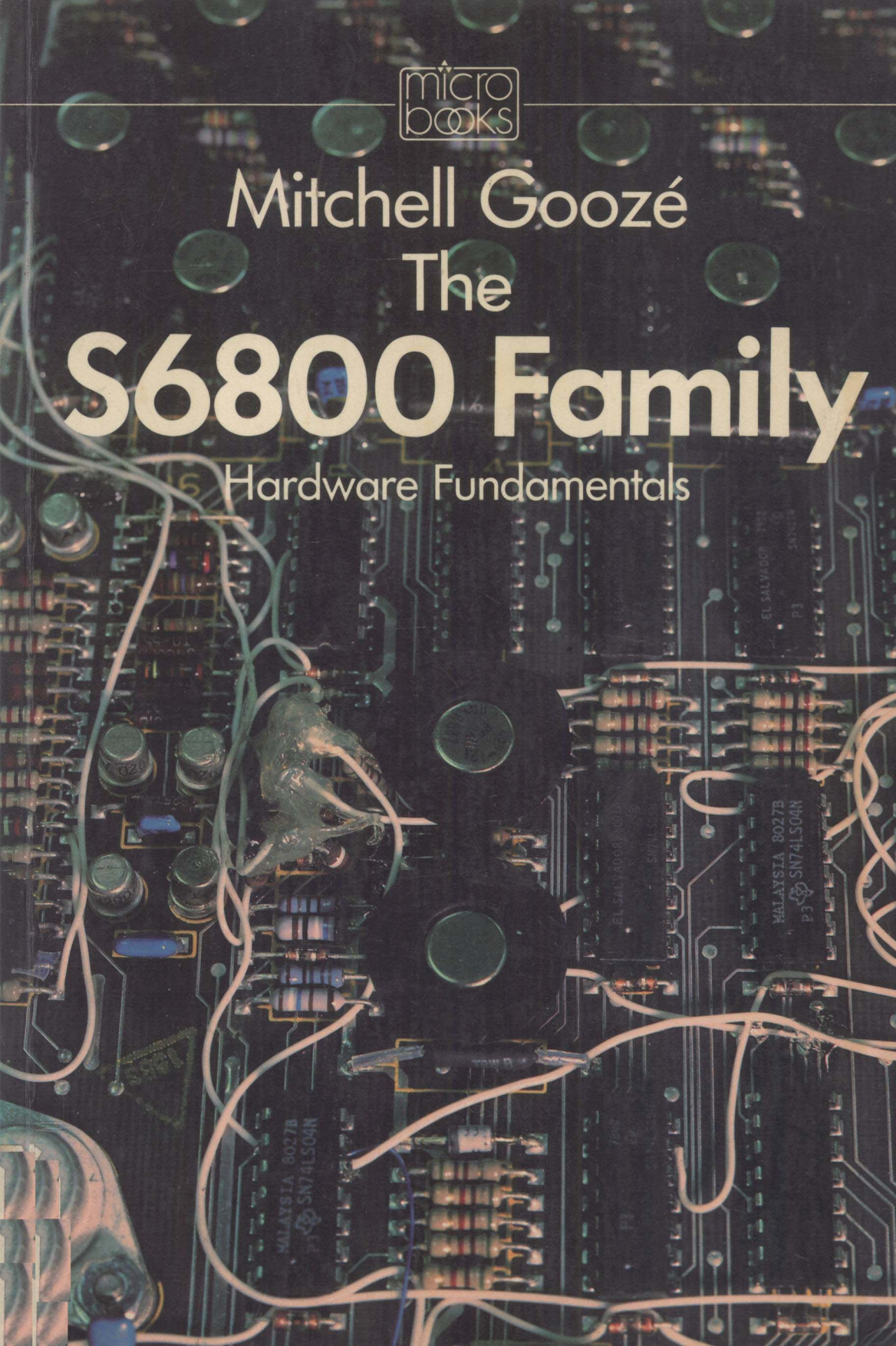

micro books
Mitchell Goozé
The
S6800 Family
Hardware Fundamentals

MITCHELL GOOZÉ
President
SOLOSystems Inc.

The S6800 Family
Hardware Fundamentals

ADDISON-WESLEY
PUBLISHING COMPANY
Reading, Massachusetts
Menlo Park, California • London • Amsterdam • Don Mills, Ontario • Sydney

This book is in the
Addison-Wesley Microbooks Technical Series

Thomas A. Bell, *Sponsoring Editor*
Marshall Henrichs, *Cover Design*

Library of Congress Cataloging in Publication Data

Goozé, Mitchell E.
The S6800 family.

