# SIGNETICS
# 8080
# EMULATOR
# MANUAL

# TABLE OF CONTENTS

# FOREWORD

Despite the numerous advances in microprocessor technology, there still remain many technological niches that have not yet been filled. One example is a fast microprocessor with both the powerful instruction set and extensive software support that is provided with an MOS machine like the 8080.

In recognition of this need, Signetics has developed an 8080 Emulator using its high speed Series 3000 microprocessor chip set. The Emulator significantly increases the speed of the 8080A system without costly redesign of system software. The Emulator is capable of achieving system speeds from two to nine times faster than the 8080A.

The Signetics 8080 Emulator bridges the gap between high-performance "custom built" bipolar CPUs and the slower, "off the shelf" MOS microprocessors. While the "custom built" bipolar machines are fast, their unique nature requires in-house development of an assembler and other support software. The MOS machines, on the other hand, offer assemblers, high-level languages, development systems and many other support features. They are, however, inherently slow. The Signetics 8080 Emulator offers the best of both worlds. It is an 8080 that runs at bipolar speeds.

To your system software, the Signetics 8080 Emulator looks just like the microcomputer it emulates. Except for timing loops, software that has been developed and debugged for an 8080 system will run directly on the 8080 Emulator.

In addition to providing industry with a Schottky-bipolar 8080 CPU system, the 8080 Emulator also gives Signetics an opportunity to present its Series 3000 microprocessor chip set in the light of a practical, accomplished design. Toward this end, this manual is more than just a reference guide for 8080 Emulator users. It also contains a wealth of information on the application, structure, and operation of bit-slice processors. Basic microprogramming concepts are first introduced and then applied to the Emulator's design. This approach is designed to give the reader a basic understanding of bit-slice CPU design and microprogramming techniques.

This manual assumes that the reader has some experience with microprocessors, particularly the 8080 system. For a detailed description of the 8080, refer to the 8080 Microcomputer Systems User's Manual published by Intel Corporation.

## FEATURES OF THE SIGNETICS 8080 EMULATOR

- Complete emulation of 7-chip 8080A CPU system
- Built with Signetics series 3000 Schottky microprocessor chip set
- Processor cycle times from 150ns to static
- Microprogrammed architecture
- Implementation of entire 8080 instruction set
- Available microprogram space for user defined macro instructions
- Multiply and Divide macro instructions
- Automatic trap for undefined or illegal op-codes (machine enters wait state)
- Instruction execution 2 to 9 times faster than 8080A
- Power on reset provided
- Single phase clock provided
- On board clock provided
- Single 5-volt supply operation
- Board dimensions and edge connector compatible with Intel's SBC 80 series
- Complete 8080 software compatibility (except for timing loops)

# CHAPTER I
# INTRODUCTION

## GENERAL

The Signetics 8080 Emulator is a bipolar Schottky microcomputer. Using the Signetics Series 3000 microprocessor chip set, a complete 8080A CPU system has been implemented. The Emulator replaces a system consisting of the 8080A microprocessor, the 8224 clock generator, the 8228 system controller, two 8226 bidirectional ports, and two 8212 bus drivers. A block diagram of the emulated system is shown in Figure 1.

The Signetics 8080 Emulator is a kit. A PC board, all of the necessary ICs and discrete components, and support documentation are provided. The Emulator kit can be assembled by a skilled technician in four to six hours using the assembly instructions presented in Chapter 4. All PROMs are preprogrammed.

## BIT SLICE ARCHITECTURE

### Advantages

The primary advantages in using bit slice architecture are:

1. The availability of microprogrammable LSI components.
2. The inherent speed advantage of these LSI components.
3. Design flexibility.

With microprogrammable LSIs, the system component count can be reduced to a manageable level, resulting in lower manufacturing costs and design simplification.

With LSI components capable of running at Schottky-bipolar speeds, it is possible to develop machines that will run with micro cycle times of less than 200 nanoseconds. Many applications require this high speed performance which cannot be achieved by MOS microprocessors.

An example of design flexibility is the ability to expand or contract the size of the microprogram both vertically and horizontally to fit the requirements of the particular application. Horizontal expansion is achieved by adding more control bits to the microinstruction word. Vertical expansion is achieved by adding microinstructions to the microprogram. Additional flexibility is provided by the cascadability of bit slice microprocessors such as the N3002. This feature makes expansion of the CPU word size possible.

### Microprogrammed CPU

Construction of a bit-slice CPU requires the development of a microprogram. A microprogram is a series of microinstructions stored in PROM (control store). For a bit-slice, microprocessor-based design (such as the 8080 Emulator), all major building blocks are controlled directly or indirectly by a microinstruction. A series of predefined microinstructions is usually required to perform a useful function, such as the
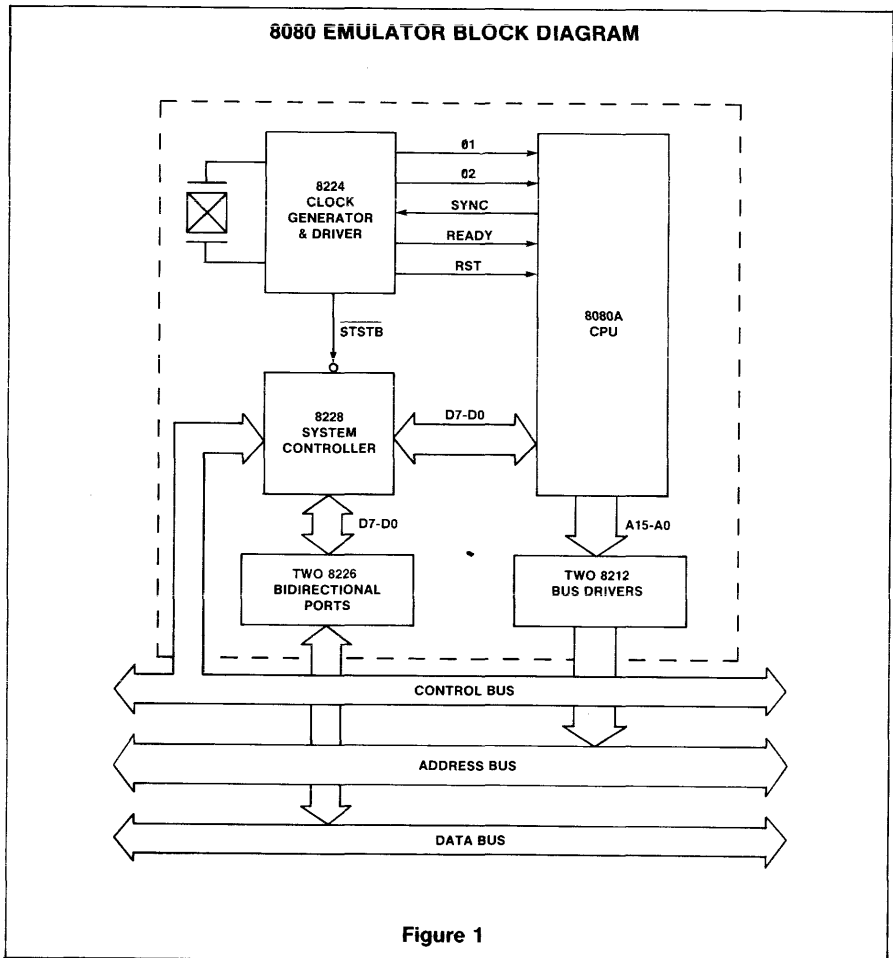


**8080 EMULATOR BLOCK DIAGRAM**

**Figure 1**

adding of two operands and depositing the sum in a specific register. This kind of useful function is called a "macro instruction." In the case of the 8080 Emulator, all of the original 8080 instructions are macro instructions. The execution of each macro instruction is accomplished by one or more microinstructions, depending on the macro instruction's complexity.

A simplified structure of a microprogrammed CPU is illustrated in Figure 2. There are five major building blocks, namely:

1. *The Central Processing Section*—This is the section where the actual logic and arithmetic operations are performed. Localized registers are available for temporary storage. This section can be implemented with the use of bit slice microprocessors such as the N3002. Cascading N3002s will yield the desired word length.

2. *Control Store*—The Control Store consists of a group of storage devices, such as ROMs or PROMs (RAMs are used in the case of writable control store). It is here where the microprogram is stored.

   The Control Store size can be varied in two ways:

   *Horizontally:* By increasing or decreasing the number of bits in each microword, the number of

parallel hardware control operations will increase or decrease, respectively.

*Vertically:* By increasing or decreasing the number of microinstructions, the capability of the CPU will increase or decrease, accordingly.

It is possible to optimize the CPU design by optimizing the size (horizontally and vertically) of the Control Store. Total flexibility is achieved through the process of optimization in the Control Store.

3. *Microprogram Sequence Control*—This function is implemented efficiently with the N3001. The N3001 controls and provides the address sequencing function for the Control Store. The address thus formed is called the "microprogram address."

4. *Macro Instruction Decode Logic*—This area performs the function of decoding the macro instruction that was fetched from main memory. As a result of the decode, an address is formed to access the Control Store (via the microprogram sequence control) at those locations which are required steps in the execution (macro decode is used throughout the instruction cycle).

5. *Microinstruction Register*—This register is commonly called a "Pipeline Register." Its key function is to hold a microinstruction so that concurrent execution of the present microinstruction and fetching of the next is possible. This architectural arrangement enhances the performance of a given technology.

## A MICROPROGRAMMED CPU
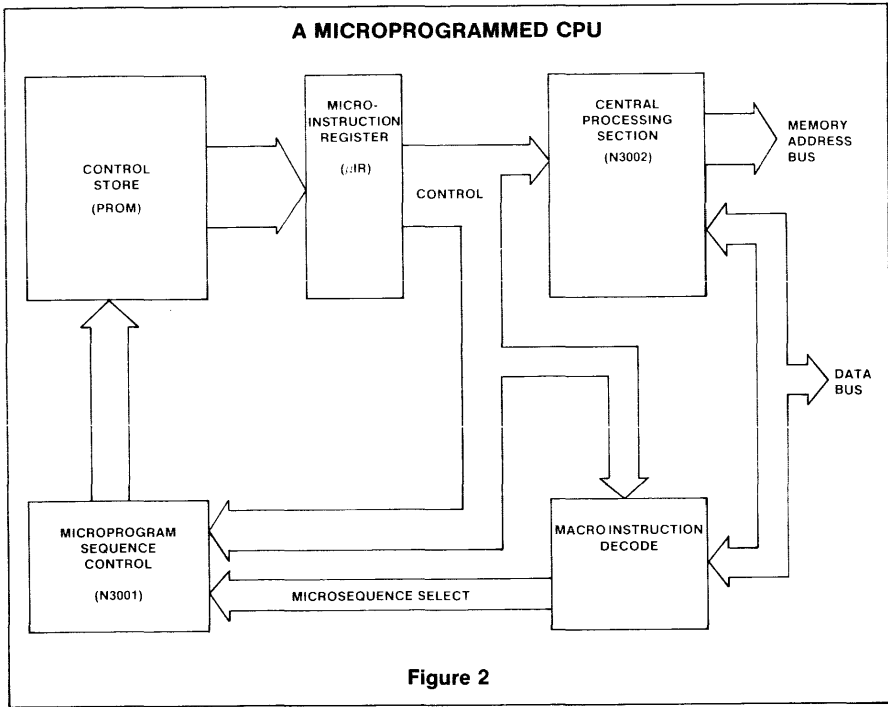


**Figure 2**

As indicated in Figure 2, the microinstruction held in the Microinstruction Register provides control fields to control all of the major building blocks in this generalized CPU. All appropriate operations in the hardware are performed with the execution of a microinstruction.

A typical microinstruction is illustrated in Figure 3. In this example, a 40-bit-wide microword is assumed. If there is more logic to be controlled, more bits can be added. Each control field is defined in relation to a specific hardware element which is controlled by the bits emerging from the specific PROM location.

