE SELECTION

The mode of operation that 6801 will operate in after Reset is determined by hardware that the user must wire on pins 10, 9, and 8 of the chip. These pins are the three LSB's (I/O 2, I/O 1, and I/O 0

respectively) of Port 2. They are latched into programmed control bits PC2, PC1, and PC0 when reset goes high. I/O Port 2 Register is shown below.

S0003	7	6	5	4	3	2	1	0
	PC2	PC1	PC0	I/O 4	I/O 3	I/O 2	I/O 1	I/O 0

An example of external hardware that could be used in the Expanded Non-Multiplexed Mode is given in Figure 16. In the Expanded Non-Multiplexed Mode, pins 10, 9 and 8 are programmed Hi, Lo, Hi respectively as shown.

Couplers between the pins on Port 2 and the peripherals attached may be required. If the lines go to devices which require signals at power up differing from the signals needed to program the 6801's mode, couplers are necessary.

The MC14066B can be used to provide this isolation between the peripheral device and the MCU during reset. Figure 17 shows the logic diagram and truth table for the MC14066B. It is bidirectional and requires no external logic to determine the direction of the information flow. The logic shown insures that the data on the peripheral will not change before it is latched into the MCU and the MCU has started the reset sequence.

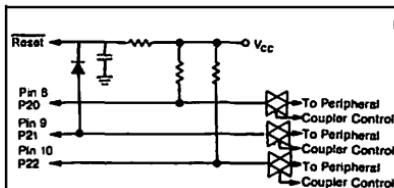


FIGURE 16 — DIODE CONFIGURATION FOR THE EXPANDED NON-MULTIPLEXED MODE

As bits 5, 8 and 7 of Port 2 are read only, the mode cannot be changed through software. The mode selections are shown in Table 3.
P20 refers to Port 2, bit 0.



MOTOROLA Semiconductor Products Inc.

FIGURE 17—MC14066B QUAD ANALOG SWITCH/MULTIPLEXER IN A TYPICAL MC6801 CIRCUIT

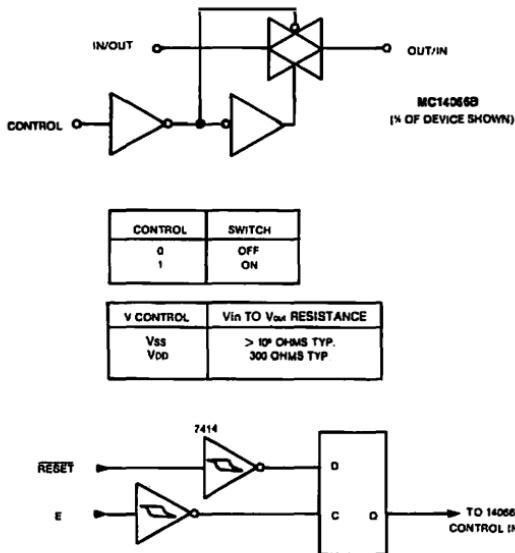
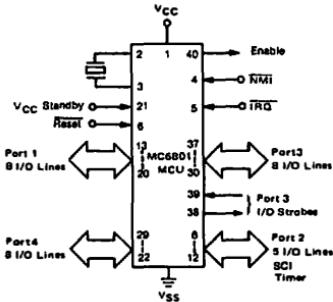


FIGURE 18 — MC6801 MCU SINGLE-CHIP MODE

**MC6801 BASIC MODES**

The MC6801 is capable of operating in three basic modes: (1) Single Chip Mode, (2) Expanded Multiplexed Mode (compatible with MC6800 peripheral family) (3) Expanded Non-Multiplexed Mode.

SINGLE CHIP MODE

In the Single Chip Mode the Ports are configured for I/O. This is shown in Figure 18 the single Chip Mode. In this mode, Port 3 will have two associated control lines, an input strobe and an output strobe for handshaking data.



MOTOROLA Semiconductor Products Inc.

EXPANDED NON-MULTIPLEXED MODE

In this mode the MC6801 will directly address M6800 peripherals with no external logic. In this mode Port 3 becomes the data bus, Port 4 becomes the A7-A0 address bus or partial address and I/O (Inputs only), Port 2 can be parallel I/O, serial I/O, Timer, or any combination thereof. Port 1 is parallel I/O

only. In this mode the MC6801 is expandable to 256 locations. The eight address lines associated with Port 4 may be substituted for I/O (Inputs only) if a fewer number of address lines will satisfy the application. (See Figure 19).

FIGURE 19 — MC6801 MCU EXPANDED NON-MULTIPLEXED MODE

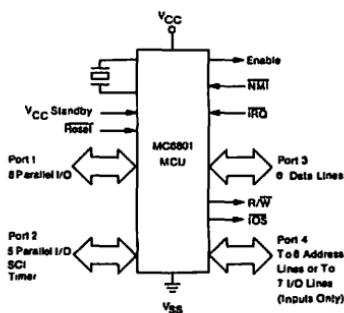
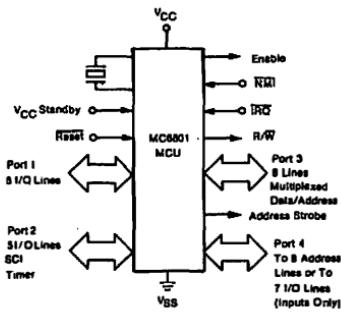


FIGURE 20 — MC6801 MCU EXPANDED MULTIPLEXED MODE



EXPANDED MULTIPLEXED MODE

In this mode Port 4 becomes higher order address lines with an alternative of substituting some of the address lines for I/O (Inputs only). Port 3 is the data bus multiplexed with the lower order address lines differentiated by an output called Address Strobe. Port 2 is 5 lines of Parallel I/O, SCI, Timer, or any combination thereof. Port 1 is 8 Parallel I/O lines. In this mode it is expandable to 65K words. (See Figure 20).



MOTOROLA Semiconductor Products Inc.

TABLE 3 — MODE SELECTS

MODE	PROGRAM	CONTROL	ROM	RAM	INTERRUPT VECTORS	BUS
7 SINGLE CHIP	HI	HI	HI	I	I	I
6 EXPANDED MULTIPLEXED	HI	HI	Lo	I	I	Ep/M
5 EXPANDED NON-MULTIPLEXED	HI	Lo	Hi	I	I	Ep
4 SINGLE CHIP TEST	HI	Lo	Lo	I(2)	I(1)	I
3 64K ADDRESS I/O	Lo	Hi	Hi	E	E	Ep/M
2 PORTS 3 & 4 EXTERNAL	Lo	Hi	Lo	E	E	Ep/M
1	Lo	Lo	Hi	I	I	Ep/M
0 TEST-DATA OUTPUTTED FROM ROM & RAM TO I/O PORT 3	Lo	Lo	Lo	I	I	Ep/M

E — EXTERNAL all vectors are external

I — INTERNAL

Ep — EXPANDED

M — MULTIPLEXED

* First two addresses read from external after reset

(1) Address for RAM XX80-XXFF

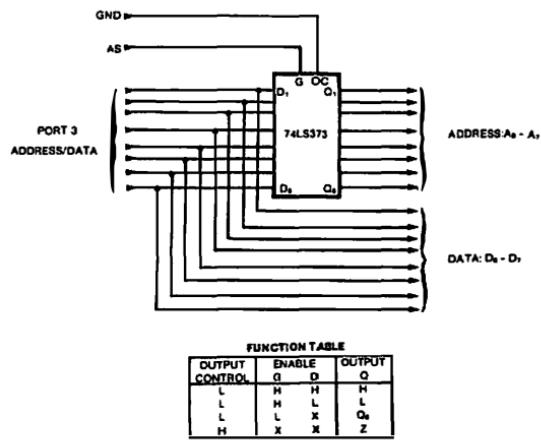
(2) ROM disabled

Lower order Address Bus Latches

Since the data bus is multiplexed with the lower order address bus in Port 3, latches are required to latch those address bits. The SN74LS373 Transparent octal D-type latch

can be used with the MC6801 to latch the least significant address byte. Figure 21 shows how to connect the latch to the MC6801. The output controls to the LS373 may be connected to ground.

FIGURE 21 — LATCH CONNECTION



MOTOROLA Semiconductor Products Inc.

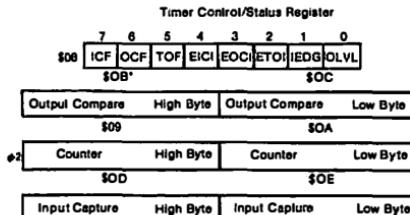
PROGRAMMABLE TIMER

The MC6801 contains an on-chip 16-bit programmable timer which may be used to perform measurements on an input waveform while independently generating an output waveform. Pulse widths for both input and output signals may vary from a few microseconds to many seconds. The timer hardware consists of

- an 8-bit control and status register,
- a 16-bit free running counter,
- a 16-bit output compare register, and
- a 16-bit input capture register

A block diagram of the timer registers is shown in Figure 22.

FIGURE 22 — BLOCK DIAGRAM OF TIMER REGISTERS



* The characters above the registers represent their address in Hex.

Free Running Counter (\$0000:00DA)

The key element in the programmable timer is a 16-bit free running counter which is driven to increasing values by the MPU φ. The counter value may be read by the MPU software at any time. The counter is cleared to zero on RESET and may be considered a read-only register with one exception. Any MPU write to the counter's address (\$09) will always result in a preset value of \$FFFB being loaded into the counter regardless of the value involved in the write. This preset feature is intended for testing operation of the part, but may be of value in some applications.

Output Compare Register (\$000B:000C)

The Output Compare Register is a 16-bit read/write register which is used to control an output waveform. The contents of this register are constantly compared with the current value of the free running counter. When a match is found a flag is set (OCF) in the Timer Control and Status Register (TCSR) and the current value of the Output Level bit (OLVL) in the TCSR is clocked to the output level register. Providing the Data Direction Register for Port 2, Bit 1 contains a "1" (output), the output level register value will appear on the pin for Port 2 Bit 1. The values in the Output Compare Register and Output Level Register may then be changed to control the output level on the next compare value. The Output Compare Register is set to \$FFFF during RESET. The Compare function is inhibited for one cycle following a write to the high byte of the Output Compare Register to insure a valid 16-bit value is in the register before a compare is made.

Input Capture Register (\$000D:000E)

The Input Capture Register is a 16-bit read-only register used to store the current value of the free running counter whenever the proper transition of an external input signal occurs. The input transition change required to trigger the counter transfer is controlled by the Input Edge bit (IEDG) in the TCSR. The Data Direction Register for Port 2 Bit 0, should be clear (zero) in order to gate in the external input signal to the edge detect unit in the timer.

*With Port 2 Bit 0 configured as an output and set to "1", the external input will still be seen by the edge detect unit.

Timer Control and Status Register (TCSR) (\$0008)

The Timer Control and Status Register consists of an 8-bit register of which all 8 bits are readable but only the low order 5 bits may be written. The upper three bits contain read-only timer status information and indicate that:

- a proper transition has taken place on the input pin with a subsequent transfer of the current counter value to the input capture register,
- a match has been found between the value in the free running counter and the output compare register, and
- when \$0000 is in the free running counter.

Each of the flags may be enabled onto the MC6801 internal bus (IRQ2) with an individual Enable bit in the TCSR. If the I-bit in the MC6801 Condition Code register has been cleared, a priority vectored interrupt will occur corresponding to the flag bit(s) set. A description for each bit follows:

7	6	5	4	3	2	1	0
Timer Control	ICF	OCF	TOF	EIC1	EOCI	ETOI	IEDG

and Status Register



MOTOROLA Semiconductor Products Inc.

- Bit 0 OLVL Output Level** — This value is clocked to the output level register on an output compare. If the DDR for Port 2 bit 1 is set, the value will appear on the output pin.
- Bit 1 IEDG Input Edge** — This bit controls which transition of an input will trigger a transfer of the counter to the input capture register. The DDR for Port 2 Bit 0 must be clear for this function to operate. IEDG = 0 Transfer takes place on negative (high-to-low transition). IEDG = 1 Transfer takes place on a positive edge (low-to-high transition).
- Bit 2 ET0F Enable Timer Overflow Interrupt** — When set, this bit enables IRQ2 to occur on the internal bus for a TOF interrupt; when clear the interrupt is inhibited.
- Bit 3 EOCl Enable Output Compare Interrupt** — When set, this bit enables IRQ2 to appear on the internal bus for an input capture interrupt; when clear the interrupt is inhibited.

- Bit 4 EIICl Enable Input Capture Interrupt** — When set, this bit enables IRQ2 to occur on the internal bus for an input capture interrupt; when clear the interrupt is inhibited.
- Bit 5 TOF Timer Overflow Flag** — This read-only bit is set when the counter contains \$0000. It is cleared by a read of the TCSR (with TOF set) followed by an MPU read of the Counter (\$09).
- Bit 6 OCF Output Compare Flag** — This read-only bit is set when a match is found between the output compare register and the free running counter. It is cleared by a read of the TCSR (with OCF set) followed by an MPU write to the output compare register (\$0B or \$0C).
- Bit 7 ICF Input Capture Flag** — This read-only status bit is set by a proper transition on the input to the edge detect unit; it is cleared by a read of the TCSR (with ICF set) followed by an MPU read of the Input Capture Register (\$0D).

SERIAL COMMUNICATIONS INTERFACE

The MC6801 contains a full-duplex asynchronous serial communications interface (SCI) on board. Two serial data formats (standard mark/space (NRZ) or Bi-phase) are provided at several different data rates. The controller comprises a transmitter and a receiver which operate independently of each other. But in the same data format and at the same data rate. Both transmitter and receiver communicate with the MPU via the data bus and with the outside world via pins 2, 3, and 4 of Port 2. The hardware, software, and registers are explained in the following paragraphs.

Wake-Up Feature

In a typical multi-processor application, the software protocol will usually contain a destination address in the initial byte(s) of the message. In order to permit non-selected MPUs to ignore the remainder of the message, a wake-up feature is included whereby all further interrupt processing may be optionally inhibited until the beginning of the next message. When the next message appears, the hardware re-enables (or "wakes-up") the for the next message. The "wake-up" is automatically triggered by a string of ten consecutive '1's which indicates an idle transmit line. The software protocol must provide for the short idle period between any two consecutive messages.

Programmable Options

The following features of the MC6801 serial I/O section are programmable:

- format — standard mark/space (NRZ) or Bi-phase
- clock — external or internal
- baud rate — one of 4 per given MPU #2 clock frequency or external clock X8 input
- wake-up feature — enabled or disabled
- interrupt requests — enabled or masked individually for transmitter and receiver data registers
- clock output — Internal clock enabled or disabled to Port 2 (Bit 2)
- Port 2 (bits 3 and 4) — dedicated or not dedicated to serial I/O individually for transmitter and receiver.

Serial Communications Hardware

The serial communications hardware is controlled by 4 registers as shown in Figure 23. The registers include:

- a 4-bit control and status register
- a 4-bit rate and mode control register (write only)
- an 8-bit read only receive data register
- an 8-bit write only transmit data register.

In addition to the four registers, the serial I/O section utilizes bit3 (serial input) and bit4 (serial output) or Port 2, Bit 2 of Port 2 is utilized if the internal-clock-out or external-clock-in options are selected.



MOTOROLA Semiconductor Products Inc.

FIGURE 23 — SERIAL I/O REGISTERS

**CONTROL AND STATUS REGISTER \$0011, READ/WRITE
EXCEPT *** (READ ONLY)**

7	6	5	4	3	2	1	0
RDRF	ORFE	TDRE	RIE	RE	TIE	TE	WU

RATE AND MODE REGISTER \$0010, WRITE ONLY

X	X	X	X	CC1	CC0	S1	S0
---	---	---	---	-----	-----	----	----

RECEIVE DATA REGISTER, \$0012, READ ONLY



PORT2 BIT3
P23 RX RECEIVE DATA SHIFT REGISTER (NOT USER ADDRESSABLE)

PORT2 BIT2
P22

EXT CLK IN/INT CLK OUT

PORT2 BIT4
P24 TX

(NOT USER ADDRESSABLE)

TRANSMIT DATA SHIFT REGISTER



TRANSMIT DATA REGISTER \$0013, WRITE ONLY

Transmit/Receive Control and Status (TRCS) Register

The TRCS register consists of an 8-bit register of which all bits may be read while only bits 0-4 may be written. The register is initialized to \$20 on RESET. The bits in the TRCS register are defined as follows:

7	6	5	4	3	2	1	0
RDRE	ORFE	TDRE	RIE	RE	TIE	TE	WU

ADDR: \$0011



MOTOROLA Semiconductor Products Inc.

- Bit 0 WU** "Wake-up" on Next Message — set by MC6801 software cleared by hardware on receipt of ten consecutive Ts.
- Bit 1 TE** Transmit Enable — set by MC6801/MC68701 to produce preamble of nine consecutive 1's and to enable gating of transmitter output to Port 2, bit 4 regardless of the DDR value corresponding to this bit; when clear, serial I/O has no effect on Port 2 bit 4.
- Bit 2 TIE** Transmit Interrupt Enable — when set, will permit an IRQ2 interrupt to occur when bit 5 (TDRE) is set; when clear, the TDRE value is masked from the bus.
- Bit 3 RE** Receiver Enable — when set, gates Port 2 bit 3 to input of receiver regardless of DDR value for this bit; when clear, serial I/O has no effect on Port 2 bit 3.
- Bit 4 RIE** Receiver Interrupt Enable — when set, will permit an IRQ2 interrupt to occur when bit 7 (RDRE) or bit 6 (ORF) is set; when clear, the interrupt is masked.

Bit 5 TDRE Transmit Data Register Empty — set by hardware when a transfer is made from the transmit data register to the output shift register. The TDRE bit is cleared by reading the status register, then writing a new byte into the transmit data register. TDRE is initialized to 1 by RESET.

Bit 6 ORFE Over-Run-Framing Error — set by hardware when an overrun or framing error occurs (receive only). An overrun is defined as a new byte received with last byte still in Data Register/Buffer. A framing error has occurred when the byte boundaries in bit stream are not synchronized to bit counter. The ORFE bit is cleared by reading the status register, then reading the Receive Data Register, or by RESET.

Bit 7 RDRE Receiver Data Register Full — Set by hardware when a transfer from the inputs shift register to the receiver data register is made. The RDRE bit is cleared by reading the status register, then reading the Receive Data Register, or by RESET.

Rate and Mode Control Register

The Rate and Mode Control register controls the following serial I/O variables:

- Baud rate
- format
- clocking source, and
- Port 2 bit 2 configuration

The register consists of 8 bits all of which are write-only and cleared on RESET. The 4 bits in the register may be considered as a pair of 2-bit fields. The two lower order bits control the bit rate for internal clocking and the remaining two bits control the format and clock select logic. The register definition is as follows:

7	6	5	4	3	2	1	0	ADDR:\$0010
X	X	X	X	CC1	CC0	S1	S0	

Bit 0 S0

Speed Select — These bits select the Baud rate for the internal clock. The four rates which may be selected are a function of the MPU $\times 2$ clock frequency. Table 4 lists the available Baud rates.

Bit 1 S1

Clock Control and Format Select — this 2-bit field controls the format and clock select logic. Table 5 defines the bit field.



MOTOROLA Semiconductor Products Inc.

TABLE 4 — SCI INTERNAL BAUD RATES

S1,S0	XTAL	4.0 MHz	4.9152 MHz	2.5476 MHz
	$\phi/2$	1.0 MHz	1.2268 MHz	0.6144 MHz
00	$\phi/2 + 16$	62.5k Bits/s	76.8k Bits/s	38.4k Bits/s
01	$\phi/2 + 128$	7,812.5 Bits/s	9,600 Bits/s	4,800 Bits/s
10	$\phi/2 + 1024$	976.6 Bits/s	1,200 Bits/s	600 Bits/s
11	$\phi/2 + 4096$	244.1 Bits/s	300 Bits/s	150 Bits/s

TABLE 5 — BIT FIELD

CC1, CC0	Format	Clock Source	Port 2 Bit 2	Port 2 Bit 3	Port 2 Bit 4
00	Bi-Phase	Internal	Not Used
01	NRZ	Internal	Not Used
10	NRZ	Internal	Output*	Serial Input	Serial Output
11	NRZ	External	Input	Serial Input	Serial Output

*Clock output is available regardless of values for bits RE and TE.

**Bit3 is used for serial input if RE = "1" in TRCS; bit4 is used for serial output if TE = "1" in TRCS.

Internally Generated Clock

- If the user wishes for the serial I/O to furnish a clock, the following requirements are applicable:
- the values of RE and TE are immaterial.
 - the values of CC1, CC0 must be set to 10
 - the maximum clock rate will be $\phi/2 + 16$.
 - the clock will beat 1X the bitrate and will have a rising edge at mid-bit.

Externally Generated Clock

- If the user wishes to provide an external clock for the serial I/O, the following requirements are applicable:
- the CC1, CC0 field in the Rate and Mode Control Register must be set to 11.
 - the external clock must be set to 8 times (X8) the desired baud rate and
 - the maximum external clock frequency is 1.3 MHZ.



MOTOROLA Semiconductor Products Inc.

SERIAL OPERATIONS

The serial I/O hardware should be initialized by the MC6801 software prior to operation. This sequence will normally consist of:

- writing the desired operation control bits to the Rate and Mode Control Register and
- writing the desired operational control bits in the Transmit/Receive Control and Status Register.

The Transmitter Enable (TE) and Receiver Enable (RE) bits may be left set for dedicated operations.

Transmit Operations

The transmit operation is enabled by the TE bit in the Transmit/Receive Control and Status Register. This bit when set, gates the output of the serial transmittal register to Port 2 Bit 4 and takes unconditional control over the Data Direction Register value for Port 2, Bit 4.

Following a RESET, the user should configure both the Rate and Mode Control Register and the Transmit/Receiver Control and Status Register for desired operation. Setting the TE bit during INIT procedure initiates the serial output by first transmitting a ten-bit preamble of 1's. Following the preamble, internal synchronization is established and the transmitter section is ready for operation.

At this point one of two situations exist:

- a) if the Transmit Data Register is empty ($TDR = 1$), a continuous string of ones will be sent indicating an idle line, or
- b) if data has been loaded into the Transmit Data Register ($TDR = 0$), the word is transferred to the output shift register and transmission of the data word will begin.

During the transfer itself, the 0 start bit is first transmitted. Then the 8 data bits (beginning with bit 0) followed by the stop bit, are transmitted. When the Transmitter Data Register has been emptied, the hardware sets the TDRE flag bit.

If the MC6801 fails to respond to the flag within the proper time, ($TDRE$ is still set when the next normal transfer from the parallel data register to the serial output register should occur) then a 1 will be sent (instead of a 0) at "Start" bit time, followed by more 1's until more data is supplied to the data register. No 0's will be sent while $TDRE$ remains a 1.

The Bi-phase mode operates as described above except that the serial output toggles each bit time, and on 1/2 bit times when a 1 is sent.

Receive Operation

The receive operation is enabled by the RE bit which gates in the serial input through Port 2 Bit 3. The receiver section operation is conditioned by the contents of the Transmit/Receive Control and Status Register and the Rate and Mode Control Register.

The receiver bit interval is divided into 8 sub-intervals for internal synchronization. In the standard, non-Bi-phase mode, the received bit stream is synchronized by the first 0 (space) encountered.

The approximate center of each bit time is strobed during the next 10 bits. If the tenth bit is not a 1 (stop bit) a framing error is assumed, and bit ORFE is set. If the tenth bit is a 1, the data is transferred to the Receiver Data Register, and interrupt flag RDRF is set. If RDRF is still set at the next tenth bit time, DRFE will be set, indicating an over-run has occurred. When the MC6801 responds to either flag (RDRF or ORFE) by reading the status register followed by reading the Data Register, RDRF (or DRFE) will be cleared.

RAM CONTROL REGISTER

This register, which is addressed at \$0014, gives status information about the standby RAM. A 0 in the RAM enable bit (RAM E) will disable the standby RAM, thereby protecting it at power down if V_{CC} is held greater than V_{SSA} volts, as explained previously in the signal description for V_{CC} Standby.

\$0014	STANDBY BIT	RAM E	X	X	X	X	X	X

Bit 0 Not Used.

Bit 1 Not Used.

Bit 2 Not used.

Bit 3 Not used.

Bit 4 Not used.

Bit 5 Not used.

Bit 6 The RAM ENABLE control bit allows the user the ability to disable the standby RAM. This bit is set to a logic "one" by reset which enables the standby RAM and can be written to one or zero under program control. When the RAM is disabled, logic "zero", data is read from external memory.

Bit 7 The STANDBY BIT of the control register, \$0014, is cleared when the standby voltage is removed. This bit is a read/write status flag that the user can read which indicates that the standby RAM voltage has been applied, and the data in the standby RAM is valid.



MOTOROLA Semiconductor Products Inc.

FIGURE 24 — MEMORY MAP

The MC6801 provides up to 65k bytes of memory for program and/or data storage. The memory map is shown in Figure 24.

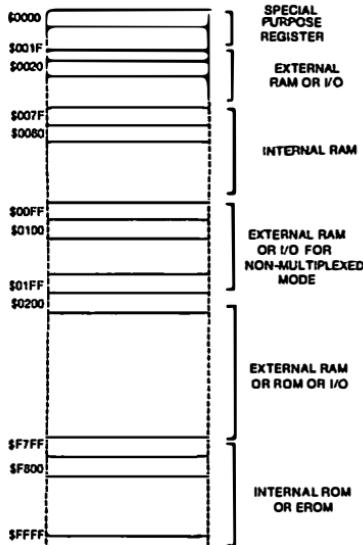


TABLE 6 — SPECIAL REGISTERS

The first 32 bytes are for the special purpose registers as shown in Table 6.

Hex Address	Register
00	Data Direction 1
01	Data Direction 2
02	I/O Port 1
03	I/O Port 2
04	Data Direction 3
05	Data Direction 4
06	I/O Port 3
07	I/O Port 4
08	TCBR
09	Counter High Byte
0A	Counter Low Byte
0B	Output Compare High Byte
0C	Output Compare Low Byte
0D	Input Capture High Byte
0E	Input Capture Low Byte
0F	I/O Port 3 C/S Register
10	Serial Rate and Mode Register
11	Serial Control and Status Register
12	Serial Receiver Data Register
13	Serial Transmit Data Register
14	RAM/EROM Control Register
15-1F Reserved	

FIGURE 25 — MEMORY MAP FOR INTERRUPT VECTORS

Highest Priority	Vector	Description	
	MS	LS	
	FFFF, FFFF		Restart
	FFFF, FFFD		Non-Maskable Interrupt
	FFFF, FFFB		Software Interrupt
	FFFF, FFF9		IRQ1/Interrupt Steepe 3
	FFFF, FFF7		IRQ2/Timer Input Capture
	FFFF, FFF5		IRQ2/Timer Output Compare
	FFFF, FFF3		IRQ2/Timer Overflow
	FFFF, FFF1		IRQ2/Serial I/O Interrupt
Lowest Priority			

Locations \$0020 through \$007F access external RAM or I/O. Internal RAM is accessed at \$0080 through \$00FF. The RAM may be alternately selected by mask programming at location \$A080. However, if the user desires to access external RAM at those locations he may do so by clearing the RAM ENABLE control bit of the RAM Control Register. In this way extra 128 bytes of external RAM are available. The first 64 bytes of the 128 bytes of on-chip RAM are provided with a separate power supply. This will maintain the 64 bytes of RAM in the power down mode as explained in the pin description for VccStandby.

Locations \$0100 through \$01FF are available in the Expanded Non-Multiplexed Mode. The eight address lines of Port 4 make

this 256 word expandability possible. Those not needed for address lines can be used as input lines instead.

The full range of addresses available to the user is in the Expanded Multiplexed Mode. Locations \$0200 through \$FFFF can be used as external RAM, external ROM, or I/O. Any higher order bits not required for addressing can be used as I/O as in the Expanded Non-Multiplexed Mode.

The internal ROM is located at \$F800 through \$FFFF. The decoder for the ROM may be mask programmed on A12 and A13 as zero or one's to provide for \$C800, \$D800, \$E800 for the ROM address. A12 and A13 may also be don't care in this decoder. The primary address for the ROM will be \$F800.



MOTOROLA Semiconductor Products Inc.

GENERAL DESCRIPTION OF INSTRUCTION SET

The MC6801 is upward object code compatible with the MC6800 as it implements the full M6800 instruction set. The execution times of key instructions have been reduced to increase throughput. In addition, new instructions have been added; these include 16-bit operations and a hardware multiply.

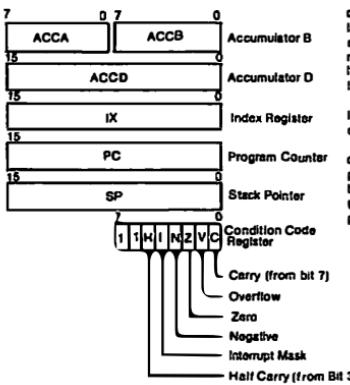
Included in the instruction set section are the following:

- MPU Programming Model (Figure 26)
- Addressing modes
- Accumulator and memory instructions — Table 7
- New instructions
- Index register and stack manipulations — Table 8
- Jump and branch instructions — Table 9
- Special operations — Figure 27
- Condition code register manipulation instructions — Table 10
- Instruction Execution times in machine cycles — Table 11
- Summary of cycle by cycle operation — Table 12

MPU PROGRAMMING MODEL

The programming model for the MC6801 is shown in Figure 26. The double (D) accumulator is physically the same as the A Accumulator concatenated with the B Accumulator so that any operation using accumulator D will destroy information in A and B.

FIGURE 26 — MCU PROGRAMMING MODEL



MPU ADDRESSING MODES

The MC6801 eight-bit microcomputer unit has seven address modes that can be used by a programmer, with the addressing mode a function of both the type of instruction and the coding within the instruction. A summary of the addressing modes for a particular instruction can be found in Table 11 along with the associated instruction execution time that is given in machine cycles. With a clock frequency of 4 MHz, these times would be microseconds.

Accumulator (ACCX) Addressing — In accumuloitor only addressing, either accumulator A or accumulator B is specified. These are one-byte instructions.

Immediate Addressing — In immediate addressing, the operand is contained in the second byte of the instruction except LDS and LDX which have the operand in the second and third bytes of the instruction. The MCU addresses this location when it fetches the immediate instruction for execution. These are two or three-byte instructions.

Direct Addressing — In direct addressing, the address of the operand is contained in the second byte of the instruction. Direct addressing allows the user to directly address the lowest 256 bytes in the machine i.e., locations zero through 255. Enhanced execution times are achieved by storing data in these locations. In most configurations, it should be a random access memory. These are two-byte instructions.

Extended Addressing — In extended addressing, the address contained in the second byte of the instruction is used as the higher eight-bits of the address of the operand. The third byte of the instruction is used as the lower eight-bits of the address for the operand. This is an absolute address in memory. These are three-byte instructions.

Indexed Addressing — In indexed addressing, the address contained in the second byte of the instruction is added to the index register's lowest eight bits in the MCU. The carry is then added to the higher order eight bits of the index register. This result is then used to address memory. The modified address is held in a temporary address register so there is no change to the index register. These are two-byte instructions.

Implied Addressing — In the implied addressing mode the instruction gives the address (i.e., stack pointer, index register, etc.). These are one-byte instructions.

Relative Addressing — In relative addressing, the address contained in the second byte of the instruction is added to the program counter's lowest eight bits plus two. The carry or borrow is then added to the high eight bits. This allows the user to address data within a range of -128 to +128 bytes of the present instruction. These are two-byte instructions.



MOTOROLA Semiconductor Products Inc.

TABLE 7—ACCUMULATOR & MEMORY INSTRUCTIONS

ACCUMULATOR AND MEMORY		ADDRESSING MODES																		
	MNEMONIC	OP	-	OP	-	OP	-	OP	-	OP	-	Boolean/Arithmetic Operation	H	I	N	Z	V	C		
Add	ADDA	6B	2	2	6B	3	2	AB	4	2	BB	4	3	A + M → B	1	*	1	1	1	
	ADDB	C9	2	2	D9	3	2	EB	4	2	FB	4	3	B + M → B	*	*	*	*	*	
Add Double	ADDO	C3	4	3	D3	5	2	E3	6	2	F3	6	3	A + B + M + I → A; B	*	*	*	*	*	
Add Accumulators	ABA											1B	2	1	A + B → A	*	*	*	*	*
Add With Carry	ADCA	89	2	2	09	3	2	A9	4	2	D9	4	3	A + M + C → A	*	*	*	*	*	
	ADC8	C9	2	2	D9	3	2	EB	4	2	FB	4	3	B + M + C → B	*	*	*	*	*	
AND	ANDA	84	2	2	04	3	2	AA	4	2	DA	4	3	A M → A	*	*	*	*	R *	
	ANDB	C4	2	2	D4	3	2	E4	4	2	FE	4	3	B M → B	*	*	*	*	R *	
Bit Test	BIT A	83	2	2	05	3	2	AS	4	2	BS	4	3	A M	*	*	*	*	R *	
	BITB	C5	2	2	05	3	2	ES	4	2	FS	4	3	B M	*	*	*	*	R *	
Clear	CLR							6F	6	2	7F	6	3	00 → M	*	*	R	S	R	
	CLRA											4F	2	1	00 → A	*	*	R	S	R
	CLRB											5F	2	1	00 → B	*	*	R	S	R
Compare	CMPA	81	2	2	01	3	2	A1	4	2	B1	4	3	A - M	*	*	*	*	*	
	CMPB	C1	2	2	D1	3	2	E1	4	2	F1	4	3	B - M	*	*	*	*	*	
Compare Accumulators	CBA											11	2	1	A - B	*	*	*	*	*
Complement, 1's	CD M							63	8	2	75	6	3	M → M	*	*	*	*	R *	
	CGMA											43	2	1	A → A	*	*	*	*	R *
	CGMB											53	2	1	B → B	*	*	*	*	R *
Complement, 2's	NEG							60	8	2	70	6	3	0C - M → M	*	*	*	*	(*)	
(Negate)	NEGA											40	2	1	00 - A → A	*	*	*	*	(*)
	NEGB											50	2	1	00 - B → B	*	*	*	*	(*)
Decimal Adjust, A	DAA											19	2	1	Converts binary add of BCD characters into BCD format				(*)	
Decrement	DEC							6A	6	2	7A	6	3	M - 1 → M	*	*	*	*	(*)	
	DECA											4A	2	1	A - 1 → A	*	*	*	*	(*)
	DEC B											5A	2	1	B - 1 → B	*	*	*	*	(*)
Exclusive OR	EORA	88	2	2	08	3	2	AB	4	2	BB	4	3	A ⊕ M → A	*	*	*	*	R *	
	EORB	C8	2	2	D8	3	2	EB	4	2	FB	4	3	B ⊕ M → B	*	*	*	*	R *	
Increment	INC							8C	6	2	7C	6	3	M + 1 → M	*	*	*	*	(*)	
	INCA											4C	2	1	A + 1 → A	*	*	*	*	(*)
	INCB											5C	2	1	B + 1 → B	*	*	*	*	(*)
Load Accumulator	LDAA	08	2	2	06	3	2	AB	4	2	BB	4	3	M → A	*	*	*	*	R *	
	LDAB	C8	2	2	D8	3	2	EB	4	2	FB	4	3	M → B	*	*	*	*	R *	
Load Double Accumulator	LDAD	CC	3	3	DC	4	2	EC	5	2	FC	5	3	M → A M + 1 → B	*	*	*	*	R *	

The Condition Code Register notes are listed after Table 10.

(Continued)



MOTOROLA Semiconductor Products Inc.