(Addison-Wesley series in joy of computing)
Includes index.
1. S6800 (Computer) I. Title. II. Series.
TK7888.3.G64 001.64'4 81-3673
ISBN 0-201-03399-2 AACR2

ISBN 0-201-03399-2
ABCDEFGHIJ-AL-898765432

Introduction

This book provides in a single volume all of the technical information required for the use of the AMI S6800 microprocessor family of parts in microcomputer systems. Although this book uses AMI part numbers throughout, the S6800 family components are identical to the other 6800 family components manufactured by Motorola, Fairchild, Hitachi, and others. The information contained in this book will be true, in general, for all 6800 family parts manufactured by any licensed alternate source.

Chapter 1 of this book describes the internal organization of the various members of the S6800 family, discussing the nature and use of the various internal registers and detailing the control functions and status conditions for the operation of the components, which are treated individually.

Chapter 2 describes the S6800 bus system and the common control signals (such as clocks, interrupts, and reset) as they affect all of the components in the family.

Chapter 3 describes the memory philosophy in the S6800 microprocessor family, with special emphasis on the Microprocessor Unit (MPU) stack, the relation between memory and Input/Output (I/O), and the instruction sequence.

Chapter 4 describes the MPU instruction set, including the various addressing modes available, and details the operation of each instruction.

Chapter 5 describes some of the ramifications of putting a system together, including details on clock driver design, bus loading, input and output considerations, and Direct Memory Access.

Chapter 6 takes a small system and follows the logic and timing through each step to illustrate the operation of the S6800 family in an actual application. The program is included and annotated so that the reader may associate the hardware with the actual program steps to go with it.

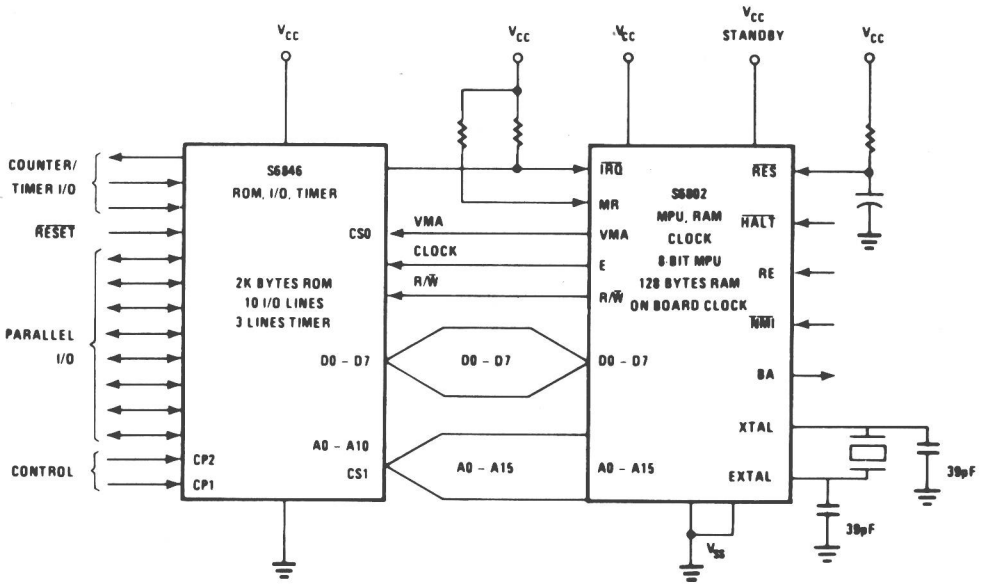
Chapter 7 analyzes a minimum terminal design using some of the newer, more complex peripherals in the S6800 family. Information similar

to that found for the system example in Chapter 6 is presented for the terminal design example in this chapter.

The AMI S6800 microprocessor series is a family of components designed to interconnect directly for system simplicity, though large and complex systems can be easily designed because of the flexible architectural characteristics of the various components. The head of the family is the S6800 MPU, which controls the activity of the other components in the system according to the plan specified in the program, which is stored in memory, as conditioned by the data that may be available on peripheral devices or stored in random-access-memory (RAM).