## TYPICAL MICROINSTRUCTION FORMAT



**Figure 3**

**signetics**
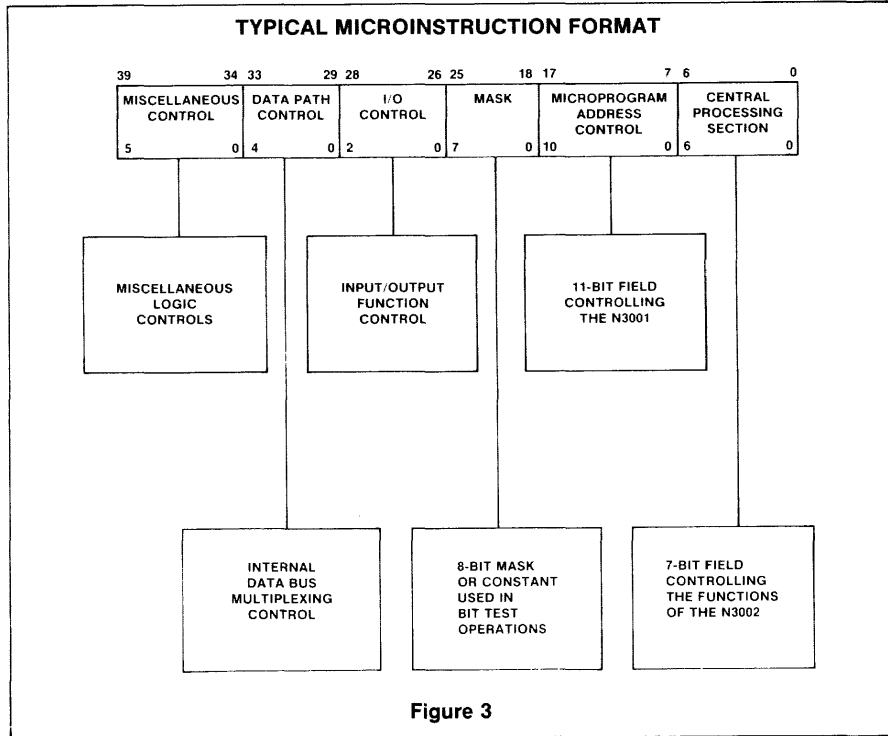
# CHAPTER 2
# THEORY OF OPERATION

## EMULATOR ARCHITECTURE

The basic architectural organization of the 8080 Emulator is shown in Figure 4. It consists of the following three major sections:

1. The Micro-Control Section
2. The ALU and Register Section
3. The I/O and Memory Interface

A detailed description of each of the above sections is presented in this chapter.

## Micro-Control Section

The overall control of the machine is provided by the micro-control section shown in Figure 4. This section is subdivided into the following functional blocks:

### Micro-Control Memory
The microprogram is stored in six 512X8 Schottky PROMs. Thus, the microinstruction is 48 bits wide, and the microprogram can include 512 microinstructions. (Only 345 microinstructions are used to implement the 8080 instruction set.)

### Micro Control Unit (N3001)
Micro control memory is addressed by the N3001 MCU. The MCU generates microprogram addresses based on the control information provided by the microprogram and the Instruction Decode PROM.

### Instruction Decode PROM
The first microaddress of every microroutine is decoded from the macroinstruction's op code.

The conversion from op-code to N3001 address data is made by the Instruction Decode PROM.

### Jump Control Logic
Conditional microprogram jumps based on the external control lines, Program Status Word (PSW) status, or microprogram status are implemented by the Jump Control Logic.

### Pipeline Register
The Pipeline Register is a latch that allows one microinstruction to be executed while the next one is being fetched from the Micro Control Memory.

### Register Control PROMs
The registers involved in any given microinstruction may be chosen by the microprogram or the macro instruction's op code. The Register Control PROMs convert a microinstruction control field and the op code into a Register Group control field for each CPE array.

## The ALU and Register Section

The ALU and Register Section, shown in Figure 4, is the operational heart of the microcomputer. All of the computational work and data manipulation are performed in this section.

The ALU and Register Section is functionally divided into the following blocks:

### CPE 3002 Array 1
Array 1 is an 8-bit ALU/Register file fabricated with four 3002 CPE 2-bit slices. The control inputs of each CPE slice are tied together so that the array behaves like a single 8-bit data processor. All arithmetic and logical functions are performed by the CPEs. They also contain the CPE's working registers.
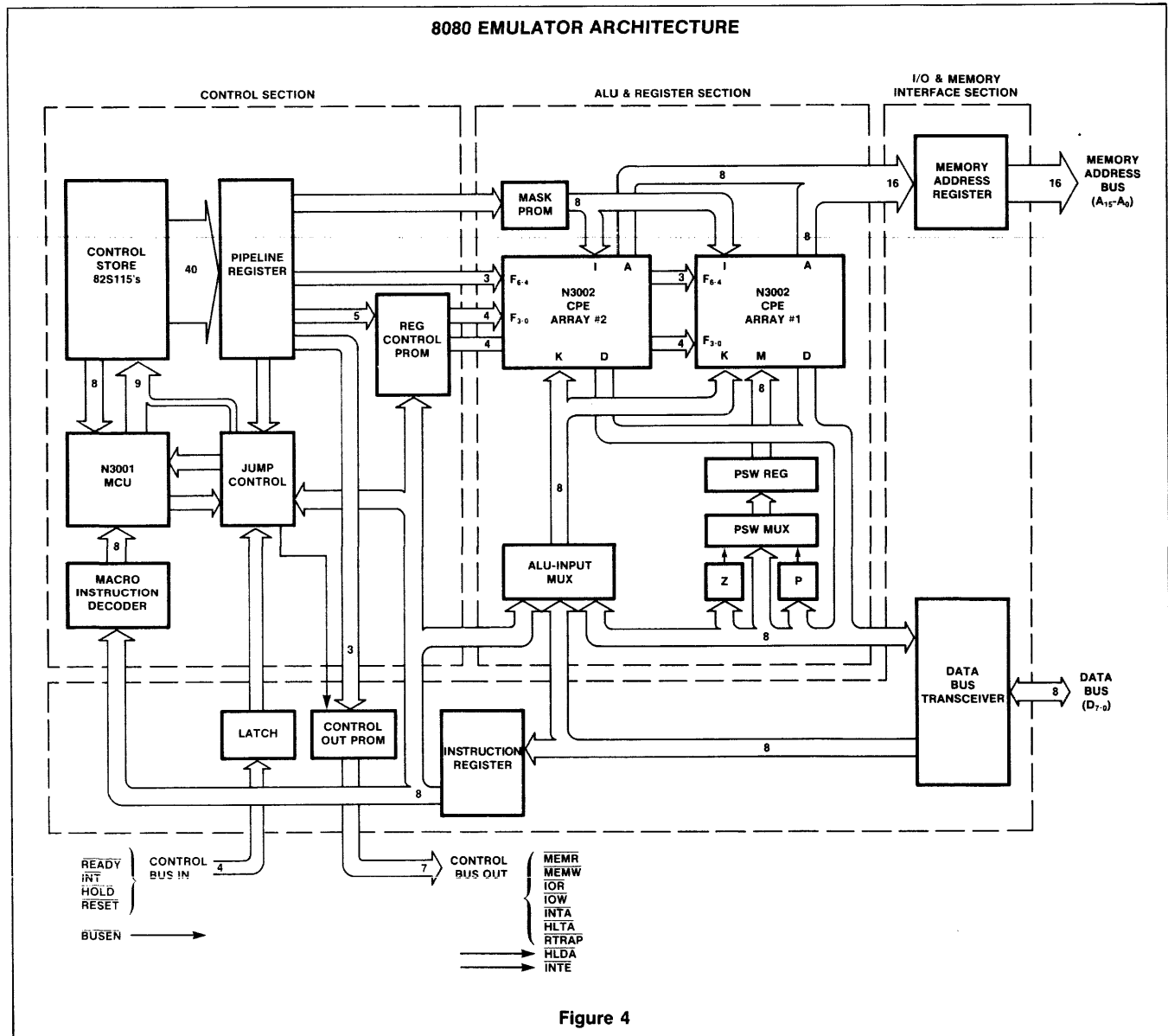


**8080 EMULATOR ARCHITECTURE**

**Figure 4**

### CPE 3002 Array 2
Functionally equivalent to Array 1, Array 2 may operate independently, or be combined with Array 1 for 16-bit operations (e.g., macro address and stack pointer calculations).

### ALU Input Multiplexer
Input data to both arrays is channeled through the ALU Input Multiplexer. Under microprogram control, the multiplexer selects from among data returned from external memory, data output from the arrays, and output from the Instruction Register. The data complement input controls the conditional complement circuit at the multiplexer output to effect either inverting or non-inverting data flow. Furthermore, the multiplexer allows the user to force the outputs to all ones (1) or all zeros (0), as desired.

### Program Status Word Latch
The Program Status Word (PSW) for the 8080 Emulator is stored in the PSW Latch. The PSW Latch is connected to the M BUS input to Array 1. This provides the microprogram with access to the PSW for pushing onto the external stack.

### PSW Multiplexer
The PSW Latch is loaded with data input from the PSW multiplexer. The PSW multiplexer selects between either current machine status or the Data Output Bus from the two arrays. The DO input facilitates a "pop" of the PSW off the external stack.

### Mask PROM
The Mask PROM, addressed by the microprogram, contains 32 masks that are used for bit masking during execution of various microinstructions. The mask is presented to the CPE arrays on the I Bus.

## Input/Output and Memory Interface
The last section of the 8080 Emulator to be considered is the I/O and Memory Interface shown in Figure 4. This section consists of the following blocks:

### External Memory Address Register
The External Memory Address Register is a 16-bit latch. It latches the A Bus outputs of the combined CPE arrays, and presents this address to the high-speed bus drivers which drive the External Address Bus.

### Tri-State Bus Transceiver
The Tri-State Bus Transceiver is an 8-bit bidirectional I/O device. This high-speed, high-current device drives the External Data Bus.

### Instruction Register
The Instruction Register is an 8-bit latch which stores op codes that have entered the 8080 Emulator via the external Data Bus. The output of the Instruction Register addresses the Instruction Decode PROM, the Jump Control PROM, the op code Register Control PROM, and is available as an input to the ALU Input Multiplexer.

### Control Signal Latch
The Control Signal Latch synchronizes external control signals input to the 8080 Emulator with the microprogram.

### Control Signal PROM
By addressing the Control Signal PROM, the microprogram provides 8080 system interface signals to the outside world.

## DESIGN CONSIDERATIONS

### ALU Structure
The schematic for the 8080 Emulator is provided in Appendix A. The Arithmetic Logic Unit (ALU) is shown in Figure A-1. The ALU consists of two 8-bit arrays. Each array is an independent 8-bit ALU with an on board register file implemented with four N3002 CPEs. For operations requiring 16-bit processing, the two arrays are combined to form a single 16-bit ALU.

Each of the 8-bit ALUs has a carry look-ahead generator which can generate a carry out of that ALU. For 16-bit operation, the outputs of the separate carry look-ahead generators are fed into a third carry look-ahead generator to produce a carry out of the 16-bit ALU. To facilitate 16-bit operation, the Carry Out of the low-order array must become the Carry In bit to the high-order array. This is accomplished by the same multiplexing scheme that controls the carry look-ahead generators. The entire operation is controlled by the microinstruction with signals CS2, CS1, CS0, and DBY.

## N3002 Bus Assignment
### Data Out (DO) Bus
Each N3002 array has an 8-bit Data Out (DO) bus that is driven by the CPE's accumulator. The two tri-state buses are wire-OR'ed together and are controlled directly by the microcode with the ED1 signal. The selected Data Out Bus drives the ALU input multiplexer and the 8-bit output transceiver. All data presented to the External Data Bus is routed via the DO Bus.

### A-BUS (Address Bus)
The Memory Address Bus Outputs from both CPE arrays are combined to form a 16-bit address for an external main memory ($A_{15}$-$A_0$). This combined 16-bit bus is latched into the external Memory Address Register. The external Memory Address Register drives three 8T97 Bus Drivers which in turn drive the Edge Connector and hence external memory.

### K-BUS (Data Input Bus)
The K-Bus is the main data input path to the N3002 CPE arrays. This is an unconventional but effective way to use the K-Bus. Normally the K-Bus is used to mask values input to the ALU.

The 8080 Emulator's use of the K-Bus as a data input path allows data to be moved into an internal register without passing through the AC or T register. This feature is essential when the contents of both registers must be saved, as is often required by certain operations.