TABLE 7 — Continued
**ACCUMULATOR AND
MEMORY** **IMMED.** **DIRECT** **INDEX** **EXTEND** **INHERENT**

Operations	MNEMONIC	OP	~	#	Boolean/ Arithmetic Operation		H	I	N	Z	V	C	S	4	3	2	1	0																				
														3D	10	1	A	X	B	→	A	+	M	→	A	+	M	→	B	→	R	+	0					
Multiply Unsigned	MUL																																					
OR, Inclusive	ORAA	6A	2	2	BA	3	2	AA	4	2	BA	4	3																									
	ORAB	CA	2	2	DA	3	2	EA	4	2	FA	4	3																									
Push Data	PSHA															36	5	1	A → M ₀	SP - 1 → SP																		
	PSHB															37	3	1	B → M ₀	SP - 1 → SP																		
Pull Data	PUJA															32	4	1	SP + 1 → SP	M ₀ → A																		
	PULB															33	4	1	SP + 1 → SP	M ₀ → B																		
Rotate Left	ROL															66	6	2	70	6	3	M	↓	0	1	0	1	0	1	0	1	0	1					
	ROLA																49	2	1	A	↓	0	1	0	1	0	1	0	1	0	1	0	1					
	ROLB															59	2	1	A	C b ₁ b ₀																		
Rotate Right	RDR															66	6	2	76	6	3	M	↑	0	1	0	1	0	1	0	1	0	1					
	RORA																46	2	1	A	C b ₁ b ₀																	
	RORB															58	2	1	B																			
Shift Left, Arithmetic	ASL															68	6	2	76	6	3	M	↓	0	1	0	1	0	1	0	1	0	1					
	ASLA																48	2	1	A	C b ₁ b ₀																	
	ASLB															58	2	1	B																			
Double Shift Left, Arithmetic	ASLD																05	3	1	C	↓ ACC _A / ACC _B = 0																	
Shift Right, Arithmetic	ASR															67	6	2	77	6	3	M	↑	0	1	0	1	0	1	0	1	0	1					
	ASRA																47	2	1	A	B ₁ B ₀																	
	ASRB															57	2	1	B																			
Shift Right, Logical	LSR															64	6	2	74	6	3	M	↑	0	1	0	1	0	1	0	1	0	1					
	LSRA																44	2	1	A	B ₁ B ₀																	
	LSRB															54	2	1	B																			
Double SHR Right Logical	LSRD																04	3	1	C	0 → ACC _A / ACC _B = 1																	
	STAA															67	3	2	A7	4	2	B7	4	3	A	→ M												
	STAB															67	3	2	E7	4	2	F7	4	3	B	→ M												
Store Double Accumulator	STAD															DD	4	2	ED	5	2	FD	5	3	A	→ M												
Subtract	SUBA	80	2	2	90	3	2	A0	4	2	B0	4	3																									
	SUBB	CO	2	2	DO	3	2	E0	4	2	FO	4	3																									
Double Subtract	SUBD	63	4	3	93	5	2	A3	6	2	B3	6	3																									
Subtract Accumulators	SBA																	10	2	1	A	- B → A																
Subtract With Carry	SBDA	62	2	2	92	3	2	A2	4	2	B2	4	3																									
	SBDB	C2	2	2	92	3	2	E2	4	2	F2	4	3																									
Transfer Accumulators	TAB																	16	2	1	A	→ B																
	TBA																	17	2	1	B	→ A																
Test Zero or Minus	TSTD															6D	6	2	7D	6	3	M	- 00															
	TSTD																SD	2	1	B	- 00																	

The Condition Code Register notes are listed after Table 10.



MOTOROLA Semiconductor Products Inc.

ADDED INSTRUCTIONS

In addition to the existing MC6800 Instruction Set, the following new instructions are incorporated in the MC6801 Microcomputer.

ABX	Adds the 8-bit unsigned accumulator B to the 16-bit X-Register taking into account the possible carry out of the low order byte of the X-Register.	$IX \leftarrow IX + ACCB$
ACDD	Adds the double precision ACCD* to the double precision value M:M+1 and places the results in ACCD	$ACCD \leftarrow (ACCD) + (M:M+1)$
ASLD	Shifts all bits of ACCAB one place to the left. Bit 0 is loaded with zero. The C bit is loaded from the most significant bit of ACCD.	
LDD	Loads the contents of double precision memory location into the double accumulator A:B. The condition codes are set according to the data.	$ACCD \leftarrow (M:M+1)$
LSRD	Shifts all bits of ACCD one place to the right. Bit 0 is loaded with zero. The C bit is loaded from the least significant bit of ACCD.	
MUL	Multiplies the 8 bits in accumulator A with the 8 bits in accumulator B to obtain a 16-bit unsigned number in A:B. ACCA contains MSB of result.	$ACCD \leftarrow ACCA * ACCB$
PBX	The contents of the index register is pushed onto the stack at the address contained in the stack pointer. The stack pointer is decremented by 2.	$((XL), SP \leftarrow (SP) - 1$ $((XL), SP \leftarrow (SP) - 1$
PULX	The index register is pulled from the stack beginning at the current address contained in the stack pointer +1. The stack pointer is incremented by 2 in total.	$SP \leftarrow (SP) + 1; IXH$ $SP \leftarrow (SP) + 1; IHL$
STD	Stores the contents of double accumulator AB in memory. The contents of ACCD remain unchanged.	$M:M+1 \leftarrow (ACCD)$
SUBD	Subtracts the contents of M:M+1 from the contents of double accumulator AB and places the result in ACCD.	$ACCAB \leftarrow (ACCD) - (M:M+1)$
<i>*ACCD is the 16bit register (A:B) formed by concatenating the A and B accumulators. The A-accumulator is the most significant byte.</i>		

TABLE 8 — INDEX REGISTER AND STACK MANIPULATION INSTRUCTIONS

POINTER OPERATIONS	MNEMONIC	IMMEDIATE						DIRECT	INDEX	EXTEND	IMPLIED	BOOLEAN/ARITHMETIC OPERATION						COND. CODE REG.						
		OP	-	#	OP	-	#					OP	-	#	OP	-	#	H	I	M	Z	V	C	
Compa Index Reg	CPI	BC	4	3	BC	5	2	AC	6	2	BC	6	2	09	3	1	X ₁ - M ₁ X ₀ - (M + 1)							
Decrement Index Reg	DEX													34	3	1	X ₁ - 1 - X ₀							
Decrement Stack Pre	DES													08	3	1	X ₁ - 1 - X ₀							
Increment Index Reg	INC													31	3	1	X ₁ - 1 - X ₀							
Increment Stack Pre	IDS													09	3	1	X ₁ - 1 - X ₀							
Load Index Reg	LDX	DE	3	3	DE	4	2	EE	5	2	FE	5	3											
Load Stack Pre	LDI	EE	3	3	EE	4	2	AE	5	2	DE	5	3											
Store Index Reg	STX	DE	3	3	DE	4	2	EF	5	2	FF	5	3											
Store Stack Pre	STS	EF	5	2	EF	7	2	BF	6	3				35	3	1	X ₁ - 1 - SP							
Index Reg → Stack Pre	TXS													30	3	1	SP - 1 - X ₀							
Stack Pre → Index Reg	TSX													30	2	1	SP - 1 - X ₀							
Ad	ABX													34	2	1	B - 1 - X ₀							
Push Data	PBX													3C	4	1	X ₁ - M ₁ SP - 1 - SP							
Push Data	PULX													39	5	1	SP - 1 - SP, M ₁ - 1 - SP							

The Condition Code Register notes are listed after Table 10



MOTOROLA Semiconductor Products Inc.

TABLE 9 — JUMP AND BRANCH INSTRUCTIONS

OPERATIONS	MNEMONIC	RELATIVE				INDEX				EXTND				IMPLD				COND. CODE REG.
		DP	-	#	OP	-	#	DP	-	#	DP	-	#	DP	-	#	DP	
Branch Always	BRA	20	6	2														1 0 1 1 0 1 1 0
Branch If Carry Clear	BCC	24	6	2														1 0 1 1 0 1 1 0
Branch If Carry Set	BCS	25	6	2														1 0 1 1 0 1 1 0
Branch If = Zero	BEQ	27	6	2														1 0 1 1 0 1 1 0
Branch If > Zero	BGE	2C	6	2														1 0 1 1 0 1 1 0
Branch If >=Zero	BGT	2E	6	2														1 0 1 1 0 1 1 0
Branch If Higher	BHI	22	6	2														1 0 1 1 0 1 1 0
Branch If < Zero	BLE	25	6	2														1 0 1 1 0 1 1 0
Branch If Lower Or Equal	BLS	23	6	2														1 0 1 1 0 1 1 0
Branch If < Zero	BLT	20	6	2														1 0 1 1 0 1 1 0
Branch If Minus	BMI	28	6	2														1 0 1 1 0 1 1 0
Branch If Not Equal Zero	BNE	26	6	2														1 0 1 1 0 1 1 0
Branch If Overflow Set	BVC	28	6	2														1 0 1 1 0 1 1 0
Branch If Plus	BPL	29	6	2														1 0 1 1 0 1 1 0
Branch To Subroutine	BTR	ED	6	2														1 0 1 1 0 1 1 0
Jump	JMP				SE	4	2	IE	3	3								See Special Operations
Add To Subroutine	JSR				AD	8	2	BD	9	3								See Special Operations
No Operation	MOP										01	2	1					AdvancesProg. Curr. Only
Return From Interrupt	RTI										28	19	1					
Return From Subroutine	RTS										29	5	1					
Software Interrupt	SII										36	12	1					
Wait For Interrupt	WAI										36	9	1					

TABLE 10 — CONDITION CODE REGISTER MANIPULATION INSTRUCTIONS

OPERATIONS	MNEMONIC	IMPLD				COND. CODE REG.						
		OP	-	#	EDOLEAR		S	4	3	2	1	0
Clear Carry	CLC	EC	2	1		0-C	H	1	0			
Clear Interrupt Mask	CLI	EE	2	1		0-I		0	0			
Clear Overflow	CLV	EA	2	1		0-V		0	0			
Set Carry	SEC	ED	2	1		1-C		0	0			
Set Interrupt Mask	SEI	EF	2	1		1-I		0	0			
Set Overflow	SEV	EB	2	1		1-V		0	0			
Accumulator A —CCR	TAP	CE	2	1		A-CCR		0	0			
CCR — Accumulator A	TPA	CF	2	1		CCR-A		0	0			

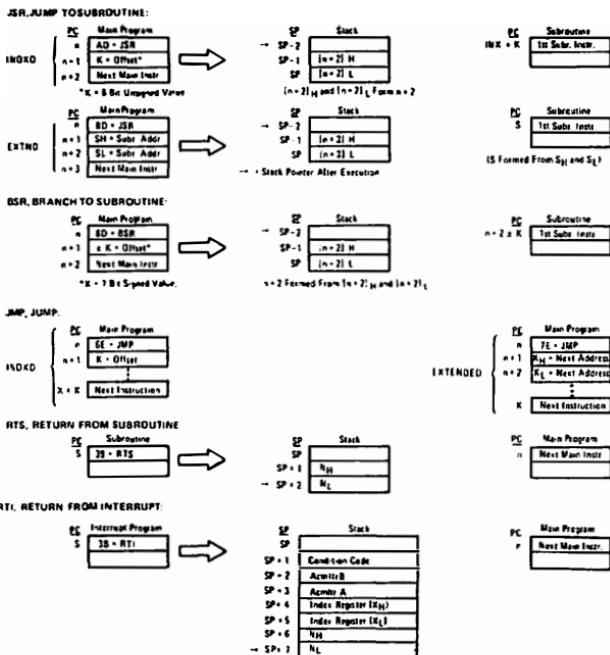
CONDITION CODE REGISTER NOTES: (DP will test as true and cleared otherwise)

1. (B1 R) Test: Result = 10000000¹
2. (B1 C) Test: Result = 0000000P
3. (B1 C) Test: Desired value of most significant ECO Character greater than zero?
(Most cleared if previously set.)
4. (B1 V) Test: Operand = 1000000 prior to subtraction²
5. (B1 R) Test: Operand = 0111111 prior to reduction³
6. (B1 V) Test: Set equal to result of NDC after shift has occurred
7. (B1 R) Test: Sign bit of most significant (MS) byte = 1?
8. (B1 V) Test: 2's complement overflow from subtraction of MS bytes?
9. (B1 R) Test: Result less than just 1010 15 - 1?
10. (A2) Load Condition Code Register from Stack. (See Special Operations)
Set when interrupt occurs. If previous set, a Non Maskable Interrupt is required to invert the next state.
11. (B1 I) Set according to the contents of Accumulator A
12. (A1) Set according to the contents of Accumulator A



MOTOROLA Semiconductor Products Inc.

FIGURE 27 — SPECIAL OPERATIONS



MOTOROLA Semiconductor Products Inc.

TABLE 11 — INSTRUCTION EXECUTION TIMES IN
MACHINE CYCLE

	ACCR	Immediate	Direct	Extended	ACCR	Immediate	Direct	Extended	Indexed	Inherent	Relative	ACCR	Immediate	Direct	Extended	Indexed	Inherent	Relative
ABA	•	•	•	•	•	•	•	•	•	•	•	INX	•	•	•	•	•	•
ABX	•	•	•	•	•	•	•	•	•	•	•	JMP	•	•	•	•	•	•
ADC	•	•	•	•	•	•	•	•	•	•	•	JSR	•	•	•	•	•	•
ADD	•	•	•	•	•	•	•	•	•	•	•	LDA	•	•	•	•	•	•
ADDD	•	•	•	•	•	•	•	•	•	•	•	LDD	•	•	•	•	•	•
AND	•	•	•	•	•	•	•	•	•	•	•	LDS	•	•	•	•	•	•
ASL	2	•	•	•	•	•	•	•	•	•	•	LDX	•	•	•	•	•	•
ASLD	•	•	•	•	•	•	•	•	•	•	•	LSR	•	•	•	•	•	•
ASR	2	•	•	•	•	•	•	•	•	•	•	LSRD	•	•	•	•	•	•
BCC	•	•	•	•	•	•	•	•	•	•	•	MUL	•	•	•	•	•	•
BCS	•	•	•	•	•	•	•	•	•	•	•	NEG	•	•	•	•	•	•
BDI	•	•	•	•	•	•	•	•	•	•	•	NOP	•	•	•	•	•	•
BGE	•	•	•	•	•	•	•	•	•	•	•	ORA	3	2	3	4	5	3
BGT	•	•	•	•	•	•	•	•	•	•	•	PSH	3	2	3	4	5	3
BHI	•	•	•	•	•	•	•	•	•	•	•	PSHX	3	2	3	4	5	3
BIT	•	2	•	•	•	•	•	•	•	•	•	PUL	4	2	3	4	5	3
BLE	•	•	•	•	•	•	•	•	•	•	•	PULX	3	2	3	4	5	3
BLS	•	•	•	•	•	•	•	•	•	•	•	ROL	3	2	3	4	5	3
BLT	•	•	•	•	•	•	•	•	•	•	•	ROR	3	2	3	4	5	3
BMI	•	•	•	•	•	•	•	•	•	•	•	RTI	3	2	3	4	5	3
BNE	•	•	•	•	•	•	•	•	•	•	•	RTS	3	2	3	4	5	3
BPL	•	•	•	•	•	•	•	•	•	•	•	SBA	3	2	3	4	5	3
BRA	•	•	•	•	•	•	•	•	•	•	•	SBC	3	2	3	4	5	3
BSR	•	•	•	•	•	•	•	•	•	•	•	SEC	6	4	5	6	7	5
BVC	•	•	•	•	•	•	•	•	•	•	•	SEI	3	2	3	4	5	3
BVS	•	•	•	•	•	•	•	•	•	•	•	SEV	3	2	3	4	5	3
CBA	•	•	•	•	•	•	•	•	•	•	•	STA	3	2	3	4	5	3
CLC	•	•	•	•	•	•	•	•	•	•	•	STD	4	3	4	5	6	4
CLI	•	•	•	•	•	•	•	•	•	•	•	STS	4	3	4	5	6	4
CLR	2	•	•	•	•	•	•	•	•	•	•	STX	4	3	4	5	6	4
CLV	•	2	•	•	•	•	•	•	•	•	•	SUB	3	2	3	4	5	3
CMP	•	2	3	•	•	•	•	•	•	•	•	SUBD	4	3	4	5	6	4
COM	2	•	•	•	•	•	•	•	•	•	•	SWI	•	•	•	•	•	•
CPX	•	4	5	•	•	•	•	•	•	•	•	TAB	•	•	•	•	•	•
DAA	•	4	5	6	•	•	•	•	•	•	•	TAP	•	•	•	•	•	•
DEC	2	•	•	•	•	•	•	•	•	•	•	TBA	2	•	•	•	•	•
DES	•	•	•	•	•	•	•	•	•	•	•	TPA	•	•	•	•	•	•
DEX	•	•	•	•	•	•	•	•	•	•	•	TST	•	•	•	•	•	•
EOR	•	2	•	•	•	•	•	•	•	•	•	TSX	•	•	•	•	•	•
INC	2	•	2	3	•	•	•	•	•	•	•	WAI	•	•	•	•	•	•



MOTOROLA Semiconductor Products Inc.

Summary of Cycle by Cycle Operation

Table 12 provides a detailed description of the information present on the Address Bus, Data Bus, and the Read/Write line (R/W) during each cycle for each instruction.

This information is useful in comparing actual with expected results during debug of both software and hardware as the control program is executed. The information is categorized in groups according to addressing mode and number of cycles per instruction. (In general, instructions with the same addressing mode and number of cycles execute in the same manner; exceptions are indicated in the table).

TABLE 12 — CYCLE BY CYCLE OPERATION

ADDRESS MODE & INSTRUCTIONS	CYCLE #	CYCLE #	ADDRESS BUS	R/W LINE	DATA BUS
IMMEDIATE					
ADC EOR	2	1	Op Code Address	1	Op Code
ADD LDA		2	Op Code Address + 1	1	Operand Data
AND ORA			Op Code Address + 2		
BIT SBC					
CMP SUB					
LDS	3	1	Op Code Address	1	Op Code
LDX		2	Op Code Address + 1	1	Operand Data (High Order Byte)
		3	Op Code Address + 2	1	Operand Data (Low Order Byte)
CPX	4	1	Op Code Address	1	Op Code
SUBD		2	Op Code Address + 1	1	Operand Data (High Order Byte)
ADDD		3	Op Code Address + 2	1	Operand Data (Low Order Byte)
		4	Address Bus FFFF	1	Low Byte of Restart Vector
DIRECT					
ADC EOR	3	1	Op Code Address	1	Op Code
ADD LDA		2	Op Code Address + 1	1	Address of Operand
AND ORA		3	Address of Operand	1	Operand Data
BIT SBC					
CMPSUB					
STA	3	1	Op Code Address	1	Op Code
		2	Op Code Address + 1	1	Destination Address
		3	Destination Address	0	Data from Accumulator
LDS	4	1	Op Code Address	1	Op Code
LDX		2	Op Code Address + 1	1	Address of Operand
LDD		3	Address of Operand	1	Operand Data (High Order Byte)
		4	Operand Address + 1	1	Operand Data (Low Order Byte)
STS	4	1	Op Code Address	1	Op Code
STX		2	Op Code Address + 1	1	Address of Operand
STD		3	Address of Operand	0	Register Data (High Order Byte)
		4	Address of Operand + 1	0	Register Data (Low Order Byte)
CPX	5	1	Op Code Address	1	Op Code
SUBD		2	Op Code Address + 1	1	Address of Operand
ADDD		3	Operand Address	1	Operand Data (High Order Byte)
		4	Operand Address + 1	1	Operand Data (Low Order Byte)
		5	Address Bus FFFF	1	Low Byte of Restart Vector
JSR	5	1	Op Code Address	1	Op Code
		2	Op Code Address + 1	1	Irrelevant Data
		3	Subroutine Address	1	First Subroutine Op Code
		4	Stack Pointer	0	Return Address (Low Order Byte)
		5	Stack Pointer + 1	0	Return Address (High Order Byte)

(continued)



MOTOROLA Semiconductor Products Inc.

TABLE 12 — CYCLE BY CYCLE OPERATION
(cont)

ADDRESS MODE & INSTRUCTIONS	CYCLES	CYCLE #	ADDRESS BUS	R/W LINE	DATA BUS
INDEXED					
JMP	3	1	Op Code Address	1	Op Code
		2	Op Code Address + 1	1	Offset
		3	Address Bus FFFF	1	Low Byte of Restart Vector
ADC EOR	4	1	Op Code Address	1	Op Code
ADD LDA		2	Op Code Address + 1	1	Offset
AND ORA		3	Address Bus FFFF	1	Low Byte of Restart Vector
BIT SBC		4	Index Register Plus Offset	1	OperandData
CMP SUB					
STA	4	1	Op Code Address	1	Op Code
		2	Op Code Address + 1	1	Offset
		3	Address Bus FFFF	1	Low Byte of Restart Vector
		4	Index Register Plus Offset	0	Operand Data
LDS	5	1	Op Code Address	1	Op Code
LDX		2	Op Code Address + 1	1	Offset
LDD		3	Address Bus FFFF	1	Low Byte of Restart Vector
		4	Index Register Plus Offset	1	Operand Data (High Order Byte)
		5	Index Register Plus Offset + 1	1	Operand Data (Low Order Byte)
STS	5	1	Op Code Address	1	Op Code
STX		2	Op Code Address + 1	1	Offset
STD		3	Address Bus FFFF	1	Low Byte of Restart Vector
		4	Index Register Plus Offset	0	Operand Data (High Order Byte)
		5	Index Register Plus Offset + 1	0	Operand Data (Low Order Byte)
ASLLSR	6	1	Op Code Address	1	Op Code
ASR NEG		2	Op Code Address + 1	1	Offset
CLR ROL		3	Address Bus FFFF	1	Low Byte of Restart Vector
COM ROR		4	Index Register Plus Offset	1	Current Operand Data
DEC TST (1)		5	Address Bus FFFF	1	Low Byte of Restart Vector
INC		6	Index Register Plus Offset	0	New Operand Data
CPX	6	1	Op Code Address	1	Op Code
SUBD		2	Op Code Address + 1	1	Offset
ADD		3	Address Bus FFFF	1	Low Byte of Restart Vector
		4	Index Register + Offset	1	Operand Data (High Order Byte)
		5	Index Register + Offset + 1	1	Operand Data (Low Order Byte)
		6	Address Bus FFFF		Low Byte of Restart Vector
JSR	6	1	Op Code Address	1	Op Code
		2	Op Code Address + 1	1	Offset
		3	Address Bus FFFF	1	Low Byte of Restart Vector
		4	Index Register + Offset	1	First Subroutine Op Code
		5	Stack Pointer	0	Return Address (Low Order Byte)
		6	Stack Pointer - 1	0	Return Address (High Order Byte)

(continued)



MOTOROLA Semiconductor Products Inc.

TABLE 12 — CYCLE BY CYCLE OPERATION
(cont)

ADDRESS MODE & INSTRUCTIONS	CYCLES	CYCLE #	ADDRESS BUS	R/W LINE	DATA BUS
EXTENDED					
JMP	3	1	Op Code Address	1	Op Code
		2	Op Code Address + 1	1	Jump Address (High Order Byte)
		3	Op Code Address + 2	1	Jump Address (Low Order Byte)
ADC EOR	4	1	Op Code Address	1	Op Code
ADD LOA		2	Op Code Address + 1	1	Address of Operand
ANOORA		3	Op Code Address + 2	1	Address of Operand (Low Order Byte)
BIT SBC		4	Address of Operand	1	Operand Data
CMP SUB					
STA A	4	1	Op Code Address	1	Op Code
STA B		2	Op Code Address + 1	1	Destination Address (High Order Byte)
		3	Op Code Address + 2	1	Destination Address (Low Order Byte)
		4	Operand Destination Address	0	Data from Accumulator
LOS	5	1	Op Code Address	1	Op Code
LOX		2	Op Code Address + 1	1	Address of Operand (High Order Byte)
LDD		3	Op Code Address + 2	1	Address of Operand (Low Order Byte)
		4	Address of Operand	1	Operand Data (High Order Byte)
		5	Address of Operand + 1	1	Operand Data (Low Order Byte)
STS	5	1	Op Code Address	1	Op Code
STX		2	Op Code Address + 1	1	Address of Operand (High Order Byte)
STD		3	Op Code Address + 2	1	Address of Operand (Low Order Byte)
		4	Address of Operand	0	Operand Data (High Order Byte)
		5	Address of Operand + 1	0	Operand Data (Low Order Byte)
ASLLSR	6	1	Op Code Address	1	Op Code
ASRNEG		2	Op Code Address + 1	1	Address of Operand (High Order Byte)
CLR ROL		3	Op Code Address + 2	1	Address of Operand (Low Order Byte)
COM ROR		4	Address of Operand	1	Current Operand Data
OEC TST (1)		5	Address Bus FFFF	1	Low Byte of Restart Vector
INC		6	Address of Operand	0	New Operand Data
CPX	6	1	Op Code Address	1	Op Code
SUBO		2	Op Code Address + 1	1	Operand Address (High Order Byte)
ADOO		3	Op code Address + 2	1	Operand Address (Low Order Byte)
		4	Operand Address	1	Operand Data (High Order Byte)
		5	Operand Address + 1	1	Operand Data (Low Order Byte)
		6	Address Bus FFFF	1	Low Byte of Restart Vector
JSR	6	1	Op Code Address	1	Op Code
		2	Op Code Address + 1	1	Address of Subroutine (High Order Byte)
		3	Op Code Address + 2	1	Address of Subroutine (Low Order Byte)
		4	Subroutine Starting Address	1	Op Code of Next Instruction
		5	Stack Pointer	0	Return Address (Low Order Byte)
		6	Stack Pointer - 1	0	Address of Operand (High Order Byte)

(continued)



MOTOROLA Semiconductor Products Inc.

TABLE 12 — CYCLE BY CYCLE OPERATION
(cont)

ADDRESS MODE & INSTRUCTIONS	CYCLES	CYCLES #	ADDRESS BUS	R/W LINE	DATA BUS
INHERENT					
ABA DAA SEC	2	1	Op Code Address	1	Op Code
ASL DEC SEI		2	Op Code Address +1	1	Op Code of Next Instruction
ASR INC SEV					
CBALSR TAB					
CLC NEG TAP					
CLI NOPTBA					
CLR ROL TPA					
CLV ROR TST					
COM SBA					
ABX	3	1	Op Code Address	1	Op Code
		2	Op Code Address +1	1	Irrelevant Data
		3	Address Bus FFFF	1	Low Byte of Restart Vector
ASLD	3	1	Op Code Address	1	Op Code
LSRD		2	Op Code Address +1	1	Irrelevant Data
		3	Address Bus FFFF	1	Low Byte of Restart Vector
DES	3	1	Op Code Address	1	Op Code
INS		2	Op Code Address +1	1	Op Code of Next Instruction
		3	Previous Register Contents	1	Irrelevant Data
INX	3	1	Op Code Address	1	Op Code
DEX		2	Op Code Address +1	1	Op Code of Next Instruction
		3	Address Bus FFFF	1	Low Byte of Restart Vector
PSHA	3	1	Op Code Address	1	Op Code
PSHB		2	Op Code Address +1	1	Op Code of Next Instruction
		3	Stack Pointer	0	Accumulator Data
ISX	3	1	Op Code Address	1	Op Code
		2	Op Code Address +1	1	Op Code of Next Instruction
		3	Stack Pointer	1	Irrelevant Data
TXS	3	1	Op Code Address	1	Op Code
		2	Op Code Address +1	1	Op Code of Next Instruction
		3	Address Bus FFFF	1	Low Byte of Restart Vector
PULA	4	1	Op Code Address	1	Op Code
PULB		2	Op Code Address +1	1	Op Code of Next Instruction
		3	Stack Pointer	1	Irrelevant Data
		4	Stack Pointer +1	1	
PSHX	4	1	Op Code Address	1	Op Code
		2	Op Code Address +1	1	Irrelevant Data
		3	Stack Pointer	0	Index Register (Low Order Byte)
		4	Stack Pointer -1	0	Index Register (High Order Byte)
PULX	5	1	Op Code Address	1	Op Code
		2	Op Code Address +1	1	Irrelevant Data
		3	Stack Pointer	1	Irrelevant Data
		4	Stack Pointer +1	1	Index Register (High Order Byte)
		5	Stack Pointer -2	1	Index Register (Low Order Byte)
BCC BHT BNE	3	1	Op Code Address	1	Op Code
BCS BLE BPL		2	Op Code Address +1	1	Branch Offset
BEQ BLS BRA		3	Address Bus FFFF	1	Low Byte of Restart Vector
BGE BLT BVC					
BGT BMT BVS					
BSR	6	1	Op Code Address	1	Op Code
		2	Op Code Address +1	1	Branch Offset
		3	Address Bus FFFF	1	Low Byte of Restart Vector
		4	Subroutine Starting Address	1	Op Code of Next Instruction
		5	Stack Pointer	0	Return Address (Low Order Byte)
		6	Stack Pointer -1	0	Return Address (High Order Byte)



MOTOROLA Semiconductor Products Inc.

FIGURE 28 — MC6801 MCU SINGLE-CHIP DUAL PROCESSOR CONFIGURATION

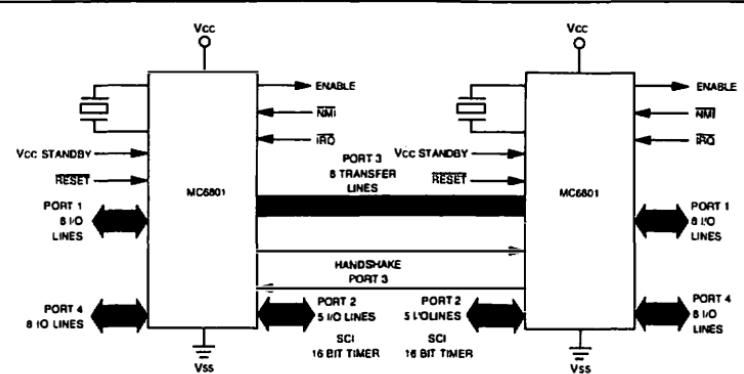


FIGURE 29 — MC6801 MCU EXPANDED NON-MULTIPLEXED MODE

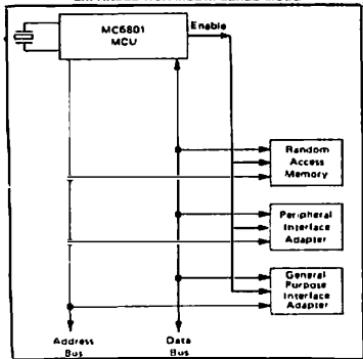
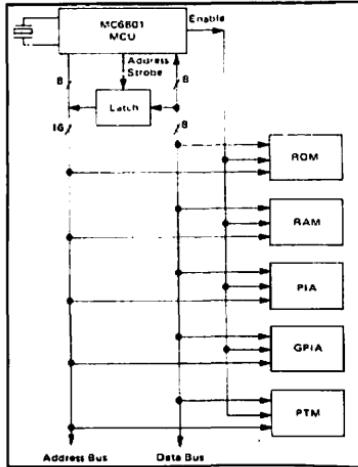
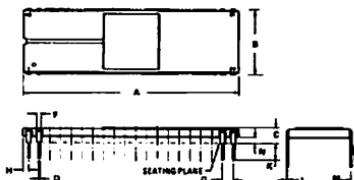
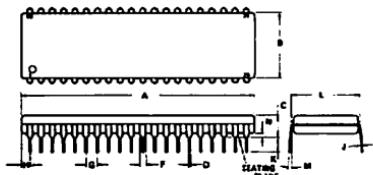


FIGURE 30 — MC6801 MCU EXPANDED MULTIPLEXED MODE



MOTOROLA Semiconductor Products Inc.

OUTLINE DIMENSIONS



	INCHES/MILLIMETERS	INCHES	MILLIMETERS	INCHES	MILLIMETERS	
ONE	MIN	MAX	MIN	MAX	MIN	MAX
A	.502	.512	12.70	12.95		
B	.154	.158	3.91	4.01		
C	.154	.158	3.91	4.01		
D	.03	.030	.76	.76		
E	.154	.158	3.91	4.01		
F	.154	.158	3.91	4.01		
G	.154	.158	3.91	4.01		
H	.154	.158	3.91	4.01		
I	.154	.158	3.91	4.01		
J	.154	.158	3.91	4.01		
K	.154	.158	3.91	4.01		
L	.154	.158	3.91	4.01		
M	.041	.152	1.04	1.09		

L DUMPFIX
CERAMIC PACKAGE
CASE 718-02

NOTE:
1. LEADS TRUE POSITION IS WITHIN .005 MM TO .015 MM AT
.050 MM (.002 IN.) (AT SEATING
PLANE), AT MAX. RATE,
CONDITION.

	INCHES	MILLIMETERS	INCHES	MILLIMETERS
A	.175	4.45	.165	4.19
B	.154	3.91	.154	3.91
C	.154	3.91	.154	3.91
D	.154	3.91	.154	3.91
E	.154	3.91	.154	3.91
F	.154	3.91	.154	3.91
G	.154	3.91	.154	3.91
H	.154	3.91	.154	3.91
I	.154	3.91	.154	3.91
J	.154	3.91	.154	3.91
K	.154	3.91	.154	3.91
L	.154	3.91	.154	3.91
M	.041	.152	.025	.64

P DUMPFIX
PLASTIC PACKAGE
CASE 711-02

- NOTES
1. LEADS TRUE POSITIONED
WITHIN .015 MM TO .025 MM AT
LEAD POSITION AND MAXIMUM
MATERIAL CONDITION
(DIM "D")
 2. DIM "L" TO CENTER OF LEADS
WHEN FORMED PARALLEL

Motorola reserves the right to make changes to any products herein to improve reliability, function or design. Although the information in this document has been carefully reviewed for broad application, Motorola does not assume any liability resulting 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



MOTOROLA SEMICONDUCTORS

3501 ED BLUESTEIN BLVD., AUSTIN, TEXAS 78721

Product Preview

MICROCOMPUTER EVALUATION CHIP FEATURES

- Three Modes of Operation — Single Chip, Expanded Non-Multiplexed, and Expanded Multiplexed
- Internal 2K ROM Programmed with a Debug Monitor Program
- Communicates with a Terminal Through the Serial Port

DESCRIPTION

The MC6801L1 is a complete microcomputer on a chip and can be operational with a minimum of external components. It may be used in either the single chip or extended mode. With the Debug Monitor Program the user can debug a program under development and evaluate the different I/O configuration and modes of the MC6801. The 6801 firmware enables the user to:

- Load a Program from Tape
- Dump the Program
- Verify That a Program Was Properly Loaded
- Examine and Change Data in a Memory Location
- Punch (Record) the Program on Tape
- Calculate the Offset for Relative Addressing
- Examine and Change Data in the User's Program Registers and Counter
- Insert, Display, and Remove Breakpoints in the Program
- Free-Run or Trace Through the User's Program

The firmware uses the serial port on the 6801 to provide input, output communications and connect to a tape or digital cassette within the terminal. The I/O functions of the firmware are down-loaded into the RAM to enable the user to change these parameters.

The MC6801L1 is used in the MC6801 Microcomputer Evaluation Module. (See preliminary information on the MEX6801EV1.)

MC6801L1 MC6801P1

MOS

(N-CHANNEL, SILICON-GATE
DEPLETION LOAD)

MICROCOMPUTER EVALUATION CHIP

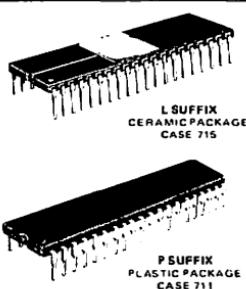
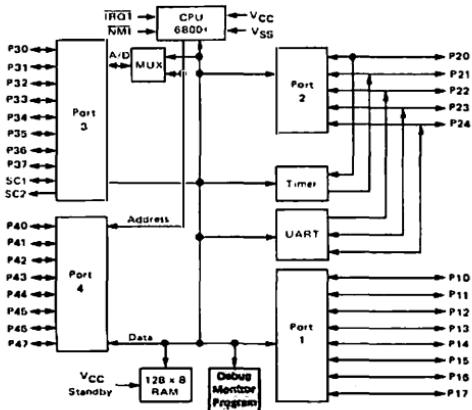
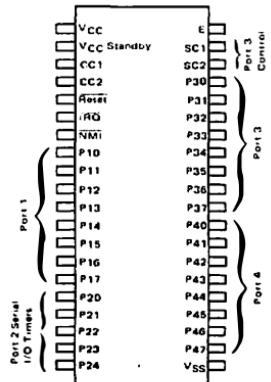


FIGURE 1 — SINGLE CHIP MICROCOMPUTER BLOCK DIAGRAM



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

FIGURE 2 — MC6801



© MOTOROLA INC., 1978

NP 95



MOTOROLA

SEMICONDUCTORS

3501 ED BLUESTEIN BLVD., AUSTIN, TEXAS 78721

Advance Information

The MC6803 is an 8-bit microcomputer which employs a multiplexed address and data system, allowing expandability to 64K words. The MC6803 is object code compatible with the M6800 instruction set and includes improved execution times of key instructions. There are several new 16-bit and 8-bit instructions including an 8 by 8 multiply with 16-bit result. The MC6803 has 128 bytes of RAM, internal clock, SCI, parallel I/O, and three function 16-bit timer all on-board. The MC6803 requires only the addition of a ROM and an external crystal for MCU operation. The MC6803 internal clock's divide by four circuitry allows for use of the inexpensive 3.58 MHz color-burst crystal. The MC6803 MCU is fully TTL compatible and requires only one +5.0 volt power supply. An external RAM is needed with the MC6803 NR.