This book covers the S6800 MPU and other MPU devices which have identical instruction sets (S6802 and S6808), along with all peripherals in the S6800 family. The book does not cover the S6801 and S6803, S6805, or S6809 MPUs. Their instruction sets contain additions to and significant deviations from the base S6800 MPU, and there are certain significant hardware differences.

N-channel, silicon-gate, MOS technology is used in the manufacture of these components. The results are high density, fast execution, and com-



BLOCK DIAGRAM OF A TYPICAL COST EFFECTIVE MICROCOMPUTER. THE MPU IS THE CENTER OF THE MICROCOMPUTER SYSTEM AND IS SHOWN IN A MINIMUM SYSTEM INTERFACING WITH A ROM COMBINATION CHIP. IT IS NOT INTENDED THAT THIS SYSTEM BE LIMITED TO THIS FUNCTION BUT THAT IT BE EXPANDABLE WITH OTHER PARTS IN THE S6800 MICROCOMPUTER FAMILY.

Figure 1

plete TTL compatibility, including a single 5-V power supply. A fully operational microcomputer can be assembled from only two parts that offers eight bidirectional I/O signal lines, two I/O control lines (used for interrupts and/or acknowledges), 128 bytes of read/write storage, 2048 bytes of read-only program, storage, and a 16-bit programmable timer/counter. This system is illustrated in Figure 1.

Larger systems are easily implemented, to a maximum capacity of 65,536 memory locations, which may use any combination for program storage, data storage, and I/O registers. Along with this ease of systems implementation, the S6800 family of MPUs is distinguished by an internal architecture that includes a true Index register for addressing memory locations not directly pointed to by the register, two accumulators, relative branches for self-relocating programs and signed binary arithmetic, with conditional branches that consider overflow in the test for signs and allow testing of all possible results ($=$, $\neq$, $<$, $>$, $\leq$, $\geq$).

Acknowledgments

I would like to thank several people without whom this book would not have been possible: Dr. Harold Stone of the University of Massachusetts for his belief in the value of this book. Ron Denchfield, former Manager of Corporate Communications at American Microsystems, Inc., for his significant effort at getting this project off the ground. Rick Orlando, Microprocessor Product Manager of American Microsystems, Inc., for his work with the S68045, which is the basis for Chapter 7. Kin Seto, former Applications Engineer at American Microsystems, Inc., for his help in working out the finer operating details of some of the peripherals. Last, but certainly not least, I would like to thank the original authors of the AMI S6800 Hardware Manual, which formed the foundation for this book.

Contents

1 S6800 Family Components

This chapter describes the general architecture of each member of the S6800 microprocessor family, giving a functional description of the component parts of each. Where appropriate, the external signal lines present on the device Input/Output (I/O) pins are defined with respect to the operation of the device, though a full description of the bus controls and the signals and timing is reserved until Chapter 2.

THE S6800 MPU

The Microprocessor Unit (MPU) may be considered the head of the S6800 family. Its function in the system is to supervise and control the operation of all the other system components.

The structure of the MPU may be most easily understood in reference to the block diagram in Figure 1-1. The basic elements of its structure include an 8-bit Arithmetic/Logic Unit (ALU), two 8-bit Accumulators, one Condition Code Register (CCR), and three 16-bit address storage registers. All of these registers are available for program use in certain defined ways. In addition, there are three registers internal to the MPU that are not accessible to the program but are used for holding the instruction to be executed, for temporary storage for address data, or for incrementing the Program Counter. There is also an instruction decoder and the logic necessary to control the processor timing and bus during instruction execution, interrupt service, and initialization.

Within the MPU, all data and address transfers between the registers, as well as to and from the ALU, are made across three internal 8-bit busses. The first is a data bus, and the other two carry the upper and lower eight bits of memory addresses, respectively.

The MPU communicates with its external memory and all I/O devices across an 8-bit bidirectional data bus (D0 through D7) and sixteen address lines (A0 through A15). In addition, there are a number of status and control signals related to the operation of the busses and the microprocessor system timing. These are discussed extensively in Chapter 2.