The K-Bus is driven by a complementing 3-to-1 multiplexer. Controlled by the microinstruction, the K-Bus multiplexer can select

any of the following:

- The Instruction Register contents (or complement).
- The external memory driven Data Bus (or complement).
- The ALU accumulator driven Data Out (DO) Bus (or complement).
- A field of all ones (or zeros).

### I-BUS (Mask Bus)
Normally, the I-Bus is used as the major data input path to the CPE array. For the 8080 Emulator design, however, the flexible nature of the I-Bus makes it suitable for inputting a mask to the CPE array. The masking (ANDing) operation occurs between the K-Bus and the I-Bus in the B multiplexer of the N3002 ALU.

Masking operations are required by four macro instructions: RST (Restart), DAA (Decimal Adjust Accumulator), MUL (Multiply) and DIV (Divide). The mask patterns for these operations are provided by a 32X8 PROM. The PROM is addressed by the microinstruction word. When one of the above instructions is not being executed, the mask PROM forces the I-Bus to all ones. Thus, when the I-Bus and the K-Bus are ANDed, the value on the K-Bus remains unchanged.

### M-BUS (PSW Bus)
Normally, the M-Bus brings in data from an external main memory. Recall that for the 8080 Emulator, data from external main memory has been multiplexed onto the K-Bus. Thus relieved of its intended function, the M-Bus has been used to bring in the Program Status Word (PSW) to the low-order CPE array. The M-Bus inputs to the high-order array are not used and have been tied to the I-Bus inputs.

The PSW bus is made available to the CPE array for the PUSH PSW operation. Via the M-Bus, the CPE array accesses the PSW and pushes its current value onto the external stack, pre-allocated in main memory.

The Program Status Word bits are defined in Table 1.

### FUNCTION BUS*
The N3002 CPE is controlled by a 7-bit field called the Function Bus. Each of the 8080 Emulator's 8-bit arrays is provided with a Function Bus.

The Function Bus is divided into two groups.

1. The F-Group: Determines the ALU function to be performed.
2. The R-Group: Determines the registers involved.

*NOTE

For a detailed description of the Function Bus, refer to the N3002 description in Appendix D.

| BIT | MNEMONIC | NAME |
|---|---|---|
| $D_0$ | CY | Carry |
| $D_1$ | 1 | Logical one |
| $D_2$ | PRTY | Parity (Even) |
| $D_3$ | 0 | Logical zero |
| $D_4$ | HC | Half carry (for BCD operations) |
| $D_5$ | 0 | Logical zero |
| $D_6$ | ZERO | Result equals zero |
| $D_7$ | SIGN | MSB |

**Table 1    PROGRAM STATUS WORD BIT DEFINITIONS**

| N3002 REGISTER | ARRAY 2 | ARRAY 1 |
|---|---|---|
| R0 | B | C |
| R1 | D | E |
| R2 | H | L |
| R3 | SPh | SPl |
| R4 | PCh | PCl |
| R5 | Not Used | Not Used |
| R6 | Not Used | Not Used |
| R7 | Not Used | Not Used |
| R8 | Not Used | Not Used |
| R9 | Working Storage | Working Storage |
| T | A | A |
| AC | Working Accumulator | Working Accumulator |

NOTE

A:               Accumulator                          SPh:    High-order stack pointer address
B, C, D, E, H, L: Working Registers                    PCl:    Low-order program counter address
SPl:            Low-order stack pointer address        PCh:    High-order program counter address

**Table 2    8080 REGISTER ASSIGNMENT**

## N3002 Register Assignment

The N3002 CPE has more registers than required for the emulation. The exact register assignment for the 8080 Emulator is shown in Table 2. The unused registers may be used for expansion purposes.

## THE 8080 EMULATOR PIPELINES

### General

Several advanced architectural techniques have been implemented in the 8080 Emulator. One of the most important of these is the concept of pipelining. Pipelining is a technique by which tasks that are normally accomplished in a serial fashion are performed in parallel. When implemented properly, pipelining can result in faster operation and better resource utilization.

In the 8080 Emulator design, the pipelining concept was implemented in two areas:

1. A multi-level pipeline is used to handle the macro instruction fetching to ensure that the next three consecutive instructions are available locally in the CPU.
2. A single-level pipeline is used to facilitate simultaneous execution of the current microinstruction and fetching of the next microinstruction.

The main advantage of this type of architecture is the resulting performance enhancement due to overlapping operations. Large scale computing machines, such as the IBM

360/195 and the CDC STAR, were implemented using similar concepts.

### Basic Concepts

The 8080 Emulator provides two excellent examples of the pipelining technique. The serial processes to be performed in parallel are the fetch and execution of instructions (both micro and macro).

With a non-pipelined CPU design, the basic machine cycle is a serial process. As an example, Table 3 shows a series of machine cycles and their respective operations:

| CYCLE X | CYCLE X+1 | CYCLE X+2 |
|---|---|---|
| Fetch Inst. N | Execute Inst. N | Fetch Inst. N+1 |

| CYCLE X+3 | CYCLE X+4 | CYCLE X+5 |
|---|---|---|
| Execute Inst. N+1 | Fetch Inst. N+2 | Execute Inst. N+2 |

**Table 3    NON-PIPELINED MACHINE CYCLES**

This type of serial machine must first fetch an instruction out of memory. Upon receipt of that instruction, execution of the instruction will take place (assuming instruction decode is part of the execution). Therefore, the whole procedure is a two-step serial operation.

The technique of pipelining is to overlap these two serial operations (i.e., the fetch and subsequent execution) into one simultaneous event, as illustrated in Table 4.

| CYCLE Y | CYCLE Y+1 | CYCLE Y+2 |
|---|---|---|
| Fetch Inst. N+1 | Fetch Inst. N+2 | Fetch Inst. N+3 |
| Execute Inst. N | Execute Inst. N+1 | Execute Inst. N+2 |

**Table 4    PIPELINED MACHINE CYCLES**

The motivation for pipeline architecture is primarily to gain speed for a given solid state technology. The gain in speed is made possible by providing dedicated hardware, such as several levels of memory address registers and instruction registers. Therefore, the primary design consideration is the tradeoff between performance and cost.

### The Micro Pipeline

The micro pipeline consists of:

1. Micro Control Memory (the microprogram)
2. A Pipeline Register

As soon as the microinstruction output from Micro Control Memory is stable, it is latched into the Pipeline Register. Once the microinstruction is latched into the Pipeline Register, it is presented as a collection of control fields to the various functional blocks of the 8080 Emulator. With the control fields thus established, execution of the microinstruction takes place. In the meantime, the next microaddress being formed by the N3001 MCU is addressing the next microinstruction. Thus, the micro pipeline is realized in that one microinstruction is executed while the next microinstruction is being accessed.

### The Macro Pipeline

The Macro pipeline structure consists of the following four dedicated registers:

    PC—16-Bit Program Counter residing in R4
    iMAR—16-Bit Internal Memory Address Register
    eMAR—16-Bit External Memory Address Register
    IR—8-Bit Instruction Register

The first three registers of the pipeline are used to maintain addresses that are eventually used to access the external main memory via the Emulator's Memory Address Bus. The last register, the Instruction Register, is used to store the op codes and data returned from memory via the 8080 Emulator's Data Bus.

The address in each of the first three registers is updated at the end of every macro instruction cycle. This operation is detailed in Table 5.

The basic operation of the macro pipeline is as follows: during any macro instruction cycle, for example, cycle (X+1), the (N)th instruction is exeucted, the (N+1)th instruction is fetched, and the iMAR and PC registers are updated.

During a jump or branch operation, the entire pipeline will be reinitiated to reflect the new address and its subsequent addresses.

## MEMORY AND I/O CONTROLS

The control signals for memory and I/O operations are generated by a PROM (82S123). These control signals include $\overline{RTRAP}$, $\overline{HLTA}$, $\overline{INTA}$, $\overline{IOW}$, $\overline{IORI}$, $\overline{MEMW}$, and $\overline{MEMRI}$. Except for $\overline{IORI}$ and $\overline{MEMRI}$, all of the above signals are presented directly to the outside world. $\overline{IORI}$ and $\overline{MEMRI}$ are strobed into separate flip-flops which output $\overline{IOR}$ and $\overline{MEMR}$, respectively.

Figure 5 illustrates the memory and I/O control signal logic implementation. The associated PROM truth table is provided in Appendix B, Table B-4 (PROM U10). Note that only 16 addresses of the 82S123 are used. The high-order address bit $A_4$ and chip enable (CE) are both tied to ground as shown in Figure 5.

The assignment of control signals for each bit of the PROM output is shown in Table 6.

## ADDRESSING MICROCONTROL MEMORY

### The Instruction Decode PROM

Each macro instruction (8080 instruction) is implemented by a sequence of microinstructions. The first microaddress of each microroutine is derived directly from the op code by the Instruction Decode PROM.

The op code fetched from external memory is latched into the Instruction Register. The Instruction Register then addresses the Instruction Decode PROM.

The outputs of the Instruction Decode PROM are loaded into the internal memory address register of the N3001 MCU via the $\overline{PX}$ and $\overline{SX}$ input buses. The N3001, under control of the microinstruction, will generate a corresponding address output on the $MA_{8-0}$ bus according to the format shown in Table 7.

When one or more microinstructions are shared by a group of macro instructions, the op code is saved in the Instruction Register and used again for a secondary decode.

The Instruction Decode PROM has 512 addressable locations. Two hundred and fifty-six locations are used for primary decode of 8080 instructions, Multiply, Divide, and user-defined macro instructions. The remaining 256 locations are used for secondary decodes. The microprogram enables the secondary half of the Instruction Decode PROM by setting the Secondary Jump ($\overline{SJM}$) bit. The $\overline{SJM}$ bit, once latched into the pipeline register, becomes the most significant bit of the Instruction Decode PROM's address field. The secondary address, thus decoded, is loaded

| | MACRO INSTRUCTION CYCLE | | |
| --- | --- | --- | --- |
| | (X) | (X+1) | (X+2) |
| Instruction being executed | N-1 | N | N+1 |
| eMAR | N | N+1 | N+2 |
| iMAR | N+1 | N+2 | N+3 |
| PC (R4) | N+2 | N+3 | N+4 |

Table 5   MACRO PIPELINE ADDRESS UPDATE SEQUENCE

| $B_7$ | $B_6$ | $B_5$ | $B_4$ | $B_3$ | $B_2$ | $B_1$ | $B_0$ |
| --- | --- | --- | --- | --- | --- | --- | --- |
| | $\overline{RTRAP}$ | $\overline{HLTA}$ | $\overline{INTA}$ | $\overline{IOW}$ | $\overline{IORI}$ | $\overline{MEMW}$ | $\overline{MEMRI}$ |
| Most-Significant Bit (not used) | | | | | | | Least-Significant Bit |