- Expanded M6800 Instruction Set
- Full Object Code Compatibility With M6800 MPU's
- Multiplexed Address and Data
- Compatible With Existing M6800 Peripherals
- 8 X 8 Multiply With 16-bit Result
- Up to 13 Parallel I/O Lines
- 128 Bytes On-Board RAM on MC6803
- On-Board RAM Retainable With V_{CC} Standby
- Serial Communications Interface On-Board
- 16-bit Timer On-Board
- Internal Clock/Divide by Four Circuitry
- Full TTL Compatibility
- Full Interrupt Capability

MC6803
MC6803NR
INO RAMI

MOS

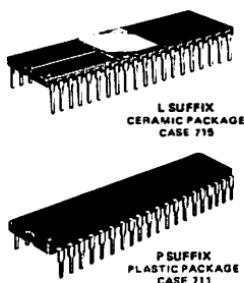
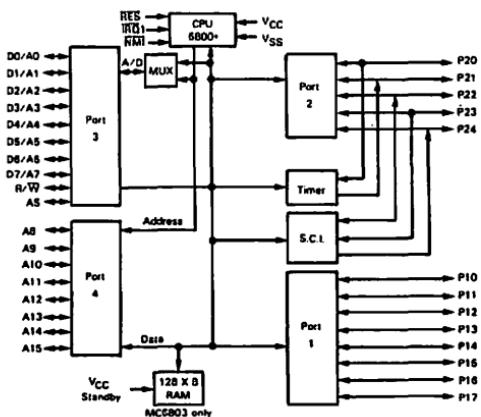
(IN-CHANNEL, SILICON-GATE,
DEPLETION LOAD)MICROPROCESSOR
WITH I/O FEATURES

FIGURE 1 — BLOCK DIAGRAM



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

FIGURE 2 — PIN ASSIGNMENT

Vss	1	40	E
XTAL1	2	38	A5
EXTAL2	3	38	R/7
RMI	4	37	00/A0
IRQ1	5	36	01/A1
RESET	6	35	02/A2
V_{CC}	7	34	03/A3
P20	8	33	04/A4
P21	9	MC6803	32
P22	10		05/A5
P23	11		06/A6
P24	12		07/A7
P10	13		08/A8
P11	14		09/A9
P12	15		10/A10
P13	16		11/A11
P14	17		12/A12
P15	18		13/A13
P16	19		14/A14
P17	20		15/A15
			V_{CC} Standby

© MOTOROLA INC. 1979

ADI-800



MOTOROLA
Semiconductors

BOX 20912 • PHOENIX, ARIZONA 85067

**MC1508L-8
MC1408L-8
MC1408L-7
MC1408L-6**

Specifications and Applications Information

EIGHT-BIT MULTIPLYING DIGITAL-TO-ANALOG CONVERTER

... designed for use where the output current is a linear product of an eight-bit digital word and an analog input voltage.

- Relative Accuracy: $\pm 0.1\%$ Error maximum (MC1508L-8, MC1408L-8)
- Seven and Six-Bit Accuracy Available (MC1408L-7, MC1408L-6)
- Fast Settling Time - 300 ns typical
- Noninverting Digital Inputs are MTTL and CMOS Compatible
- Output Voltage Swing - +0.4 V to -5.0 V
- High-Speed Multiplying Input Slew Rate 4.0 mA/μs
- Standard Supply Voltages: +5.0 V and -5.0 V to -15 V

EIGHT-BIT MULTIPLYING DIGITAL-TO-ANALOG CONVERTER

SILICON MONOLITHIC INTEGRATED CIRCUIT

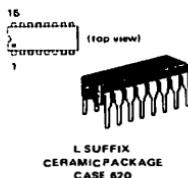


FIGURE 1 - D-to-A TRANSFER CHARACTERISTICS

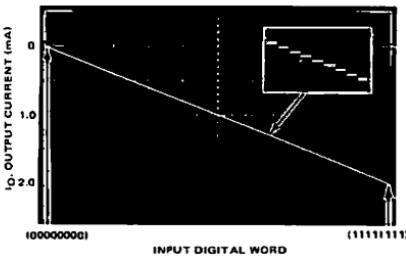
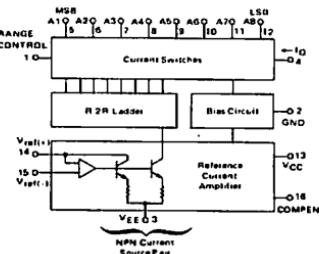


FIGURE 2 - BLOCK DIAGRAM



TYPICAL APPLICATIONS

- Tracking A-to-D Converters
- Successive Approximation A-to-D Converters
- 2 1/2 Digit Panel Meters and DVM's
- Waveform Synthesis
- Sample and Hold
- Peak Detector
- Programmable Gain and Attenuation
- CRT Character Generation
- Audio Digitizing and Decoding
- Programmable Power Supplies
- Analog-Digital Multiplication
- Digital-Digital Multiplication
- Analog-Digital Division
- Digital Addition and Subtraction
- Speech Compression and Expansion
- Stepping Motor Drive

MTTL is a trademark of Motorola Inc.

© MOTOROLA INC., 1975

DS 9238 R1

(MC1508 - Page 1)

MAXIMUM RATINGS ($T_A = +25^\circ\text{C}$ unless otherwise noted.)

Rating	Symbol	Value	Unit
Power Supply Voltage	V_{CC}	+5.5	Vdc
	V_{EE}	-16.5	
Digital Input Voltage	V_5 thru V_{12}	0 to +5.5	Vdc
Applied Output Voltage	V_O	+0.5 to -5.2	Vdc
Reference Current	I_{R14}	5.0	mA
Reference Amplifier Inputs	V_{14}, V_{15}	$V_{CC} - V_{EE}$	Vdc
Operating Temperature Range	T_A	-55 to +125 0 to +75	$^\circ\text{C}$
MC1508L8 MC1408L Series			
Storage Temperature Range	T_{STG}	-65 to +150	$^\circ\text{C}$

ELECTRICAL CHARACTERISTICS ($V_{CC} = +5.0\text{Vdc}$, $V_{EE} = -15\text{Vdc}$, $R_{14} = 2.0\text{mA}$, MC1508L8: $T_A = -55^\circ\text{C}$ to $+125^\circ\text{C}$, MC1408L Series: $T_A = 0$ to $+75^\circ\text{C}$ unless otherwise noted. All digital inputs at high logic level.)

Characteristic	Figure	Symbol	Min	Typ	Max	Unit
Relative Accuracy (Error relative to full scale (I_O)) MC1508L8, MC1408L8 MC1408L7, See Note 1 MC1408L6, See Note 1	4	E_r	-	-	± 0.19 ± 0.39 ± 0.78	%
Settling Time to within $\pm 1/2$ LSB (includes τ_{PLH} at $T_A = +25^\circ\text{C}$) See Note 2	5	t_S	-	300	-	ns
Propagation Delay Time $T_A = +25^\circ\text{C}$	5	$\tau_{PLH} - \tau_{PHL}$	-	30	100	ns
Output Full Scale Current Drift		TC_{IO}	-	-20	-	PPM/ $^\circ\text{C}$
Digital Input Logic Levels (MSB) High Level, Logic "1" Low Level, Logic "0"	3	V_{IH} V_{IL}	2.0 - -	-	0.8	Vdc
Digital Input Current (MSB) High Level, $V_{IH} = 5.0\text{V}$ Low Level, $V_{IL} = 0.8\text{V}$	3	I_{IH} I_{IL}	- -	0 -0.4	0.04 -0.8	mA
Reference Input Bias Current (Pin 15)	3	I_{I5}	-	-1.0	-5.0	μA
Output Current Range $V_{EE} = -5.0\text{V}$ $V_{EE} = -15\text{V}$, $T_A = 25^\circ\text{C}$	3	I_{OR}	0 0	2.0 2.0	2.1 4.2	mA
Output Current $V_{ref} = 2.000\text{V}$, $R_{14} = 1000\Omega$	3	I_O	1.9	1.99	2.1	mA
Output Current (All bits low)	3	$I_O(\text{min})$	-	0	4.0	μA
Output Voltage Compliance ($ E_r \leq 0.10\%$ at $T_A = +25^\circ\text{C}$) Pin 1 grounded Pin 1 open, V_{EE} below -10V	3	V_O	- -	-	-0.55, +0.4 -5.0, +0.4	Vdc
Reference Current Slew Rate	6	$SR I_{ref}$	-	4.0	-	$\text{mA}/\mu\text{s}$
Output Current Power Supply Sensitivity		$PSRR_{II-I}$	-	0.5	2.7	$\mu\text{A}/\text{V}$
Power Supply Current (All bits low)	3	I_{CC} I_{EE}	- -	+13.5 -7.5	+22 -13	mA
Power Supply Voltage Range ($T_A = +25^\circ\text{C}$)	3	V_{CCR} V_{EER}	+4.5 -4.5	+5.0 -15	+5.5 -16.5	Vdc
Power Dissipation All bits low, $V_{EE} = -5.0\text{Vdc}$ $V_{EE} = -15\text{Vdc}$ All bits high $V_{EE} = -5.0\text{Vdc}$ $V_{EE} = -15\text{Vdc}$	3	P_D	- - -	105 190 160	170 305 -	mW

Note 1. All current switches are tested to guarantee at least 50% of rated output current.

Note 2. All bits switched.



MOTOROLA Semiconductor Products Inc.

(MC1508 - Page 2)

TEST CIRCUITS

FIGURE 3 – NOTATION DEFINITIONS TEST CIRCUIT

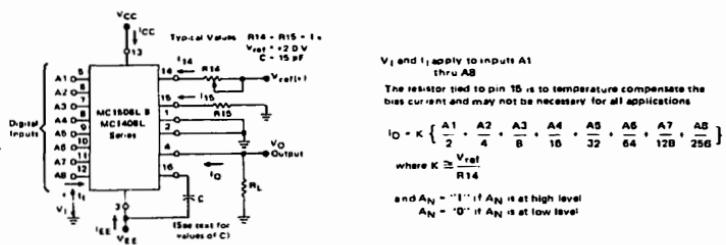


FIGURE 4 – RELATIVE ACCURACY TEST CIRCUIT

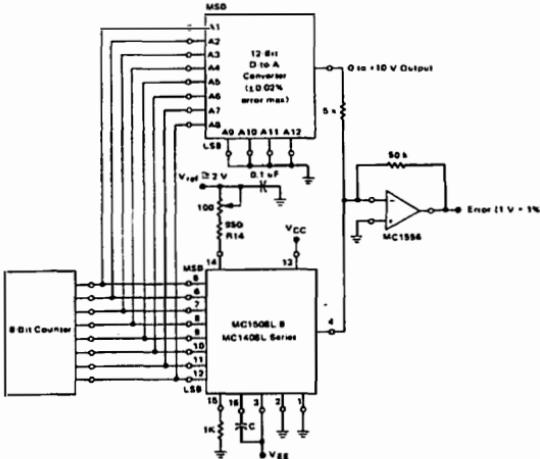
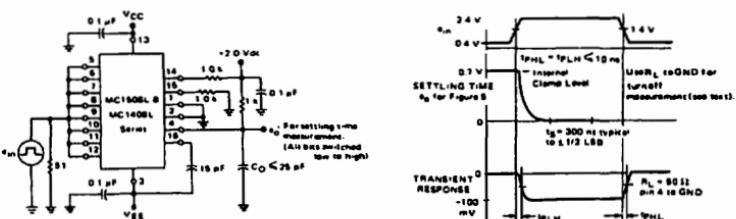


FIGURE 5 – TRANSIENT RESPONSE and SETTLING TIME

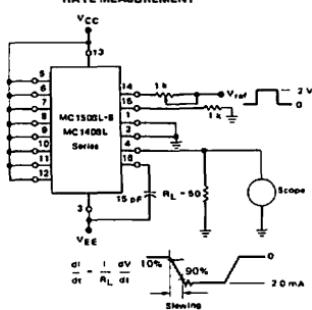
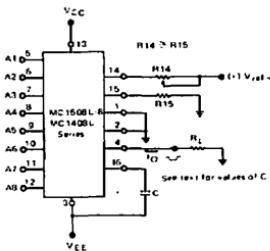
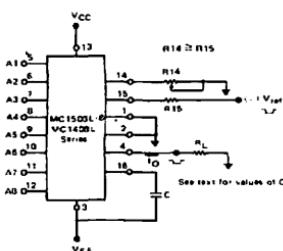


MOTOROLA Semiconductor Products Inc.

(MC1500 – Page 3)

TEST CIRCUITS (continued)

FIGURE 6 - REFERENCE CURRENT SLEW RATE MEASUREMENT

FIGURE 7 - POSITIVE V_{ref}FIGURE 8 - NEGATIVE V_{ref}

THERMAL INFORMATION

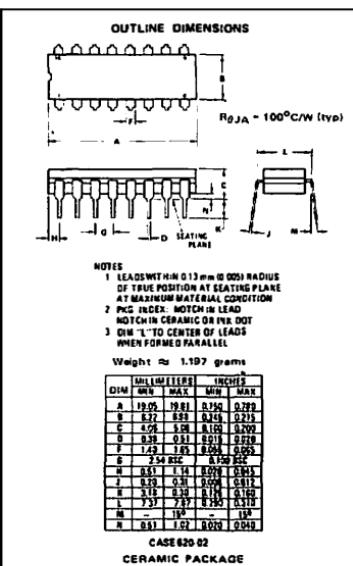
The maximum power consumption an integrated circuit can tolerate at a given operating ambient temperature, can be found from the equation:

$$P_{D(T_A)} = \frac{T_{J(\max)} - T_A}{R_{JJA}(T_{Vd})}$$

Where: $P_{D(T_A)}$ = Power Dissipation allowable at a given operating ambient temperature. This must be greater than the sum of the products of the supply voltages and supply currents at the worst case operating condition.

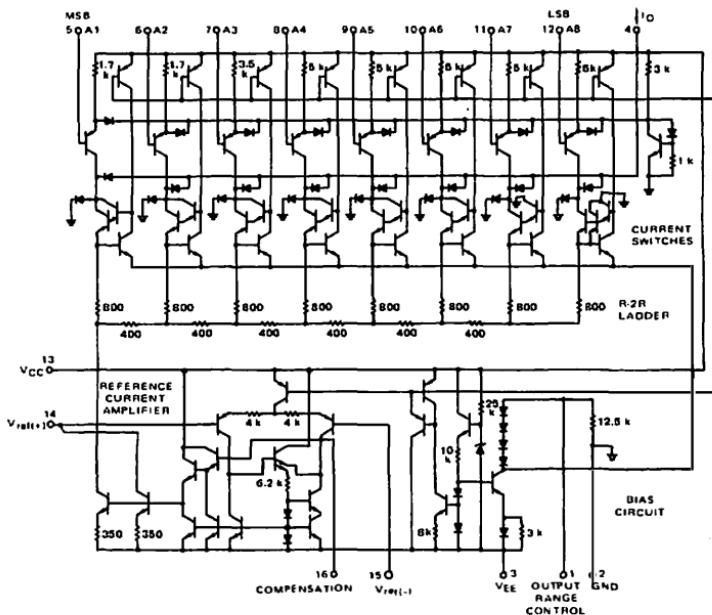
$T_{J(\max)}$ = Maximum Operating Junction Temperature as listed in the Maximum Ratings Section
 T_A = Maximum Desired Operating Ambient Temperature

$R_{JJA}(T_{Vd})$ = Typical Thermal Resistance Junction to Ambient



MOTOROLA Semiconductor Products Inc.

FIGURE 9 – MC1508L-8/MC1408L SERIES EQUIVALENT
CIRCUIT SCHEMATIC
DIGITAL INPUTS



CIRCUIT DESCRIPTION

The MC1508L-8 consists of a reference current amplifier, an R-2R ladder, and eight high-speed current switches. For many applications, only a reference resistor and reference voltage need be added.

The switches are noninverting in operation, therefore a high state on the input turns on the specified output current component. The switch uses current steering for high speed, and a termination amplifier consisting of an active load gain stage with unity gain feedback. The termination amplifier holds the parasitic capacitance of the ladder at a constant voltage during switching, and provides

a low impedance termination of equal voltage for all legs of the ladder.

The R-2R ladder divides the reference amplifier current into binary related components, which are fed to the switches. Note that there is always a remainder current which is equal to the least significant bit. This current is shunted to ground, and the maximum output current is 255/256 of the reference amplifier current, or 1992 mA for a 20 mA reference amplifier current if the NPN current source pair is perfectly matched.



MOTOROLA Semiconductor Products Inc.

(MC1508 - Page 5)

GENERAL INFORMATION

Reference Amplifier Drive and Compensation

The reference amplifier provides a voltage at pin 14 for converting the reference voltage to a current, and a turn-around circuit or current mirror for feeding the ladder. The reference amplifier input current, I_{14} , must always flow into pin 14 regardless of the ratio method or reference voltage polarity.

Connections for positive reference voltage are shown in Figure 7. The reference voltage source supplies the full current I_{14} . For bipolar reference signals, as in the multiplying mode, R15 can be tied to a negative voltage corresponding to the minimum input level. It is possible to eliminate R15 with only a small sacrifice in accuracy and temperature drift. Another method for bipolar inputs is shown in Figure 26.

The compensation capacitor value must be increased with increases in R_{14} to maintain proper phase margin; for R_{14} values of 1.0, 2.5 and 5.0 kilohms, minimum capacitor values are 15, 37, and 75 pF. The capacitor should be tied to V_{EE} as this increases negative supply rejection.

A negative reference voltage may be used if R_{14} is grounded and the reference voltage is applied to R15 as shown in Figure 8. A high input impedance is the main advantage of this method. Compensation involves a capacitor to V_{EE} on pin 16, using the values of the previous paragraph. The negative reference voltage must be at least 3.0 volts above the V_{EE} supply. Bipolar input signals may be handled by connecting R14 to a positive reference voltage equal to the peak positive input level at pin 15.

When a reference voltage is used, capacitive bypass to ground is recommended. A 0.01 μ farad bypass is not recommended as a reference voltage will oscillate 5.0 V supply which drives logic to be used as the reference. R14 should be decoupled by connecting it to +5.0 V through another resistor and bypassing the junction of this two resistors with 0.1 μ F to ground. For reference voltage greater than 5.0 V, a clamping diode is recommended between pin 14 and ground.

If pin 14 is driven by a high impedance such as a transistor current source, none of the above compensation methods apply and the amplifier must be heavily compensated, decreasing the overall bandwidth.

Output Voltage Range

The voltage on pin 4 is restricted to a range of -0.65 to +0.4 volts at +25°C, due to the current switching methods employed in the MC1508L-8. When a current switch is turned "off", the positive voltage on the output terminal can turn "on" the output diodes and increase the output current level. When a current switch is turned "on", the negative output voltage range is restricted. The base of the termination circuit Darlington transistor is one diode voltage below ground when pin 1 is grounded, so a negative voltage below the specified safe level will drive the low current device of the Darlington into saturation, decreasing the output current level.

The negative output voltage compliance of the MC1508L-8 may be extended to -5.0 volt by opening the circuit at pin 1. The negative supply voltage must be more negative than -10 volts. Using a full scale current of 1.992 mA and load resistor of 2.5 kilohms between pin 4 and ground will yield a voltage output of -250 millivolts from 0 and -4.880 volts. Floating pin 1 does not affect the conversion speed or power dissipation. However, the value of the load resistor determines the switching time due to increased voltage swing. Values of R_L up to 500 ohms do not significantly affect performance, but a 2.5-kilohm load increases "worstcase" settling time to 1.2 μ s (when all bits are switched on).

Refer to the subsequent text section on Settling Time for more details on output loading.

If a power supply value between -5.0 V and -10 V is desired, a voltage of between 0 and -5.0 V may be applied to pin 1. The value of this voltage will be the maximum allowable negative output swing.

Output Current Range

The output current maximum rating of 4.2 mA may be used only for negative supply voltages typically more negative than -8.0 volts, due to the increased voltage drop across the 350-ohm resistors in the reference current amplifier.

Accuracy

Absolute accuracy is the measure of each output current level with respect to its intended value, and is dependent upon relative accuracy and full scale current drift. Relative accuracy is the measure of the output current level as a function of the output scale current. The relative accuracy of the MC1508L-8 is essentially constant with temperature due to the excellent temperature tracking of the monolithic resistor ladder. The reference current may drift with temperature, causing a change in the absolute accuracy of output current. However, the MC1508L-8 has a very low full scale current drift with temperature.

The MC1508L-8/MC1408L Series is guaranteed accurate to within $\pm 1/2$ LSB at +25°C at a full scale output current of 1.992 mA. This corresponds to a reference amplifier output current drive to the ladder network of 2.0 mA, with the loss of one LSB = 8.0 μ A which is the ladder remainder shunted to ground. The input current to pin 14 has a guaranteed value of between 1.9 and 2.1 mA, allowing some mismatch in the NPN current source pair. The accuracy test circuit is shown in Figure 4. The 12-bit converter is calibrated for a full scale output current of 1.992 mA. This is an optional step since the MC1508L-8 accuracy is essentially the same between 1.5 and 2.5 mA. Then the MC1508L-8 circuit's full scale current is trimmed to the same value with R14 so that a zero value appears at the error amplifier output. The counter is activated and the error band may be displayed on an oscilloscope, detected by comparators, or stored in a peak detector.

Two 8-bit D-to-A converters may not be used to construct a 16-bit accurate D-to-A converter. 16-bit accuracy implies a total error of $\pm 1/2$ of one part in 65,536, or $\pm 0.0007\%$, which is much more accurate than the $\pm 0.1\%$ specification provided by the MC1508L-8.

Multiplying Accuracy

The MC1508L-8 may be used in the multiplying mode with eight-bit accuracy when the reference current is varied over a range of 256:1. The major source of error is the bias current of the termination amplifier. Under "worst case" conditions, these eight amplifiers can contribute a total of 1.8 μ A extra current at the output terminal. If the reference current in the multiplying mode ranges from 10 μ A to 4.0 mA, the 1.8 μ A contributes an error of 0.1 LSB. This is well within eight-bit accuracy referenced to 4.0 mA.

A monotonic converter is one which supplies an increase in current for each increment in the binary word. Typically, the MC1508L-8 is monotonic for all values of reference current above 0.5mA. The recommended range for operation with a dc reference current is 0.5 to 4.0 mA.



MOTOROLA Semiconductor Products Inc.

GENERAL INFORMATION (Continued)

Setting Time

The "worst case" switching condition occurs when all bits are switched "on", which corresponds to a low-to-high transition for all bits. This time is typically 300 ns for settling to within 1/2 LSB, for 8-bit accuracy, and 200 ns to 1/2 LSB for 7 and 6-bit accuracy. The turn off is typically under 100 ns. These times apply when $R_L \leq 500$ ohms and $C_O \leq 25$ pF.

The slowest single switch is the least significant bit, which turns "on" and settles in 250 ns and turns "off" in 80 ns. In applications where the D-to-A converter functions in a positive-going ramp mode, the "worst case" switching condition does not occur, and a settling time of less than 300 ns may be realized. Bit A7 turns "on" in 200 ns and "off" in 80 ns, while bit A6 turns "on" in 150 ns and "off" in 80 ns.

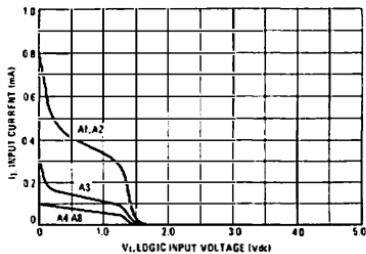
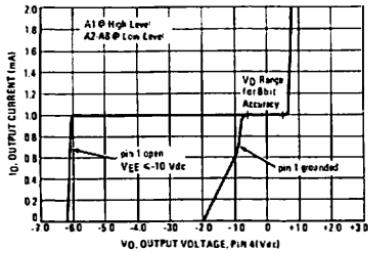
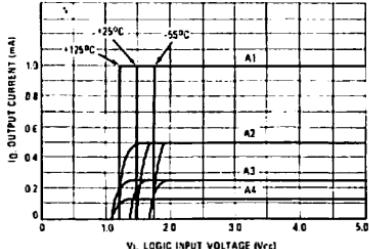
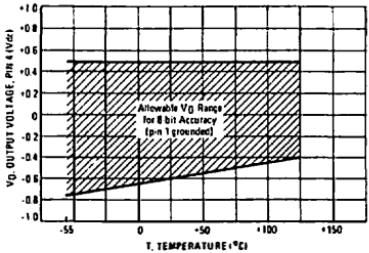
The test circuit of Figure 5 requires a smaller voltage swing for the current switches due to internal voltage clamping in the MC1508L-B. A 1.0-kilohm load resistor from pin 4 to ground gives a typical settling time of 400 ns. Thus, it is voltage swing and not the output RC time constant that determines settling time for most applications.

Extra care must be taken in board layout since this is usually the dominant factor in satisfactory test results when measuring settling time. Short leads, 100 μ F supply bypassing for low frequencies, and minimum scope lead length are all mandatory.

TYPICAL CHARACTERISTICS

($V_{CC} = +5.0$ V, $V_{EE} = -15$ V, $T_A = +25^\circ\text{C}$ unless otherwise noted.)

FIGURE 10 – LOGIC INPUT CURRENT versus INPUT VOLTAGE

FIGURE 12 – OUTPUT CURRENT versus OUTPUT VOLTAGE
(See text for pin 1 restrictions)FIGURE 11 – TRANSFER CHARACTERISTIC versus TEMPERATURE
(A5 thru AB thresholds lie within range for A1 thru A4)FIGURE 13 – OUTPUT VOLTAGE versus TEMPERATURE
(Negative range with pin 1 open is -5.0 Vdc over full temperature range)

MOTOROLA Semiconductor Products Inc.

TYPICAL CHARACTERISTICS (continued)
 $(V_{CC} = +5.0\text{ V}$, $V_{EE} = -15\text{ V}$, $T_A = +25^\circ\text{C}$ unless otherwise noted)

FIGURE 14 — REFERENCE INPUT FREQUENCY RESPONSE

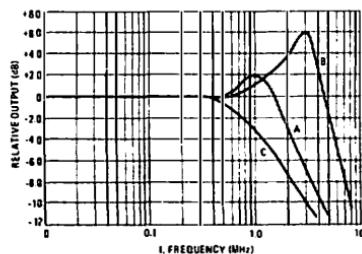


FIGURE 15 — TYPICAL POWERSUPPLY CURRENT
 versus TEMPERATURE (all bits low)

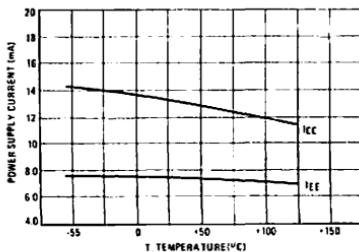
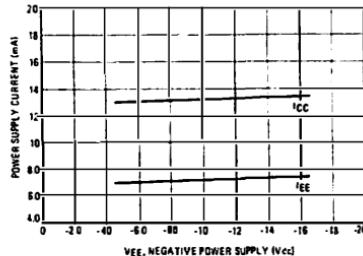
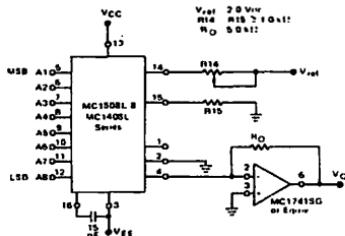


FIGURE 16 — TYPICAL POWER SUPPLY CURRENT
 versus V_{EE} (all bits low)



APPLICATIONS INFORMATION

FIGURE 17 — OUTPUT CURRENT TO VOLTAGE CONVERSION



Theoretical V_O

$$V_O = V_{ref} \left(\frac{R_{14}}{R_{14} + R_{15}} \right) [A_1, A_2, A_3, A_4, A_5, A_6, A_7, A_8]$$

Adjust V_{ref} , R_{14} or R_{15} so that V_O with all digital inputs high
 Input is equal to 0.931 VOLTS.

$$\begin{array}{c|ccccccccccccc}
2 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
V_O & 1 & 1.5 & 2 & 3 & 4 & 6 & 10 & 16 & 32 & 64 & 128 & 256 \\
10^4 & 255 & & & & & & & & & & & \\
& 268 & & & & & & & & & & & \\
& & 0.931\text{ V} & & & & & & & & & &
\end{array}$$



MOTOROLA Semiconductor Products Inc.

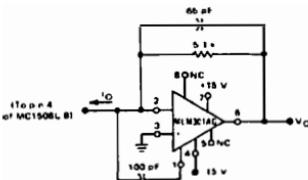
APPLICATIONS INFORMATION (continued)

Voltage outputs of a larger magnitude are obtainable with this circuit which uses an external operational amplifier as a current to voltage converter. This configuration automatically keeps the output of the MC1508L-B at ground potential and the operational amplifier can generate a positive voltage limited only by its positive supply voltage. Frequency response and settling time are primarily determined by the characteristics of the operational amplifier. In addition, the operational amplifier must be compensated for unity gain, and in some cases overcompensation may be desirable.

Note that this configuration results in a positive output voltage only, the magnitude of which is dependent on the digital input.

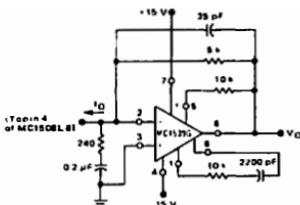
The following circuit shows how the MML301AG can be used in a feedforward mode resulting in a full scale settling time on the order of 2.0 μ s.

FIGURE 18



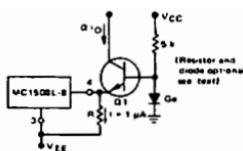
An alternative method is to use the MC1539G and input compensation. Response of this circuit is also on the order of 2.0 μ s. See Motorola Application Note AN-459 for more details on this concept.

FIGURE 19



The positive voltage range may be extended by cascading the output with a high beta common base transistor, Q1, as shown.

FIGURE 20 — EXTENDING POSITIVE VOLTAGE RANGE



The output voltage range for this circuit is 0 volts to BV_{CEO} of the transistor. If pin 1 is left open, the transistor base may be grounded, eliminating both the resistor and the diode. Variations in beta must be considered for wide temperature range applications. An inverted output waveform may be obtained by using a load resistor from a positive reference voltage to the collector of the transistor. Also, high-speed operation is possible with a large output voltage swing, because pin 4 is held at a constant voltage. The resistor (R₁) to V_{EE} maintains the transistor emitter voltage when all bits are "off" and insures fast turn-on of the least significant bit.

Combined Output Amplifier and Voltage Reference

For many of its applications the MC1508L-B requires reference voltage and an operational amplifier. Normally the operational amplifier is used as a current to voltage converter and its output need only go positive. With the popular MC1723G voltage regulator both of these functions are provided in a single package with the added bonus of up to 150 mA of output current. See Figure 21. The MC1723G uses both a positive and negative power supply. The reference voltage of the MC1723G is then developed with respect to the negative supply and applied to a common-mode input to the reference amplifier as a D-to-A converter. This allows use of its output amplifier as a class AB current-to-voltage converter with the non-inverting input grounded.

Since ± 15 V and ± 5.0 V are normally available in a combination digital-to-analog system, only the -5.0 V need be developed. A resistor divider is sufficiently accurate since the allowable range on pin 5 is from -2.0 to -8.0 volts. The 5.0 kilohm pulldown resistor on the amplifier output is necessary for fast negative transitions.

Full scale output may be increased to as much as 32 volts by increasing R_Q and raising the +15 V supply voltage to 35 V maximum. The resistor divider should be altered to comply with the maximum limit of 40 volts across the MC1723G. C_Q may be decreased to maintain the same R_QC_Q product if maximum speed is desired.



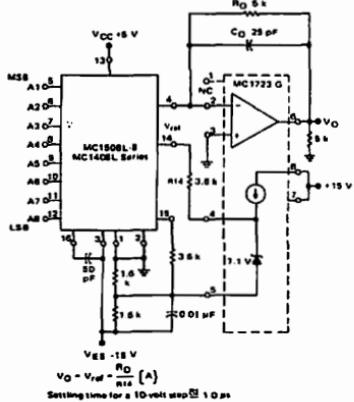
MOTOROLA Semiconductor Products Inc.

APPLICATIONS INFORMATION (continued)

Programmable Power Supply

The circuit of Figure 21 can be used as a digitally programmed power supply by the addition of thumbwheel switches and a BCD-to-binary converter. The output voltage can be scaled in several ways, including 0 to +25.5 volts in 0.1-volt increments, ± 0.05 volt; or 0 to 5.1 volts in 20 mV increments, ± 10 mV.

FIGURE 21 — COMBINED OUTPUT AMPLIFIER and VOLTAGE REFERENCE CIRCUIT



Bipolar or Negative Output Voltage

The circuit of Figure 22 is a variation from the standard voltage output circuit and will produce bipolar output signals. A positive input signal is converted into the summing junction to offset the output voltage in the negative direction. For example, if approximately 1.0 mA is used a bipolar output signal results which may be described as a 8-bit "Y" complement offset binary. V_{ref} may be used as this auxiliary reference. Note that R_O has been doubled to 10 kilohms because of the anticipated 20 V(p-p) output range.

FIGURE 22 — BIPOLE OR NEGATIVE OUTPUT VOLTAGE CIRCUIT

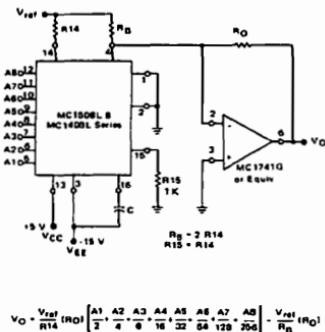
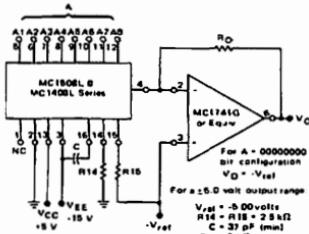


FIGURE 23 — BIPOLE OR INVERTED NEGATIVE OUTPUT VOLTAGE CIRCUIT



Decrease R_O to 2.5 k Ω for a 0 to -5-V output range. This application provides somewhat lower speed, as previously discussed in the Output Voltage Range section of the General Information.



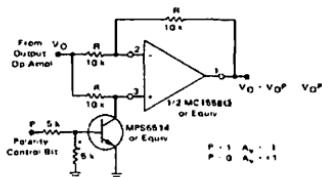
MOTOROLA Semiconductor Products Inc.

APPLICATIONS INFORMATION (continued)

Polarity Switching Circuit, 8-Bit Magnitude Plus Sign D-to-A Converter

Bipolar outputs may also be obtained by using a polarity switch circuit. The circuit of Figure 24 gives 8-bit magnitude plus a sign bit. In this configuration the operational amplifier is switched between a gain of +1.0 and -1.0. Although another operational amplifier is required, no more space is taken when a dual operational amplifier such as the MC1558G is used. The transistor should be selected for a very low saturation voltage and resistance.

FIGURE 24 – POLARITY SWITCHING CIRCUIT (8-Bit Magnitude Plus Sign D-to-A Converter)



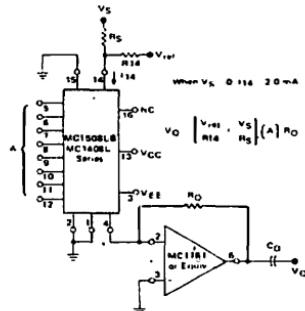
Programmable Gain Amplifier or Digital Attenuator

When used in the multiplying mode the MC1508L-B can be applied as a digital attenuator. See Figure 25. One advantage of this technique is that if $R_g = 50\text{m}\Omega$, no compensation capacitor is needed. The small and large signal bandwidths are now identical and are shown in Figure 14.

The best frequency response is obtained by not allowing I_{14} to reach zero. However, the high impedance node, pin 15, is clamped to prevent saturation and insure fast recovery when the current through R_{14} goes to zero. R_g can be set for a $\pm 1.0\text{ mA}$ variation in relation to I_{14} . I_{14} can never be negative.

The output current is always unipolar. The quiescent dc output current levels change with the digital word which makes coupling necessary.

FIGURE 25 – PROGRAMMABLE GAIN AMPLIFIER OR DIGITAL ATTENUATOR CIRCUIT



Panel Meter Readout

The MC1508L-B can be used to read out the status of BCD or binary registers or counters in a digital control system. The current output can be used to drive directly an analog panel meter. External meter shunts may be necessary if a meter of less than 2.0 mA full scale is used. Full scale calibration can be done by adjusting R_{14} or R_{15} .

FIGURE 26 – PANEL METER READOUT CIRCUIT

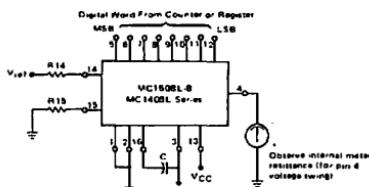
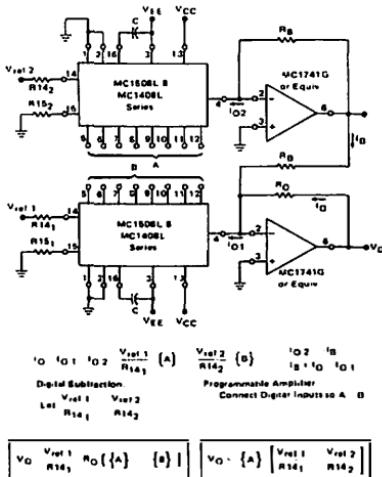


FIGURE 27 – DC COUPLED DIGITAL ATTENUATOR and DIGITAL SUBTRACTION



MOTOROLA Semiconductor Products Inc.