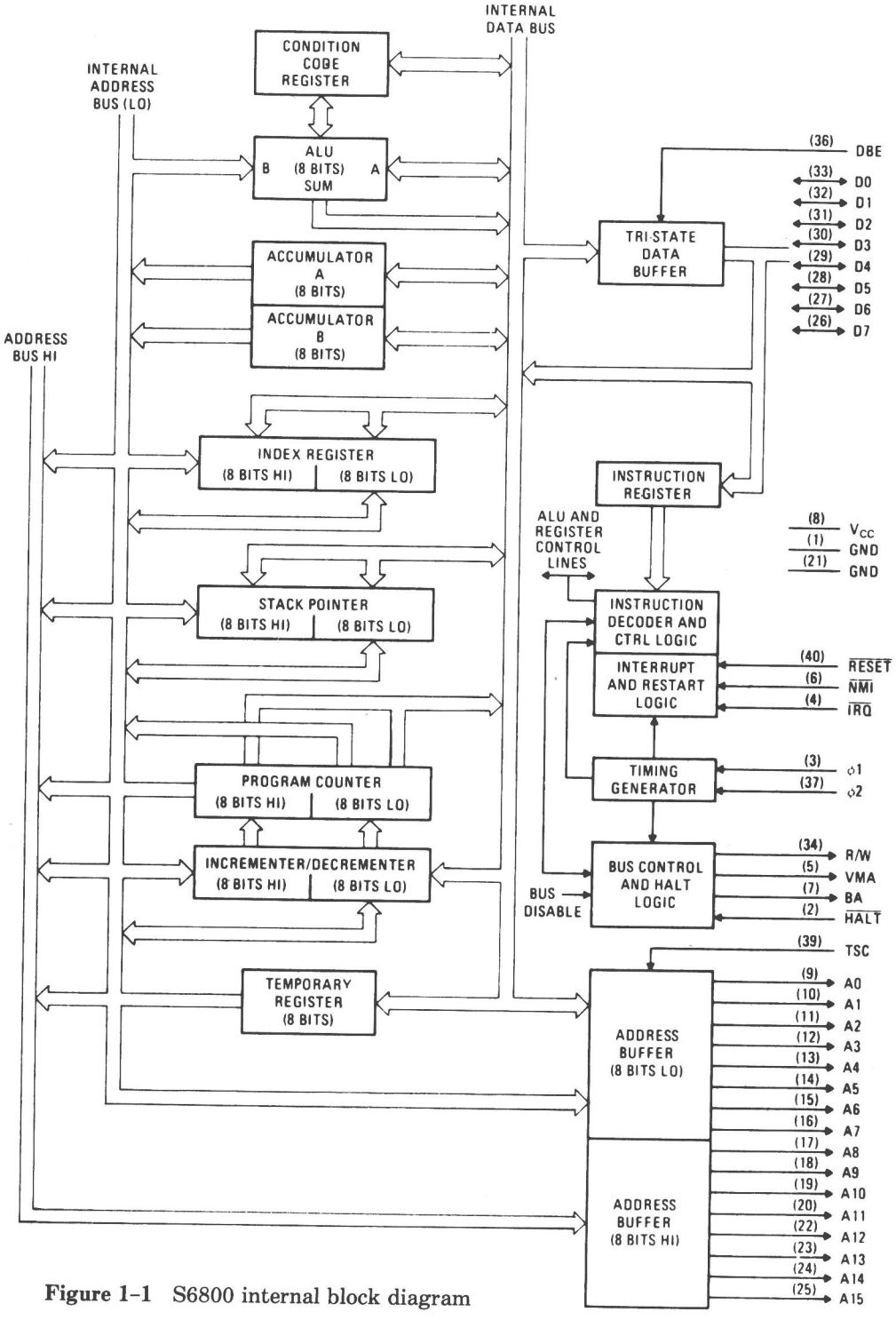


Figure 1–1 S6800 internal block diagram

Three of the program-accessible registers of the MPU are used for addressing memory; they thus are able to hold 16-bit addresses. Each of these three registers has a different function within the operation of the MPU.

The **Program Counter** (PC) register generally points to the next instruction byte in the memory. As each instruction is fetched from memory, the PC is incremented to point to the next instruction byte. Multiple-byte instructions are fetched from memory in successive memory cycles, and the PC is incremented for each byte fetched. Certain of the instructions cause the PC to be loaded with an arbitrary value fetched from memory or derived from the previous contents of the PC or Index register and a displacement value fetched from memory; these are Jump, Branch, and Return instructions. Some instructions cause the PC to be stored in memory; these are the subroutine calling instructions and the Wait for Interrupt and Software Interrupt instructions. The service of a processor interrupt also stores the PC, as well as some of the other registers, into memory.