Table 6   ASSIGNMENT OF CONTROL SIGNALS



**GENERATION OF MEMORY AND I/O CONTROL SIGNALS**

NOTES

1. Refer to PROM truth table in Appendix B, Table 4.
2. Refer to signal description in Appendix A.
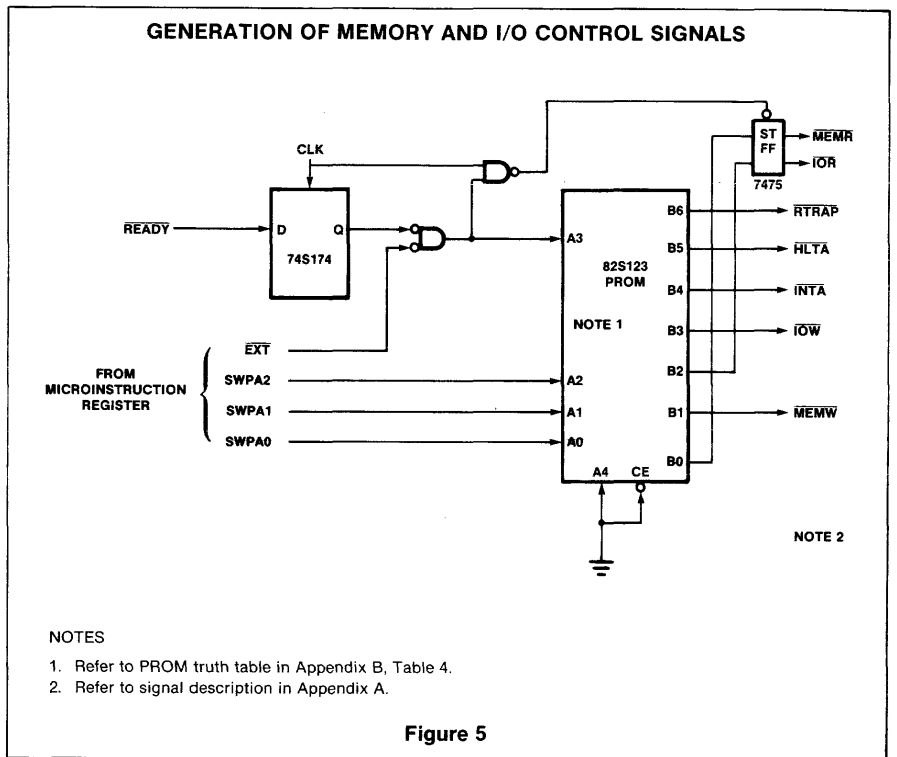
Figure 5

| ADDRESS TO INSTRUCTION DECODE PROM | INPUT TO N3001 | OUTPUT OF N3001 | (MICROPROGRAM ADDRESS) |
| --- | --- | --- | --- |
| IR$_{(7-0)}$ and SJM | $PX_4$ | $MA_0$ | |
| | $PX_5$ | $MA_1$ | |
| | $PX_6$ | $MA_2$ | |
| | $PX_7$ | $MA_3$ | |
| | $SX_0$ | $MA_4$ | |
| | $SX_1$ | $MA_5$ | |
| | $SX_2$ | $MA_6$ | |
| | $SX_3$ | $MA_7$ | |
| | | $MA_8$ | (set to zero) |

Table 7   GENERATING ADDRESS OUTPUT ON THE $MA_{8-0}$ BUS

into the N3001 to generate the microprogram's next address.

### N3001 Address Control

Once the starting address for a microroutine has been determined by the Instruction Decode PROM and loaded onto the

Microaddress (MA) Bus (via the N3001), all subsequent microaddresses in the routine are specified by the microinstructions. Each microinstruction generates an Address Control (AC) field for the N3001 MCU. The AC Bus determines what function the MCU will perform on the current address to

produce the next MA value. (For a detailed description of the AC functions, refer to the N3001 data sheet in Appendix D.)

Jumps required by the microprogram are implemented with supplemental control of the $MA_0$ and $MA_4$ bits. $MA_0$ is driven by a multiplexer that selects from among the $MA_0$ output of the N3001, Ready, and Hold. The Ready and Hold inputs are used to create dynamic wait loops while the microprogram is waiting for those signals. The $MA_4$ bit is an AND term of the N3001's $MA_4$ output and a control signal that goes false (thus forcing $MA_4$ to a logical zero) when both Interrupt Strobe (IST) and Hold are true.

Jumps required by the macro program are implemented by the Jump Control PROM (U-37). The Jump Control PROM is addressed by the 8080 Program Status Word bits $\overline{Zero}$, $\overline{Carry}$, $\overline{Parity}$, $\overline{Sign}$; three bits of the Instruction Bus $(IR_{(5-3)})$; and SJM. The output of the Jump control PROM (1 bit) is OR ed with LD2 to generate the $SX_3$ input to the N3001 MCU. This feature allows conditional jumps to be executed when the Load function of the N3001 is performed.

## MICROINSTRUCTION FORMAT

### General

The microprogram is realized as a series of microinstructions. All microinstructions for the 8080 Emulator have the same format, namely, a 48-bit word consisting of the control fields shown in Figure 6.

To fully describe the functions of each one of these control fields, the following procedure has been adopted:

1. A pictorial overview of the microword broken down into six groups of eight consecutive bits is presented. Field names and their control functions are briefly described.
2. Detailed descriptions of all control fields are provided.

### Microinstruction Control Field Descriptions

AC (7 bits)  **ADDRESS CONTROL FIELD**
The Address Control Field determines the jump function executed by the N3001 MCU. See Figure 7.

LD (1-bit)  **LOAD**
Load enables the load function of the N3001 MCU. Load initiates a microroutine based on a primary or secondary decode of the instruction op-code (see SJM and LD2 and Figure 8).

CS (3 bits)  **CARRY SELECT CONTROL FIELD**
This field controls the Carry In and Carry Out of the CPE arrays. Specifically, the Carry Select Control Field (see Figure 8):
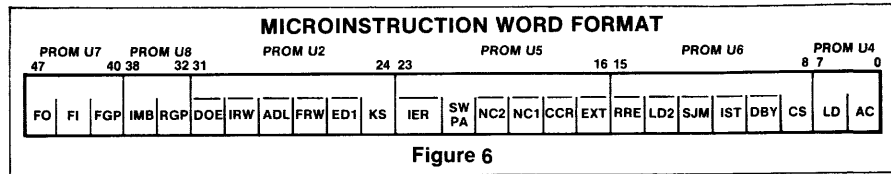


Figure 6
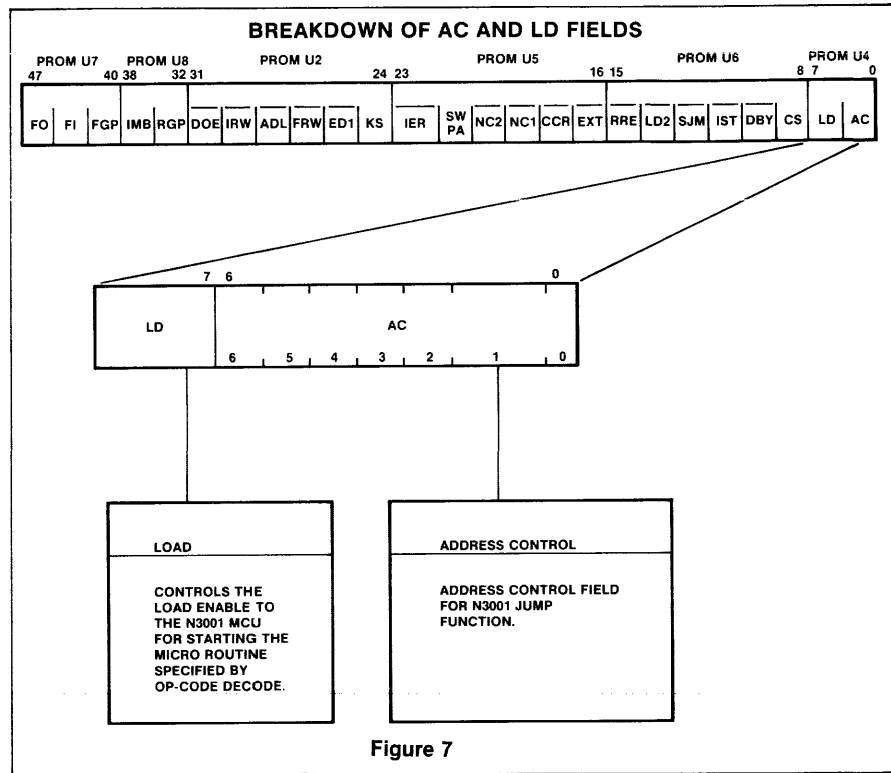


Figure 7
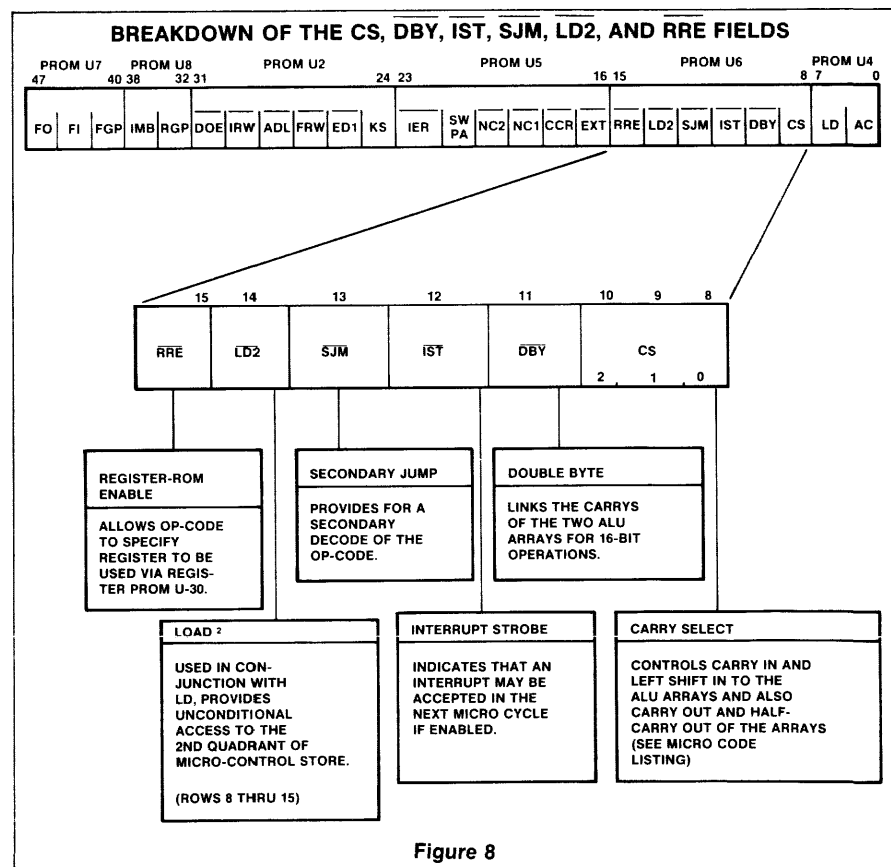


Figure 8

1. Determines 8 or 16-bit operation.
2. Determines whether Carry In to the arrays is complemented or not.
3. Multiplexes the carry bits for shift operations.
4. Controls the carry and half carry input to the Program Status Word register.

**$\overline{DBY}$ (1-bit)  DOUBLE BYTE**

$\overline{DBY}$ specifies the configuration of the carry look-ahead logic. When $\overline{DBY}$ is high, the carry look-ahead is computed for each 8-bit array individually. When $\overline{DBY}$ is low, the carry look-ahead is computed over the 16-bit ALU. See Figure 8.

**$\overline{IST}$ (1-bit)  INTERRUPT STROBE**

This control bit enables the interrupt control circuitry for the 8080 Emulator. When $\overline{IST}$ is true, $\overline{INT}$ and $\overline{HOLD}$ are allowed to interrupt the N3001 MCU. See Figure 8.

$\overline{IST}$ goes true during the fetch cycle following all macro instructions except:
1. Enable Interrupt (EI)
2. Disable Interrupt (DI)
3. The Fetch of an Interrupt Vector

**$\overline{SJM}$ (1-bit)  SECONDARY JUMP**

This control bit addresses the Instruction Decode PROM. $\overline{SJM}$ thus divides the Instruction Decode PROM into two fields. The first field is the primary jump field, while the second field is the secondary jump field. In many cases the primary decode defines a general class of instructions that share the same beginning micro routine. $\overline{SJM}$ then allows a secondary decode of the macro instruction that calls up the specific microroutine to complete execution of the macro instruction. See Figure 8.

**$\overline{LD2}$ (1-bit)  LOAD 2**

$\overline{LD2}$ can force the $\overline{SX3}$ input to the N3001 to go low. It gives the microprogram control of the jump destination when the Load function is used for microaddress generation. $\overline{LD2}$ also permits an op-code decode into the second quadrant of micro control storage (locations $080_{16}$ through $0FF_{16}$). See Figure 8.