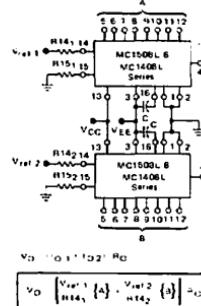
(MC1508 – Page 11)

APPLICATIONS INFORMATION (continued)

This digital subtraction application is useful for indicating when one digital word is approaching another in value. More information is available than with a digital comparator.

Bipolar inputs can be accepted by using any of the previously described methods, or applied differentially to R141 and R142 or R151 and R152. V_{ref} will be a bipolar signal defined by the input configuration. Note that the circuit shown accepts bipolar differential signals but does not have a negative common mode range. A very useful method is to connect R141 and R142 to a positive reference higher than the most positive input, and drive R151 and R152 by a common mode range.

FIGURE 28 - DIGITAL SUMMING and CHARACTER GENERATION



In a character generation system one MC150BL 8 circuit uses a fixed reference voltage and its digital input defines the starting point for a stroke. The second converter circuit has a ramp input for the reference and its digital input defines the slope of the stroke. Note that this approach does not result in a 16 bit D-to-A converter (see Accuracy Section).

FIGURE 29 - POSITIVE PEAK DETECTING SAMPLE and HOLD IF features indefinite hold time and optional digital output.]

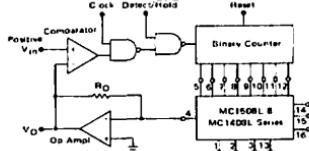


FIGURE 30 - NEGATIVE PEAK DETECTING SAMPLE AND HOLD

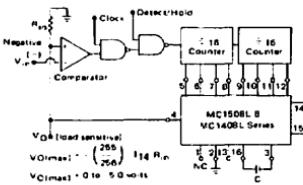
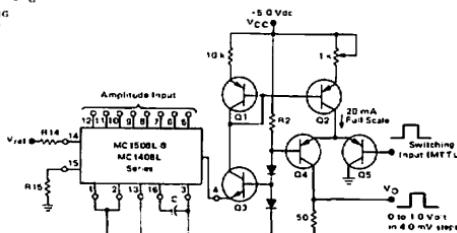
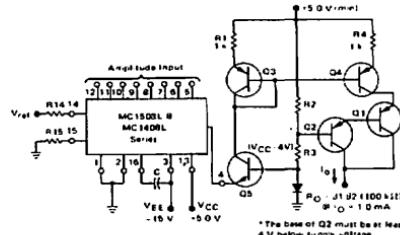


FIGURE 31 - PROGRAMMABLE PULSE GENERATOR



Fast rise and fall times require the use of high-speed switching transistors for the differential pair, Q4 and Q5. Linear ramps and sine waves may be generated by the appropriate reference input.

FIGURE 32 - PROGRAMMABLE CONSTANT CURRENT SOURCE



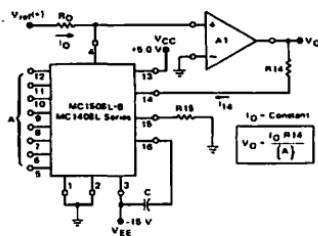
Current pulses, ramps, staircases, and sine waves may be generated by the appropriate digital and reference inputs. This circuit is especially useful in curve tracer applications.



MOTOROLA Semiconductor Products Inc.

APPLICATIONS INFORMATION (continued)

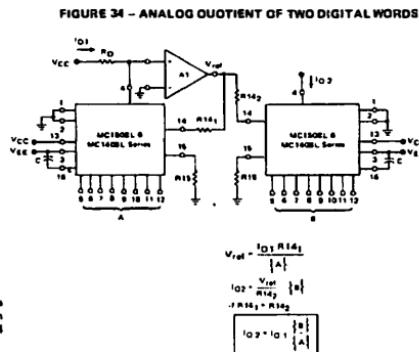
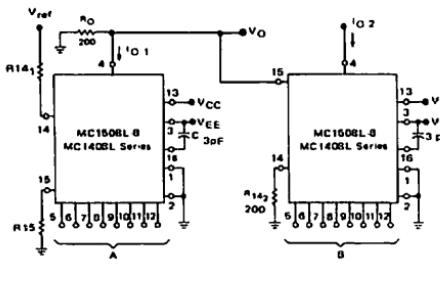
FIGURE 33 - ANALOG DIVISION BY DIGITAL WORD



This circuit yields the inverse of a digital word scaled by a constant. For minimum error over the range of operation, I_0 can be set at 16 μ A so that I_{14} will have a maximum value of 3.984 mA for a digital bit input configuration of 000000001.

Compensation is necessary for loop stability and depends on the type of operational amplifier used. If a standard 1.0 MHz operational amplifier is employed, it should be overcompensated when possible. If the MC1723 or another wideband amplifier is used, the reference amplifier should always be overcompensated.

FIGURE 34 - ANALOG QUOTIENT OF TWO DIGITAL WORDS

FIGURE 35 - ANALOG PRODUCT OF TWO DIGITAL WORDS
(High-Speed Operation)

$$VO = -I_{01} \cdot R_{14} = \frac{V_{ref}}{R_{141}} \cdot |A| \cdot R_{14}$$

$$I_{02} = \frac{|B|}{R_{142}} \cdot |V_{01}| = \frac{|B|}{R_{142}} \left[R_{14} \left(\frac{V_{ref}}{R_{141}} \right) \cdot |A| \right]$$

$$\text{Since } R_{14} = R_{142} \text{ and } K = \frac{V_{ref}}{R_{141}}$$

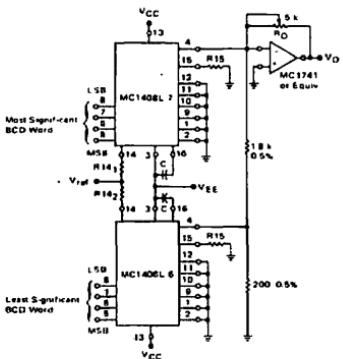
$$I_{02} = K \cdot |A| \cdot |B| \quad K \text{ can be an analog variable}$$



MOTOROLA Semiconductor Products Inc.

APPLICATIONS INFORMATION (continued)

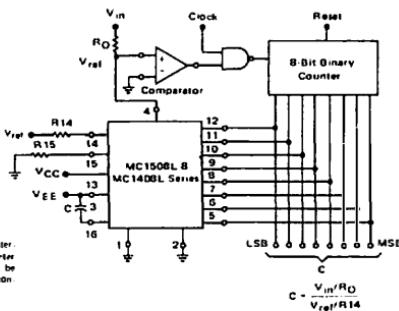
FIGURE 36 - TWO-DIGIT BCD CONVERSION



Two 8-bit, D-to-A converters can be used to build a two digit BCD-Dto-A or A-to-D converter. If both outputs feed the virtual ground of an operational amplifier, 10:1 current scaling can be achieved with a resistive current divider. If current output is desired, the units may be operated at full scale current levels of

4.0mA and 0.4mA with the output connected to sum the currents. The error of the D-to-A converter handling the least significant bits will be scaled down by a factor of ten and thus an MC1408L-6 may be used for the least significant word.

FIGURE 37 - DIGITAL QUOTIENT OF TWO ANALOG VARIABLES or ANALOG-TO-DIGITAL CONVERSION



Circuit diagrams utilizing Motorola products are included as a means of illustrating typical semiconductor applications. Consequently, complete information sufficient for construction purposes is not necessarily given. The information has been carefully checked and

is believed to be entirely reliable. However, no responsibility is assumed for inaccuracies. Furthermore, such information does not convey to the purchaser of the semiconductor devices described any license under the patent rights of Motorola Inc. or others.



MOTOROLA Semiconductor Products Inc.

BOX 20412 • PHOENIX, ARIZONA 85024 • A SUBSIDIARY OF MOTOROLA INC.

DATA SHEET 24 (2/72) REVISED 1/74 (MC1508 - Page 14)

55-282-24

INTERSIL

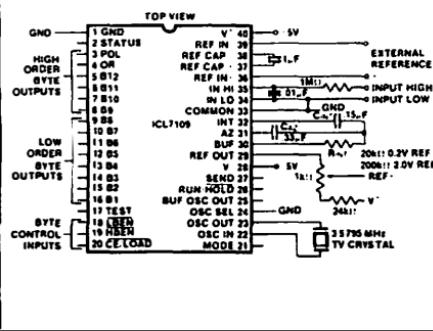
ICL7109 12 Bit Binary A/D Converter for Microprocessor Interfaces

FEATURES

- 12 bit binary (plus polarity and overrange) dual slope integrating analog-to-digital converter.
- Byte-organized TTL-compatible three-state outputs and UART handshake mode for simple parallel or serial interfacing to microprocessor systems.
- RUN/HOLD input and STATUS output can be used to monitor and control conversion timing.
- True differential input and differential reference.
- Low noise typically $15\mu V$ peak-to-peak.
- 1pA typical input current.
- Operates at up to 30 conversions per second.
- On-chip oscillator operates with inexpensive 3.58MHz TV crystal giving 7.5 conversions per second for 60Hz rejection, or may be operated as an RC oscillator for other clock frequencies.
- Fabricated using MAX-CMOS™ technology combining analog and digital functions on a single low power LSI CMOS chip.
- All inputs fully protected against static discharge; no special handling precautions necessary.

PIN CONFIGURATION AND TEST CIRCUIT:

See Figure 1 for typical connection to a UART or Microcomputer.



ORDERING INFORMATION

Part	Package	Temp. Range	Order Part #
7109	40 pin ceramic DIP	-25°C to -85°C	ICL7109JL
7109	40 pin ceramic DIP	0°C to -70°C	ICL7109CPL

GENERAL DESCRIPTION

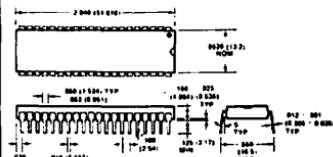
The ICL7109 is a high performance, low power integrating A/D converter designed to easily interface to microprocessors

The output data (12 bits, polarity and overrange) may be directly accessed under control of two byte enable inputs and a chip select input for a simple parallel bus interface. A UART handshake mode is provided which allows the ICL7109 to work with industry-standard UARTs to provide serial data transmission, ideal for remote data logging applications. The RUN/HOLD input and STATUS output allow monitoring and control of conversion timing.

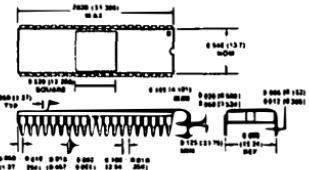
The ICL7109 provides the user the high accuracy, low noise, low drift, versatility and economy of the dual-slope integrating A/D converter. Features like true differential input and reference, zero drift of less than $1\mu V/\text{°C}$ max., input bias current of 10pA max., and typical power consumption of 20mW make the ICL7109 an attractive per-channel alternative to analog multiplexing for many data acquisition applications.

PACKAGE DIMENSIONS

40 Pin Plastic Dual-in-Line Package



40 Pin Ceramic Dual-in-Line Package



INTERSIL, INC., 10710 N TANTAUAVE, CUPERTINO, CA 95014

Printed in USA

408/996-5000 TWX 910-338-0171

1 of 16

ABSOLUTE MAXIMUM RATINGS

Positive Supply Voltage (GND to V ⁺)	-62V
Negative Supply Voltage (GND to V ⁻)	-5V
Analog Input Voltage (Lo or Hi) (Note 1)	V ⁻ to V ⁺
Reference Input Voltage (Lo or Hi) (Note 1)	V ⁻ to V ⁺
Digital Input Voltage (Pins 2-27) (Note 2)	V ⁻ + 0.3V GND - 0.3V
Power Dissipation (Note 3)	1W @ 85°C
Ceramic or Cerdip Package	500mW @ 70°C
Plastic Package	
Operating Temperature	
Ceramic or Cerdip Package	-25°C ≤ T _A ≤ 85°C
Plastic Package	0°C ≤ T _A ≤ 70°C
Storage Temperature	-55°C ≤ T _A ≤ 125°C
Lead Temperature (soldering, 60 sec.)	300°C

Absolute maximum ratings define stress limitations which if exceeded may permanently damage the device. These ratings are not continuous duty ratings. For continuous operation these devices must be operated under the conditions defined under "Operating Characteristics".

TABLE I OPERATING CHARACTERISTICS

All parameters with V⁺ = +5V, V⁻ = -5V GND = 0V, T_A = 25°C, unless otherwise indicated.
Test circuit as shown on page 1.

ANALOG SECTION

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Zero Input Reading	V _{IN}	V _{IN} = 0V Full scale = 409.6mV	-0000s	=0000s	-0000s	Octal Reading
Ratiometric Reading	V _{IN}	V _{IN} = V _{REF} V _{REF} = 204.8mV	3777s	3777s 4000s	4000s	Octal Reading
Non-Linearity (Max deviation from best straight line fit)		Full scale = 409.6mV or 4.096V	-1	-2	+1	Counts
Roll-over Error (Difference in reading for equal pos. and neg. inputs near full scale)			-1	-2	+1	Counts
Common Mode Rejection Ratio	V _{CMR}	V _{CM} ± 1V V _{IN} = 0V Full Scale = 409.6mV		50		µV/V
Noise (p-p value not exceeded 95% of time)	V _{IN}	V _{IN} = 0V Full Scale = 409.6mV		15		µV
Leakage Current at Input	I _{IN}	V _{IN} = 0V		1	10	pA
Zero Reading Drift	V _{IN}	V _{IN} = 0V		0.2	1	µV/°C
Scale Factor Temperature Coefficient		V _{IN} = 409.6mV => 7770s reading Ext. Ref. 0 ppm/°C		1	5	ppm/°C
Supply Current V ⁺ to GND	I _{DL}	V _{IN} = 0, Crystal Osc. 3.58MHz test circuit		700	1500	µA
Supply Current V ⁺ to V ⁻	I _{DA}	Pins 2-21, 25, 26, 27, 29, open		700	1500	µA
Ref Out Voltage	V _{REF}	Referred to V ⁻ . 25kΩ between V ⁻ and REF OUT	-2.4	-2.8	-3.2	V
Ref Out Temp. Coefficient		25kΩ between V ⁻ and REFOUT	80			ppm/°C

2 of 16

DIGITAL SECTION

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Output High Voltage	V _{OH}	I _{OUT} = 100 μ A Pins 2-16, 18, 19, 20	3.5	4.3		V
Output Low Voltage	V _{OL}	I _{OUT} = 1.6mA		0.2	0.4	V
Output Leakage Current		Pins 3-16 high impedance		$\pm .01$	± 1	μ A
Control I/O Pullup Current		Pins 18, 19, 20 V _{OUT} = V ₊ -3V MODE input at GND		5		μ A
Control I/O Loading		HBN Pin 19 LBEN Pin 18			50	pF
Input High Voltage	V _{IH}	Pins 18-21, 26, 27 referred to GND	2.5			V
Input Low Voltage	V _{IL}	Pins 18-21, 26, 27 referred to GND			1	V
Input Pull-up Current		Pins 26, 27 V _{OUT} = V ₊ -3V		5		μ A
Input Pull-up Current		Pins 17, 24 V _{OUT} = V ₊ -3V		25		μ A
Input Pull-down Current		Pin 21 V _{OUT} = GND +3V		5		μ A
Oscillator Output Current	High O _{OH}	V _{OUT} = 2.5V		1		mA
Low O _{OL}	V _{OUT} = 2.5V			1.5		mA
Buffered Oscillator Output Current	High B _{O_{OH}}	V _{OUT} = 2.5V		2		mA
Low B _{O_{OL}}	V _{OUT} = 2.5V			5		mA
MOOE Input Pulse Width			50			ns

Note 1: Input voltages may exceed the supply voltages provided the input current is limited to $\pm 100\mu$ A

Note 2: Due to the SCR structure inherent in the process used to fabricate these devices, connecting any digital inputs or outputs to voltages greater than V₊ or less than GND may cause destructive device latchup. For this reason it is recommended that no inputs from sources not on the same power supply be applied to the ICL7109 before its power supply is established, and that in multiple supply systems the supply to the ICL7109 be activated first.

Note 3: This limit refers to that of the package and will not be obtained during normal operation

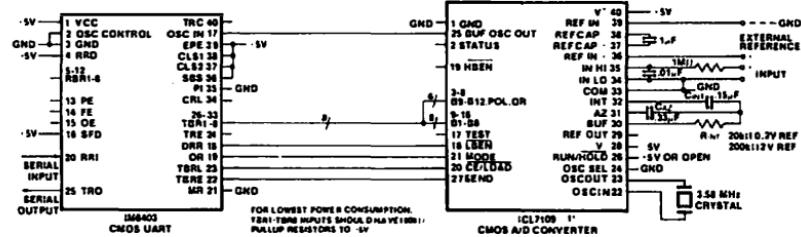


Figure 1A. Typical Connection Diagram UART Interface - Transmits Every Conversion

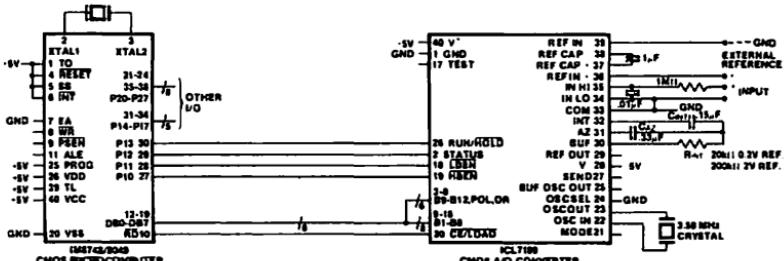


Figure 1B. Typical Connection Diagram Parallel Interface With 8748/8048 Microcomputer

TABLE 2 - Pin Assignment and Function Description

PIN	SYMBOL	DESCRIPTION	PIN	SYMBOL	DESCRIPTION
1	GND	Digital Ground 0V. Ground return for all digital logic	21	MODE	Input Low - Direct output mode where CE:LOAD iPin 21; HBEN iPin 19; and LBEN iPin 18; act as inputs directly controlling byte outputs Input Pulsed High - Causes immediate entry into handshake mode and output of data as in Figure 9 Input High - Enables CE:LOAD Pin 20; HBEN Pin 19; and LBEN Pin 18 as outputs. handshake mode will be entered and data output as in Figures 7 and 8 at conversion completion
2	STATUS	Output - High during integrate and deintegrate until data is latched - Low when analog section is in Auto-Zero configuration	22	OSC IN	Oscillator Input
3	POL	Polarity Three-State Output	23	OSC OUT	Oscillator Output
4	IOR	Over-range Three-State Output	24	OSC SEL	Oscillator Select - Input high configures OSC IN OSC OUT, BUF OSC OUT as RC oscillator - clock will be same phase and duty cycle as BUF OSC OUT - Input low configures OSC IN OSC OUT for crystal oscillator - clock frequency will be 1/8 of frequency at BUF OSC OUT
5	IB12	Bit 12 Most Significant Bit	25	BUF OSC OUT	Buffered Oscillator Output
6	IB11	Bit 11	26	RUN HOLD	Input High - Conversions continuously performed every 8192 clock pulses Input Low - Conversion in progress completed, converter will stop in Auto-Zero 7 counts before integrate
7	IB10	Bit 10	27	SEND	Input - Used in handshake mode to indicate ability of an external device to accept data
8	IB9	Bit 9	28	V	Analog Negative Supply - Normally 5V with respect to GND iPin 1
9	IB8	Bit 8	29	REF OUT	Reference Voltage Output - Normally 28V down from V- iPin 40
10	IB7	Bit 7	30	BUFFER	Buffer Amplifier Output
11	IB6	Bit 6 Data Bits, Three-State Output	31	AUTO-ZERO	Auto-Zero Node - Inside foil of Caz
12	IB5	Bit 5	32	INTEGRATOR	Integrator Output - Outside foil of Caz
13	IB4	Bit 4	33	COMMON	Analog Common - System is Auto-Zeroed to COMMON
14	IB3	Bit 3	34	INPUT LO	Differential Input Low Side
15	IB2	Bit 2	35	INPUT HI	Differential Input High Side
16	IB1	Bit 1 Least Significant Bit	36	REF IN -	Differential Reference Input Positive
17	TEST	Input High - Normal Operation Input Low - Forces all bit outputs high Note: This input is used for test purposes only	37	REF CAP -	Reference Capacitor Positive
18	LBEN	Low Byte Enable - With Mode iPin 21 low, and CE:LOAD iPin 20 low taking this pin low activates low order byte outputs Bit-B8 - With Mode iPin 21 high, this pin serves as a low byte flag output used in handshake mode. See Figures 7, 8, 9	38	REF CAP	Reference Capacitor Negative
19	HBEN	High Byte Enable - With Mode iPin 21 low, and CE:LOAD iPin 20 low taking this pin low activates high order byte outputs B9-B12, POL OR - With Mode iPin 21 high, this pin serves as a high byte flag output used in handshake mode. See Figures 7, 8, 9	39	REF IN	Differential Reference Input Negative
20	CE:LOAD	Chip Enable Load - With Mode iPin 21 low, CE:LOAD serves as a master output enable When high, B1-B12, POL, OR outputs are disabled - With Mode iPin 21 high, this pin serves as a load strobe used in handshake mode See Figures 7, 8, 9	40	V	Positive Supply Voltage - Normally +5V with respect to GND iPin 1

DETAILED DESCRIPTION

Analog Section

Figure 2 shows the equivalent circuit of the Analog Section of the ICL7109. When the RUN/ROLD input is left open or connected to V+, the circuit will perform conversions at a rate determined by the clock frequency (8192 clock periods per cycle). Each measurement cycle is divided into three phases as shown in Figure 3. They are (1) Auto-Zero (AZ), (2) Signal Integrate (INT) and (3) Deintegrate (DE).

1. Auto-Zero Phase

During auto-zero three things happen. First, input high and low are disconnected from their pins and internally shorted to analog common. Second, the reference capacitor is charged to the reference voltage. Third, a feedback loop is closed around the system to charge the auto-

zero capacitor Caz to compensate for offset voltages in the buffer amplifier, integrator, and comparator. Since the comparator is included in the loop, the accuracy is limited only by the noise of the system. In any case, the offset referred to the input is less than $10\mu\text{V}$.

2. Signal Integrate Phase

During signal integrate the auto-zero loop is opened, the internal short is removed and the internal input high and low are connected to the external pins. The converter then integrates the differential voltage between input high and input low for a fixed time of 2048 clock periods. At the end of this phase, the polarity of the integrated signal is determined.

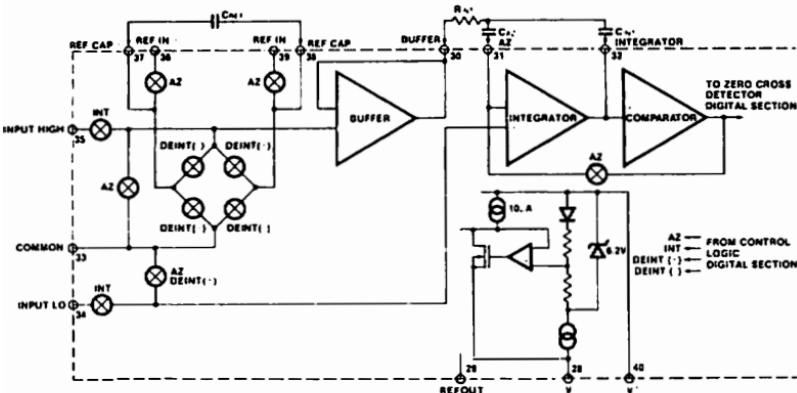


Figure 2: Analog Section

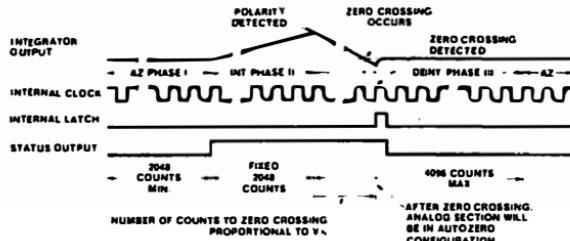


Figure 3: Conversion Timing

3 Deintegrate Phase

The final phase is deintegrate, or reference integrate. Input low is internally connected to analog common and input high is connected across the previously charged (during auto-zero) reference capacitor. Circuitry within the chip ensures that the capacitor will be connected with the correct polarity to cause the integrator output to return to the zero crossing established in Auto Zero with a fixed slope. Thus the time for the output to return to zero (represented by the number of clock periods counted) is proportional to the input signal.

Differential Input

The input can accept differential voltages anywhere within the common mode range of the input amplifier, or specifically from 0.5 volts below the positive supply to 1.0 volt above the negative supply. In this range the system has a CMRR of 86dB typical. However, since the integrator also swings with the common mode voltage, care must be exercised to assure the integrator output does not saturate. A worst case condition would be a large positive common mode voltage with a near full-scale negative differential input voltage. The negative input signal drives the integrator

positive when most of its swing has been used up by the positive common mode voltage. For these critical applications the integrator swing can be reduced to less than the recommended 4V full scale with some loss of accuracy. The integrator output can swing within 0.3 volts of either supply without loss of linearity.

The ICL7109 has, however, been optimized for operation with analog common near digital ground. With power supplies of -5V and +5V, this allows a 4V full scale integrator swing positive or negative maximizing the performance of the analog section.

Differential Reference

The reference voltage can be generated anywhere within the power supply voltage of the converter. The main source of common mode error is a roll-over voltage caused by the reference capacitor losing or gaining charge to stray capacity on its nodes. If there is a large common mode voltage, the reference capacitor can gain charge (increase voltage) when called up to deintegrate a positive signal but lose charge (decrease voltage) when called up to deintegrate a negative input signal. This difference in reference for + or - input voltage will give a roll-over error. However, by

selecting the reference capacitor large enough in comparison to the stray capacitance, this error can be held to less than 0.5 count for the worst case condition (see Component Values Selection below).

The roll-over error from these sources is minimized by having the reference common mode voltage near or at analog common.

Component Value Selection

For optimum performance of the analog section, care must be taken in the selection of values for the integrator capacitor and resistor, auto-zero capacity, reference voltage, and conversion rate. These values must be chosen to suit the particular application.

The most important consideration is that the integrator output swing (for full-scale input) be as large as possible. For example, with ± 5 V supplies and COMMON connected to GND, the nominal integrator output swing at full scale is ± 4 V. Since the integrator output can go to 0.3V from either supply without significantly affecting linearity, a 4V integrator output swing allows 0.7V for variations in output swing due to component value and oscillator tolerances. With ± 5 V supplies and a common mode range of ± 1 V required, the component values should be selected to provide ± 3 V integrator output swing. Noise and roll-over errors will be slightly worse than in the ± 4 V case. For larger common mode voltage ranges, the integrator output swing must be reduced further. This will increase both noise and roll-over errors. To improve the performance, supplies of ± 6 V may be used.

1. Integrating Resistor

Both the buffer amplifier and the integrator have a class A output stage with 100μ A of quiescent current. They supply 20μ A of drive current with negligible non-linearity. The integrating resistor should be large enough to remain in this very linear region over the input voltage range, but small enough that undistortion requirements are not placed on the PC board. For 4.096 volt full scale, $200\text{k}\Omega$ is near optimum and similarly a $20\text{k}\Omega$ for a 409.6mV scale. For other values of full scale voltage, R_{INT} should be chosen by the relation $R_{INT} = \frac{\text{full scale voltage}}{20\mu\text{A}}$

2. Integrating Capacitor

The integrating capacitor C_{INT} should be selected to give the maximum integrator output voltage swing without saturating the integrator (approximately 0.3 volt from either supply). For the ICL7109 with ± 5 volt supplies and analog common connected to GND, a ± 3.5 to ± 4 volt integrator output swing is nominal. For 7-1/2 conversions per second (61.72KHz clock frequency) as provided by the crystal oscillator, nominal values for C_{INT} and C_2 are $0.15\mu\text{F}$ and $0.33\mu\text{F}$, respectively. If different clock frequencies are used, these values should be changed to maintain the integrator output voltage swing. In general, the value of C_{INT} is given by

$$C_{INT} = \frac{(2048 \times \text{clock period}) (20\mu\text{A})}{\text{integrator output voltage swing}}$$

An additional requirement of the integrating capacitor is that it have low dielectric absorption to prevent roll-over errors. While other types of capacitors are adequate for this application, polypropylene capacitors give undetectable errors at reasonable cost.

3. Auto-Zero Capacitor

The size of the auto-zero capacitor has some influence on the noise of the system. A big capacitor, giving less noise. However, it cannot be increased without limits since it, in parallel with the integrating capacitor forms an R-C time constant that determines the speed of recovery from overloads and more important the error that exists at the end of an auto-zero cycle. For 409.6mV full scale where noise is very important and the integrating resistor small, a value of C_{AZ} twice C_{INT} is optimum. Similarly for 4.096V full scale where recovery is more important than noise, a value of C_{AZ} equal to half of C_{INT} is recommended.

For optimal rejection of stray pickup, the outer foil of C_{AZ} should be connected to the R-C summing junction and the inner foil to pin 31. Similarly the outer foil of C_{INT} should be connected to pin 32 and the inner foil to the R-C summing junction.

4. Reference Capacitor

A $1\mu\text{F}$ capacitor gives good results in most applications. However, where a large common mode voltage exists (i.e., the reference is not at analog common) and a 409.6mV scale is used, a larger value is required to prevent roll-over error. Generally $10\mu\text{F}$ will hold the roll-over error to 0.5 count in this instance.

5. Reference Voltage

The analog input required to generate a full scale output of 4096 counts is $V_{IN} = 2V_{REF}$. Thus for a normalized scale, a reference of 2.048V should be used for a 4.096V full scale, and 20.48mV should be used for a 409.6mV full scale. However, in many applications where the A/D is sensing the output of a transducer, there will exist a scale factor other than unity between the absolute output voltage to be measured and a desired digital output. For instance, in a weighing system, the designer might like to have a full scale reading when the voltage from the transducer is 0.682V. Instead of dividing the input down to 409.6mV, the input voltage should be measured directly and a reference voltage of 0.341V should be used. Suitable values for integrating resistor and capacitor are 34k and $0.15\mu\text{F}$. This avoids a divider on the input. Another advantage of this system occurs when a zero reading is desired for non-zero input. Temperature and weight measurements with an offset or tare are examples. The offset may be introduced by connecting the voltage output of the transducer between common and analog high, and the offset voltage between common and analog low, observing polarities carefully. However, in processor-based systems using the ICL7109, it may be more efficient to perform this type of scaling or tare subtraction digitally using software.

6. Reference Sources

The stability of the reference voltage is a major factor in the overall absolute accuracy of the converter. The resolution of the ICL7109 at 12 bits is one part in 4096, or 244ppm. Thus if the reference has a temperature coefficient of $80\text{ppm}/^{\circ}\text{C}$ (onboard reference) a temperature difference of 3°C will introduce a one-bit absolute error. For this reason, it is recommended that an external high-quality reference be used where the ambient temperature is not controlled or where high-accuracy absolute measurements are being made.

The ICL7109 provides a Reference Output (pin 29) which may be used with a resistive divider to generate a suitable reference voltage. This output will sink up to about 20mA without significant variation in output voltage, and is provided with a pullup bias device which sources about 10 μ A. The output voltage is nominally 2.8V below V_{cc}, and has a temperature coefficient of +80ppm/ $^{\circ}$ C typ. When using the onboard reference, Ref Out (pin 29) should be connected to Ref- (pin 39), and Ref+ should be connected to the wiper of a precision potentiometer between Ref Out and V_{cc}. The circuit for a 204.8mV reference is shown in the test circuit. For a 2.048V reference, the fixed resistor should be removed, and a 25k Ω precision potentiometer between Ref Out and V_{cc} should be used.

DETAILED DESCRIPTION

Digital Section

The digital section includes the clock oscillator and scaling circuit, a 12-bit binary counter with output latches and TTL-compatible three-state output drivers, polarity, over-range and control logic, and UART handshake logic, as shown in the Block Diagram Figure 4.

Throughout this description, logic levels will be referred to as "low" or "high". The actual logic levels are defined in Table 1 "Operating Characteristics". For minimum power consumption, all inputs should swing from GND (low) to V_{cc} (high). Inputs driven from TTL gates should have 3.5k Ω pulldown resistors added for maximum noise immunity.

MODE Input

The MODE input is used to control the output mode of the converter. When the MODE pin is connected to GND or left open, this input is provided with a pulldown resistor to ensure a low level when the pin is left open. The converter is in its "Direct" output mode, where the output data is directly accessible under the control of the chip and byte enable

inputs. When the MODE input is pulsed high, the converter enters the UART handshake mode and outputs the data in two bytes, then returns to "direct" mode. When the MODE input is left high, the converter will output data in the handshake mode at the end of every conversion cycle (See section entitled "Handshake Mode" for further details).

STATUS Output

During a conversion cycle, the STATUS output goes high at the beginning of Signal Integrate (Phase III), and goes low one-half clock period after new data from the conversion has been stored in the output latches. See Figure 3 for details of this timing. This signal may be used as a "data valid" flag (data never changes while STATUS is low) to drive interrupts, or for monitoring the status of the converter.

RUN/HOLD Input

When the RUN/HOLD input is connected to V_{cc} or left open (this input has a pullup resistor to ensure a high level when the pin is left open), the circuit will continuously perform conversion cycles, updating the output latches at the end of every Deintegrate (Phase III) portion of the conversion cycle (See Figure 3). In this mode of operation, the conversion cycle will be performed in 6192 clock periods, regardless of the resulting value.

If the RUN/HOLD input goes low (and stays there) during Integrate (Phase II) or Deintegrate (Phase III) before the zero crossing is detected, the converter will complete the conversion in progress, update the output latches, and then terminate Phase III, jumping to Auto-Zero (Phase II). If RUN/HOLD stays low, the converter will ensure a minimum Auto-Zero time, and wait in Auto-Zero until the RUN/HOLD input goes high. The converter will begin the Integrate (Phase II) portion of the next conversion (and the STATUS output will go high seven clock periods after the high level is detected at RUN/HOLD). See Figure 5 for details.

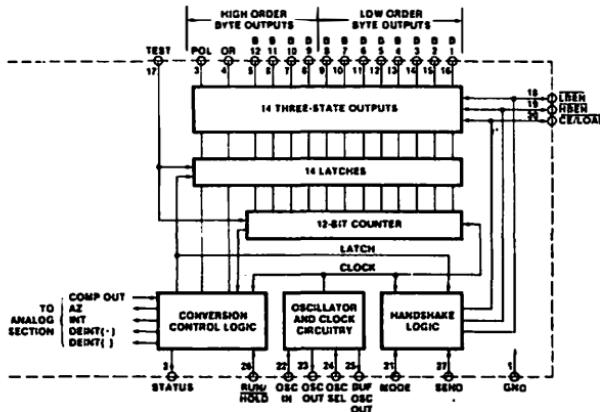


Figure 4: Digital Section

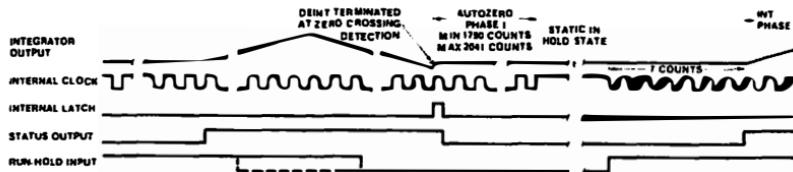


Figure 5: Run Hold Operation

Using the RUN/HOLD input in this manner allows an easy "convert on demand" interface to be used. The converter may be held at idle in auto-zero with RUN/HOLD low. When RUN/HOLD goes high the conversion is started, and when the STATUS output goes low the new data is valid for transferred to the UART - see Handshake Mode. RUN/HOLD may now go low terminating Demegrate and ensuring a minimum Auto-Zero time before stopping to wait for the next conversion.

If RUN/HOLD goes low at any time during Demegrate (Phase II) after the zero crossing has occurred, the circuit will immediately terminate Demegrate and jump to Auto-Zero. This feature can be used to "short-cycle" the converter by eliminating the time spent in Demegrate after the zero crossing. The required activity on the RUN/HOLD input can be provided by connecting it to the Buffered Oscillator Output. In this mode the conversion time is dependent on the input value measured. Also refer to Intersil Application Bulletin A030 for a discussion of the effects this will have on Auto-Zero performance.

If the RUN/HOLD input goes low and stays low during Auto-Zero (Phase II), the converter will simply stop at the end of Auto-Zero and wait for RUN/HOLD to go high. As above, Integrate (Phase III) begins seven clock periods after the high level is detected.