The **Stack Pointer** (SP) register is used to address memory sequentially for the temporary storage of data and processor registers. When the stack pointer is used to address memory for write operations, the stack is decremented after each write operation; these are called 'Push' operations, and the data written into memory is said to be 'pushed onto the stack.' When the stack pointer is used to address memory for read operations, it is incremented first before addressing the memory; these are called 'Pull' operations, and the data read from memory is said to have been 'pulled from the stack.' The pre-increment, post-decrement feature of the stack pointer permits its use for temporary storage of data and program parameters as follows: A Push operation followed by a Pull operation returns the original data; several consecutive Push operations store the successive data bytes in consecutive memory locations, with decreasing addresses; an equal number of consecutive Pull operations retrieves the same data in the reverse order; Push and Pull operations may be mixed in any sequence, provided that each Pull is matched by a previously occurring Push.

The Stack Pointer holds a full 16-bit address, so the amount of data which may be pushed onto the stack is limited only by the size of the system memory. In addition to instructions which push and pull the contents of processor registers onto and off of the stack, the service of a processor interrupt pushes the entire processor state (that is, all of the registers accessible to the program except the Stack Pointer itself) onto the stack. Other instructions are provided for the direct manipulation of the Stack Pointer; it may be incremented or decremented without accessing memory, or it may be loaded from or stored into memory or the Index register.

The **Index** register (X) is used as a base address in the indexed addressing mode of certain of the MPU instructions. In this mode, the second byte

of the instruction provides a displacement, which is added to the contents of the Index register to form the effective memory address of the instruction operand; the index register contents are not altered by this operation. Like the Stack Pointer, the Index register may be incremented, decremented, or loaded from or stored into memory or the Stack Pointer. The Indexed Addressing Mode is discussed more fully in Chapter 3.

The two **Accumulators** (A and B) receive the results of most of the ALU operations and also provide one of the source operands of the operations. Most of the instructions that operate on an accumulator may operate on either accumulator, as specified in the instruction operation code, although there are three instructions (TAP, TPA, and DAA) that apply to the A accumulator only. In addition to the ALU operations, the accumulators may be shifted, incremented, or loaded from or stored into memory or the other accumulator.

Most of the MPU instructions pass the data through the ALU, and accordingly, many of them set the condition code register flip-flops according to the results. Besides addition and subtraction, the ALU is able to perform the logical functions And, Or, and Exclusive Or, and Left and Right Shifts. The status of the data coming out of the ALU is normally stored into the CCR flags, N, V, Z, and, except for transfers and logical operations, the C flag. The addition instructions also affect the setting of the H flag, which may then be used for decimal correction.

There are six bits in the CCR. When this register is transferred into the A accumulator or pushed onto the stack, the remaining two bits of the generated byte of data are set to ones (1's). When the CCR is loaded from memory or from the A accumulator, the two excess bits are ignored. There are also special instructions for setting and clearing the V, C, and I bits of the CCR. The six flags of the CCR are as follows:

H. The Half Carry is used only in connection with BCD (packed decimal) addition; it represents the carry out of bit 3 and into bit 4 of the sum in the three ALU Add instructions (ABA, AAD, and ADC). Together with the C bit and the current state of the A accumulator, this flag is used to adjust the result of a binary addition of decimal operands to correspond to the correct decimal representation of the sum. This operation is discussed more completely in Chapter 4 in connection with the DAA instruction.

I. The Interrupt Mask bit of the CCR controls the MPU recognition of the maskable Interrupt Request ($\overline{IRQ}$). When the I flag is 0 (clear), interrupts are enabled; when it is 1 (set), interrupts are masked (disabled), and the processor may only be interrupted by the Non-Maskable Interrupt ($\overline{NMI}$). This bit may be directly set or cleared, or it may be altered when the CCR is loaded. It is also set to 1 by the service of a reset or processor interrupt or by the SWI (Software Interrupt) instruction. After the Interrupt mask bit is cleared to zero from a one state, at least one machine cycle will

pass before the MPU can recognize an interrupt. Thus except after the RTI instruction, whenever the I bit is newly cleared to zero, at least one more instruction will be executed before the next maskable interrupt will be serviced. This does not apply when the I bit was already zero. Instructions which set the Interrupt mask bit to one take effect immediately, and no subsequent maskable interrupts are possible until after the I bit has been cleared again to zero.