**$\overline{RRE}$ (1-bit)  REGISTER ROM ENABLE**
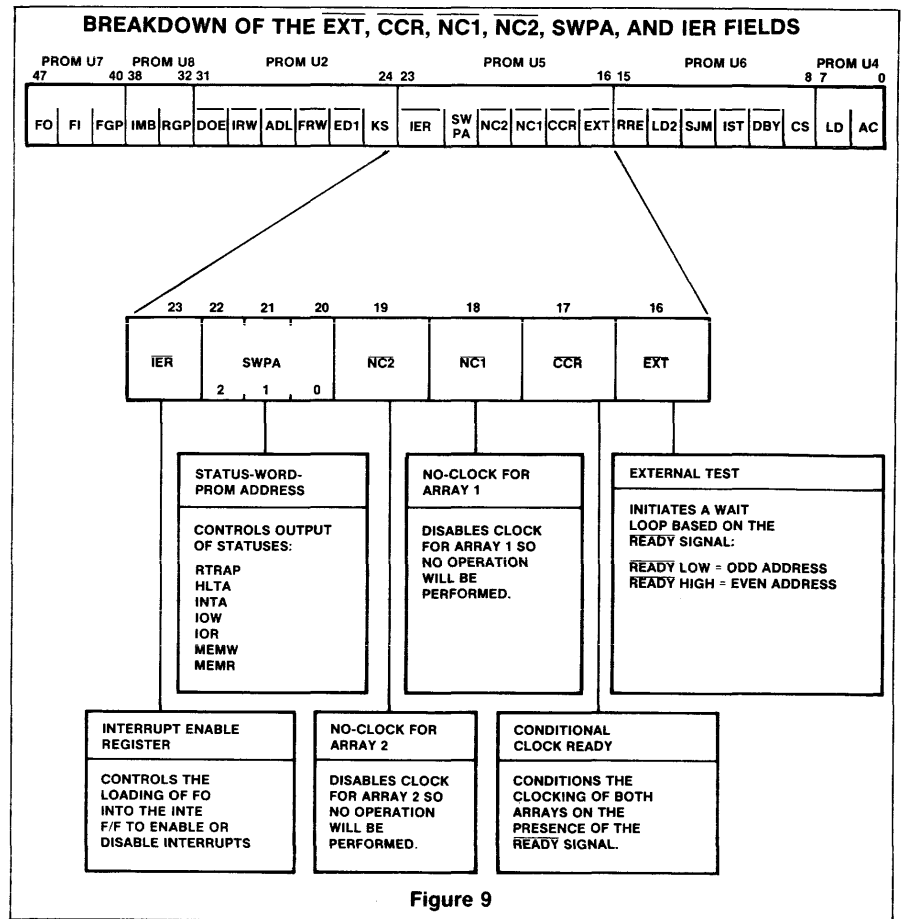


Figure 9

This bit determines whether the N3002 array register group is specified by the microprogram or the Instruction Bus. See Figure 8.

**$\overline{EXT}$ (1-bit)  EXTERNAL TEST**

External Test is a control bit that the microprogram activates when it is waiting for Ready to return from external memory or I/O. While the microprogram is waiting for Ready, it remains in the idle loop based on the value of MA0. MA0 is controlled by the MA0 multiplexer which is monitoring Ready. The MA0 multiplexer is switched between the value of Ready or the MCU output by the $\overline{EXT}$ signal. $\overline{EXT}$ also disables the $\overline{HLDA}$ signal and enables Ready addressing of the Control Signal PROM. See Figure 9.

**$\overline{CCR}$ (1-bit)  CONDITIONAL CLOCK READY**

This control bit is used in the dynamic wait loop used for external memory and I/O interfacing. When the microprogram is waiting for Ready to go negative true, this bit disables the clock to both Array 1 and Array 2. Thus disabled, the contents and status of the CPE arrays remains undisturbed until the requested data is ready for processing (see $\overline{EXT}$). See Figure 9.

**$\overline{NC1}$ (1-bit)  NO CLOCK ARRAY 1**

This signal disables the clock to Array 1. With this bit, the microprogram can selectively enable or disable Array 1. See Figure 9.

**$\overline{NC2}$ (1-bit)  NO CLOCK ARRAY 2**

This signal is identical to $\overline{NC1}$ but operates on Array 2. See Figure 9.

**SWPA (3 bits)  STATUS WORD PROM ADDRESS**

This control field provides three bits of the 4-bit address field for the Control Signal PROM. It also provides for the output of the following status signals: $\overline{RTRAP}$, $\overline{HLTA}$, $\overline{INTA}$, $\overline{IOW}$, $\overline{IOR}$, $\overline{MEMW}$, and $\overline{MEMR}$. The fourth bit of the address, $\overline{EXT}$. Ready, disables all of the above seven output status signals. See Figure 9.

**IER (1-bit)  INTERRUPT ENABLE REGISTER**

This signal controls the

Interrupt Enable signal (INTE) by allowing a one or a zero to be written into it from the Flag Output signal (FO). See Figure 9.

**KS (3 bits)**      **K-BUS SELECT**
This field controls the origin of the data for the main input bus to the ALU arrays (K-Bus) (See Figure 10).

KO (000) = All 0's
KD (001) = ALU Data Out
KM (010) = Memory Data
KIR (011) = Instruction Register
K1 (100) = All 1's
KND (101) = Complemented ALU Data Out
KNM (110) = Complemented Memory Data
KNIR (111) = Complemented Instruction Register

**$\overline{ED1}$ (1-bit)**      **ENABLE DATA 1**
The Data Out buses of the two ALU arrays are tied together into one 8-bit bus. $\overline{ED1}$ determines which CPE array will drive this bus. $\overline{ED1}$ low enables array 1 data; $\overline{ED1}$ high enables array 2 data. See Figure 10.

**$\overline{FRW}$ (1-bit)**      **FLAG REGISTER WRITE**
The newly calculated Zero, Parity and Sign flag bits are loaded into the Program Status Word (PSW) latch with this control signal. See Figure 10.

**$\overline{ADL}$ (1-bit)**      **ADDRESS LOAD**
This 1-bit control loads the external memory address register from the memory address register internal to the N3002 CPE arrays. This address is then used to access memory or I/O. See Figure 10.

**$\overline{IRW}$ (1-bit)**      **INSTRUCTION REGISTER WRITE**
Data input over the bidirectional data bus is loaded into the instruction register with this control line. [Active upon termination of Memory Read (MEMR), Input-Output Read (IOR), and while receiving the interrupt vector ($\overline{INTA}$)]. See Figure 10.

**$\overline{DOE}$ (1-bit)**      **DATA OUT ENABLE**
Enables ALU data output onto the bidirectional data bus during Memory Write ($\overline{MEMW}$) and Input-Output Write ($\overline{IOW}$) operations. See Figure 10.

**RGP (5 bits)**      **REGISTER GROUP CONTROL FIELD**



**BREAKDOWN OF THE KS, $\overline{EDI}$, $\overline{FRW}$, $\overline{ADL}$, $\overline{IRW}$, AND $\overline{DOE}$ FIELDS**

**Figure 10**



**BREAKDOWN OF THE RGP AND IMB FIELDS**
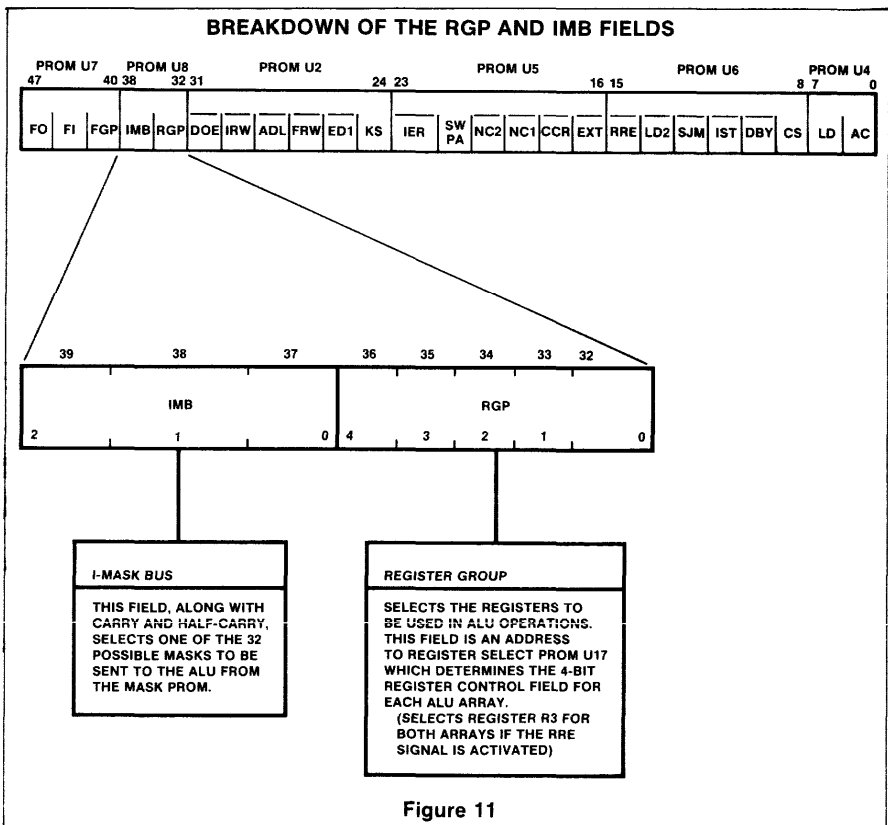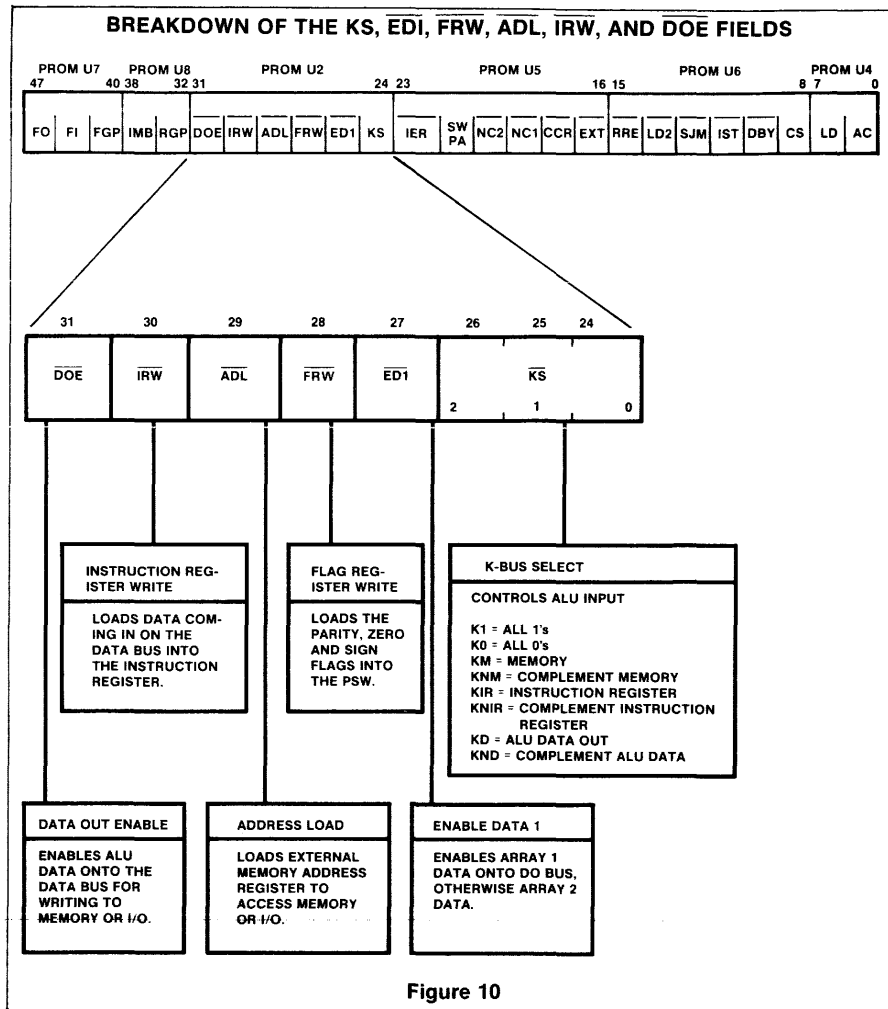
**Figure 11**

The Register Group Control Field input to each CPE array is generated by two Register Control PROMs (see Figure 11):