Direct Mode

When the MODE pin is left at a low level, the data outputs (bits 1 through 8 low order byte, bits 9 through 12, polarity and over-range high order byte) are accessible under control of the byte and chip enable terminals as inputs. These three inputs are all active low, and are provided with pullup resistors to ensure an inactive high level when left open. When the chip enable input is low, taking a byte enable input low will allow the outputs of that byte to become active (three-state'd out). This allows a variety of parallel data accessing techniques to be used, as shown in the section entitled "Interfacing". The timing requirements for these outputs are shown in Figure 6 and Table 3.

Table 3 - Direct Mode Timing Requirements

SYMBOL	DESCRIPTION	MIN	TYP	MAX	UNITS
t _{BEA}	Byte Enable Width	200	500	ns	
t _{DAB}	Data Access Time from Byte Enable	150	300	ns	
t _{DHB}	Data Hold Time from Byte Enable	150	300	ns	
t _{CCE}	Chip Enable Width	300	500	ns	
t _{DC}	Data Access Time from Chip Enable	200	400	ns	
t _{DHC}	Data Hold Time from Chip Enable	200	400	ns	

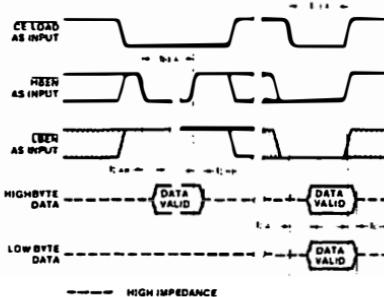


Figure 6: Direct Mode Output Timing

It should be noted that these control inputs are asynchronous with respect to the converter clock - the data may be accessed at any time. Thus it is possible to access the data while it is being updated, which could lead to scrambled data. Synchronizing the access of data with the conversion cycle by monitoring the STATUS output will prevent this. Data is never updated while STATUS is low.

Handshake Mode

The handshake output mode is provided as an alternative means of interfacing the ICL7109 to digital systems, where the A/D converter becomes active in controlling the flow of data instead of passively responding to chip and byte enable inputs. This mode is specifically designed to allow a direct interface between the ICL7109 and industry-standard UARTs such as the Intersil CMOS UARTs, IM6402/3 with no external logic required. When triggered into the handshake mode, the ICL7109 provides all the control and flag signals necessary to sequence the two bytes of data into the UART and initiate their transmission in serial form. This greatly eases the task and reduces the cost of designing remote data acquisition stations using serial data transmission to minimize the number of lines to the central controlling processor.

Entry into the handshake mode is controlled by the MODE pin. When the MODE terminal is held high, the ICL7109 will enter the handshake mode after new data has been stored in the output latches at the end of every conversion performed. (See Figures 7 and 8). The MODE terminal may also be used to trigger entry into the handshake mode on demand. At any time during the conversion cycle, the low-to-high transition of a short pulse at the MODE input will cause immediate entry

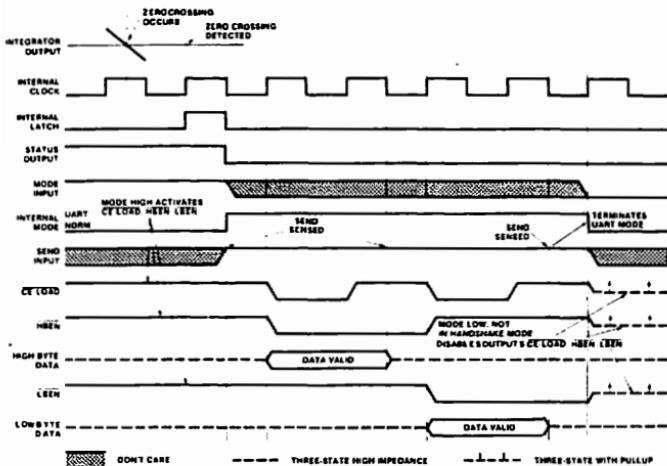


Figure 7: Handshake With Send Hold Positive

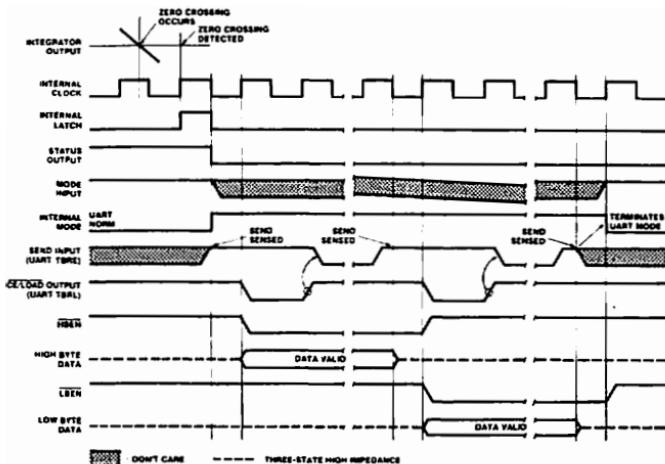


Figure 8: Handshake - Typical UART Interface Timing

ICL7109

INTERSIL

into the handshake mode. If this pulse occurs while new data is being stored, the entry into handshake mode is delayed until the data is stable. While the converter is in the handshake mode, the MODE input is ignored, and although conversions will still be performed, data updating will be inhibited (See Figure 9) until the converter completes the output cycle and clears the handshake mode.

When the converter enters the handshake mode, or when the MODE input is high, the chip and byte enable terminals become TTL-compatible outputs which provide the control signals for the output cycle (See Figures 7, 8, and 9).

In handshake mode, the SEND input is used by the converter as an indication of the ability of the receiving device (such as a UART) to accept data.

Figure 7 shows the sequence of the output cycle with SEND held high. The handshake mode (INTERNAL MODE high) is entered after the data latch pulse (since MODE remains high). The CE/LOAD, LBEN and HBEN terminals are active as outputs. The high level at the SEND input is sensed on the same high to low internal clock edge. On the next low to high internal clock edge, the CE/LOAD and the HBEN outputs assume a low level, and the high-order byte (bits 9 through 12, POL, and ORI outputs are enabled. The CE/LOAD output remains low for one full internal clock period only, the data outputs remain active for 1-1/2 internal clock periods, and the high byte enable remains low for two clock periods. Thus the CE/LOAD output low level or low to high edge may be used as a synchronizing signal to ensure valid data, and the

byte enable as an output may be used as a byte identification flag. With SEND remaining high the converter completes the output cycle using CE/LOAD and LBEN while the low order byte outputs bits 1 through 8 are activated. The handshake mode is terminated when both bytes are sent.

Figure 8 shows an output sequence where the SEND input is used to delay portions of the sequence, or handshake, to ensure correct data transfer. This timing diagram shows the relationships that occur using an industry-standard IM6402/3 CMOS UART to interface to serial data channels. In this interface, the SEND input to the ICL7109 is driven by the TBRE (Transmitter Buffer Register Empty) output of the UART, and the CE/LOAD terminal of the ICL7109 drives the TBRL (Transmitter Buffer Register Load) input to the UART. The data outputs are paralleled into the eight Transmitter Buffer Register inputs.

Assuming the UART Transmitter Buffer Register is empty, the SEND input will be high when the handshake mode is entered after new data is stored. The CE/LOAD and HBEN terminals will go low after SEND is sensed, and the high order byte outputs become active. When CE/LOAD goes high at the end of one clock period, the high order byte data is clocked into the UART Transmitter Buffer Register. The UART TBRE output will now go low, which halts the output cycle with the HBEN output low, and the high order byte outputs active. When the UART has transferred the data to the Transmitter Register and cleared the Transmitter Buffer Register, the TBRL returns high. On the next ICL7109

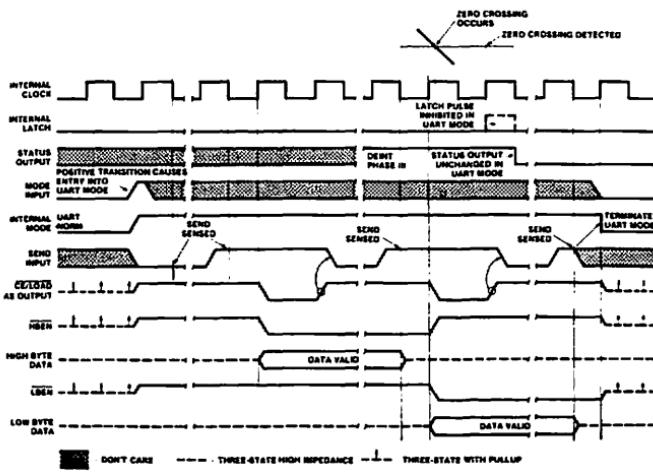


Figure 8: Handshake Triggered By Mode

10 of 16

ICL7109

INTEGRAL

internal clock high to low edge, the high order byte outputs are disabled, and one-half internal clock later, the HBEN output returns high. At the same time, the CE/LOAD and LBEN outputs go low, and the low order byte outputs become active. Similarly, when the CE/LOAD returns high at the end of one clock period, the low order data is clocked into the UART Transmitter Buffer Register, and TBRE again goes low. When TBRE returns to a high it will be sensed on the next ICL7109 internal clock high to low edge, disabling the data outputs. One-half internal clock later, the handshake mode will be cleared, and the CE/LOAD, HBEN, and LBEN terminals return high and stay active as long as MODE stays high.

With the MODE input remaining high as in these examples, the converter will output the results of every conversion except those completed during a handshake operation. By triggering the converter into handshake mode with a low to high edge on the MODE input, handshake output sequences may be performed on demand. Figure 5 shows a handshake output sequence triggered by such an edge. In addition, the SEND input is shown as being low when the converter enters handshake mode. In this case, the whole output sequence is controlled by the SEND input, and the sequence for the first (high order) byte is similar to the sequence for the second byte. This diagram also shows the output sequence taking longer than a conversion cycle. Note that the converter still makes conversions, with the STATUS output and RUN/HOLD input functioning normally. The only difference is that new data will not be latched when in handshake mode, and is therefore lost.

Oscillator

The ICL7109 is provided with a versatile three terminal oscillator to generate the internal clock. The oscillator may be overdriven, or may be operated as an RC or crystal oscillator. The OSCILLATOR SELECT input changes the internal configuration of the oscillator to optimize it for RC or crystal operation.

When the OSCILLATOR SELECT input is high or left open the input is provided with a pullup resistor, the oscillator is configured for RC operation, and the internal clock will be of the same frequency and phase as the signal at the BUFFERED OSCILLATOR OUTPUT. The resistor and capacitor should be connected as in Figure 10. The circuit will oscillate at a frequency given by $f = 1/(45RC)$. A 100k Ω resistor is recommended for useful ranges of frequency. For optimum 60Hz line rejection, the capacitor value should be chosen such that 2048 clock periods is close to an integral multiple of the 60Hz period.

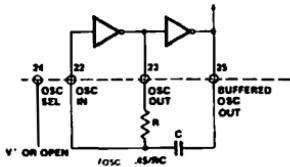


Figure 10: RC Oscillator

When the OSCILLATOR SELECT input is low a feedback device and output and input capacitors are added to the oscillator. In this configuration, as shown in Figure 11, the

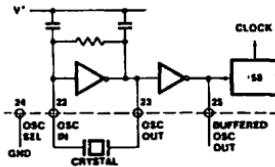


Figure 11: Crystal Oscillator

oscillator will operate with most crystals in the 1 to 5MHz range with no external components. Taking the SELECT input low also inserts a fixed +58 divider circuit between the BUFFERED OSCILLATOR OUTPUT and the internal clock. Using an inexpensive 3.58MHz TV crystal, this division ratio provides an integration time given by:

$$T = (2048 \text{ clock periods}) \times \left(\frac{58}{3.58 \text{ MHz}} \right) = 33.18 \text{ ms}$$

This time is very close to two 60Hz periods or 33.33ms. The error is less than one percent, which will give better than 40dB 60Hz rejection. The converter will operate reliably at conversion rates of up to 30 per second, which corresponds to a clock frequency of 245.8kHz.

If at any time the oscillator is to be overdriven, the overdriving signal should be applied at the OSCILLATOR INPUT, and the OSCILLATOR OUTPUT should be left open. The internal clock will be of the same frequency, duty cycle, and phase as the input signal when OSCILLATOR SELECT is left open. When OSCILLATOR SELECT is at GND, the clock will be a factor of 58 below the input frequency.

When using the ICL7109 with the IM6403 UART, it is possible to use one 3.58MHz crystal for both devices. The BUFFERED OSCILLATOR OUTPUT of the ICL7109 may be used to drive the OSCILLATOR INPUT of the UART, saving the need for a second crystal. However, the BUFFERED OSCILLATOR OUTPUT does not have a great deal of drive, and when driving more than one slave device, external buffering should be used.

Test Input

When the TEST input is taken to a level halfway between V' and GND, the counter output latches are enabled, allowing the counter contents to be examined anytime.

When the TEST input is connected to GND, the counter outputs are all forced into the high state, and the internal clock is disabled. When the input returns to the 1/2(V' - GND) voltage or to V' and one clock is input, the counter outputs will all be clocked to the negative state. This allows easy testing of the counter and its outputs.

INTERFACING

Direct Mode

Figure 12 shows some of the combinations of chip enable and byte enable control signals which may be used when interfacing the ICL7109 to parallel data lines. The CE/LOAD input may be tied low, allowing either byte to be controlled by its own enable as in Figure 12A. Figure 12B shows a configuration where the two byte enables are connected together. In this configuration, the CE/LOAD serves as a chip enable, and the HBEN and LBEN may be connected to GND or serve as a second chip enable. The 14 data outputs will all be enabled simultaneously. Figure 12C shows the HBEN and LBEN as flag inputs, and CE/LOAD as a master enable, which could be the READ strobe available from most microprocessors.

ICL7109

INTERSIL

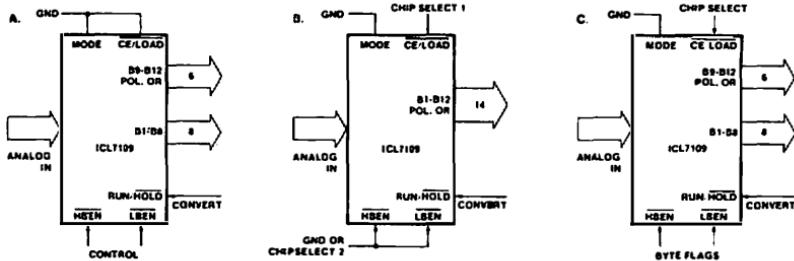


Figure 12: Direct Mode Chip and Byte Enable Combinations

Figure 13 shows an approach to interfacing several ICL7109s to a bus, ganging the HBEN and LBEN signals to several converters together, and using the CE/LOAD inputs (perhaps decoded from an address) to select the desired converter.

Some practical circuits utilizing the parallel three-state output capabilities of the ICL7109 are shown in Figures 14 through 19. Figure 14 shows a straightforward application to the Intel MCS-48, -80 and -85 systems via an 8255PPI, where the ICL7109 data outputs are active at all times. The I/O ports of an 8155 may be used in the same way. This interface can be used in a read-anytime mode, although a read performed while the data latches are being updated will lead to scrambled data. This will occur very rarely, in the proportion of setup-skew times to conversion time. One way to overcome this is to read the STATUS output as well, and if it is high, read the data again after a delay of more than 1/2 converter clock period. If STATUS is now low, the second reading is correct, and if it is still high, the first reading is correct. Alternatively, this timing problem is completely avoided by using a read-after-update sequence, as shown in Figure 15. Here the high to low transition of the STATUS output drives an interrupt to the microprocessor causing it to

access the data. This application also shows the RUN-HOLD input being used to initiate conversions under software control.

A similar interface to Motorola MC6800 or MOS Technology MCS650X systems is shown in Figure 16. The high to low transition of the STATUS output generates an interrupt via the Control Register B CB1 line. Note that CB2 controls the RUN-HOLD pin through Control Register B, allowing software-controlled initiation of conversions in this system also.

Figure 17 shows an interface to the Intersil IM6101 CMOS microprocessor family using the IM6101 PIE to control the data transfers. Here the data is read by the microprocessor in an 8-bit and a 6-bit word, directly from the ICL7109 to the microprocessor data bus. Again the high to low transition of the STATUS output generates an interrupt leading to a software routine controlling the two read operations. As before, the RUN-HOLD input to the ICL7109 is shown as being under software control.

The three-state output capability of the ICL7109 allows direct interfacing to most microprocessor busses. Examples of this are shown in the Typical Connection Diagram on

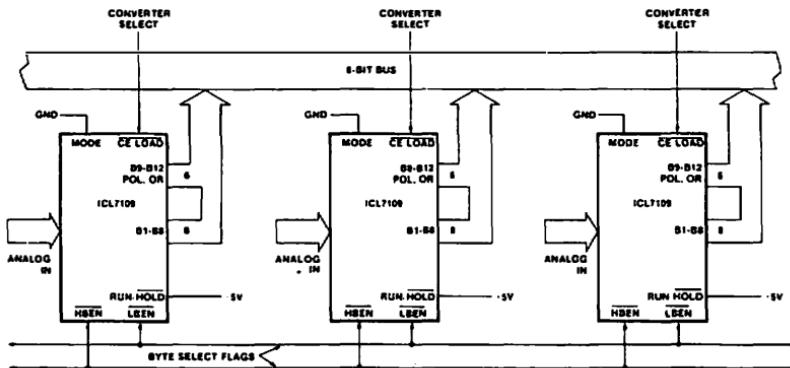


Figure 13: Three-staging Several 7109's to a Small Bus

12 of 16

Page 3 and in Figures 18 and 19. It is necessary to carefully consider the system timing in this type of interface, to be sure that requirements for setup and hold times, and minimum pulse widths are met. Note also the drive limitations on long busses. Generally this type of interface is only favored if the

memory peripheral address density is low so that simple address decoding can be used. Interrupt handling can also require many additional components, and using an interface device will usually simplify the system in this case.

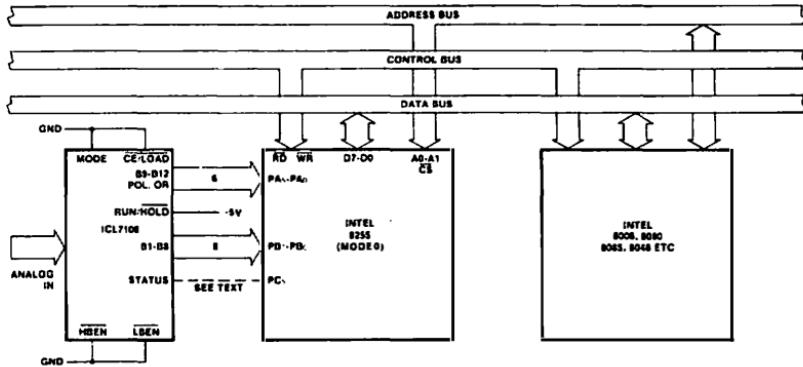
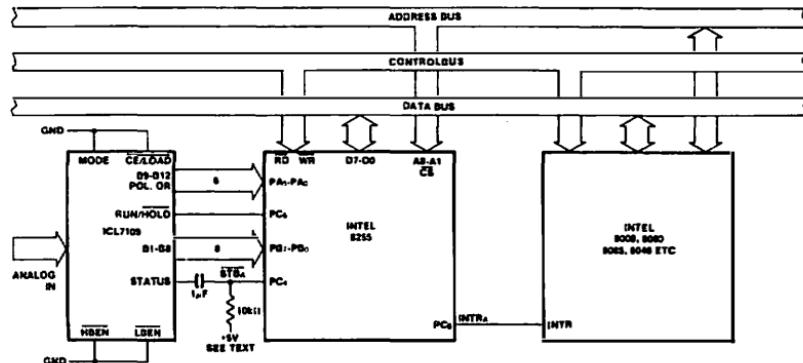


Figure 14: Full-time Parallel Interface to INTEL Microcomputer Systems



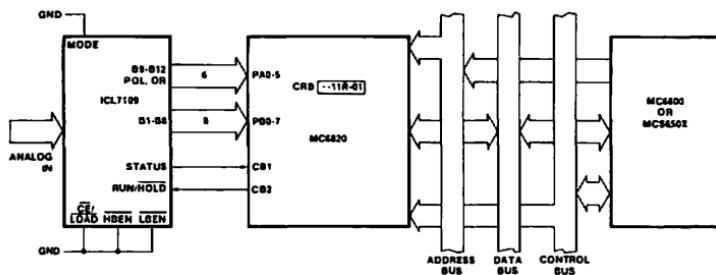


Figure 16: Full-time Parallel Interface to MC6800 or MCS650X Microprocessors

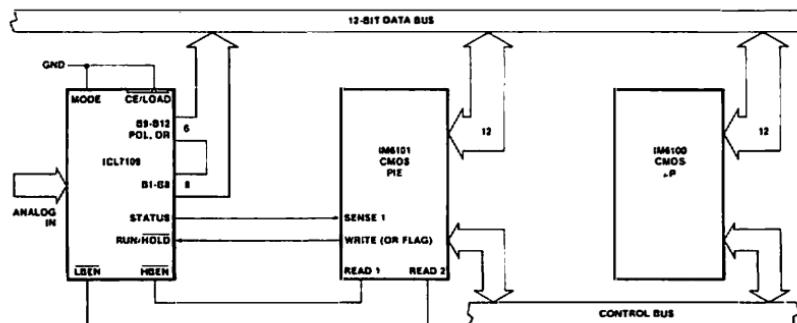


Figure 17: ICL7109-IM6100 Interface Using IM6101 PIE

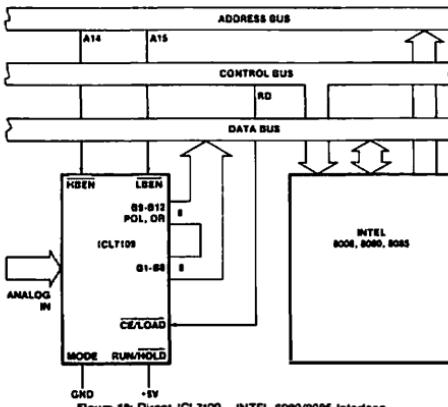


Figure 18: Direct ICL7109 - INTEL 8080/8085 Interface

14 of 15

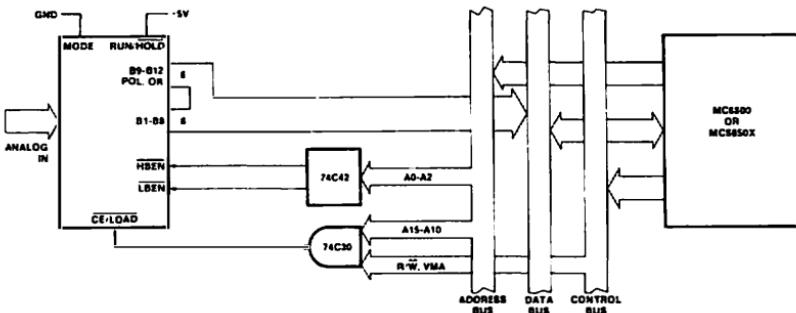


Figure 19: Direct ICL7109 - MC6800 Bus Interface

Handshake Mode

The handshake mode allows ready interface with a wide variety of external devices. For instance, external latches may be clocked by the rising edge of CE/LOAD, and the byte enables may be used as byte identification flags or as load enables.

Figure 20 shows a handshake interface to Intel microprocessors again using an 8255PPI. The handshake operation with the 8255 is controlled by inverting its Input Buffer Full (IBF) flag to drive the SEND input to the ICL7109, and using the CE/LOAD to drive the 8255 strobe. The internal control register of the PPI should be set in MODE 1 for the port used if the 7109 is in handshake mode and the 8255 IBF flag is low, the next word will be strobed into the port. The strobe will cause IBF to go high (SEND goes low), which will keep the enabled byte outputs active. The PPI will generate an interrupt when executed will result in the data being read. When the byte is read, the IBF will be reset low, which causes the ICL7109 to sequence into the next byte. This figure shows the MODE input to the ICL7109 connected to a control line on the PPI. If this output is left high, or tied high

separately, the data from every conversion (provided the data access takes less time than a conversion will be sequenced in two bytes into the system).

If this output is made to go from low to high, the output sequence can be obtained on demand, and the interrupt may be used to reset the MODE bit. Note that the RUN/HOLD input to the ICL7109 may also be driven by a bit of the 8255 so that conversions may be obtained on command under software control. Note that one port of the 8255 is not used, and can service another peripheral device. The same arrangement can also be used with the 8155.

Figure 21 shows a similar arrangement with the MC6800 or MC6850X microprocessors, except that both MODE and RUN/HOLD are tied high to serve port outputs.

The handshake mode is particularly convenient for directly interfacing to industry standard UARTs (such as the Intersil IM6402/6403 or Western Digital TR1602) providing a minimum component count means of serially transmitting converted data. A typical UART connection is shown on page 3. In this circuit, any word received by the UART causes

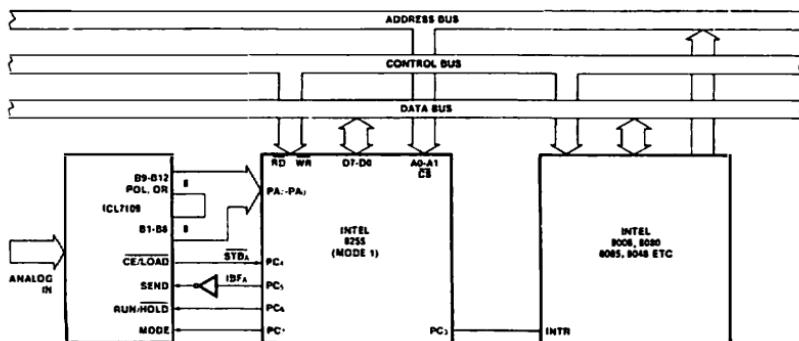


Figure 20: Handshake Interface - ICL7109 to INTEL MCS-48 -8D, 85

ICL7109

INTERSIL

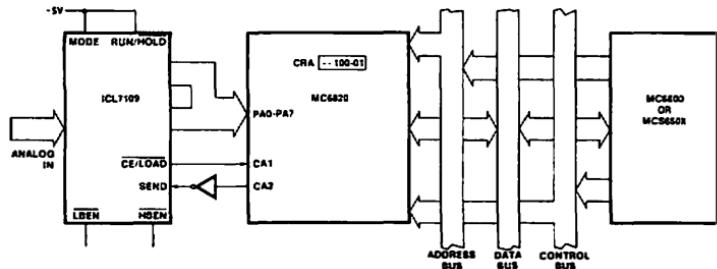


Figure 21: Handshake Interface - ICL7109 to MC6800, MCS6850X

the UART DR (Data Ready) output to go high. This drives the MODE input to the ICL7109 high, triggering the ICL7109 into handshake mode. The high order byte is output to the UART first, and when the UART has transferred the data to the Transmitter Register, TBRE (SEND) goes high and the second byte is output. When TBRE (SEND) goes high again, LBEN will go high, driving the UART DRR (Data Ready Reset) which will signal the end of the transfer of data from the ICL7109 to the UART.

Figure 22 shows an extension of the one converter - one UART scheme of the Typical Connection to several ICL7109s with one UART. In this circuit, the word received by the UART available at the RBR outputs when DR is high!

is used to select which converter will handshake with the UART. With no external components, this scheme will allow up to eight ICL7109s to interface with one UART. Using a few more components to decode the received word will allow up to 256 converters to be accessed on one serial line.

The applications of the ICL7109 are not limited to those shown here. The purpose of these examples is to provide a starting point for users to develop useful systems, and to show some of the variety of interfaces and uses of the ICL7109. Many of the ideas suggested here may be used in combination; in particular the uses of the STATUS, RUN/HOLD, and MODE signals may be mixed.

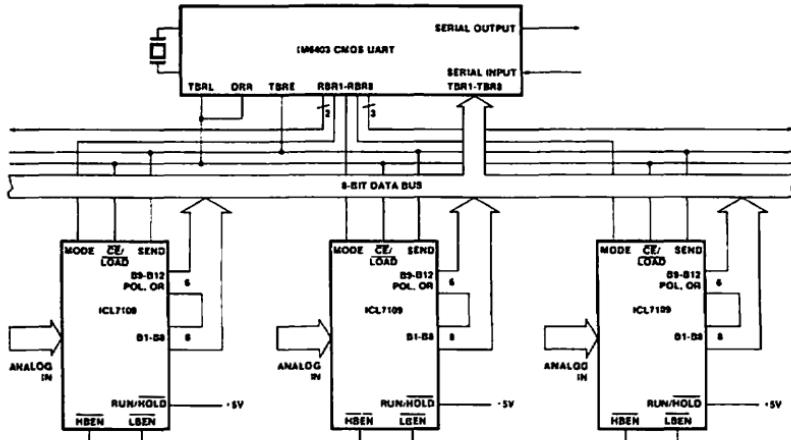


Figure 22: Multiplexing Converters with Mode Input

INTERSIL

10710 N. Tantau Ave., Cupertino, CA 95014 (408) 996-5000 TWX: 910-338-0171

Intersil cannot assume responsibility for use of any circuitry described other than circuitry entirely embodied in an Intersil product. No other circuit patent licenses are implied. Intersil reserves the right to change the circuitry and specifications without notice at any time.
9-78-00A

8-BIT μ P-COMPATIBLE D/A CONVERTER

NE5018

PRELIMINARY SPECIFICATION

NE5018-F.N

DESCRIPTION

The NE5018 is a complete 8-bit digital to analog converter subsystem on one monolithic chip. The data inputs have input latches controlled by a latch enable pin. The data and latch enable inputs are ultra-low loading for easy interfacing with all logic systems. The latches appear transparent when the LE input is in the low state. When LE goes high, the input data present at the moment of transition is latched and retained until LE again goes low. This feature allows easy compatibility with most microprocessors.

The chip also comprises a stable voltage reference (5V nominal) and a high slew rate buffer amplifier. The voltage reference may be externally trimmed with a potentiometer for easy adjustment of full scale, while maintaining a low temperature coefficient.

The output of the buffer amplifier may be offset so as to provide bipolar as well as unipolar operation.

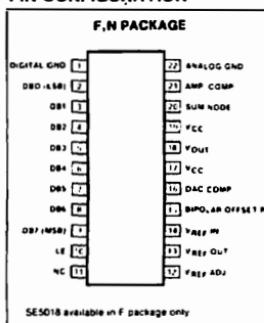
FEATURES

- 8-bit resolution
- Input latches
- Low-loading data inputs
- On-chip voltage reference
- Output buffer amplifier
- Accurate to $\pm 1/2$ LSB
- Monotonic to 8 bits
- Amplifier and reference both short-circuit protected
- Compatible with 2850, 8080 and many other μ Ps.

APPLICATIONS

- Precision 8-bit D/A converters
- A/D converters
- Programmable power supplies
- Test equipment
- Measuring instruments
- Analog-digital multiplication

PIN CONFIGURATION



NE5018 available in F package only

ABSOLUTE MAXIMUM RATINGS

PARAMETER	RATING	UNIT
V _{CC+}	Positive supply voltage	18
V _{CC-}	Negative supply voltage	-18
V _{IN}	Logic input voltage	0 to 18
V _{REFIN}	Voltage at V _{REF} input	12
V _{REFADJ}	Voltage at V _{REF} adjust	0 to V _{REF}
V _{SUM}	Voltage at sum node	12
I _{REFSC}	Short-circuit current to ground at V _{REF} OUT	Continuous
I _{OUTSC}	Short-circuit current to ground or either supply at V _{OUT}	Continuous
I _{REF}	Reference input current	5
P _D	Power dissipation*	
	-N package	800
	-F package	1000
T _A	Operating temperature range	mW
	SE5018	0 to +125
	NE5018	0 to +70
T _{STG}	Storage temperature range	°C
T _{SOLD}	Lead soldering temperature (10 seconds)	-65 to -150
		°C
		300

*NOTE

For N package derate at 12°C/°C above 35°C

For F package derate at 75°C W above 75°C

8-BIT μ P-COMPATIBLE D/A CONVERTER

NE5018

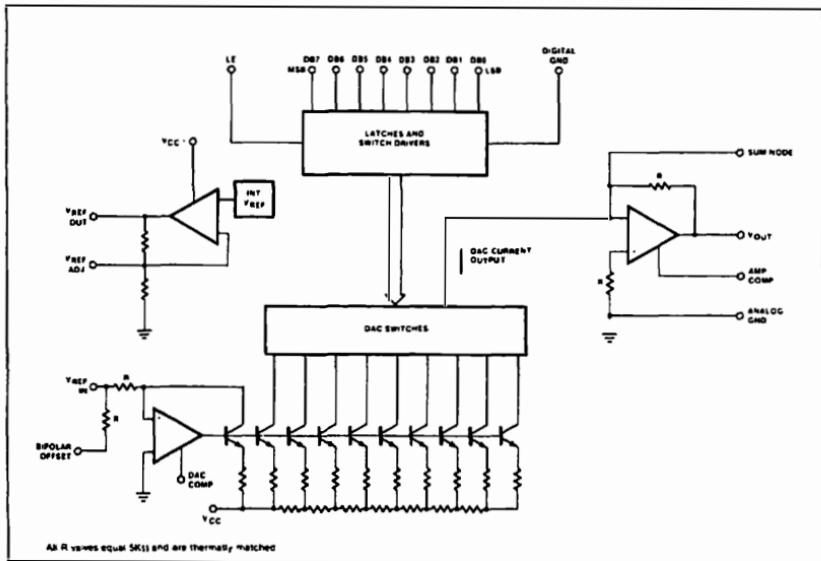
PRELIMINARY SPECIFICATION

NE5018-F.N

DC ELECTRICAL CHARACTERISTICS $V_{CC+} = +15V$, $V_{CC-} = -15V$, $SE5018$ $-55^{\circ}C \leq T_A \leq 125^{\circ}C$,
 $NE5018$ $0^{\circ}C \leq T_A \leq 70^{\circ}C$ unless otherwise specified

PARAMETER	TEST CONDITIONS	SE5018			NE5018			UNITS
		Min	Typ	Max	Min	Typ	Max	
V_{CC+}	Positive supply voltage			15		15		V
V_{CC-}	Negative supply voltage			-15		-15		V
Resolution			8		8		8	bits
Relative accuracy				± 0.19			± 0.19	%
T_s	Settling time		2		2		2	μs
PSRR	Power supply rejection ratio		± 1		± 1		± 1	mV/V
I_{CC+}	Positive supply current			8		8		mA
I_{CC-}	Negative supply current			-10		-10		mA
$I_{IN(0)}$	Logic "0" input current		5		5		0.8	μA
$V_{IN(0)}$	Logic "0" input voltage			0.8		2.0		V
$V_{IN(1)}$	Logic "1" input voltage					2.0	400	V
T_{PWLE}	Latch enable pulse width						400	ns

BLOCK DIAGRAM



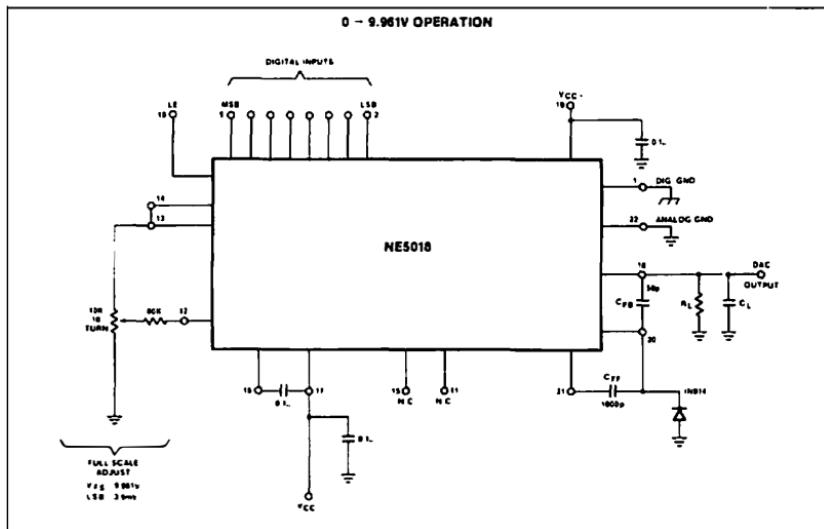
8-BIT μ P-COMPATIBLE D/A CONVERTER

NE5018

PRELIMINARY SPECIFICATION

NE5018-F.N

EQUIVALENT SCHEMATIC



sgnetics



MOTOROLA

MC14066B

QUAD ANALOG SWITCH/QUAD MULTIPLEXER

The MC14066B consists of four independent switches capable of controlling either digital or analog signals. This quad bilateral switch is useful in signal gating, chopper, modulator, demodulator and CMOS logic implementation.