N. The Negative result bit of the CCR generally designates the state of the most significant (or sign) bit of the result of an operation. In Two's Complement notation, a negative number is indicated by the state of the most significant bit being 1.

Z. The Zero result bit of the CCR generally is a record of the fact that the previous operation had a zero result.

V. The Two's Complement Overflow bit in the CCR normally indicates that the result of an ALU operation fell outside the defined range for signed two's complement results (seven bits + sign for one-byte operands). If the sum of two positive 7-bit numbers produces a carry into the eighth bit, it is defined to be an overflow condition; similarly, if the sum of two negative 7-bit numbers (that is, the sign in the eighth bit is set to 1) incurs a borrow (which is the complement of the carry), it is defined to be overflow. Thus the general form for overflow is given by the Boolean equation

$$V = C_6 \cdot \overline{A_7} \cdot \overline{B_7} \cdot + \overline{C_6} \cdot A_7 \cdot B_7$$

where C_6 represents the carry into the most significant bit of the adder, and A_7 and B_7 represent the most significant bits of the two operands going into the adder. The overflow condition may also be the carry going into bit 7 of the result and the carry coming out of bit 7, as in the following formula:

$$V = C_6 \oplus C_7$$

This formula is logically equivalent to the previous equation. Logical operations generate no carries in the ALU, so according to this formula the overflow bit is always cleared. In the case of shift instructions, the overflow bit V is arbitrarily set by the formula

$$V = C \oplus N$$

which is equivalent to the previous formula for a left shift, if the shift is understood as the sum of the operand and itself. The V bit in the CCR may also be directly set or cleared by instructions defined for that purpose. Some of the conditional branch instructions permit the overflow bit to be tested in conjunction with the negative bit to determine whether the result of operation on two's complement operands is positive or negative, regardless of whether overflow actually occurred or not.

C. The Carry bit of the CCR is used to hold the carry out of sums in the ALU and to hold the borrow incurred by subtraction. The borrow in this case represents the complement of the carry for the equivalent complementary sum, that is, the borrow incurred by the subtraction $a + (-b)$. In the case of the shift instructions, the carry flag is used to hold the bit shifted off the end of the datum. In order to permit multiple-precision (that is, more than byte-wide) addition, subtraction, and shifts, instructions are available that accept the previous contents of the carry into the operation. There are also instructions for setting and clearing the carry flag directly. When subtracting unsigned binary numbers, the carry bit (actually representing the Borrow condition) may be tested to determine which of the two is greater.

There are 72 instructions that the MPU recognizes in the execution of the program. Most of these instructions operate on the data in one or more of the MPU registers or on data stored in memory, altering the data in some defined way or transferring the data to some other register or to memory or to an I/O register. Some of the instructions test conditions left by previous instructions to alter the flow of the program execution. The complete definition of the S6800 instruction set is given in Chapter 4.

S6802 MPU, CLOCK, AND RAM

The S6802 is a monolithic 8-bit microprocessor that contains all of the registers and accumulators of the S6800 plus an internal clock oscillator and driver and 128 bytes of Random-Access-Memory (RAM) on the same chip. Figure 1-2 shows a block diagram of the S6802. The MPU on the S6802 is functionally identical to the S6800 MPU, and the functions previously described pertaining to the internal workings of the S6800 are identical on the S6802. The S6802 is therefore also software-compatible at the machine-code level with the S6800. All of the information found in subsequent chapters regarding the internal and external functioning of the S6800 MPU is equally true for the S6802 unless specifically stated. All bus timing, system interface, and instruction sequences are identical.

The on-chip clock available for use with the S6802 consists of an internal oscillator that is controlled by an external crystal. The crystal is connected between pin 38 (xtal) and pin 39 (extal), forming a parallel resonant circuit as shown in Figure 1-3 along with other required crystal characteristics. Pin 39 can be driven from an external TTL-compatible clock source if desired. In either case, crystal or external clock, the internal circuitry on the S6802 divides the fundamental frequency by 4 to obtain the actual operating clock of the MPU. Thus a 4-MHz crystal is used to obtain a 1-MHz system clock.