1. The op code Register Control PROM
2. The Microcode Register Control PROM

The PROM controlled directly by the microinstruction is the Microcode Register Control PROM. The address for this PROM is the RGP control field. The RGP field addresses 32 pairs of Register Group control fields (one for each array). For macro instructions that call for specific 8080 registers, the output of the Microcode Register Control PROM is combined with the output from the op code Register Control PROM. Addressed by the Instruction Register Bus, the op code Register Control PROM's output is wire OR'ed with the Microcode Register Control PROM's output. This Register Group control field generation scheme is illustrated in Figure 12.
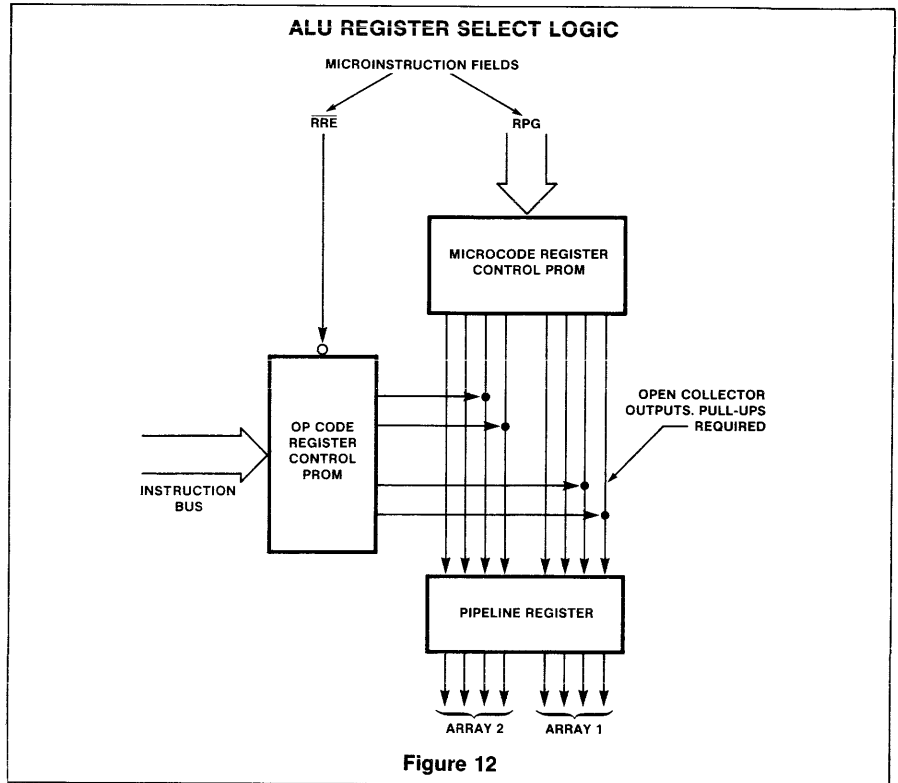
**IMB (3 bits)** | **I MASK BUS CONTROL FIELD**
The I Bus is generated by the I Mask PROM. This 32X8 PROM, an 82S123, is addressed by the 3-bit I Mask Bus control field in conjunction with the carry bit and the half carry bit. The I bus output from the I Mask PROM is logically AND'ed with the K-Bus or used as an addend to N3002 registers. See Figure 12.

**FGP (4 bits)** | **FUNCTION GROUP CONTROL FIELD**
The 3-bit Function Group (F-6, F-5, and F-4) input to each array is determined by this field (see Figure 13). Table 7 details how each Function Group is generated.

Separate F-5 control bits allow the two arrays to perform separate ALU functions during the same microcycle. This feature results in significant microinstruction savings in terms of microinstructions required to accomplish a specific task. See Figure 13.



**ALU REGISTER SELECT LOGIC**

**Figure 12**



**BREAKDOWN OF FGP, FI, AND FO FIELDS**
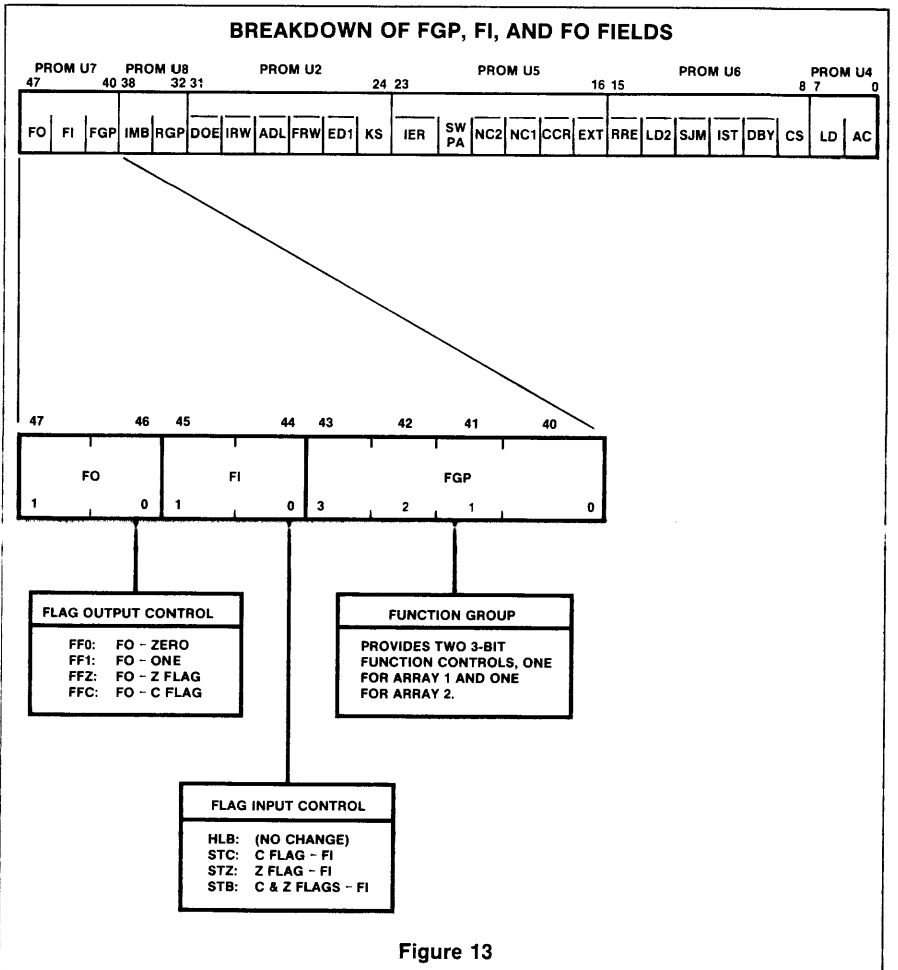
**Figure 13**

| FGP CONTROL | F GROUP ARRAY 2 | | |
|---|---|---|---|
| | F-6 | F-5 | F-4 |
| F-6 | X | | |
| F1-5 | | | |
| F2-5 | | X | |
| F-4 | | | X |

| FGP CONTROL | F GROUP ARRAY 1 | | |
|---|---|---|---|
| | F-6 | F-5 | F-4 |
| F-6 | X | | |
| F1-5 | | X | |
| F2-5 | | | |
| F-4 | | | X |

NOTE

(X indicates where each of the four signals is used).

**Table 7  F GROUP GENERATION**

**FI (2 bits)    FLAG INPUT CONTROL FIELD**
Determines how the Z Flag and the C Flag of the N3001 MCU are loaded with data from the Flag Input ($\overline{FI}$). Either one or both of the flags may be set to the value of $\overline{FI}$. Alternatively, both flags may be held constant. See Figure 13.

**FO (2 bits)    FLAG OUTPUT CONTROL FIELD**
Controls the Flag Output ($\overline{FO}$) of the N3001 MCU. FO may reflect the current value of either the Z Flag or the C Flag. $\overline{FO}$ may also be forced to a logical one or zero. See Figure 13.

# IMPLEMENTING 8080 INSTRUCTIONS— A DETAILED EXAMPLE

## Implementing the MOV A,H Instruction

The MOV A,H instruction transfers the contents of register H to the Accumulator (without affecting the condition flags). Execution of this macro instruction by the 8080 Emulator requires fetching of the op code from main memory and then moving the contents of R2 of Array 2 to the T register of both arrays 1 and 2 (A ← H). A simplified flowchart of the microroutine is shown in Figure 14. A detailed description of the operation is described below.
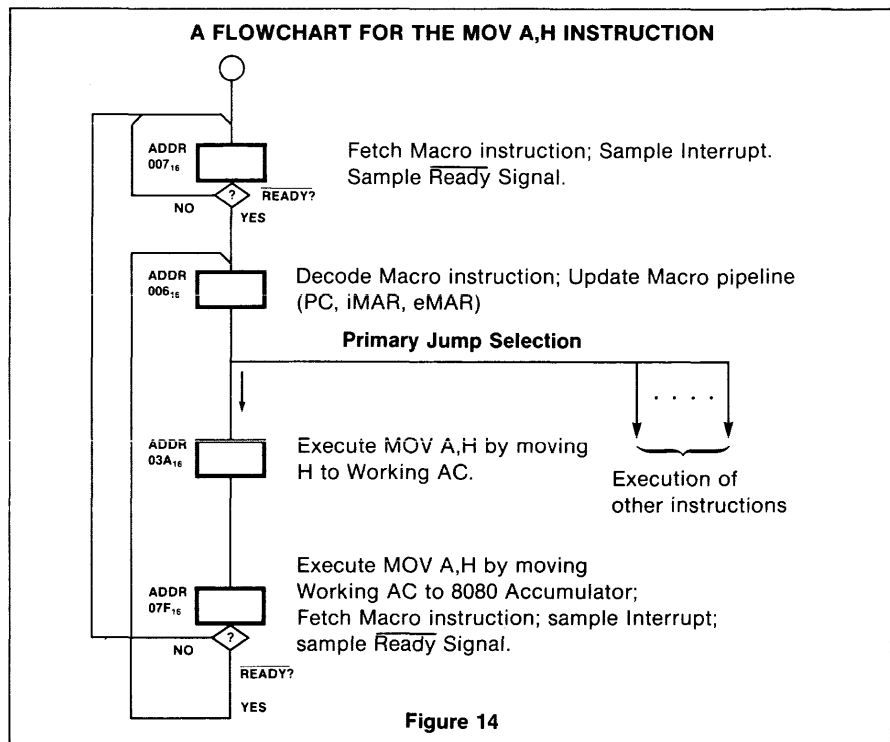
### THE FETCH
Every microroutine returns the microprogram to microaddress $006_{16}$. However, until $\overline{Ready}$ is returned from external memory (indicating valid data on the Data Bus), the microinstruction actually being executed is located at $007_{16}$. This is because MA0 is driven by a multiplexer which, in fetch sequences, is routing ReadyQ to the MA0 line. Until Ready goes true, the microprogram executes $007_{16}$. Microinstruction $007_{16}$ is illustrated in Figure 15.

First note that the Address Control (AC) field specifies a jump to current row (Row 0), Column 6. But, as long as the $\overline{Ready}$ line is false, the microinstruction will jump to itself. This is the dynamic wait loop for Ready.

While this single microinstruction loop at $007_{16}$ is being executed, $\overline{IRW}$ will repeatedly latch the Data Bus into the Instruction Register. $\overline{EXT}$ is enabling $\overline{Ready}$ to control MA0, and $\overline{IST}$ is enabling the interrupt logic.

When $\overline{Ready}$ goes true, the microprogram moves on to $006_{16}$. Microinstruction $006_{16}$ is presented in Figure 15.

Microinstruction $006_{16}$ performs two basic functions.

1. Maintains the microaddress pipeline.
2. Translates the op-code into a beginning address for the MOV microroutine.

If it is assumed that the MOV A,H instruction was fetched from macro memory location N, then the current status of the macro pipeline is as follows:

| LOCATION | | CONTENTS |
|---|---|---|
| PC (R4) | = | N+2 |
| 3002 MAR | = | N+1 |
| Ext MAR | = | N |

Microinstruction $006_{16}$ must update the pipeline to:

| LOCATION | | CONTENTS |
|---|---|---|
| PC (R4) | = | N+3 |
| 3002 MAR | = | N+2 |
| Ext MAR | = | N+1 |

The PC (R4) is updated by performing a double byte increment on R4. The requisite microinstruction control fields are:

°°DBY Places the CPE array in the 16-bit operand



**A FLOWCHART FOR THE MOV A,H INSTRUCTION**

ADDR $007_{16}$  Fetch Macro instruction; Sample Interrupt. Sample $\overline{Ready}$ Signal.