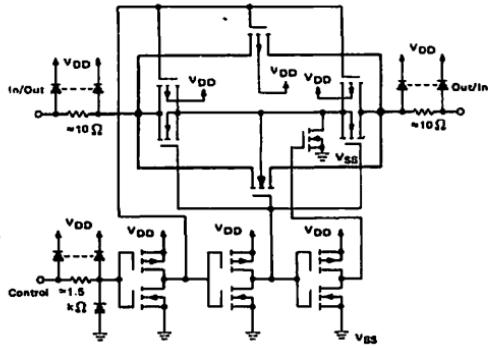
The MC14066B is designed to be pin-for-pin compatible with the MC14066B, but has much lower ON resistance. Input voltage swings as large as the full supply voltage can be controlled via each independent control input.

- High On/Off Output Voltage Ratio - 65 dB typical
- Quiescent Current = 0.5 nA/package typical @ 5 Vdc
- Low Crosstalk Between Switches -50 dB typical @ 8 MHz
- Diode Protection on All Inputs
- Supply Voltage Range = 3.0 Vdc to 18 Vdc
- Transmits Frequencies Up to 65 MHz @ 10 Vdc
- Linearized Transfer Characteristics, $\Delta R_{ON} < 60 \Omega$ for $V_{IN} = V_{DD}$ to V_{SS} (at 15V)
- Low Noise - 12 nV/V/Cycle, f > 1 kHz typical
- Pin-for-Pin Replacement for CD4016, CD4066, MC14016

MAXIMUM RATINGS (Voltages referenced to V_{SS})

Rating	Symbol	Value	Unit
DC Supply Voltage	V_{DD}	-0.5 to +18	Vdc
Input Voltage, All Inputs	V_{IN}	-0.5 to $V_{DD} + 0.5$	Vdc
Through Current	I	25	mA/dc
Operating Temperature Range - AL Device	T_A	-55 to +125	°C
CL/CP Device		-40 to +85	
Storage Temperature Range	T_{STG}	-65 to +150	°C

CIRCUIT SCHEMATIC
(1/4 OF DEVICE SHOWN)



CMOS SSI

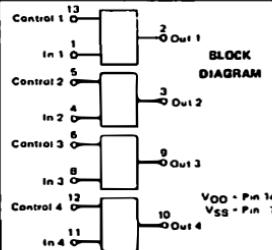
(LOW-POWER COMPLEMENTARY MOS)

QUAD ANALOG SWITCH QUAD MULTIPLEXER

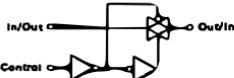


ORDERING INFORMATION

MC14XXX	Suffix	Denotes
	L	Ceramic Package
	P	Plastic Package
	A	Extended Operating Temperature Range
	C	Limited Operating Temperature Range



LOGIC DIAGRAM AND TRUTH TABLE
(1/4 OF DEVICE SHOWN)



Control	Switch	Logic Diagram Restrictions
0	DFF	$V_{SS} \leq V_{IN} \leq V_{DD}$
1	ON	$V_{SS} \leq V_{OUT} \leq V_{DD}$

V _{Control}	V _{IN} to V _{OUT} Resistance
V _{SS}	> 10 ⁹ Ohms typ
V _{DD}	3 × 10 ² Ohms typ

ELECTRICAL CHARACTERISTICS

Characteristic	Symbol	V _{DD} Vdc	T _{low} [*]		25°C			T _{high} [*]		Unit
			Min	Max	Min	Typ	Max	Min	Max	
Input Voltage (Control) "0" Level IV ₀ = 4.5 or 0.5 Vdc IV ₀ = 9.0 or 1.0 Vdc IV ₀ = 13.5 or 1.5 Vdc	V _{IL}		5.0	-	1.5	-	2.25	1.5	-	1.5
			10	-	3.0	-	4.50	3.0	-	3.0
			15	-	3.75	-	6.75	3.75	-	3.75
IV ₀ = 0.5 or 4.5 Vdc IV ₀ = 1.0 or 9.0 Vdc IV ₀ = 1.5 or 13.5 Vdc	V _{IH}		5.0	3.5	-	3.5	2.75	-	3.5	Vdc
			10	7.0	-	7.0	5.50	-	7.0	
			15	11.25	-	11.25	8.25	-	11.25	
Input Current (AL Device) Control I _{in}	I _{in}	15	-	±0.1	-	±0.0001	±0.1	-	±1.0	nAdc
Input Current (CL/CP Device) Control I _{in}	I _{in}	15	-	±0.3	-	±0.0001	±0.3	-	±1.0	nAdc
Input Capacitance IV _{IN} = 0	C _{IN}									pF
Control Input Switch Inputs		10	-	-	-	5.0	7.5	-	-	
				-	-	8.0	15	-	-	
Output Capacitance C _{OUT}	C _{OUT}	10	-	-	-	8.0	--	-	-	pF
Feedthrough Capacitance C _{IN OUT}	C _{IN OUT}	10	-	-	-	0.5	-	-	-	pF
Quiescent Current (AL Device) (Per Package)	I _Q	5.0	-	0.25	-	0.0005	0.25	-	7.5	μAdc
		10	-	0.50	-	0.0010	0.50	-	15	
		15	-	1.00	-	0.0015	1.00	-	30	
Quiescent Current (CL/CP Device) (Per Package)	I _Q	5.0	-	1.0	-	0.0005	1.0	-	7.5	μAdc
		10	-	2.0	-	0.0010	2.0	-	15	
		15	-	4.0	-	0.0015	4.0	-	30	
ON Resistance (AL Device)	R _{ON}	5.0	-	800	-	250	1050	-	1200	Ω
		10	-	400	-	120	500	-	520	
		15	-	220	-	80	280	-	300	
ON Resistance (CL/CP Device)	R _{ON}	5.0	-	800	-	250	1050	-	1300	Ω
		10	-	450	-	120	500	-	550	
		15	-	250	-	80	280	-	320	
Δ ON Resistance Between Any Two of Four Switches	ΔR _{ON}	5.0	-	-	-	25	-	-	-	Ω
		10	-	-	-	10	-	-	-	
		15	-	-	-	5.0	-	-	-	
Input/Output Leakage Current Switch OFF (AL Device)	-	15	-	±100	-	±0.01	±100	-	±1000	nAdc
Input/Output Leakage Current Switch OFF (CL/CP Device)	-	15	-	±300	-	±0.01	±300	-	±1000	nAdc

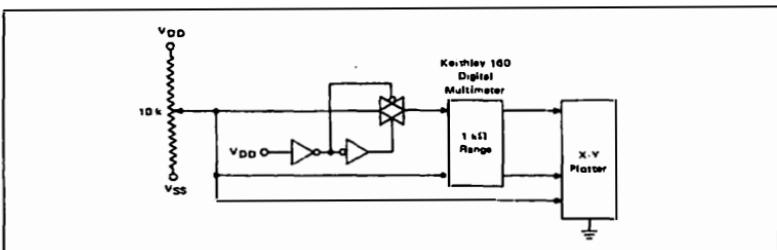
^{*}The formulas given are for the typical characteristics only.T_{low} = -55°C for AL Device, -40°C for CL/CP Device.T_{high} = +125°C for AL Device, +85°C for CL/CP Device.

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. For proper operation it is recommended that V_{in} and V_{out} be constrained to the range V_{SS} < V_{in} or V_{out} < V_{DD}. Unused inputs must always be tied to an appropriate logic voltage level (e.g., either V_{SS} or V_{DD}).

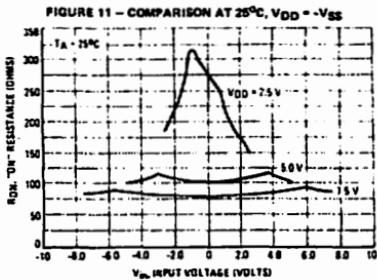
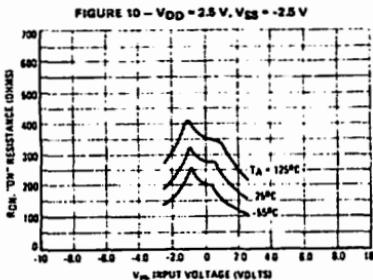
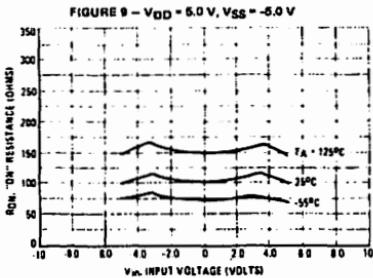
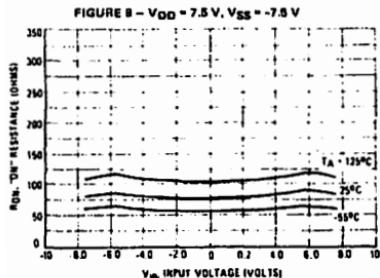
SWITCHING CHARACTERISTICS* ($C_L = 50 \text{ pF}$, $T_A = 25^\circ\text{C}$ unless otherwise noted.)

Characteristic	Symbol	V _{DD} Vdc	Min	Typ	Max	Unit
Propagation Delay Times V _{SS} = 0 Vdc						
Input to Output ($R_L = 10 \text{ k}\Omega$)	t _{PLH} , t _{PHL}	5.0 10 15	— — —	20 10 7.0	40 20 15	ns
t _{PLH} - t _{PHL} = 0.17 ns/pFl C _L + 15.5 ns						
t _{PLH} , t _{PHL} = 10.08 ns/pFl C _L + 6.0 ns						
t _{PLH} , t _{PHL} = 10.08 ns/pFl C _L + 4.0 ns						
Control to Output ($R_L = 1 \text{ k}\Omega$)						
Output "1" to High Impedance	t _{PZH}	5.0 10 15	— — —	40 35 30	80 70 60	ns
Output "0" to High Impedance	t _{PZL}	5.0 10 15	— — —	40 35 30	90 70 60	ns
High Impedance to Output "1"	t _{PZH}	5.0 10 15	— — —	60 20 15	120 40 30	ns
High Impedance to Output "0"	t _{PZL}	5.0 10 15	— — —	60 20 15	120 40 30	ns
Sine Wave Distortion V _{SS} = -5 Vdc	—	5.0	—	0.1	—	%
IV _{in} = 1.77 Vdc, RMS Centered @ 0.0 Vdc, R _L = 10 kΩ, f = 1.0 kHz						
Frequency Response (Switch ON)	V _{SS} = -5 Vdc	—	5.0	—	65	—
IR _L = 1 kΩ, 20 Log ₁₀ $\frac{V_{out}}{V_{in}}$ = -3 dB						MHz
Feedthrough Attenuation (Switch OFF)	V _{SS} = -5 Vdc	—	5.0	—	1.0	—
(R _L = 1 kΩ, 20 Log ₁₀ $\frac{V_{out}}{V_{in}}$ = -50 dB)						MHz
Crosstalk Between Any Two Switches	V _{SS} = -5 Vdc	—	5.0	—	8.0	—
(R _L = 1 kΩ, 20 Log ₁₀ $\frac{V_{out(B)}}{V_{in(A)}}$ = -50 dB, (Switch A ON, Switch B OFF)						MHz
Crosstalk, Control Input to Signal Output	V _{SS} = -5 Vdc	—	5.0	—	300	—
Maximum Control Input Frequency	V _{SS} = 0 Vdc	—	5.0	—	6.0	—
(20 Log ₁₀ $\frac{V_{out}}{V_{in}}$ = -6 dB)			10 15	— —	8.0 8.5	—

*The formulas given are for the typical characteristics only.

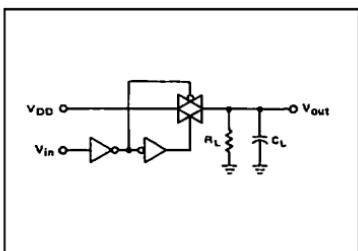
FIGURE 7 - CHANNEL RESISTANCE (R_{ON}) TEST CIRCUIT

TYPICAL RESISTANCE CHARACTERISTICS

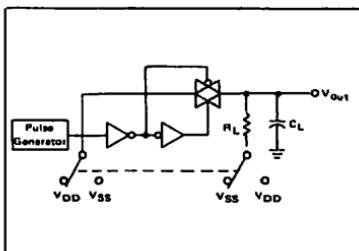


TEST CIRCUITS

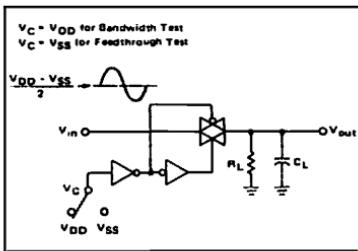
FIGURE 1 – INPUT VOLTAGE



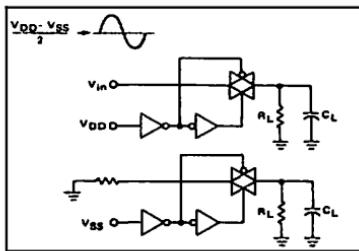
**FIGURE 2 – PROPAGATION DELAY TIME,
CONTROL TO OUTPUT**



**FIGURE 3 – BANDWIDTH AND
FEEDTHROUGH ATTENUATION**



**FIGURE 4 – CROSSTALK BETWEEN
ANY TWO SWITCHES**



**FIGURE 5 – CROSSTALK,
CONTROL TO OUTPUT**

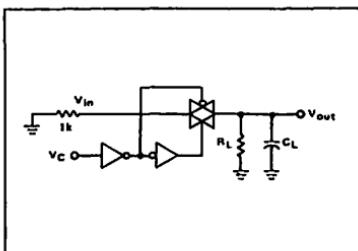
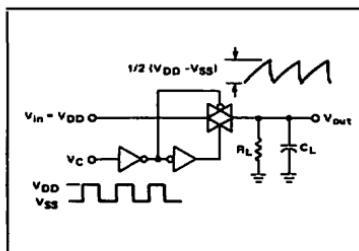


FIGURE 6 – MAXIMUM CONTROL FREQUENCY





MOTOROLA Semiconductors

3001 EDISON ROAD, AUSTIN, TEXAS 78721

MC14584B

HEX SCHMITT TRIGGER

The MC14584B hex Schmitt Trigger is constructed with MOS P-channel and N-channel enhancement mode devices in a single monolithic structure. These devices find primary use where low power dissipation and/or high noise immunity is desired. The MC14584B may be used in place of the MC14069B hex inverter for enhanced noise immunity or to "square up" slowly changing waveforms.

- Quiescent Current = 0.5 nA typ/pkg @ 5 Vdc
- Supply Voltage Range = 3.0 Vdc to 18 Vdc
- Capable of Driving Two Low-Power TTL Loads, One Low-Power Schottky TTL Load or Two HTL Loads Over the Rated Temperature Range
- Double Diode Protection on All Inputs
- Pin-for-Pin Replacement for CD40106B and MM74C14
- Can Be Used to Replace MC14069B

McMOS SSI

(LOW-POWER COMPLEMENTARY MOS)

HEX SCHMITT TRIGGER



L SUFFIX
CERAMIC PACKAGE
CASE 632

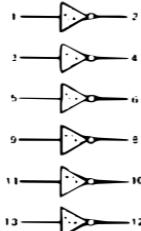


P SUFFIX
PLASTIC PACKAGE
CASE 646

ORDERING INFORMATION

MC14XXXB	Suffix Denotes
	L Ceramic Package
	P Plastic Package
	A Extended Operating Temperature Range
	C Limited Operating Temperature Range

LOGIC DIAGRAM

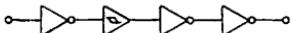


V_{DD} = Pin 14
V_{SS} = Pin 7

MAXIMUM RATINGS (Voltages referenced to V_{SS})

Rating	Symbol	Value	Unit
DC Supply Voltage	V _{DD}	-0.5 to +18	Vdc
Input Voltage, All Inputs	V _{in}	-0.5 to V _{DD} + 0.5	Vdc
DC Current Drain per Pin	I	10	mA/dc
Operating Temperature Range - AL Device	T _A	-55 to +125	°C
CL/CP Device		-40 to +85	
Storage Temperature Range	T _{stg}	-65 to +150	°C

EQUIVALENT CIRCUIT SCHEMATIC (1/6 OF CIRCUIT SHOWN)



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. For proper operation it is recommended that V_{in} and V_{out} be constrained to the range V_{SS} ≤ V_{in} or V_{out} ≤ V_{DD}. Unused inputs must always be tied to an appropriate logic voltage level (e.g., either V_{SS} or V_{DD}).

McMOS is a trademark of Motorola Inc

©MOTOROLA INC. 1978

DS9406

ELECTRICAL CHARACTERISTICS

Characteristic	Symbol	V _{DD} V _{Ds}	T _{low} ^a		25°C			T _{high} ^a		Unit
			Min	Max	Min	Typ	Max	Min	Max	
Output Voltage ^b V _{in} = V _{DD} or 0	V _{OL}	5.0	-	0.05	-	0	0.05	-	0.05	Vdc
		10	-	0.05	-	0	0.05	-	0.05	
		15	-	0.05	-	0	0.05	-	0.05	
	V _{OH}	5.0	4.95	-	4.95	5.0	-	4.95	-	Vdc
		10	9.05	-	9.95	10	-	9.95	-	
		15	14.95	-	14.95	15	-	14.95	-	
Input Voltage ^c V _Q = "0" Level (V _Q = 4.5 or 0.5 Vdc) (V _Q = 9.0 or 1.0 Vdc) (V _Q = 13.5 or 1.5 Vdc)	V _{IL}	5.0	-	1.5	-	2.25	1.5	-	1.5	Vdc
		10	-	3.0	-	4.50	3.0	-	3.0	
		15	-	4.0	-	6.75	4.0	-	4.0	
	V _{IH}	5.0	3.5	-	3.5	2.75	-	3.5	-	Vdc
		10	7.0	-	7.0	5.50	-	7.0	-	
		15	11.0	-	11.0	8.25	-	11.0	-	
Output Drive Current (AL Device) (V _{OH} = 2.5 Vdc) Source (V _{OH} = 4.8 Vdc) (V _{OH} = 9.5 Vdc) (V _{OH} = 13.5 Vdc)	I _{OH}	5.0	-3.0	-	-2.4	-4.2	-	-1.7	-	μAdc
		5.0	0.64	-	-0.51	-0.88	-	-0.38	-	
		10	-1.6	-	-1.3	-2.25	-	-0.9	-	
		15	-4.2	-	-3.4	-8.8	-	-2.4	-	
	I _{OL}	5.0	0.64	-	0.51	0.68	-	0.36	-	μAdc
		10	1.6	-	1.3	2.25	-	0.9	-	
Output Drive Current ICL/CP Device (V _{OH} = 2.5 Vdc) Source (V _{OH} = 4.8 Vdc) (V _{OH} = 9.5 Vdc) (V _{OH} = 13.5 Vdc)	I _{OH}	5.0	-2.5	-	-2.1	-4.2	-	-1.7	-	μAdc
		5.0	-0.52	-	-0.44	-0.88	-	-0.36	-	
		10	-1.3	-	-1.1	-2.25	-	-0.9	-	
		15	-3.6	-	-3.0	-8.8	-	-2.4	-	
	I _{OL}	5.0	0.52	-	0.44	0.88	-	0.36	-	μAdc
		10	1.3	-	1.1	2.25	-	0.9	-	
Input Current (AL Device)	I _{in}	15	-	1.01	-	+0.00001	+0.1	-	+1.0	μAdc
	I _{in}	15	-	+0.3	-	+0.00001	+0.3	-	+1.0	μAdc
	C _{in}	-	-	-	-	5.0	7.5	-	-	pF
Quiescent Current (AL Device) (Per Package)	I _{DD}	5.0	-	0.25	-	0.0005	0.25	-	7.5	μAdc
		10	-	0.50	-	0.0010	0.50	-	15	
		15	-	1.00	-	0.0015	1.00	-	30	
Quiescent Current (CL/CP Device) (Per Package)	I _{DD}	5.0	-	1.0	-	0.0005	1.0	-	7.5	μAdc
		10	-	2.0	-	0.0010	2.0	-	15	
		15	-	4.0	-	0.0015	4.0	-	34	
Total Supply Current ^d (Dynamic plus Quiescent, Per Package) (C _L = 50 pF on all outputs, all buffers switching)	I _T	5.0	-	-	I _T = [1.8 μA/kHz] f + I _{DD}	-	-	-	-	μAdc
		10	-	-	I _T = [3.6 μA/kHz] f + I _{DD}	-	-	-	-	
		15	-	-	I _T = [5.4 μA/kHz] f + I _{DD}	-	-	-	-	
Hysteresis Voltage	V _H ^e	5.0	0.32	1.0	0.30	0.55	1.0	0.26	1.0	Vdc
		10	0.36	1.3	0.30	0.70	1.2	0.26	1.2	
		15	0.90	1.7	0.70	1.1	1.5	0.58	1.4	
Threshold Voltage Positive-Going	V _{T+}	5.0	1.9	3.5	1.8	2.7	3.4	1.7	3.4	Vdc
		10	3.4	7.0	3.3	5.3	6.9	3.2	6.9	
		15	5.2	10.8	5.2	8.0	10.5	5.2	10.5	
Negative-Going	V _{T-}	5.0	1.6	3.3	1.8	2.1	3.2	1.5	3.2	Vdc
		10	3.0	6.7	3.0	4.6	6.7	3.0	6.7	
		15	4.5	8.7	4.8	6.9	9.8	4.7	9.9	

^aT_{low} = -55°C for AL Device, -40°C for CL/CP Device.
T_{high} = +125°C for AL Device, +85°C for CL/CP Device.

^bNoise immunity specified for worst-case input combination.

^cNoise Margin for both "1" and "0" level =

$$1.0 \text{ Vdc min} \oplus V_{DD} = 5.0 \text{ Vdc}$$

$$2.0 \text{ Vdc min} \oplus V_{DD} = 10 \text{ Vdc}$$

$$2.5 \text{ Vdc min} \oplus V_{DD} = 15 \text{ Vdc}$$

^dTo calculate total supply current at loads other than 50 pF:

$$I_T [C_L] = I_T [50 \text{ pF}] + 1 \times 10^{-3} |I_C| \cdot 50 \text{ pF} \cdot V_{DD}$$

where: I_T is mA (per package), C_L in pF, V_{DD} in Vdc, and f in kHz is input frequency.

^eThe formulas given are for the typical characteristics only at 25°C.
ΔV_H = V_{T+} - V_{T-}. (But maximum variation of V_H is specified as less than V_{T+max} - V_{T-min}).



MOTOROLA Semiconductor Products Inc.

SWITCHING CHARACTERISTICS ($C_L = 50 \text{ pF}$, $T_A = 25^\circ\text{C}$)

Characteristic	Symbol	V _{DD} V _{dc}	Min	Typ	Max	Unit
Output Rise Time	t _r	5.0	—	100	200	ns
		10	—	50	100	ns
		15	—	40	80	ns
Output Fall Time	t _f	5.0	—	100	200	ns
		10	—	50	100	ns
		15	—	40	80	ns
Propagation Delay Time	t _{PLH} , t _{PHL}	5.0	—	125	250	ns
		10	—	50	100	ns
		15	—	40	80	ns

FIGURE 1 – SWITCHING TIME TEST CIRCUIT AND WAVEFORMS

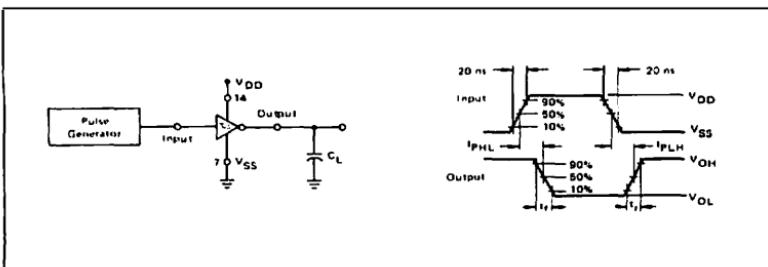
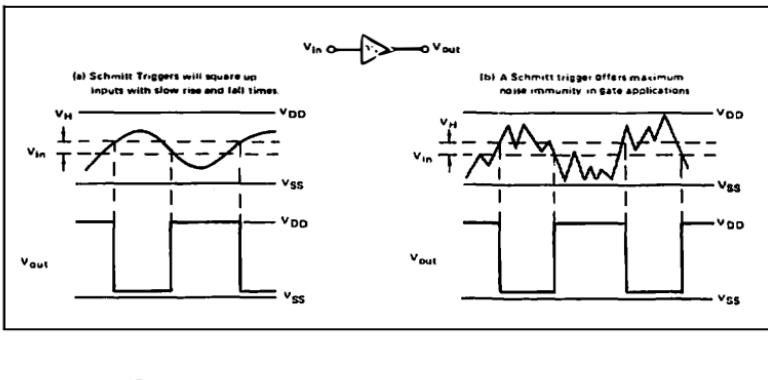
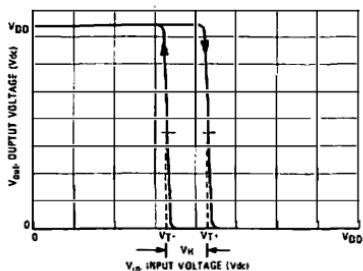


FIGURE 2 – TYPICAL SCHMITT TRIGGER APPLICATIONS



MOTOROLA Semiconductor Products Inc.

FIGURE 3 - TYPICAL TRANSFER CHARACTERISTICS



MOTOROLA Semiconductor Products Inc.

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

DATA SHEET 3501 ED-BLUESTEIN BLVD. AUSTIN, TEXAS 78721

DS-1000



MC6801/03 PORT EXPANSION

Prepared by
Arnold J. Morales
Application Engineer

MC6801/MC6803 PORT EXPANSION

The I/O capabilities of the MC6801/03 can be easily expanded. In effect, the number of I/O ports available can be increased to accommodate user needs with simple designs utilizing relatively inexpensive parts.

This application note describes several methods of port expansion.

MC6801/03 EXPANDED MULTIPLEXED MODE PORT EXPANSION

Figure 1 illustrates several methods for increasing the I/O capability of the MC6801/03 in the expanded multiplexed modes.* All these methods utilize ICs interfaced to the data bus.

Figure 1a shows a means of using the SN74LS374 as a latched input port. A strobe from a remote peripheral or MPU is used to latch data into the LS374. The MC6801/03 can read the latched contents of the LS374 by pulling OE low utilizing E/I/O Select as shown. When not selected the LS374 is held in a high-impedance state, thus eliminating data bus contention.

The SN74LS374 can also be used as a latched output port, as shown in Figure 1b. Data from the MC6801/03 is latched into the LS374 on the falling edge of E when I/O Select is high. Latched data can be continuously presented to the remote peripheral or MPU by tying OE low, as shown, or can be gated by the remote peripheral by an appropriate "I/O Select" logic control of OE.

An unlatched input port can be designed using a SN74LS244, as shown in Figure 1c. The octal three-state buffer presents data to the data bus with E/I/O Select and is particularly suited for MPU read of switches using polling. Once again, when not "selected," this peripheral is held in a high-impedance state, thus eliminating bus contention on the data bus.

A more straightforward approach to port expansion is illustrated in Figure 1d. Here an MC6821 is used to yield a net gain of two handshaking bidirectional ports. The MC6801/03 is compatible with all 6800 family peripherals, including the MC6821. Programming, pinout, and interface information for the MC6821 is provided in its data sheet, readily available from Motorola field offices and distributors.

*The MC6803 is limited to expanded multiplexed (modes 2 and 3) operation.

MC6801/03 SCI PORT EXPANSION

Port expansion is possible using the MC6801/03 SCI (Serial Communications Interface) operated in the standard NRZ format. This format consists of a start bit (low), eight data bits, and one stopbit (high). When no data is being sent the line is held high, indicating an idle line.

All the SCI port expansion schemes described in this application note utilize the SCI clock brought out externally from the MC6801/03 as provided by a software option.

SCI PORT OUTPUT EXPANSION

Figure 2 shows a means of expanding the SCI to 16 sets of 4-bit nibbles yielding 64 I/O output lines.

The key element of this circuit is the start-bit recognition system. The serial bit stream is inverted before reaching the SN74LS164 serial-in, parallel-out shift register. An idle line, therefore, loads the shift register. The two LS74's (U2, U3) with zeros. The SN74LS154 decoder is consequently disabled with E high, keeping its outputs high. During this time no clocking is provided to the SN74LS175's, and their data outputs are unchanged.

Data transmitted out of the SCI is preceded by a startbit. This low start bit is inverted and clocked through the LS164 and the LS74's as a high. One-half clock time after the start bit is clocked into F/F U2, it is clocked to F/F U3. At this time the eight SCI data bits are in the LS164. The LS164 is enabled and decodes the four SLB's of the data (transmitted first). The four MSB's of the SCI data byte are I/O output data and are tied to all LS175's with appropriate buffering.

Decoding by the LS154 clocks the addressed 175, latching in the data.

One-half SCI clock time later, the LS164 is cleared by driving the MR pin low. F/F U2 is also cleared. The system is now ready for another SCI data byte. If no more data is transmitted, it's are transmitted indicating an idle line and output data remains unchanged.

Figure 3 illustrates a scheme for expanding the SCI port to an 8-bit output port.

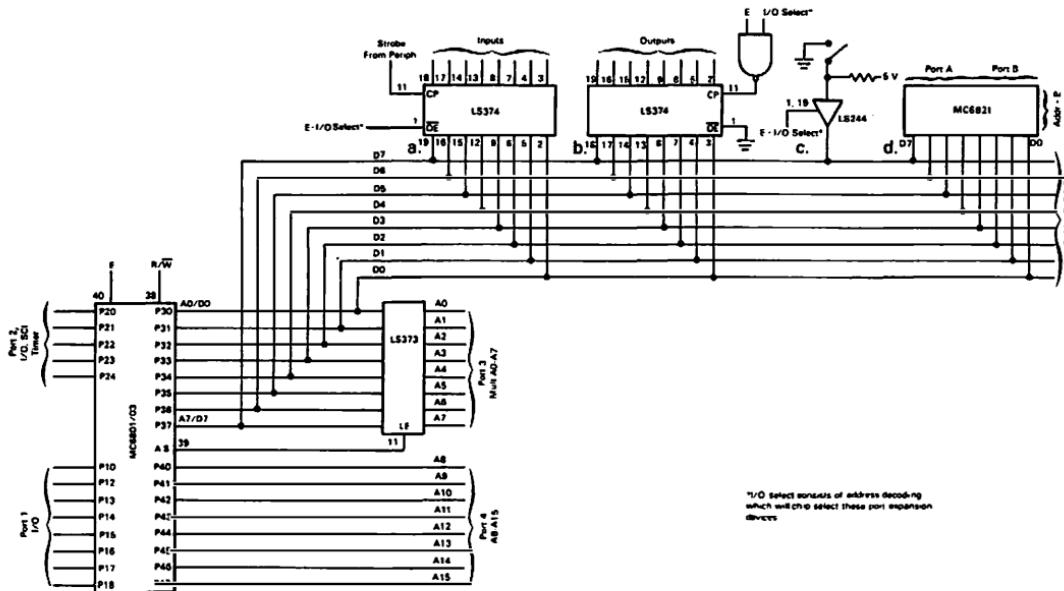


FIGURE 1 — MC6801/MC6803 Expanded Multiplexed Mode Port Expansion

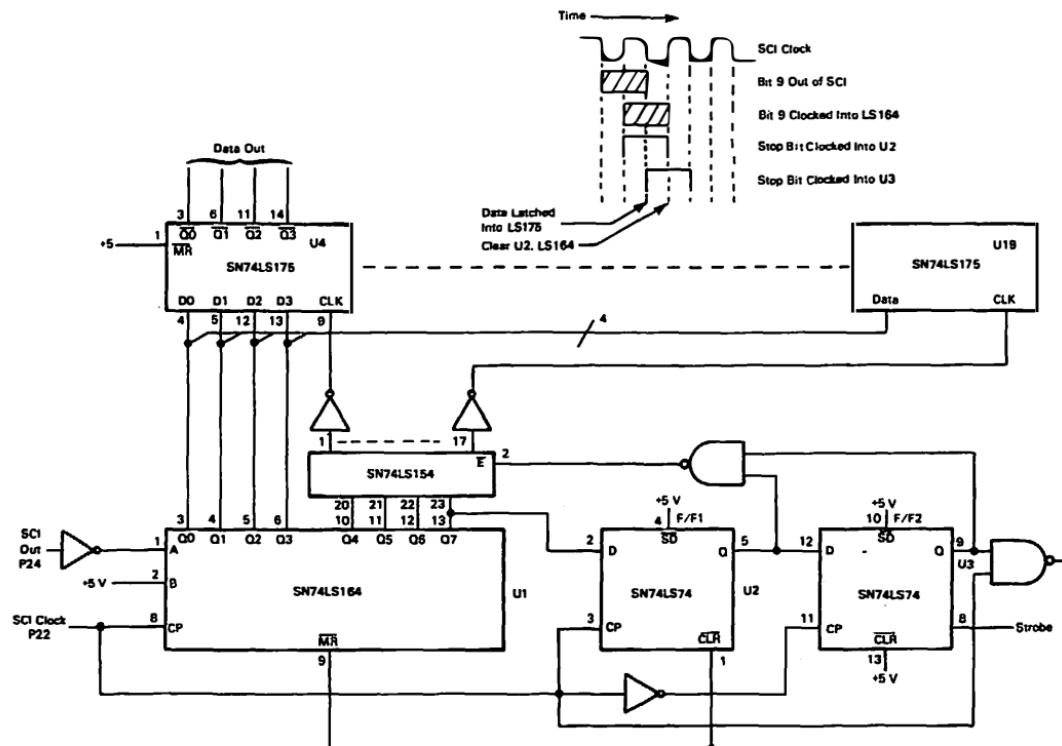
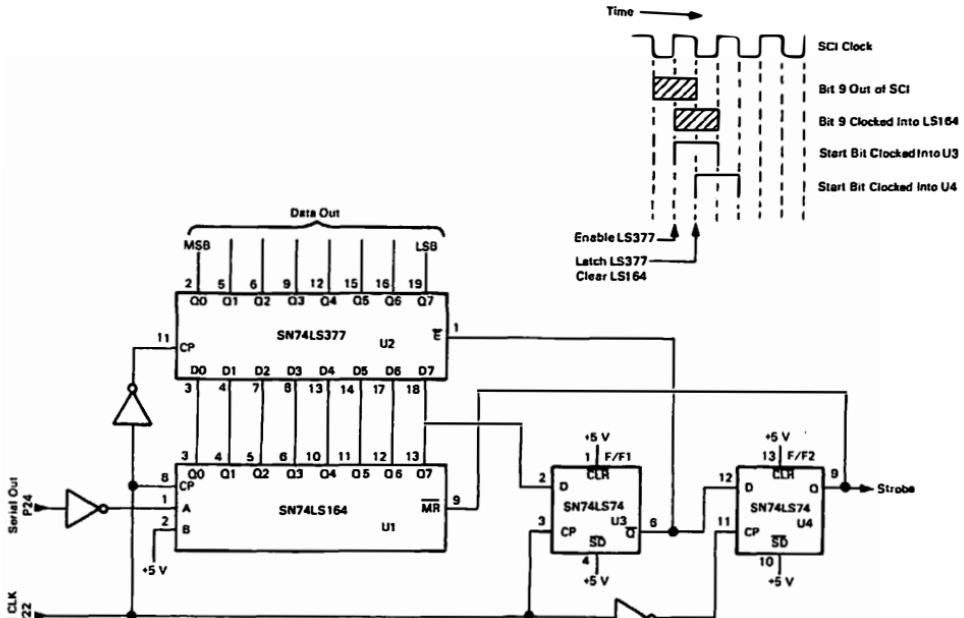


FIGURE 2 — Latched SCI Output Expansion



Note: Data Out is Inverted

FIGURE 3 — SCI Latched Port Expansion

Once again, the SCI bit stream is inverted and an idle line (clocks zeros [lows] into the LS164 and LS74's. In this idle state the SN74LS377 octal "D" F/F is disabled with E high so no new data can be latched in, and the MR pin of the LS164 is held high, enabling clocking of serial input data.

When SCI data is transmitted the start bit is inverted and clocked through the LS164 as a one/low. When the start bit is clocked into the first F/F (U3), the LS377 is enabled. One-half SCI clock time later, the eight databits in the LS164 are latched into the LS377. At the same time the start bit is clocked into the second F/F (U4) and its Q output, with appropriate propagation delay, goes low, pulling LS164 MR low, thus resetting it to all zeros. At this time the system is ready for a new SCI data byte. If the SCI line goes idle (no new SCI data bytes), the LS377 output remains unchanged.

The output of the LS377 is inverted. Non-inverted outputs can be effected by simply complementing data for MC6801/03 SCI transmission, using the appropriate "complement" op code.

SCI PORT INPUT EXPANSION

Figure 4 illustrates a parallel-to-serial interface, designed to input keyboard ASCII characters and clock the data serially into the MC6801/03 SCI port.