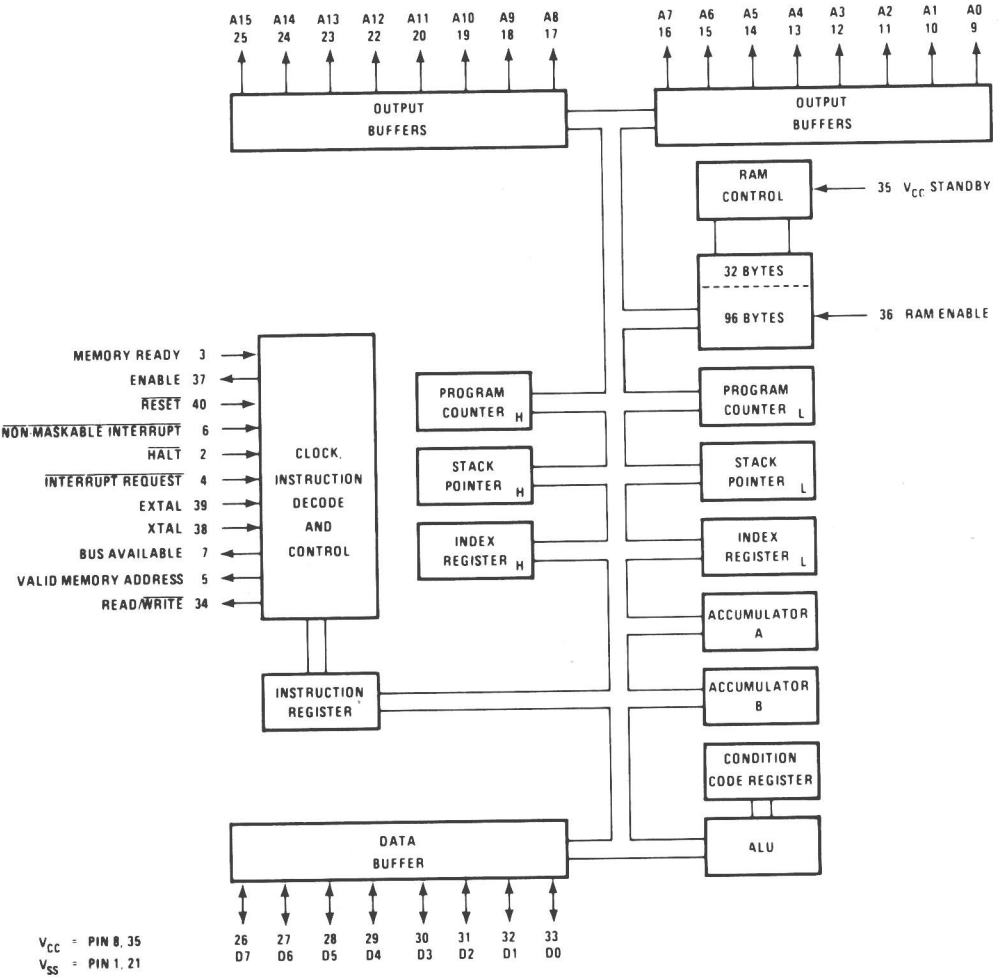
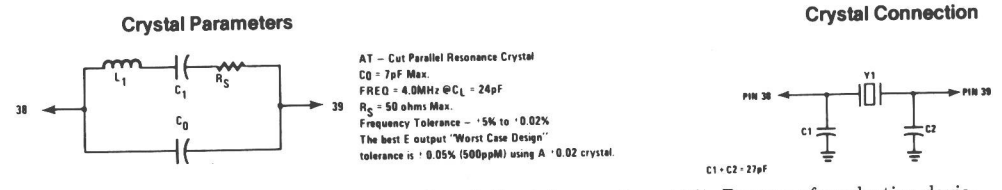


Figure 1–2 S6802 internal block diagram



Tolerance Note: Critical timing loops may require a better tolerance than ±5%. Because of production deviations and the Temperature Coefficient of the S6802, the best "worst case design" tolerance is ±0.05%. (500 ppm) using a ±0.02% crystal. If the S6802 is not going to be used over its entire temperature range of 0°C to 70°C, a much tighter overall tolerance can be achieved.

Figure 1–3 S6802/S6808 crystal information

The on-chip RAM in the S6802 is fully static and organized as a 128×8 bit array addressable in 8-bit (byte-wide) increments. This is identical to the S6810 (see below). The on-chip RAM is located in memory locations 0000_{16} to $007F_{16}$ and is fully decoded. Pin 36 (RE), the RAM enable pin, enables and disables the on-chip RAM. The RAM is enabled when this pin is high and disabled when the pin is taken low.