ADDR $006_{16}$  Decode Macro instruction; Update Macro pipeline (PC, iMAR, eMAR)

**Primary Jump Selection**

ADDR $03A_{16}$  Execute MOV A,H by moving H to Working AC.

Execution of other instructions

ADDR $07F_{16}$  Execute MOV A,H by moving Working AC to 8080 Accumulator; Fetch Macro instruction; sample Interrupt; sample $\overline{Ready}$ Signal.

**Figure 14**

**MICROCODE LISTING FOR MOV, A,H**

| ADDR | FO | FI | FGP | IMB | RGP | DOE | $\overline{IRW}$ | ADL | $\overline{FRW}$ | $\overline{ED1}$ | KS | IER | SWPA | $\overline{NC2}$ | $\overline{NC1}$ | CCR | $\overline{EXT}$ | $\overline{RRE}$ | LD2 | SJM | $\overline{IST}$ | DBY | CS | LD | AC | ; COMMENTS |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| (007H): | | | NOP | | RFF | | IRW | | | | K1 | | | | | | EXT | | | | IST | | NAN | | JCR(006H) | ; FETCH |
| (006H):FF1 | | | LMI | | R44 | | | ADL | | | K1 | | | | | | | | | SJM | | DBY | AN | LD | JPX | ; FETCH |
| (03AH):FF1 | | | ILR | | R33 | | | | | | K1 | | | | | | | RRE | | | | | NAN | LD | | ; |
| (07FH):FF1 | | | LDI | | REE | | IRW | | | | KD | | | | | | EXT | | | | IST | | AN | | JZR06 | ;MOV A, (B,D,H) |

**Figure 15**

mode.

°°FF1 Forces a one to be output on the N3001 $\overline{FO}$ pin.

°°AN Directs the value of $\overline{FO}$ to the $\overline{CI}$ input of the double array uncomplemented.

°°R44 Selects register R4 of both arrays 1 and 2.

°°K1 Forces the negative true K-Bus to all ones.

°°LM1 Forces the ALU function to be performed by the double byte array.

The functional equations for LMI are:

MAR ← Rn and Rn ← Rn + CI.

For a functional description of N3002 microfunctions, refer to Appendix D.

Given the conditions listed above, these equations become:

MAR ← $R_4$ and $R_4$ ← $R_4$ + 1.

The first equation takes place on the first half of the microcycle. This moves the old PC value, N+2, into the N3002 MAR. During the second half of the microcycle, the second equation is executed, updating the PC to N+3.

The last macro pipeline maintenance function, moving the old N3002 MAR value (N+1) into the external MAR, is accomplished at the very beginning of the microcycle by $\overline{ADL}$. $\overline{ADL}$ latches the 16-bit array's AB Bus into the external MAR.

The second major task to be accomplished during microinstruction $006_{16}$ is to direct the microprogram to the MOV microroutine. This is done by using the Instruction Decode PROM output ($03A_{16}$) as a beginning address to the two word execution program which accomplishes the MOV A,H Macro.

The op code joins with the $\overline{SJM}$ bit to address the Instruction Decode PROM. In the present case this value addresses $C5_{16}$ (Truth Table for Instruction Decode PROM in Appendix B) which becomes the $\overline{PX}$ and $\overline{SX}$ inputs to the N3001, which, in turn, becomes the next microaddress as follows:

| INSTRUCTION DECODE PROM ADDRESS | 3001 INPUT | 3001 OUTPUT |
|---|---|---|
| $\overline{SJM}$, $\overline{IR}_{(7,0)}$ | $\overline{PX}_{(7-4)}$, $\overline{SX}_{(3-0)}$ | $MA_{(8-0)}$ |
| | $1C5_{16}$ | $03A_{16}$ |

For MOV A,H, the $006_{16}$ microinstruction determines the next microaddress to be:

$$MA_8\text{-}MA_0 = 03A_{16}$$

At address $03A_{16}$ (Figure 15) the actual execution of the Move instruction begins.

Microinstruction $03A_{16}$ accomplishes two major tasks:

1. Moves R2(16) to AC, the working accumulator,
2. Determines Secondary Jump Destination.

The intermediate move to the Working Accumulator AC is affected by the microinstruction fields:



**ALU REGISTER SELECT LOGIC**
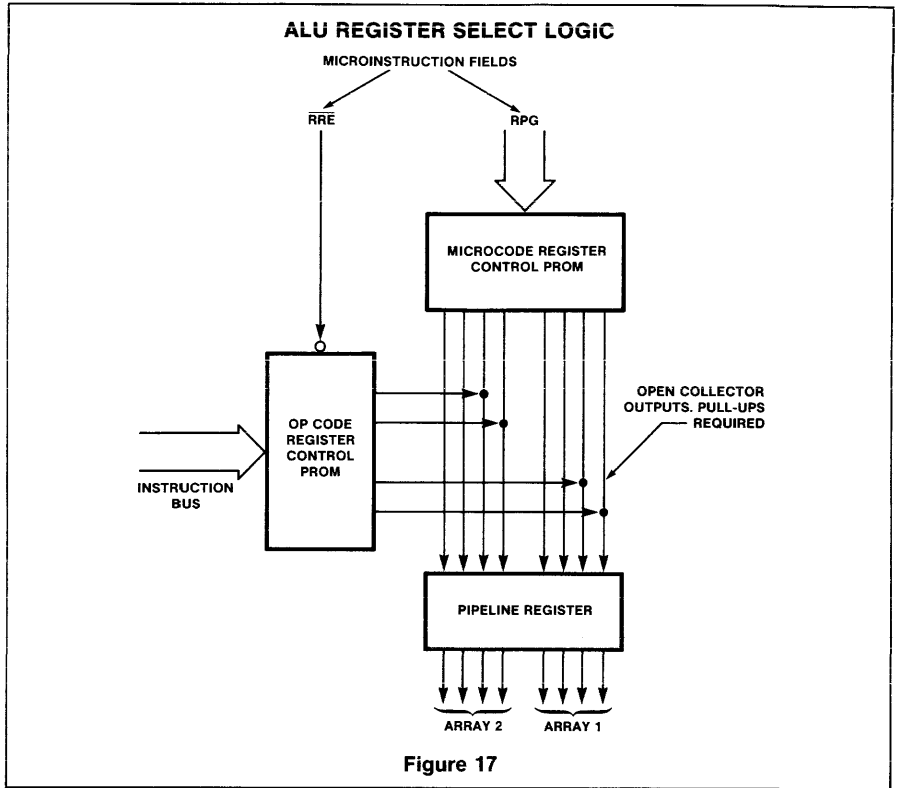
**Figure 17**

°°FF1 Forces a one on the N3001's $\overline{FO}$ pin.

°°NAN Complements $\overline{FO}$ and delivers it as $\overline{CI}$ to both arrays.

°°K1 Forces K-Bus to all ones.

°°R33 Selects Register 3 for each array.

°°ILR CPE function to move R3 to AC.

The functional equation of ILR is:

Rn, AC ← Rn + CI.
where $\overline{CI}$ is false under the control of $\overline{FO}$.

The microinstruction located at $03A_{16}$ executes the first intermediary move (AC ← Rn) for a large group of macro instructions.

The op code controls the two low-order bits of the Register Group inputs with the op code Register Control PROM, which is addressed by the Instruction Register. RRE enables this PROM. The two high-order bits of both array's Register Groups are provided by the microinstruction addressed Micro Code Register Control PROM as always. This approach is presented in Figure 17.

As indicated by the microcode listing for $03A_{16}$, the two 4-bit fields output by the microinstruction to the Microcode Register Control PROM are $3_{16}$ and $3_{16}$. Note that the two low-order bits of each array's R Group are pulled up. With this arrangement, the op code Register Control PROM outputs may go low if so directed by the Instruction Bus.

Referring to Appendix B for the truth tables for the two Register Control PROMs (Tables 2 and 5), we find that for MOV A,H:

| | ARRAY 1 | ARRAY 2 |
|---|---|---|
| Register Selected | R0 (C) | R2 (H) |

Thus, ILR resolves to:

$AC_2$, R2(H) ← R2(H), in array 2

which is the desired result. (The R0 selection for Array 1 is a default condition, and the resulting move, $AC_1$ ← R0(C), is ignored.)

The op code controls the next microaddress because $03A_{16}$ activates the Load signal. The same op code addresses the Instruction Decode PROM as in the previous microinstruction with the notable exception of the Secondary Jump (SJM) bit. At the end of the last microinstruction ($006_{16}$), SJM was latched. Now the active SJM signal selects the secondary jump half of the Instruction Decode PROM. The following address derivation results:

| 3001 OUTPUT | 3001 INPUT | INSTRUCTION DECODE PROM ADDRESS |
|---|---|---|
| $MA_{(8-0)}$ ← $PX_{(7-4)}$, | $SX_{(3-0)}$ ← | SJM, $IR_{(7-0)}$ |
| $07F_{16}$ ← $8_{16}$, $0_{16}$ | ← | 0, $83_{16}$ |

That is, the secondary jump has directed the microprogram to location $07F_{16}$ (Figure 15).

Microinstruction $07F_{16}$ has two major tasks:

1. Complete the move (A ← AC).
2. Return the microprogram to $006_{16}$ if $\overline{Ready}$ is low or else to $007_{16}$.

The move is completed with the microinstruction fields:

°°FF1 $\overline{\text{FO}}$ ← 1
°°AN CI ← $\overline{\text{FO}}$ for both arrays.
°°$\overline{\text{REE}}$ Selects the T register of both arrays.
°°KD Selects the Data Out of the arrays as input to the K-Bus.
°°$\overline{\text{ED1}}$ By default (ED2) selects Array 2 as source of Data Out Bus.
°°LDI CPE function.

The functional equation for LDI is:

$$T \leftarrow (I \quad K) - 1 + CI.$$

Since CI to both arrays is a one, the last two terms cancel.

The I-Bus has been forced to all ones, which leaves the K-Bus unaltered. As the K-Bus is being driven by the DO Bus of the high-order array, the LDI function actually performed looks like:

$T$ (8080 Acc) ← $DO_2$ (containing the value of H).

Thus, LDI completes the functional execution of MOV A,H. All that remains is to return the microprogram to its Fetch cycle ($006_{16}$). $07F_{16}$ has already enabled the fetch signals:

°°$\overline{\text{EXT}}$ Selects $\overline{\text{ReadyQ}}$ as source for $MA_0$.
°°$\overline{\text{IRW}}$ Latches external Data Bus into Instruction Register.
°°$\overline{\text{IST}}$ Enables interrupt acknowledge logic.

With the signals on the previous page already active for one microcycle, $\overline{\text{Ready}}$ may be returned immediately, and the microprogram may proceed directly to $006_{16}$. If $\overline{\text{Ready}}$ is not returned immediately, the next microinstruction executed will be $007_{16}$, and the microprogram will wait for the external memory to respond with the next op code.

A summary of the MOV A,H macro instructions is presented in Figure 15 in the form of a microcode listing.

# CHAPTER 3
# SIGNETICS
# MICROASSEMBLER

## THE ASSEMBLY PROCESS

The basic purpose of an assembler is to translate a microprogram written in symbolic assembly language into executable binary form. The assembly language provides a convenient form for symbolically expressing the microprogram using mnemonics, symbols, and delimiters. A microprogram coded in assembly language is easier to implement and easier to understand, and the assembly language text provides an important documentation element for the microprogram. The assembly language form of the microprogram is known as the source program. The binary form of the microprogram which is produced by the assembler can be loaded directly onto the appropriate PROMs, ROMs and RAMs for execution. The binary form of the microprogram is known as the object program.

The assembly language form of a microprogram consists of a sequence of statements. Each assembly statement requests a specific action from the assembler. A statement may specify microinstructions or data for the object program, define symbols used in other statements, define instruction fields and special mnemonics known as "microps" for use in microinstruction statements, or control other aspects of the assembly, such as listing and object generation, listing spacing, and page headings.

The assembler processes the assembly language source program and produces a binary object program. The object program is a format suitable for PROM, ROM, etc., loaders and programmers. The input to the assembler is the source program. The output of the assembler is the object program and a listing. The assembly listing contains source and object information and serves as the primary documentation of the microprogram.