The SN74LS165 "serial in" line is tied high so that during an idle period (no keyboard data) 1's are clocked through the LS165 and F/F U4 into the SCI. The SCI thus sees an idle line. F/F U1 and F/F U2 are cleared at this time.

When a key is punched the keyboard strobe clocks a high into F/F U1. This high is clocked into F/F U2 on the falling edge of the SCI clock. When the SCI clock next goes high, F/F U1 and F/F U4 are cleared and LS165 (U3) P/L is driven low, latching the keyboard data. The output of F/F U4, a low, is the start bit. U2 is driven low on the next high-to-low SCI clock transition.

Data is now clocked into the SCI by the SCI clock. Ones (highs) are clocked into the SCI after the eight data bits are clocked in, indicating an idle line. At this time the interface is ready for more data.

Care must be taken to insure that "repeat" characters are not sent by the keyboard while characters are being clocked into the SCI.

MC6801 PORT 3 OUTPUT EXPANSION WITH HANDSHAKING

Port 3 of the MC6801 operated in the single chip mode can be easily expanded using SN74LS377 octal D flip-flops. Figure 5 illustrates one method of expanding port 3 to two 8-bit output ports with handshaking

The "D" inputs of each LS377 are tied to port 3. Port 4 bit 0 (P40) is used to select one of the LS377's and deselect the other by controlling their respective E inputs.

When data is written to port 3, the port 3 strobe (OS3) clocks both LS377's. Only the selected LS377, however, will latch in the data. Nand gates are used to generate the appropriate strobe, OS3A or OS3B.

Further expansion is possible using two or more port 4 outputs for LS377 selection. Software initialization must include setting bit 4 in the Port 3 Control/Status register (60F), so that the OS3 strobe is generated by a write to port 3.

MC6801 PORT 3 INPUT EXPANSION WITH HANDSHAKING

Input expansion of port 3 is illustrated in Figure 6 with the MC6801 operated in the single chip mode.

Initially, F/F U3 and U4 are set. When data is strobed into one of the LS374's, one of the LS74 flip-flops is also strobed, clocking its Q output low. When either LS374 is strobed, the IS3 pin of the MC6801 is driven low, setting the IS3 flag in the Port 3 Control/Status Register. Setting the IS3 IRQ1 Enable bit in the Port 3 Control/Status Register enables the IS3 flag to generate an IRQ1 interrupt, vectored the MPU to a routine for servicing port 3.

When data is strobed into port A, F/F U4 is clocked and its Q output goes low, enabling the output of U1 and making the output of U2 high impedance. When data is strobed into port B the Q output of F/F U4 remains high, enabling the output of U2 and making the output of U1 a high impedance. The Q output of F/F U4 is also tied to port 4 bit 0, programmed as an I/O input.

The MC6801 software responds to the IS3 flag by polling port 4 bit 0 to determine which port has data. The software then reads port 3 data, thus generating an OS3 strobe which sets F/F U3 and U4. The system is now ready for new data.

Software initialization must include initializing the Port 3 Control/Status Register (60F). The IS3 interrupt enable bit (bit 0) may be set if desired. The Output Strobe Select bit (bit 4), cleared during reset, must remain cleared so that the OS3 strobe is generated by an MPU read of port 3. The latch enable bit (bit 3), also cleared during reset, must remain cleared so the port 3 latches remain transparent. Port 3 latching is external in this circuit.

Initialization should also include a read of port 3 to set F/F U3 and U4.

Measures must be taken to insure that data is not strobed into one port while data in another port is being processed. One approach is to inhibit peripheral writes to the port while the output of U5 is low, indicating that port 3 servicing is in progress. Further arbitration logic must be included if the possibility exists of data being strobed into both ports simultaneously.

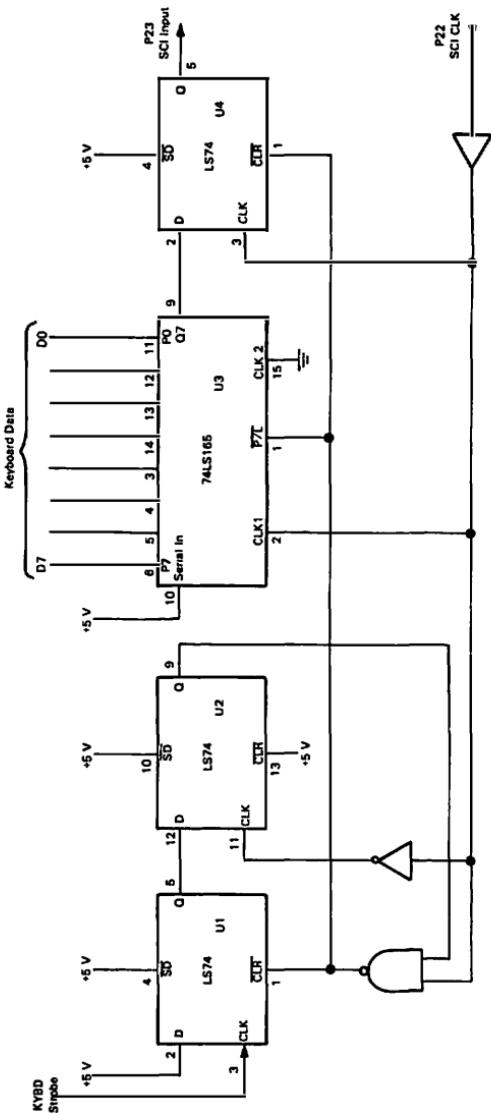


FIGURE 4 – SCI Keyboard Interface

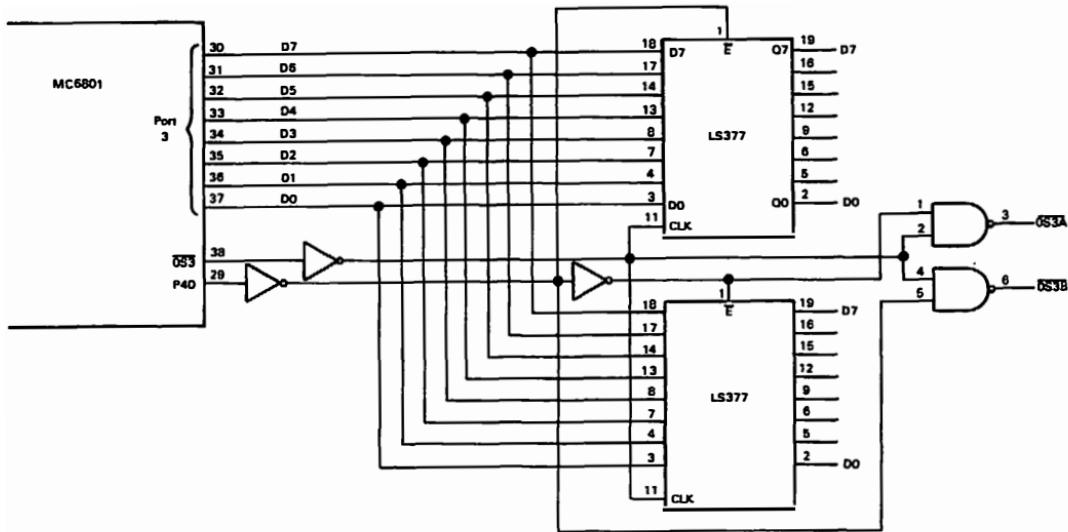


FIGURE 5 — MC6801 Port 3 Expansion (Output)

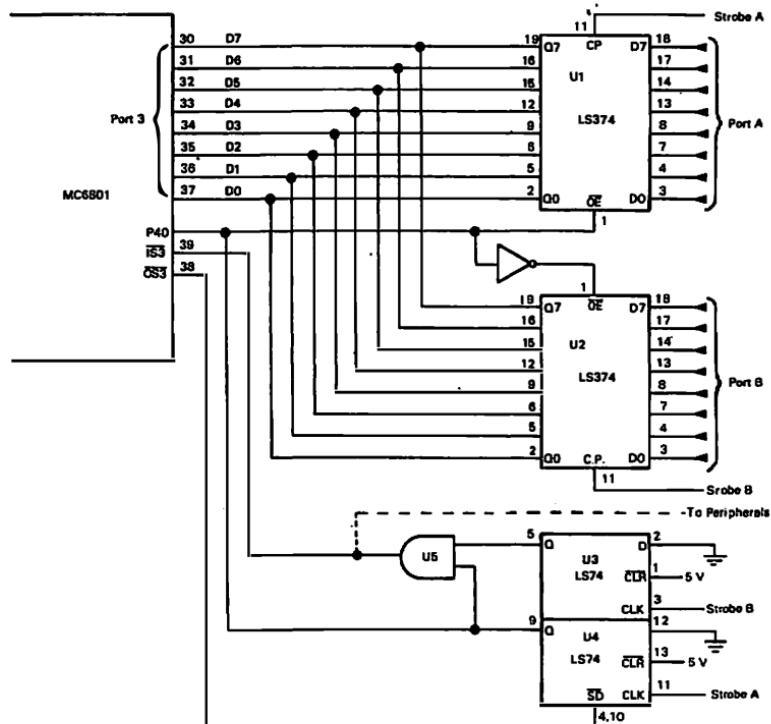


FIGURE 8 — MC6801 Port 3 Expansion (Input)

Motorola reserves the right to make changes to any products or specifications 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.

3601 ED BLUESTEIN BLVD., AUSTIN, TEXAS 78721 • A SUBSIDIARY OF MOTOROLA INC

APPENDIX C

LILbug Monitor Description

The following pages contain a description of the LILbug Monitor. These pages are provided through the courtesy of Motorola Semiconductor Products, Inc., Austin, TX, Copyright 1979.

1.0 MC6801 OVERVIEW

Successful design begins with a clearly stated set of objectives. The principle design objectives driving the MC6801 development have remained constant throughout its evolution: to provide the user with the finest single-chip microcomputer possible today with sufficient flexibility and power to deal with his problems extending into the 1980's. Three clear trends are emerging with respect to future applications: users will continue to strive for minimization of part counts, make increasing use of distributed processing, and automate processes heretofore unimaginable. The MC6801 has demanded the highest skills and talents from its designers in meeting these formidable objectives.

1.1 ARCHITECTURAL INNOVATIONS

The MC6801 represents the state-of-the-art in single-chip microcomputer development. The MC6801 MPU has been enhanced from the M6800 with respect to both its capability and internal architecture. In addition to the MPU, the MC6801 has incorporated an on-chip oscillator driver, an on-chip programmable timer, a serial communications capability, two types of memory and 8 modes of operation. The single most noteworthy accomplishment of the architecture is probably the systems integration of these components into a powerful single-chip microcomputer.

The MC6801 microcomputer may be considered an entire family of processors all of which have been integrated into one common design. This approach was taken at the risk of overwhelming the user with the myriad of options available. The advantages to be gained from the approach are significant:

- a single microcomputer can be used in an extremely wide and vastly different variety of applications.
- the user can build upon his experience with the microcomputer without the fear of early obsolescence, end
- the user's investment in learning and support tools may be amortized over a longer period of time than with a number of more custom designs, hence it also will free the user from a heavy burden of inventory.

1.2 SOFTWARE INNOVATIONS

The M6800 programmer will feel very much at ease with the MC6801 instruction set. All of the familiar instructions of the M6800 have been implemented and function in exactly the same manner. In addition to the M6800 instructions, several new and powerful instructions have been added to ease the problems associated with 16-bit arithmetic problems. The A and B-accumulators have been concatenated into a single, double-accumulator (D) for which a full set of 16-bit arithmetic instructions (including load, store, add, subtract, and shift) are included. These new double accumulator instructions have been implemented with all of the addressing modes available for single accumulator operations.

Indexing has been greatly enhanced by adding a "Add B-accumulator to X" instruction along with the capability to both push and pull the Index register. A hardware multiply instruction has also been added to further aid in arithmetic problems. Improvements in the internal architecture also have resulted in faster cycle times for many instructions.

The MC6801 MPU may be considered an enhanced M6800 MPU which capitalizes on the user's investment with the M6800 Family of Parts. New users will appreciate the widely acclaimed easy-to-learn instruction set.

1.3 SUMMARY OF FEATURES

Hardware

- M6800 Bus Compatible
- Single 5V Power Supply
- 8-Bit Word Size
- 16-Bit Address Field
- TTL-Compatible Inputs and Outputs
- On-Chip Oscillator/Driver
- On-Chip 16-Bit Dual Function (Input and Output)
 - Programmable Timer
- On-Chip Serial Input/Output
- On-Chip 128 Byte RAM
- On-Chip 2K Byte ROM
- Vectored Priority Interrupts for Timer and Serial I/O
- Four Programmable Input/Output Ports
- Eight Hardware Programmable Modes of Operation
- Mask Option for External Clock Input
- Peripheral Controller Mask Option
- EROM version for all Mask Options
- Mask Relocatable ROM Address
- Mask Relocatable RAM Address
- Programmable External Address Space to 64K
- Multiplexed Address/Data Bus
- Valid Address Strobe
- On-Chip Standby RAM for 64 Bytes
- Vectored Restart
- Maskable Vectored Interrupt
- Separate Non-Maskable Interrupt
- Full Duplex Programmable Serial I/O for Either NRZ or Bi-phase
- Four Baud Rate Programmable Selection for Serial I/O

Software

- MC6800 Upward Compatible Architecture
 - Two 8-Bit Accumulators
 - One 16-Bit Index Register
 - One 16-Bit Stack Pointer
- MC6800 Upward Compatible Instruction Set
 - All M6800 Instructions Included (Many contain fewer cycles)
 - M6800 Object Code Compatible
 - 8 x 8-Bit Unsigned Hardware Multiply
 - 16-Bit Arithmetic Capabilities (Load, Store, Add, Subtract, Shift)
 - Additional Index Register Instructions (Push, Pull, Add B to X)

2.0 LILBUG MONITOR OVERVIEW

The standard MC6801 ROM pattern contains the LILbug monitor. This ROM monitor is used to evaluate and debug a program under development. The LILbug monitor commands enable the user to:

- Load a program
- Verify that a program was properly loaded
- Dump the program
- Punch (record) the program on tape
- Examine and change data in a memory location (including I/O)
- Calculate the offset for relative addressing

- Examine and change data in the user's program registers
 - Insert, display, and remove breakpoints in the program
 - Free run or trace through the user's program
- The 6801 LILbug monitor is designed to enable a user to modify the system to meet his specific hardware and software needs. The main areas of flexibility are:
- I/O device independence
 - Interrupt vector independence
 - Command table expandability

3.0 COMMANDS

3.1 COMMAND FORMAT

The general command format is:

<Command> <Delimiter> | <Arg> <Delimiter> | <Arg>]

The command name is followed by a delimiter of space, comma, or carriage return. If there are no arguments, carriage return terminates the command; otherwise a space or a comma separates the command from its argument. A space or comma separates arguments from each other. The ":" and "/" are special quick commands that do not need carriage returns.

3.2 COMMAND DESCRIPTION

LOAD L
L <Offset>

Load a tape file. Figure 3.2 shows the monitor's tape format. Load with offset is available by specifying the hexadecimal <offset>. The offset is added to the address on tape to get the load address. All offsets are positive, but wrap around memory modulo 64K. For example, to offset a load by minus \$1000 locations, use \$F000 as offset. An attempt to load into ROM, protected RAM, or non-existent memory terminates the command after printing "?". A checksum error also prints "?" and terminates. On some tape readers, several extraneous characters are transmitted before the tape reader is turned off. These characters will sometimes print after the prompt on the console.

Example:
 12012 500 400 080 300 440 BF0
 2018 00 1A0 7F0 650 EE0
 !L
 12012 50 40 05 05 30 44 BF 1A 7F
 13012-42 05 03 03 C0 60 FD F0 FF
 !L 1000
 13012 50 40 08 08 30 44 BF 1A 7F

VERIFY V
V <Offset>

Verify or compare the contents of memory to the tape data. A hexadecimal <offset> can be specified. A checksum error, a bad compare, or non-existent memory prints "?" and then terminates

Example:

```
!V 1000
!V
!
```

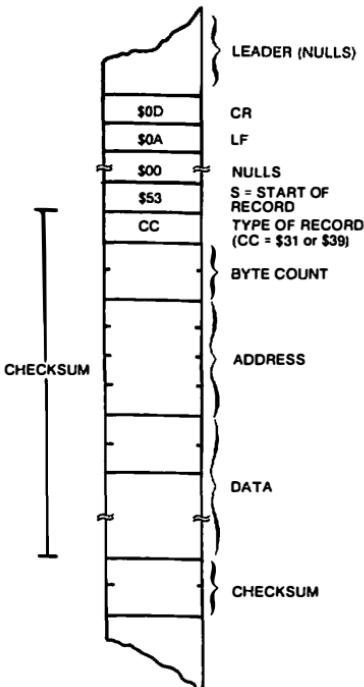


Figure 3.2 — Tape Format

PUNCH P <Adr1>, <Adr2>

Punch or record formatted binary object tape as shown in figure 3.2. Start at <Adr1> and punch through <Adr2>. This routine fails to diagnose an end address smaller than the beginning address. With low speed the values punched to tape may also print to console. High speed may cause some garbage characters to print to the console during the punch.

Example:

!P 2012 2046

```
$11B2012604008083044BF1A7FEE63EEE4A604480100038911819EEF71
$11B202AFFAFB3EEFB1603852031003912FCFFB91DFF7F7BAF7FE031114
$11B20420818123432608F65E8F7FCEF2FE716904280109F0751EEFEF1E
$11B205RA#FFFF798926C0801A00C802C4DDA2FFF6FFESDFE737101346
$11B20722008151280888ABFE85CD8D7DANED0040001F8B462A94DFFF51
$11B208AFFAEC9EF4BFF0206C30802511708FA7F6IDFFEEED2FBB40447D
$10820A2814508629570
$90200000FC
```

DISPLAY D <Adr1>, <Adr2>

```
D <Adr1>
D
```

Display contents of memory in hexadecimal followed by the literal ASCII characters. Start at <Adr1> and print thru <Adr2>. Ignore the command if the beginning address is larger than the end address. The command prints blocks of 16 bytes — <Adr1> is masked back to the beginning of a block and <Adr2> is extended to the end of a block. If only one argument is specified, display a block of 48 words surrounding address <Adr1>. If no argument is given display a block of 48 words containing the last location used in memory examine.

Example:

ID 2012 2056

	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F		
\$010	01	1A	63	40	00	00	30	44	FF	1A	7F	6E	43	EE	E4	46	
\$020	04	4A	63	01	00	00	99	11	81	9E	EF	FF	4F	4B	46	H.....3..V	
\$030	02	55	63	81	00	00	39	12	FC	FF	99	EB	F3	F7	4B1.Y...8=..T..	
\$040	05	11	63	18	10	54	30	00	EF	99	89	80	F2	FC	EF	2F	E7
\$050	16	90	46	80	10	9F	01	91	EF	EF	FF	FF	FF	79	89	26B....0..+....8

ID 2045

	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F		
\$030	03	85	20	21	00	00	39	10	FC	FF	89	F1	F5	F7	EA	F7	FE
\$040	03	11	63	18	10	54	30	00	EF	69	E1	F7	FC	EF	2F	E7TF...3...V
\$050	16	90	46	80	10	9F	07	91	EF	EF	FF	FF	FF	79	89	26B....0..+....8

MEMORY M <Adr>(Space) <Adr>/

Initiate the memory examine/change function. Print the value at <Adr> and maintain a pointer to that address. <Adr> is 1 to 4 hex digits not counting leading zeroes, and it must be followed by a delimiter of space or slash. The form <Adr>/ cannot start with characters 'A' through 'F'. If the address would normally start with 'A' thru 'F', zero should precede the address. For example: 0FFE/ instead of FFFE/. The user may execute a combination of the following sub-commands to modify the current memory location, look at consecutive words of memory, move the memory pointer, or terminate memory examine.

- <DATA> — Enter one byte of data to replace the value at the current pointer. There is nosizecheck on the data. If <DATA> is more than two digits, the right-most byte of data becomes the new value.
 - SPACE — Increment the current pointer and print the value at the new address on the same line.
 - . — (Comma) Increment the pointer to the next memory location with no print of address or value. This allows insertion of data without seeing current values in memory.
 - LF — (Line feed) Increment the pointer to the next memory location, print the address and value at that address on the next line.
 - UA — (Up arrow) Decrement the pointer to the previous memory location. Print the new address and its value on the next line.
 - / — (Slash) Print the current address and the value on the next line
 - CR — (Carriage return) Terminate the memory examine command
- Any other input terminates the memory examine command.
- Example:**
- ```

IM 2000 30 * MEMORY EXAMINE
!2000/30
!FFE/
? * Address cannot start with
 * "A" thru "F"
!OFFFE/30 00 * Zero should precede the address
!2000/3024 3248 3021 * Change 30 to 24 at address 2000
 * Space moves pointer to address 2001
 * where 32 is changed to 48

!2005/201,2,3,4,5 * At location 2005 start inserting
 * values 1, 2, 3, 4, and 5

!2005/01 02 03 04 05 * Look at values just inserted

!2005/01
2006 02 * Line feed moves pointer to next
2007 03 * memory location

!2007/03^ * Up arrow moves pointer to previous
2006 02^ * memory location
2005 01

!2005/01/ * Slash recalls current address
2005 01/ * printing address and value
2005 01

! * Carriage return terminates Memory
 * Examine - prompt indicates ready
 * for next command

```

**SLASH /**

Recall the location last referenced in memory examine and print that address and the value at that address. No carriage return or delimiter is necessary.

Example:

```
!/2005 01 * Slash recalls current address,
 * printing address and value
```

**OFFSET O <Adr1><Adr2> {CR}**

Calculate the offset (second byte) of a branch instruction from <Adr1> to <Adr2>. <Adr1> should be the address of the second byte of a branch instruction. <Adr2> is the absolute address for the branch. The command prints the offset if it is within the -127 to +128 byte range, otherwise it prints ? and returns to main control loop. If the range is valid, <Adr1> is printed and the user can modify that location.

Example:

```
!O 2034 209A
65
2034 0065
!O 2034 2012
DD
2034 65DD
!O 2034 3000
?
!
```

**REGISTER R**

Display the contents of the program registers and counter. The registers are printed in the format:

P-XXXX X-XXXX A-XX B-XX C-XX S-XXXX

P-

where:

P = program counter

X = index register

A = accumulator A

B = accumulator B

C = condition code register

S = stack pointer

XX = values in registers

Register P is ready for update. New values may be inserted or the register change function can be terminated.

The following inputs are valid:

<DATA> — Insert new value for the current register. There is no validity check on the length of the data, so the right-most byte or double byte, whichever is appropriate for the register becomes the new register value.

SPACE — Print register value unless <DATA> changed its value. Move the register pointer to the next register, printing the mnemonic and a dash.

Any other input characters terminate the register command. The command terminates after examining and/or updating the stack pointer.

Example:

```
!P
P-0000 X-0000 A-00 B-00 C-D0 I-BB
P-2000 X- 0000 A-123456 B- 00 C- D0 I-A0
!P
P-2000 X-0000 A-56 B-00 C-D0 I-A0
```

**BREAKPOINT B <Adr>**

B-<Adr>

B

B-

Enter a breakpoint at address <Adr> and print current breakpoints. A maximum of four are allowed.

A question mark is printed if 4 breakpoints are already set.

B-<Adr>

Remove the breakpoint at address <Adr> and print the remaining breakpoints.

B

Display all breakpoints.

B-

Remove all breakpoints and show empty breakpoint table.

Examples:

```
!B 2002
2002 0000 0000 0000
!B 2045
2002 2045 0000 0000
!B -2002
0000 2045 0000 0000
!B 1234
1234 2045 0000 0000
!B -
1234 2045 0000 0000
!B -
0000 0000 0000 0000
!
```

**GO G <Adr>**

G

Execute starting from <Adr>. Execution will continue until a breakpoint (3F) is detected.

G

Execute starting at the current program counter setting.

**CALL C <Adr>**

C

Call and execute a user routine as a subroutine starting at address <Adr>. A return address to the monitor is stored in the users stack. Breakpoints are recognized and trace can be used after initiating a "C" command. When the user "RTS" is executed, the monitor gets control, prints registers and returns to the main calling routine. The user PC than points into the monitor.

C

Same as C <Adr> except start executing from the current program counter.

**TRACE T <Hex number>**

Trace the specified number of instructions where <hex number> is a maximum of \$FFFF. The opcode, program registers, and counter are printed after each instruction.

The format of the print line is:

OP-<Op> P-XXXX X-XXXX A-XX B-XX C-XX S-XXXX

where: <Op> = opcode of last instruction executed

P = program counter

X = index register

A = accumulator A

B = accumulator B

C = condition code register

S = stack pointer

XX = value in register

**Example:**

```
P-2002 X-1 0A1 A-00 B-2F C-D0 S-00A0
!-3
DP-01 P-2003 X-1 001 A-00 B-2F C-D0 S-00A0
DP-01 P-2004 X-1 001 A-00 B-2F C-D0 S-00A0
DP-01 P-2005 X-1 001 A-00 B-2F C-D0 S-00A0
```

**NEXT .**

The character '.' (period) with no carriage return will trace the next instruction and print the trace line.

Example:

```
!-3
DP-4C P-2006 X-1 001 A-01 B-2F C-D0 S-00A0
DP-7E P-2000 X-1 001 A-01 B-2F C-D0 S-00A0
```

**CONTROL X**

Return control to the main command loop. Control 'X' is recognized during display and trace print out, and whenever the monitor is reading console input.

**CONTROL W**

Halt print and wait until input character signals continuation of print. This command is recognized during trace print and during display memory print. Print is halted until any character is input, then print continues.

**HIGHER HI**

Set speed to 1200 baud (+1024), set padding and set bits in on-chip serial I/O RMCR (rate and mode control register). When changing speeds, tape readers and punches should be off-line to prevent accidental read or write during the transition.

**HIGHER YET HY**

Set speed to 9600 baud (+128), set zero padding. This is for CRT terminals. Tape recorders and punches should be off-line when changing speeds.

**4.0 MONITOR OPTIONS**

Since versatility is built into the MC6801, versatility, expandability, and ease of use were the primary design criteria of LILbug. LILbug can be used in either the single chip or expanded modes of the MC6801. It can be configured to use on-chip I/O, standard external devices (e.g., ACIA, VDG, PTM), or user defined external I/O (e.g., keyboards, displays). All interrupt vectors are accessible to the user and can be redefined to point to user written interrupt routines. Furthermore the user can define additional variable length commands or redefine existing ones to suit his needs. This flexible design will hopefully allow LILbug to become a de facto standard for the basic monitor of all MC6801 based systems.

Often versatility is traded for ease of use. This is not true for LILbug. The monitor initializes all variables to convenient defaults so that a user who wishes to become familiar with the MC6801 and use its on-chip I/O need not even be aware of the more esoteric features of LILbug. All he needs to do is build the simple circuit explained in section 5.1, find a RS232 compatible terminal, push reset and his system is up and running.

**4.1 IO INDEPENDENCE**

Monitor I/Os are indirect. All I/O calls are to a devicetype:

1. single-byte input (console keyboard)
2. single-byte output (console print)
3. high speed print
4. high speed punch/load device

Each device has three routines associated with it:

1. initialization
2. data in/out
3. device termination

Before a device is used by the monitor it will issue a call to the routine to initialize that device. Normal data transfer is done thru the data in/out routines and devices are terminated thru the device termination routine. Data transfers can be of two types: serial or block. In serial transfers the byte of data is passed to or from the data in/out routines in the A-register. For the high speed print and high speed punch/load, the address of a data packet is passed to the in/out routine. See Appendix B routines HSDTA and BDSDTA for details of the format of the data packet.

The addresses for these device routines are reordered in a table. The address of the table is stored in 'IOPTR' at RAM location \$0FC. The monitor calls an I/O routine by setting register B as an index into the I/O table, and then calling routine IO. Routine IO retrieves the address from the table and does a subroutine call.

The LILbug monitor default to the on-chip serial input and output routines for the Silent 700S. The monitor's high speed routine writes blocks of data to the console output device. The default high speed punch/load device routines are for a Silent 700S cassette. The monitor's RESTART code initializes 'IOPTR' to point to its default table. Following is the default I/O table as defined in LILbug.

Silent 700 is a registered trademark of Texas Instruments.

```

00171 ##### IO TABLE #####
00172 ■ ROUTINE IO IS CALLED WITH
00173 ■ INDEX INTO IO TABLE CI OR INTO USER IO TABLE
00174 ■ IOPTR POINTS TO THE IO TABLE TO BE USED
00175 ■ THE INDEX TABLE DEFINES ORDER OF IO ROUTINES IN IO TABL

00176A F85B F8C8 A C1 FDB CION,CIDTA,CIOFF
 A F85D F891 A
 A F85F F8D1 A
00177A F861 F8D2 A FDB COON,CODTA,COOFF
 A F863 F8A8 A
 A F865 F8D1 A
00178A F867 F8D1 A FDB HSOM,HSDTA,HSOFF
 A F869 FDEC A
 A F86B F8D1 A
00179A F86D FEAF A FDB BSON,BSDTA,BSOFF
 A F86F FEC3 A
 A F871 FEBA A

00181 ■ THE FOLLOWING ARE INDICES INTO IO TABLE
00182 0000 A C1.ON EQU 0 INIT INPUT DEVICE
00183 0002 A C1.DTA EQU 2 INPUT A CHAR W/HO WAIT
00184 0004 A C1.OFF EQU 4 DISABLE INPUT DEVICE
00185 0006 A C0.ON EQU 6 INIT OUTPUT DEVICE
00186 0008 A C0.DTA EQU 8 OUTPUT A CHAR W/PADDING
00187 000A A C0.OFF EQU 10 DISABLE OUTPUT DEVICE
00188 000C A HS.ON EQU 12 INIT HIGH SPEED OUTPUT DEVICE
00189 000E A HS.DTA EQU 14 OUTPUT BLOCK OF DATA
00190 0010 A HS.OFF EQU 16 DISABLE HIGH SPEED DEVICE
00191 0012 A BS.ON EQU 18 INIT PUNCH/LOAD
00192 0014 A BS.DTA EQU 20 WRITE DATA BLK TO PNCH/LOAD
00193 0016 A BS.OFF EQU 22 DISABLE PUNCH/LOAD
00194 ■

```

The user can redefine the I/O by writing his own routines and putting the addresses of the routines in his table. The routine addresses should be in the same order as shown in the previous table. The user may wish to redefine only part of the I/O routines. In this case, he should fill in the table with the monitor's addresses for those routines he does not change. He must then initialize 'IOPTR' with the address of his table.

The program — USERIO (Appendix C) — is an example of redefining the I/O to use an ACIA instead of the on-chip serial I/O. USERIO defines the initialization and character input/output routines using an ACIA. The other I/O routines used are from the monitor.

#### 4.2 INTERRUPT VECTORS

The 6801 modes define internal or external vectors. If the vectors are internal, they are in the LILbug ROM. The monitor is designed to allow any of the ROM vectors except RESTART to be user defined. The interrupt

address is retrieved indirectly from a table whose address is in 'VECPTR' at RAM location \$0E. The table must contain addresses or dummy addresses for seven interrupts. The monitor's restart code initializes 'VECPTR' but the user may define his own vector table in ROM or RAM by changing 'VECPTR'.

| Word | Vector Table   |
|------|----------------|
| \$0  | Serial         |
| 2    | Timer Overflow |
| 4    | Timer Output   |
| 6    | Timer Input    |
| 8    | IRQ1 "         |
| A    | SWI            |
| C    | NMI            |

Following is a listing of the vector table defined in LILbug.

```

01621 ##### VECTORS #####
01622 ■ VECTOF. INDEPENDENCE
01623 ■ ALSO SAVE ON RAM USAGE
01624 ■ VECPTF - RAM PINTP TO VECTOR TABLE
01625 ■ VECTOR TABLE - ADR OF INTERRUPT VECTORS
01626 ■ MAY BE REDEFINED BY USER TABLE IN SAME FORM
01627A FFCA FCE1 A SERIAL FDE DUMMY NOT USED BY MONITOR
01628A FFCC FCE1 A TIMEOUT FDE DUMMY
01629A FFCC FCE1 A TIMEOUT FDE DUMMY
01630A FFCE FCE1 A TIMIN FDE DUMMY
01631A FFD0 FCE1 A IRQ1 FDB DUMMY
01632A FFD2 F821 A SWI FDB INT.SWI
01633A FFD4 F823 A NM1 FDB INT.NMI
01634 ■ DUMMY IS AN RTI

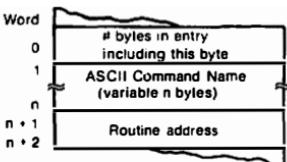
```

The monitor defines SWI and NMI. All other interrupts default to an 'RTI' instruction. The LILbug SWI is used for the monitor breakpoint command. The NMI is used by trace and breakpoint. The NMI vector controls the type of hardware tracing, either on-chip timer, or PTM (Programmable Timer Module).

The internal vector default uses the on-chip timer entry. Address \$F803 is the NMI vector for trace using the on-chip timer, and address \$F800 is for the PTM trace processing. This vector should not be changed after monitor initialization. The initialization defines the timer according to the address in the NMI vector. The chip must be in an expanded mode to use a PTM. SWI and NMI can be defined for other uses, if breakpoint and trace are not used.

#### 4.3 USER-DEFINED COMMANDS

The LILbug monitor has a command table consisting of entries in the form:



The monitor reads a command and does a search of the command table to find a match with the ASCII command name. If a match is found, a subroutine call is made to the routine address following the command name. If no match is found, a question mark is printed. The user can define new commands and have his command table searched prior to the monitor command table. The address of the command table is stored in 'FCTPTR' at RAM location \$0FA. Command names are of variable length. Decimal numbers are not allowed in command names. The characters '/' and '.' are special quick commands that cannot be redefined. A terminating word marks the end of the table. A negative one indicates the monitor table is also to be searched. A negative two means the monitor commands are not being used. Since the user table is always searched first, the user can redefine existing commands. Program USERIO (Appendix C) is an example of redefining LILbug commands. USERIO defines ACIA I/O and redefines the commands, HI and HY, to change output padding for ACIA baud rate change.

Figure 4.3 is the command table as defined by LILbug.

```

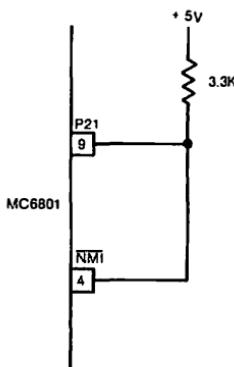
00110
00111
00112
00113
00114
00115
00116
00117
00118
00119
00120
00121
00122
00123
00124
00125
00126
00127
00128
00129
00130A F824 A FCTABL EOU X
00131A F825 A FCB 4 ■
00132A F826 F898 A FCC /B/ ■
00133A F828 A FDB BRKPNT
00134A F829 A FCB 4 ■
00135A F82A FC0F A FCC /C/
00136A F82C A FCB CALL
00137A F82D A FCB 4 ■
00138A F82E FD86 A FDB /D/
00139A F830 A FCB 4 ■
00140A F831 A FCC /G/
00141A F832 FC11 A FDB GOXQT
00142A F834 A FCB 4 ■
00143A F835 A FCC /L/
00144A F836 FE9A A FDB LOAD
00145A F838 A FCB 4 ■
00146A F839 A FCC /M/
00147A F83A FASB A FDB MEMORY
00148A F83C A FCB 4 ■
00149A F83D A FCC /O/
00150A F83E FAB4 A FDB OFFSET
00151A F849 A FCB 4 ■
00152A F841 S0 A FCC /P/
00153A F842 FE72 A FDB PUNCH
00154A F844 A FCB 4 ■
00155A F845 S2 A FCC /R/
00156A F846 FB1E A FDB REGSTR
00157A F848 S5 A FCB 5 ■
00158A F849 A FCC /HI/
A F84A 49 A
00159A F84B F8EB A FDB S120
00150A F84D S5 A FCB 5 ■
00161A F84E A FCC /HY/
A F84F 59 A
00162A F859 F8F1 A FDB HY
00163A F852 S4 A FCB 4 ■
00164A F853 S4 A FCC /T/
00165A F854 FC46 A FDB TRACE
00166A F856 S4 A FCB 4 ■
00167A F857 S6 A FCC /V/
00168A F858 FEAB A FDB VERF
00169A F859 FE A FCB -2 ■END OF TABLE

```

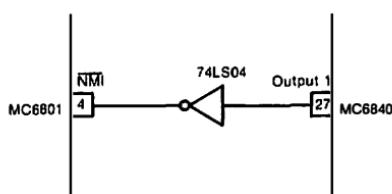
Figure 4.3

#### 4.4 HARDWARE TRACE

Hardware trace permits a user's program to be executed one instruction at a time. Trace and breakpoint functions need this ability. The trace will operate on programs in either RAM or ROM. The circuitry as diagrammed below consists of NMI tied to the output of the on-chip timer or to the inverted output of a PTM (Programmable Timer Module). Figure 4.4 shows the output waveform and timing for the on-chip timer and the PTM.