In addition, the first 32 bytes (memory locations 0000_{16} to $001F_{16}$) have a separate DC voltage supply pin: pin 35, V_{cc} standby. This optional power pin supplies power to this portion of memory and the RAM enable logic in power-up, power-down, and standby conditions if desired. Maximum current drain at 5.25 V is 8 mA. If this power save feature is not desired, pin 35 is connected to pin 8, V_{cc}.

S6808 MPU AND CLOCK

The S6808 is absolutely identical to the S6802, with the exception that the on-chip RAM has been removed. The MPU and the on-chip clock function identically to those on the S6802. The RE pin on the S6802 has been changed to V_{ss} on the S6808, and the V_{cc} standby pin on the S6802 is simply V_{cc} on the S6808.

S6810 RAM

The S6810 is a static read/write Random-Access-Memory (RAM) designed to be compatible with the S6800 microprocessor family. It is organized as 128 words of eight bits each, so that in a minimum system one RAM may serve as a scratch pad memory for the stack and other data storage, with no other interfacing parts required. It is totally static and requires no clocks or cell refresh.

The internal structure of the S6810 RAM is most easily seen in the block diagram in Figure 1-4. Six chip select or enable lines are provided with this RAM to facilitate its use in microprocessor systems. These may be attached to the outputs of address decoders in large memory arrays or to individual address lines in small memory systems, eliminating the need for extra decoders. The techniques for memory address mapping are discussed more fully in Chapter 5.

The RAM may be thought of as 128 addressable data registers for the MPU operation using the direct addressing mode, or one or more RAMs may be defined in the system for processor stack storage. If the PC in the MPU contains the address of data in a RAM, that data will be interpreted as instructions by the MPU; thus a program might be set up to alter parts of itself dynamically during the course of its execution, by putting those parts into RAM.

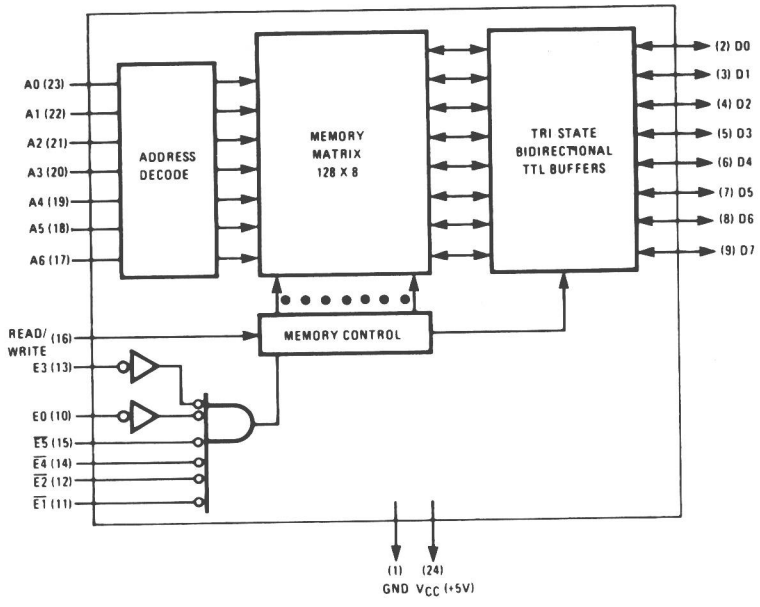


Figure 1–4 S6810 internal block diagram

S6820/S6821 PIA

The S6821 replaced the S6820 in 1978 as the preferred PIA device in the S6800 family. (This is true of all 6800 family manufacturers.) The S6821 and S6820 function identically with only minor changes in the I/O input and output drive specifications. Since the S6820 has been out of production for over four years, it is unlikely the reader will encounter the part.

The S6821 Peripheral Interface Adapter (PIA) encompasses in a single LSI circuit most of the data and control requirements for 8-bit parallel input and output. The PIA is compatible with the S6800 microprocessor family bus, and it appears to the MPU as a four-location RAM (4×8), although there are actually six addressable registers in it.

The structure of the PIA may be most easily understood by referring to the block diagram in Figure 1–5. It is divided into two sections, A and B, which are similar in function and operation and which are programmed in the same way. Each section is equipped with eight parallel bidirectional peripheral data lines, two programmable interrupt status and control lines, an interrupt request line, and three 8-bit registers for data and control. Common circuitry allows the selection of the two sections, the various registers within each section, and the interface to the microprocessor busses.