Hardware Trace Using the MC6801 Timer



Hardware Trace Using the PTM

Using the on-chip timer, Port 2 is set for output and the output level bit in the TCSR (Timer Control/Status Register) is set low. The on-chip clock is read. That time is adjusted and stored in the OCREG (Output Compare Register) such that the compare occurs after the first cycle of the next user instruction. After storing the OCREG, the monitor does an "RTI". When the compare to the OCREG occurs, the output wave goes low (corresponding to low setting of level bit in TCSR). The low transition causes an NMI. NMI vectors back to the monitor. The output waveform is brought high by setting the level bit in TCSR and by setting OCREG for another compare.

If a PTM is selected for trace instead of the on-chip timer, timer 1 is used. NMI is tied to the inverted output. To execute one user instruction, latches are set such that the output level changes after the first cycle of the user instruction. The PTM output level returns to its initial level after one cycle.

#### 4.5 INITIALIZATION

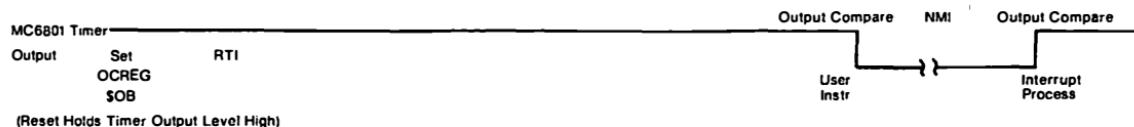
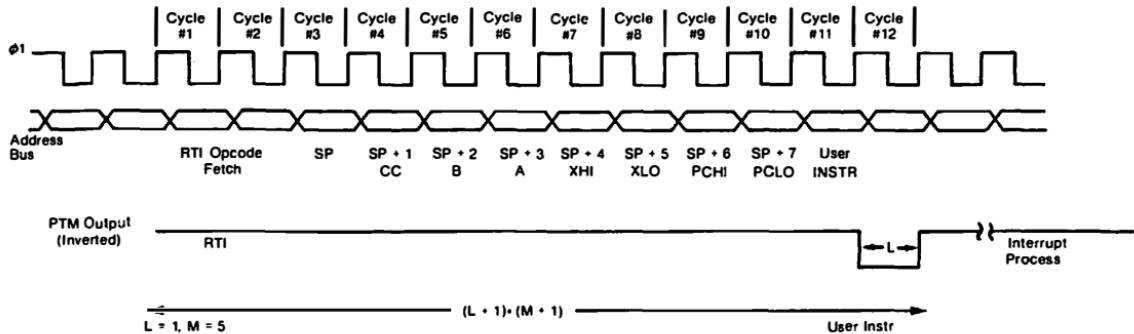
An MC6801 mode that best allocates resources for the user's software must be selected. The MC6801 has eight hardware selectable modes controlling on-chip use of ROM, RAM, interrupt vectors, and I/O ports. Figures 4.5.1 and 4.5.2 are the mode select chart and mode overview.

Modes 2, 3, and 4 cannot use the monitor since the internal ROM is not used. Mode 0 is a test mode designed to check on-chip memory. Mode 7 is minimal having only the on-chip ROM (monitor) and 128 bytes of on-chip RAM. The monitor uses about half of the on-chip RAM. The monitor stack is at address \$0CF, and RAM variables are from \$0D0 to \$0FF. The user has the low addresses in RAM. Modes 1, 5, and 6 allow development of large programs since they can have expanded memory. In modes 5 and 6, the user must program port 1 as output to have external memory. Modes 5 and 6 have internal vectors while mode 1 has external vectors. Mode 1 is most easily used with programs that use PTM trace, or programs that redefine I/O devices. For these programs, external devices (and addresses) are necessary. Also, external vectors allow the user to jump to his own initialization routine at RESET.

If the 6801 is in a mode that has internal vectors and LILbug enabled, the RESET vector initializes the monitor's on-chip serial I/O and on-chip timer for hardware trace and breakpoint. The vector table, command table, and I/O tables are initialized. Calls are made to turn on console reader and printer and I/O devices are initialized.

If the user selects a monitor option, such as adding user-defined commands to the command table, a mode with external vectors is usually more convenient. The RESET vector would then jump directly to a user initialization routine. The user needs to set the command table pointer, and then jump into the monitor RESET code such that the table pointer is not redefined to the monitor default. This may cause omitting other initialization which the user routine needs to do prior to entering the monitor. An example is USRCMD in Appendix C. The routine defines a conversion command. The program sets the pointer to the user-command table, and also the pointer to the monitor I/O table. USRCMD then jumps to the monitor code following the monitor setting of the command table pointer.

When the user selects mode 1, which allows external vectors with the internal ROM, the MC6801 will input data from the external data bus for the addresses \$FFF0 through \$FFFF. This function allows the user to define his own vector table in the MC6801 memory map. A full decode of the external memory is not required, due to the architecture of the MC6801, which selects either the internal data bus or external data bus to read the data for these sixteen memory locations.



Hardware Trace  
NMI Tied To PTM or On-Chip  
Timer Output  
Figure 4.4

| MODE | PC2 | PC1 | PC0 | ROM  | RAM  | INTERRUPT VECTORS | BUS MODE | OPERATING MODE                 |
|------|-----|-----|-----|------|------|-------------------|----------|--------------------------------|
| 7    | H   | H   | H   | I    | I    | I                 | I        | Single Chip                    |
| 6    | H   | H   | L   | I    | I    | I                 | MUX (6)  | Multiplexed/Partial Decode     |
| 5    | H   | L   | H   | I    | I    | I                 | NMUX (6) | Non-Multiplexed Partial Decode |
| 4    | H   | L   | L   | I(2) | I(1) | I                 | I        | Single Chip Test               |
| 3    | L   | H   | H   | E    | E    | E                 | MUX      | Multiplexed/No RAM or ROM      |
| 2    | L   | H   | L   | E    | I    | E                 | MUX      | Multiplexed/RAM                |
| 1    | L   | L   | H   | I    | I    | E                 | MUX      | Multiplexed/RAM & ROM          |
| 0    | L   | L   | L   | I    | I    | I(3)              | MUX      | Multiplexed Test               |

#### LEGEND:

I — Internal      MUX — Multiplexed      L — Logic "0"  
 E — External      NMUX — Non-Multiplexed      H — Logic "1"

#### NOTES:

- (1) Internal RAM is addressed at \$XX80
- (2) Internal ROM is disabled
- (3) RESET vector is external for 2 cycles after RESET goes high
- (4) Addresses associated with Ports 3 and 4 are considered external in Modes 3 and 4
- (5) Addresses associated with Port 3 are considered external in Modes 5 and 6
- (6) Port 4 default is user data input; address output is optional by writing to Port 4 Data Direction Register

MC6801 Mode Selects  
Figure 4.5.1

#### OVERVIEW OF MC6801 OPERATING MODES

##### COMMON TO ALL MODES:

- Reserved Register Area
- I/O Port 1 Operation
- I/O Port 2 Operation
- Timer Operation
- Serial I/O Operation

##### SINGLE CHIP-MODE 7

- 128 bytes of on-chip RAM; 2048 bytes of on-chip ROM
- Port 3 is parallel I/O handshaking port
- Port 4 is parallel I/O port

##### EXPANDED MEMORY SPACE/NON-MULTIPLEXED BUS-MODE 5

- 128 bytes of on-chip RAM; 2048 bytes of on-chip ROM
- Port 3 is 8-bit data bus
- Port 4 is optional 8-bit address bus
- 256 bytes of external memory space

##### • External memory space select output (IOS)

##### EXPANDED MEMORY SPACE/MULTIPLIED BUS-MODES 1, 2, 3, 6

- Port 3 is multiplexed address/data bus
- Port 4 is address bus
- 4 memory space options
  - (1) No Internal RAM or ROM (Mode 3)
  - (2) Internal RAM (Mode 2)
  - (3) Internal RAM and ROM (Mode 1)
  - (4) Internal RAM, ROM with partial address bus (Mode 6)

##### TEST-MODES 0 and 4

- Expanded Test-Mode 0
  - May be used to test Internal RAM and ROM
- Single Chip and Non-Multiplexed Test-Mode 4
  - (1) May be changed to Mode 5 without RESET
  - (2) May be used to test Ports 3 and 4 as I/O ports

Figure 4.5.2

#### 4.6 JUMP TABLE

LILbug subroutines and entries frequently used are in a jump table at the beginning of LILbug. These entries should be used whenever possible to insure compatibility with future versions of the monitor. Following is the jump table as defined in LILbug:

|                                           |              | ORG    | FF800                        |
|-------------------------------------------|--------------|--------|------------------------------|
| <b>000956</b> * JUMP TABLE TO SUBROUTINES |              |        |                              |
| 00097A F800 7E FC7E                       | A EX.NMI JMP | H.NMI  | NMI VECTOR FOR PTM           |
| 00098A F803 7E FC79                       | A IN.NMI JMP | C.NMI  | NMI VECTOR FOR ON-CHIP TIMER |
| 00099A F806 7E F873                       | A INCWNP JMP | INCH1  | INPUT 1 CHAR W/ NO PARITY    |
| 00100A F809 7E F88A                       | A OUTCH JMP  | OUTCH1 | OUTPUT 1 CHAR W/PADDING      |
| 00101A F80C 7E FB07                       | A JMP        | PDATA1 | PRINT DATA STRING            |
| 00102A F80F 7E FB0E                       | A JMP        | PDATA2 | PR CR/LF, DATA STRING        |
| 00103A F812 7E FAD8                       | A JMP        | OUT2HS | PR 2 HEX + SP (X)            |
| 00104A F815 7E FAP8                       | A JMP        | OUT4HS | PR 4 HEX + SP (X)            |
| 00105A F818 7E FB12                       | A JMP        | PCRLF  | PRINT CR/LF                  |
| 00106A F81B 7E FADD                       | A JMP        | SPACE  | PRINT A SPACE                |
| 00107A F81E 7E F8F6                       | A STRT JMP   | START  | RESTART ADDRESS              |
| 00108A F821 7E FD5F                       | A IN.SHI JMP | H.SHI  | SHI VECTOR                   |

Following is a description of the jump table entries and their associated subroutines.

| Entry  | Address | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|--------|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| M.NMI  | \$F800  | NMI entry for hardware trace using a PTM. The LILbug monitor initializes the PTM if the NMI vector is this address.                                                                                                                                                                                                                                                                                                                                                                                             |
| C.NMI  | \$F803  | NMI entry for hardware trace using on-chip timer. The LILbug monitor initializes the on-chip timer if the NMI vector is this address. This is the default NMI vector.                                                                                                                                                                                                                                                                                                                                           |
| INCH1  | \$F806  | Call I/O routine CIDTA to input one character into register A. Carry bit is clear until data read, then it is set. INCH1 waits for input, clears parity, and ignores rereads. OUTCH1 is called to echo input if OUTSW is clear.                                                                                                                                                                                                                                                                                 |
| CIDTA  | ---     | I/O routine that reads one character of data into register A using on-chip serial I/O. This is a read with no waiting for data. Return with carry bit set if data is read, otherwise carry bit is clear.                                                                                                                                                                                                                                                                                                        |
| OUTCH1 | \$F809  | Call I/O routine to output one character from register A. Saves register B. Default LILbug I/O routine called by OUTCH1. Call OUTC to use on-chip serial interface to output one character. Writes nulls (padding) after output character. OUTSW and CHRNL control padding. OUTSW is the tape flag; set to 0 if not tape, set to number of nulls to follow character if tape. The upper six bit value of CHRNL is the padding for a carriage return. The low two bit value is the padding for other characters. |
| CODTA  | ----    | Call OUTC to use on-chip serial interface to output one character. Writes nulls (padding) after output character. OUTSW and CHRNL control padding. OUTSW is the tape flag; set to 0 if not tape, set to number of nulls to follow character if tape. The upper six bit value of CHRNL is the padding for a carriage return. The low two bit value is the padding for other characters.                                                                                                                          |
| PDATA1 | \$F80C  | Print data string pointed to by register X. Last character of data string should be \$04. Registers A and X are modified. Call OUTCH1 to print one character at a time.                                                                                                                                                                                                                                                                                                                                         |
| PDATA  | \$F80F  | Call PCRLF to print carriage return and line feed. Then call PDATA1 to print data string pointed to by register X. Registers A and X are modified.                                                                                                                                                                                                                                                                                                                                                              |
| OUT2HS | \$F812  | Output two hexadecimal characters pointed to by register X. Then output a space. Modifies register A. Calls OUT2H and SPACE.                                                                                                                                                                                                                                                                                                                                                                                    |
| OUT2H  | ----    | Print two hexadecimal characters pointed to by register X. Calls OUTHL and OUTHR to convert and print a byte.                                                                                                                                                                                                                                                                                                                                                                                                   |
| OUTHL  | ----    | Convert left four bits of a byte to display code and output. Calls OUTCH1 to print.                                                                                                                                                                                                                                                                                                                                                                                                                             |
| OUTHR  | ----    | Convert right four bits of a byte to display code and output. Calls OUTCH1 to print.                                                                                                                                                                                                                                                                                                                                                                                                                            |
| OUT4HS | \$F815  | Call OUT2H and OUT2HS to print four hexadecimal characters pointed to by register X.                                                                                                                                                                                                                                                                                                                                                                                                                            |

|       |        |                                                                           |
|-------|--------|---------------------------------------------------------------------------|
| PCRLF | \$F818 | Call OUTCH1 to output a line feed and then a carriage return.             |
| SPACE | \$F81B | Call OUTCH1 to output a space.                                            |
| START | \$F81E | Reset entry point. This is the default RESTART vector.                    |
| M.SWI | \$F821 | Default SWI vector in LILbug monitor. This routine processes breakpoints. |

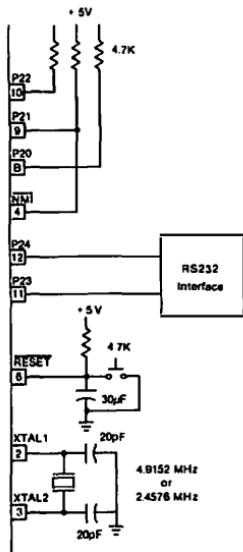
## 5.0 HARDWARE REQUIREMENTS

### 5.1 MINIMUM SYSTEM

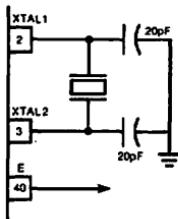
The minimum system for the MC6801 requires only a clock generator, serial communication interface, and reset logic. This minimum system is shown in Figure 5.1. CLOCK— The two connections XTAL1 and XTAL2 can either be used for a parallel resonant crystal or an external clock source as shown in Figures 5.2 and 5.3.

The frequency of the crystal or clock source should be either 4.9152 MHz or 2.4576 MHz to give a standard baud rate for the serial communication interface. The following table shows the baud rates available with these crystals.

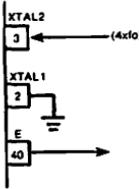
| XTAL  | 4.9152 MHz | 2.4576 MHz |
|-------|------------|------------|
| φ2    | 1.2288 MHz | .6144 MHz  |
| HY    | 9,600 Baud | 4,800 Baud |
| HI    | 1,200 Baud | 600 Baud   |
| Reset | 300 Baud   | 150 Baud   |



Minimum System Requirements  
Figure 5.1



Connections For A Parallel Resonant Crystal  
Figure 5.2



Connections For An External Clock  
Figure 5.3

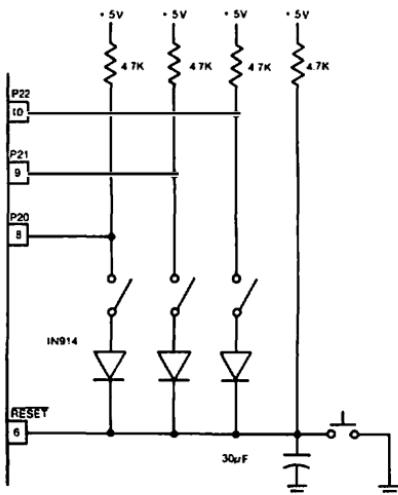
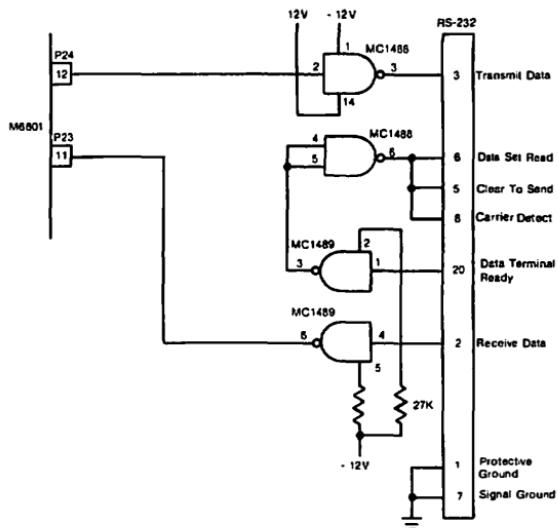
### Serial Communication Interface

The MC1488 and MC1489A devices provide the system with a RS-232C interface capability. The circuit in Figure 5.4 interfaces the serial input/output of the MC6801 directly to an RS-232C terminal without any additional hardware.

### Reset

The Reset Function should be performed when power is first applied and if the LILbug Firmware loses program control. The Reset Function is also used to select the mode of operation of the MC6801. The circuit shown in Figure 5.5 is a simple reset function. The diodes are used to select the different modes of operation during reset.

**Serial Communication Interface**  
Figure 5.4



**Simple Reset Logic**  
Figure 5.5

## MC6801 MONITOR COMMAND SUMMARY

L [<offset>] Load a program from tape. Add <offset>, if specified, to load address on tape.  
V [<offset>] Verify that a program was properly loaded. Add <offset>, if specified, to load address on tape.  
D [<adr1> [,<adr2>]] Display memory from <adr1> to <adr2>.  
P <adr1> , <adr2> Punch or record contents of memory from <adr1> to <adr2>.  
M <adr> Set pointer to <adr> and print value at that memory location.  
Next input:  
    <data> Change one byte in memory to <data>  
    (LF) Increment pointer and print address and value of new pointer.  
    (UA) Up arrow, decrement pointer, print address and value of pointer.  
    ( ) Space, increment pointer and print new value on same line.  
    , Comma, increment pointer with no print of address or value.  
    / Slash, print address and value of current pointer.  
    (CR) End memory examine command.  
<adr>/ Same as M <adr>; <adr> must start with 0-9, zero may precede hex address.  
/ Print address and value of location last referenced by memory examination.  
O <adr1> <adr2> Calculate relative offset from <adr1> to <adr2> for branch instructions.  
B Display all breakpoint addresses.  
B <adr> Set a breakpoint at address <adr> and display breakpoints.  
B-<adr> Remove the breakpoint at address <adr> and display breakpoints.  
B- Remove all breakpoints and display breakpoints.  
G [<adr>] <Adr>, if specified, becomes the new program counter. Execute from program counter.  
R Display/modify user's MPU registers.  
Next input:  
    <data> Change value of register to <data>  
    ( ) Space, display register value and move pointer to next register.  
    (CR) Terminate register change command.  
Period. Trace one instruction. (No carriage return after period).  
T [<hex value>] Trace <hex value> number of commands.  
C [<adr>] <Adr>, if specified, becomes the new program counter. Execute code as subroutine starting  
at address in program counter. User 'RTS' returns to monitor.  
HI Set speed for on-chip I/O to 1200 baud (+1024).  
HY Set speed for on-chip I/O to 9600 baud (-128).

# Index

## A

- ABX (inherent), 28
- Accumulators A, B, and D, 22-23
- Acquisition, remote data, 185
- ADDD (immediate, direct, extended, indexed), 28
- Addressing**
  - direct, 24
  - extended, 24-25
  - indexed, 25
  - indicating instruction, 37-38
  - modes and instruction set, 6801, 24-38
  - relative, 25
  - switch, 145
- Analog
  - signals, 144
  - to-digital converters, interfacing with, 164-166
- Angle, timing, 181-183
- ASLD (inherent), 26
- Asynchronous
  - serial data transmission format, 127
  - transmission, 125
- Automobile applications, 175-184
- Automobile engine
  - monitor process flow, 178
  - rpm, dwell, and timing angle flowchart, 183
  - signal
    - detection and conditioning, 179
    - measurements and calculations, 178
  - requirements, 107

## B

- Basic timer modes of operation, 109
- Baud rate, 126
- Biphase format, 129
- Bit
  - parity, 126
  - rate generator, 131
- Branch destination, 25
- BRN (relative), 33-34
- Byte
  - mask, 146
  - status, 146

## C

- CHCK command, 88
- Chip layout of functional 6801, 13
- Circuit, debouncing, 149
- Clock, 18
  - signal, serial, 131
- Common
  - anode LED display, 155
  - cathode LED display, 155
- Communications, serial, 125-128
- Condition codes register (CCR), 24
- Controller, remote data-acquisition, 184
- Counter (0009:000A), 103
- Coupler control circuit, 61
- CPU, 14
- CPX instruction, 34

## D

- Data register, receive, 132

**D**ebouncing  
circuit, 149  
switch, 145  
**D**ecimal adjust accumulator  
instruction, 25  
**D**ecoding, switch, 145  
**D**estination, branch, 25  
**D**etecting switch closure, 145  
**D**etection and conditioning, signal,  
176-180  
**D**evice, temperature monitoring,  
173-175  
**D**igital-to-analog converters, inter-  
facing with, 162-164  
**D**irect addressing, 24  
**D**isplays, interfacing with, 155-162

## E

**E**cho monitor circuit, 120  
**E**nable  
    input capture interrupt, 105  
    output compare interrupt, 106  
    timer overflow interrupt, 106  
**E**ncoded keyboard, 153  
**E**PROM, 76  
**E**xamples  
    continuous pulse generation,  
        111-112  
    decoding a switch column, 36, 151  
    detecting switch closure and pro-  
        viding software debouncing,  
        148-149  
    event counting, 114-115  
    event timing (interval and  
        frequency measurement),  
        116-117  
    input handshaking via port 3, 67  
    input to output echo, 119-120  
**NE5018** data transfer program, 163  
    output handshaking via port 3, 70  
    output of 7-segment LED character  
        designations via port 1, 157  
    polling routine for encoded key-  
        boards, 154  
    7-segment LED character designa-  
        tion look-up table routine,  
        158  
    7-segment LED multiplexing sub-  
        routine, 161  
    using the multiply (MUL)  
        instruction, 35  
**E**xpanded  
    chip structure, 6801, 14-19  
    multiplexed mode, 16, 50-52  
        address strobe (pin 39), 52

**E**xpanded—cont  
*multiplexed mode*  
    enable (pin 40), 52  
    port 3 (pins 30-37), 51  
    port 4 (pins 22-29), 50-51  
    read/write (pin 38), 51-52  
nonmultiplexed mode, 16, 48-50  
    enable (pin 40), 50  
    i/o select (pin 39), 50  
    port 3 (pins 30-37), 49-50  
    port 4 (pins 22-29), 49  
    read/write (pin 38), 50  
**E**xtended addressing, 24-25

## F

**F**lag  
    input capture, 104  
    output capture, 105  
    timer overflow, 105

**F**ull duplex, 128

**F**undamental instructions, 6801, 25

## G

**G**eneral features of the 6801, 12-14  
**G**enerating an output waveform,  
    110-113

**G**enerator, bit rate, 131

## H

**H**alf duplex, 128

**H**exadecimal display representations,  
    159

## I

**I**ndexed addressing, 25

**I**ndex register (IX), 23

**I**ndicating instruction addressing,  
    37-38

**I**nherent instructions, 24

**I**nput

    capture flag, 104  
    capture register (000D-000E), 104  
    device handshaking-single chip  
        mode, 68

    edge bit, 107

**I**nstruction(s)

    CPX, 34

    decimal adjust accumulator, 25

    inherent, 24

    JSR, 25

    RTS, 25

    store accumulator D, 35

**I**nterfacing

    on encoded keyboard, 153

    Intersil ICL 7109 a/d converter, 165

- I**
- Interfacing—cont
    - Motorola MC1408-8/MC1508-8
      - d/a converter, 164
      - single 7-segment display, 160
    - Signetics NE5018 d/a converter, 163
    - switch column to the 6801, 37
    - Teledyne 8703 a/d converter, 165
    - to a switch
      - column, 150
      - matrix, 152
    - with
      - analog-to-digital converters, 164-166
      - digital-to-analog converters, 162-164
      - displays, 155-162
      - switches and keyboards, 145-155
  - Internal chip structure, 15
  - Interrupt vectors memory map, 106
  - I/O ports, 16-17
  - IRQ, 14
  - IRQ2/timer overflow, 106
  - IS3
    - enable—input strobe enable (bit 6), 65
    - flag—input strobe flag (bit 7), 65
- J**
- JSR instruction, 25
- K**
- Keyboards, 151-155
    - encoded, 153
    - unencoded, 153
- L**
- Latch
    - enable (bit 3), 64
    - octal-D type, 72
  - LDD (immediate, direct, extended, indexed), 28-28
  - LED display, 155
    - common-anode, 155
    - common-cathode, 155
  - LILbug, 85
  - Logic
    - negative, 156
    - positive, 155
  - Look-up table, 156
  - Loop-to-EIA converter circuit, 173
  - LSRD (inherent), 28
- M**
- Mask byte, 146
- M**
- Mark/space format, 129
  - Memory map, 6801, 77-82
  - Metallization, 84
  - Mode
    - expanded multiplexed, 16, 50-52
    - expanded nonmultiplexed, 16, 48-50
    - single-chip, 16, 41-48
  - MUL (inherent), 28
  - Multiplexing LED displays, 157
- N**
- Negative logic, 156
  - NMI, 14
- O**
- Octal D-type latch, 72
  - Operation, SCI, 138-141
  - Output
    - capture flag, 105
    - compare register (000B:000C), 103-104
    - device handshaking—single-chip mode, 69
    - level bit, 107
    - strobe select (bit 4), 64
    - waveform, generating, 110-113
  - Overflow, IRQ2/timer, 106
  - Overrun framing error, 132-133
- P**
- Parity bit, 126
  - Performing measurements on an input waveform, 114-121
  - Peripheral isolation during mode selection, 60-62
  - PGE, 83
  - PGE/PLE truth table, 92
  - PLE, 83
  - Pointer, stack, 24
  - Point signal measurements, 181
  - Port(s)
    - access and control, 6801, 62-64
    - i/o, 16-17
      - 1, 16
      - 2, 16-17
      - 3, 17
      - 4, 17
  - Positive logic, 155
  - PRObug commands, 88
  - PROG command, 88
  - Program counter (PC), 23
  - Programming, 68701 EPROM, 86-93
  - PSHX (inherent), 28-30
  - PULX (inherent), 30

**R**

RAME, 82  
Rate and mode control register, 135-137  
Receive  
  data register, 132  
  operation flowchart, 140  
Receiver  
  enable, 134  
  interrupt enable, 133  
Register  
  condition codes, 24  
  index, 23  
  rate and mode control, 135-137  
  transmit data, 132  
  transmit/receive control and status, 132-135  
Relative addressing, 25  
Remote data acquisition, 185  
  controller, 184  
RESET circuit, 68701 EPROM  
  programming, 87  
ROM, 16  
Rpm and point dwell, 180-181  
RS-232C, serial communication, 170-173  
RTS instruction, 25  
R/W memory, 14  
R/W memory/EPROM control  
  register, 82

**S**

Schmitt trigger, 61  
SCI  
  internal structure, 130-137  
  modes and data formats, 128-130  
  operation, 138-141  
  registers, 131  
Serial clock signal, 131  
Serial communications, 125-128  
  interface, 18, 125  
    bit times and band rates, 130  
    data formats, 129  
    via RS-232C, 170-173  
Signal detection and conditioning, 176-180  
Single-chip mode, 16, 41-48  
  enable (pin 40), 48  
  interrupt request (pin 5), 44  
  nonmaskable interrupt (pin 4), 43  
  port 1 (pins 13-20), 46-47  
  port 2 (pins 8-12), 46  
  port 3 (pins 30-37), 47-48  
  port 3 input strobe (pin 39), 48  
  port 3 output strobe (pin 38), 48

Single-chip mode—cont  
  port 4 (pins 22-29), 47  
  RESET (pin 6), 44  
  V<sub>cc</sub> (pin 7), 45  
  V<sub>cc</sub> Standby (pin 21), 47  
  V<sub>m</sub> (pin 1), 42-43

6801  
  addressing modes and instruction set, 24-38  
  expanded chip structure, 14-19  
  fundamental instructions, 25  
  general features of, 12-14  
  interrupt vector map, 45  
  masked programming ROM, 84-85  
  memory map, 77-82  
  mode selection and additional  
    operating modes, 56-60  
    modes 1, 2, and 3, 58  
    modes 0 and 4, 58-60  
  port access and control, 62-64  
  programming internal register  
    format, 22-24  
  serial communications interfacing  
    modes, 128  
6801/6803 alphabetic mnemonic op-code listing, 31-33  
6803, 93-97  
  functional block diagram and pin  
    assignments, 95  
6803/6803NR  
  memory maps, 96  
  special purpose register functions, 97  
68701, 85-93  
  EPROM programming, 86-93  
    flowchart, 94  
Software interrupt (SWI), 25  
Special purpose register functions, 78  
Stack pointer (SP), 24  
STANDBY PWR bit, 83  
Status byte, 148  
STD (direct, extended, indexed), 30  
Store accumulator D instruction, 35  
Structure, timer, 102-109  
SUBD (immediate, direct, extended,  
  indexed), 33  
Switches, 145-151  
  addressing, 145  
  closure, detecting, 145  
  debouncing, 145  
  decoding, 145  
Synchronous transmission, 125

**T**

Table, look-up, 156

- Temperature monitoring device, 173-175  
Time line diagram of continuous output pulse, 113  
Timer, 18  
    control and status register (0008), 104-105  
    examples, 110-121  
    overflow flag, 105  
    structure, 102-109  
Timing angle, 181-183  
    measurement—multiplexing circuit, 182  
Transmission  
    asynchronous, 125  
    synchronous, 125  
Transmit  
    data register, 132  
    empty, 133  
    enable, 134  
    interrupt enable, 134  
    receive control and status register, 132-135
- TSCR, 102  
20-mA current loop, 172  
Typical 7-segment LED display, 156
- U**
- UART, 124  
Unencoded keyboard, 153  
Using the  
    on-chip ROM, 84-85  
    on-chip R/W memory, 82-83  
    6801 to multiplex displays, 160
- USRT, 124
- V**
- VERF command, 90  
VLSL, 12
- W**
- Wake up on next message, 135
- Z**
- Z-flag, 146

## **READER SERVICE CARD**

To better serve you, the reader, please take a moment to fill out this card, or a copy of it, for us. Not only will you be kept up to date on the Blacksburg Series books, but as an extra bonus, we will randomly select five cards every month, from all of the cards sent to us during the previous month. The names that are drawn will win, absolutely free, a book from the Blacksburg Continuing Education Series. Therefore, make sure to indicate your choice in the space provided below. For a complete listing of all the books to choose from, refer to the inside front cover of this book. Please, one card per person. Give everyone a chance.

In order to find out who has won a book in your area, call (703) 953-1861 anytime during the night or weekend. When you do call, an answering machine will let you know the monthly winners. Too good to be true? Just give us a call. Good luck.

If I win, please send me a copy of:

---

I understand that this book will be sent to me absolutely free, if my card is selected.

For our information, how about telling us a little about yourself. We are interested in your occupation, how and where you normally purchase books and the books that you would like to see in the Blacksburg Series. We are also interested in finding authors for the series, so if you have a book idea, write to The Blacksburg Group, Inc., P.O. Box 242, Blacksburg, VA 24060 and ask for an Author Packet. We are also interested in TRS-80, APPLE, OSI and PET BASIC programs.

My occupation is \_\_\_\_\_

I buy books through/from \_\_\_\_\_

Would you buy books through the mail? \_\_\_\_\_

I'd like to see a book about \_\_\_\_\_

Name \_\_\_\_\_

Address \_\_\_\_\_

City \_\_\_\_\_

State \_\_\_\_\_ Zip \_\_\_\_\_

**MAIL TO: BOOKS, BOX 715, BLACKSBURG, VA 24060  
!!!!PLEASE PRINT!!!!**

## The Blacksburg Group

According to Business Week magazine (Technology July 6, 1976) large scale integrated circuits or LSI "chips" are creating a second industrial revolution that will quickly involve us all. The speed of the developments in this area is breathtaking and it becomes more and more difficult to keep up with the rapid advances that are being made. It is also becoming difficult for newcomers to "get on board."

It has been our objective, as The Blacksburg Group, to develop timely and effective educational materials and aids that will permit students, engineers, scientists and others to quickly learn how to apply new technologies to their particular needs. We are doing this through a number of means, textbooks, short courses, and through the development of educational "hardware" or training aids.

Our group members make their home in Blacksburg, found in the Appalachian Mountains of southwestern Virginia. While we didn't actively start our group collaboration until the Spring of 1974, members of our group have been involved in digital electronics, minicomputers and microcomputers for some time.

Some of our past experiences and on-going efforts include the following:

—The development of the Mark-8 computer, an 8008-based device that was featured in Radio-Electronics magazine in 1974, and generally recognized as the first widely available hobby computer. We have also designed several 8080-based computers, including the Mini-Micro Designer (MMD-1). More recently we have been working with 8085-based computers and the TRS-80.

—The Blacksburg Continuing Education Series™ covers subjects ranging from basic electronics through microcomputers, operational amplifiers, and active filters. Test experiments and examples have been provided in each book. We are strong believers in the use of detailed experiments and examples to reinforce basic concepts. This series originally started as our Bugbook series and many titles are now being translated into Chinese, Japanese, German and Italian.

—We have pioneered the use of small self-contained computers in hands-on courses aimed at microcomputer users. The solderless breadboarding modules developed for use in circuit design and development make it easy for people to set up and test digital circuits and computer interfaces. Some of our technical products are marketed by Group Technology, Ltd., Check, VA 24072, USA. (703) 651-3153.

—Our short course programs have been presented throughout the world, covering digital electronics through TRS-80 computer interfacing. Programs are offered through the Blacksburg Group and the Virginia Tech Extension Division. Each course offers a mix of lectures and hands-on laboratory sessions. Courses are presented on a regular basis in Blacksburg, and at various times to open groups, companies, schools, and other sponsors.

For additional information about course offerings, we encourage you to write or call Dr. Chris Titus at The Blacksburg Group, Box 242, Blacksburg, VA 24060, (703) 951-9030, or Dr. Linda Leffer at the Center for Continuing Education, Virginia Tech, Blacksburg, VA 24061, (703) 961-5241.

Mr. David Larsen is on the faculty of the Department of Chemistry at Virginia Polytechnic Institute and State University. Dr. Jonathan Titus and Dr. Christopher Titus are with The Blacksburg Group, Inc., all of Blacksburg, Virginia.

# **6801, 68701, & 6803 MICROCOMPUTER PROGRAMMING & INTERFACING**

This book has been written to provide you with a detailed discussion of the 6801 single-chip microcomputer and its various versions, the 68701 and 6803. The first chapter is a general discussion on the 6801, 68701 and 6803 microcomputers. Software is covered in Chapter 2. The various i/o configurations and operating modes of the 6801, 68701, and 6803 are discussed in Chapters 3 and 4. Chapters 5, 6 and 7 detail "how to use" the 6801, 68701, and 6803 internal R/W memory, ROM (EPROM), timer, and serial communications interface. In Chapters 8 and 9, you will learn how to interface the 6801 to switches, keyboards, displays, digital-to-analog converters, and analog-to-digital converters. Specific temperature monitoring, automobile, and data-acquisition applications are provided in Chapter 9.

Review questions and answers are provided at the end of each chapter to test the reader on the material presented in that chapter. Three appendices are featured in the book. They cover the 6801/68701 and 6803 Instruction Set, Specifications Sheets, and the LILbug Monitor.



**Andrew C. Staugaard, Jr.** is an experienced engineering/educator in the field of microelectronics. He is presently Assistant Professor of Electrical Engineering at Jamestown Community College, a State University of New York campus. In 1977 he was the recipient of the Faculty Award for Excellence in Teaching at Jamestown Community College.

Prior to entering the education field, Professor Staugaard was employed as a quality engineer in microelectronics processing by the Bendix Corporation, Kansas City Division. He is the author of another SAMS book, **How to Program and Interface the 6800**. He is coauthor of monthly columns on the Motorola 6800 chip family that appear in several US and foreign magazines.

**Howard W. Sams & Co., Inc.**  
4300 WEST 62ND ST. INDIANAPOLIS, INDIANA 46268 USA

14.95

ISBN: 0-672-21726-0