## THE MICROASSEMBLY LANGUAGE

### Introduction

The microassembly language is a symbolic language for microprogramming. A microprogram is coded as a sequence of microassembly language statements. This set of statements is input for the microassembler and is known as the source microprogram. The allowable microassembly statements, their structure (syntax) and their meaning or function (semantics) are described in subsequent sections.

The source program input to the microassembler consists of a file of records in character format. The placement of statements on source records is free-form, that is, the meaning of statement elements is not tied to their position in the record. Several records may be used for a single statement or several statements may be placed on a single record. The standard source record length for the microassembler is 80 characters.

## Assembly Language Elements

Each microassembly statement consists of characters grouped into microassembly language elements. The basic elements of the language are symbols, numbers (numeric constants), quoted strings, and delimiters (special characters). These basic elements are combined into expressions, operands, statement labels, statement bodies, statements and blocks.

## Symbols

Symbols are 1-to-28 characters long and consist of alphabetic characters, numeric characters, and the special character, at sign (@). The first character of a symbol must be alphabetic or an at sign. Symbols are used for reserved words and for names. Reserved words are special symbols used to identify statements and statement operands. Symbols are used as names for the following program information:

- Values, addresses
- Fields in microinstructions
- Microps
- Memory Blocks

These symbols are used to name user information in the source program and are defined and given values with the appropriate assembly language statements.

## Self-Defining Constants

Self-defining constants are used to specify constant values. The value of a self-defining constant is determined from its representation. Self-defining constants may be any number of characters in length, but the first character must be a numeric character or a quote. Two types of self-defining constants are used: numeric and character constants.

### Numeric Constants

The first character of a numeric self-defining constant is always a numeric character (0 through 9). The numeric constant has the following format: "nnnnr". "r" is an alphabetic character which defines the valid characters for "nnnn" and defines the radix of the constant, as follows:

B— Binary, "nnnn" characters are 0 and 1.
O or Q—Octal, "nnnn" characters are 0 through 7.
D— Decimal, "nnnn" characters are 0 through 9.
H— Hexadecimal, "nnnn" characters are 0 through 9, A through F. A through F represent values 10 through 15, respectively.

If "r" is omitted, the radix of the constant is D (decimal). A hexadecimal constant may contain alphabetic characters, but the first character must be numeric. This can be accomplished by adding leading zeros as required.

### Character Constants

The character self-defining constant is a string of ASCII characters enclosed in quotes. The first and last characters of a character constant must be a quote ('). A quote within a character string is represented by two quotes. The binary value of a character constant is determined by converting each character to 8-bit ASCII (7-bit ASCII with a high-order zero bit appended).

## Expressions

Self-defining constants and symbols which name values and addresses may be combined with operators into expressions to compute values. The operators are the special characters: + (add) and – (subtract).

The operands of each operator may be a symbol or a constant. In addition, an expression operand may be a subexpression enclosed in parentheses. The subtract operator (–) may be used as the first character of an expression indicating that the negative value of the operand following the operator is to be used. An expression operand may also be a reference to the current location counter. The assembler location counter is defined below under data statements. The location counter is referenced with the special character: $ (dollar sign).

Expressions may be used anywhere in a statement where a value is required. Expressions are used to specify the absolute location of microinstructions, the value of a symbol, the length of an instruction field, the value of an instruction field, etc.

## Statements

The basic elements of the microassembly language are combined to form expressions, expression lists, and operands. These are combined to form statements. Statements are the primary language structure of the microassembly language.

Each statement is a command to the microassembler. A statement tells the microassembler to perform a specific action, such as, define a field in a microinstruction, name a value with a symbol, establish a memory block, or specify data to be placed in the object file. The source input to the microassembly is a sequence of statements that request actions by the microassembler. The ultimate purpose of these actions is the production of the listing and object files.

Statements are placed on the source records in free format. They may begin anywhere on a record and may occupy several successive records. Blanks may be interspersed anywhere except within symbols and numeric constants. Each statement is terminated by a semicolon (;). The next statement begins at the semicolon, terminating the previous statement. Multiple statements may be placed on one record.

Comments may also be interspersed within statements. Comments are enclosed in double quotes. The first and last characters of a comment must be a double quote (").

Within the double quotes, any character may be used except the double quote. Comments may be placed anywhere a blank may be used.

The function of each assembly language statement is described in the next section.

## MICROASSEMBLY LANGUAGE STATEMENTS

### Data Statements

The primary microassembly language statements are data statements. These statements produce the object program. Each statement specifies object data for one or more words of the object memory chips. There are two types of data statements, the DCL and the microinstruction statement. The DCL statement specifies a single binary value for one or more object words. The microinstruction statement specifies data in instruction format for object memory.

A data statement may specify the object address for its data, or the assembler location counter may be used. The assembler location counter provides for linear assignment of addresses. A data statement which doesn't specify an object address is assigned the current location counter value as an address, and the location counter is incremented by the length of the data. Subsequent data statements will be assigned to successive memory addresses.

When the object address *is* specified in a data statement, it must be the first operand of the statement. It has the following format:

( <expression> ) :

The value of the expression is the object address for the data statement.

Data statements may also be labeled. The value of a symbol naming a data statement is the object address of the data. Label symbols must follow the object address operand (if any). They are specified with the following format:

<symbol> :

A DCL statement specifies an object data value as a single expression. The number of object memory words (if more than 1) to be used for the value may also be specified in the DCL statement. If the object value is not specified in the DCL statement, the statement reserves memory space and does not produce object data.

The microinstruction statement specifies object instructions. The body of the microinstruction statement is a list of operands. Microinstruction operands assign values to instruction fields.

The format of object instructions is defined using definition statements. These are described below. An instruction format is divided into bit fields. A microinstruction statement specifies an object instruction by assigning values to the fields of the instruction.

The values are assigned to fields with field assign operands. A field assign operand has the following format:

<field-name> = <expression>

The value of the expression is assigned to the named instruction field.

In addition to field assign operands, an operand of a microinstruction statement may also be a reference to a microp. Microps are defined using definition statements. A microp is a shorthand method of assigning values to fields.

When the microp is defined, a list of field assign operands are specified. When the microp is referenced in a microinstruction statement, these pre-defined field assignments are made. A reference to a microp consists of the microp name.

Microps may also have arguments. The arguments are a list of expressions separated by commas. The argument list (if any) follows the microp name and is enclosed in parentheses. The argument values are used in the field assign expressions of the microp.

### Memory Block Statements

Preceding any data statements in the source program is the PROGRAM statement. The PROGRAM statement specifies the length of the object memory word and the maximum number of object words in the microprogram. The PROGRAM statement also initializes the assembler location counter to zeros. The PROGRAM statement defines a block of object memory and gives the block a name. The subsequent data statements specify data for the memory. A memory block is terminated by a PROGRAM statement for a second block of memory or the END statement. The END statement is always the last statement of the source program.

### Definition Statements

All definition statements must precede the memory block statements. There are two types of definition statements, the microp statement and instruction definition statements. A microp statement defines a microp.

An instruction format is defined with a set of statements in the following format:

<instruction-statement> ;
<field-statement> ;
<field-statement> ;
.
.
END INSTRUCTION;

The instruction statement specifies the width of instruction in bits. The field statement names each field and specifies the field width. Fields are assigned to successive bits in the instruction beginning at the high-order (leftmost) bit. A field statement may also specify a default value. The default value is assigned to the field when no value

is assigned in a microinstruction statement.

## Directive Statements

### EQU Statement

The EQU statement defines symbols and assigns values to them. A symbol defined in an EQU statement may be used as an operand in an expression.

### SET Statement

The SET statement is similar to an EQU statement in that it assigns a value to symbols. The difference is that values of symbols defined in SET statements may be redefined by subsequent SET statements. EQU symbols may not be redefined.

### ORG Statement

The ORG statement sets the assembler location counter to a new value (address).

### OBJECT Statement

The OBJECT statement allows or suppresses output of the object program by the assembler.

### LIST Statement

The LIST statement allows or suppresses listing output of the assembler. It also may suppress listing of object information while allowing listing of source information.

### SPACE Statement

The SPACE statement generates blank lines (spaces) in the listing output of the assembler.

### EJECT Statement

The EJECT statement causes a new page with page headings in the listing output of the assembler.

### TITLE Statement

The TITLE statement specifies user text to be placed in the page heading of the listing output. The TITLE statement also causes a new page with the updated page heading.

## Using the Microassembler

The microassembler is composed of two separate programs written in FORTRAN: the microassembly program and the microformat program. The microassembly program has one input file and two output files. The input file for the microassembly program is the source program. The two output files are the assembly listing file and the intermediate object file. The listing file includes a cross-reference listing of all symbols in the source program. The object file contains the object program in an intermediate object format. The intermediate object output from the microassembly program is input to the microformat program.

The microformat program has two input files and two output files. The two input files are intermediate object files from the microassembly program and a file of control statements. The two output files are the loadable object file and a listing of the control input to the microformat program. The microformat program control statements specify the format of the loadable

object output of the microformat program. The format of the loadable object can be tailored for the programmer or loader which is to be used for the memory chips. The loadable object will be in proper format for input to a PROM, ROM, RAM loader/programmer. The control statements also specify allocation of instruction fields to individual memory chips, inversion of fields and separate output for each PROM.

The microassembly program and the microformat program are written in ANSI FORTRAN and may be compiled and executed on any computer system supporting standard FORTRAN. These programs will also be available on the NCSS, TYMSHARE, and General Electric timesharing services. For a detailed description of the microassembler, refer to the Signetics Microassembler Manual.

# CHAPTER 4
# 8080 EMULATOR
# KIT ASSEMBLY

**signetics**

## KIT ASSEMBLY

The following checklist is provided to aid the technician in an orderly assembly of the 8080 Emulator. Please refer to Parts List and Assembly Drawing in Appendix A.

A. Inventory the parts against the parts list.

B. Install supplied integrated circuit sockets as follows:
   1. 5 each 16-pin sockets at U10, U17, U24, U30 and U37.
   2. 7 each 24-pin sockets at U2, U4, U5, U6, U7, U8, and U29.
   3. 8 each 28-pin sockets at U31, U32, U38, U39, U43, U44, U51 and U53.
   4. 1 each 40-pin socket at U12.
   (Additional sockets may be installed to enhance the ease of checkout.)

C. Install discrete components (resistors, capacitors, diodes) being careful to observe proper polarity on C1, CR1, CR2 and the three 22uf bypass capacitors.

D. Install integrated circuits U1 through U61 with pin 1 toward the U5 end of the PC board.

E. Install clock jumper to pads 1, 2 and 3 located between U1 and U9.

1. For internal clock, connect a jumper between pads 1 and 2.
2. For external clock (from P1 pin 31), connect a jumper between pads 2 and 3.

## CHECKOUT PROCEDURE

A. The first step in checkout is to provide the 8080 Emulator with an external memory. A suggested approach is to place a PROM containing a diagnostic program at the bottom of the 16K memory space (that is, beginning at address $0000_{16}$). The 82S115 (512X8) Schottky PROM is ideal for this purpose. Complete checkout also requires that some RAM be provided. The 82S09 (64X9) RAM is suggested as it enables the 8080 Emulator to run at full speed while minimizing the checkout hardware required. The placement of RAM within the memory space is not critical but it must correspond to the RAM reference addresses contained in the diagnostic program. (Note: Remember that the Address and Data Buses are negative true logic.)

B. With the clock jumper wired for external clocking, a pulse generator may be used as the clock input to edge connector P1 (pin 31). This allows the system clock to be adjusted from one-shot operation to the maximum clock frequency of 6.6Mhz (for minimum positive and negative pulse widths, refer to the Electrical Specifications in Appendix A).

C. When power is applied to the Emulator, the Power On Reset circuit forces the microprogram to either microaddress 1FF or 1FE. Both 1FF and 1FE send the microprogram to an initializing routine. The Power On Reset microroutine fetches a macro instruction from address $0000_{16}$ in external memory.

D. The execution of macro instructions returned from external memory can be traced by following the microinstruction sequences as presented in the microcode listing (Appendix E). The location of the microprogram is determined by the value of the MA Bus. Monitoring the MA Bus with a logic analyzer may prove very helpful in debugging any assembly errors.

# APPENDICES

## 8080 EMULATOR SPECIFICATIONS

**LOGIC DIAGRAM**
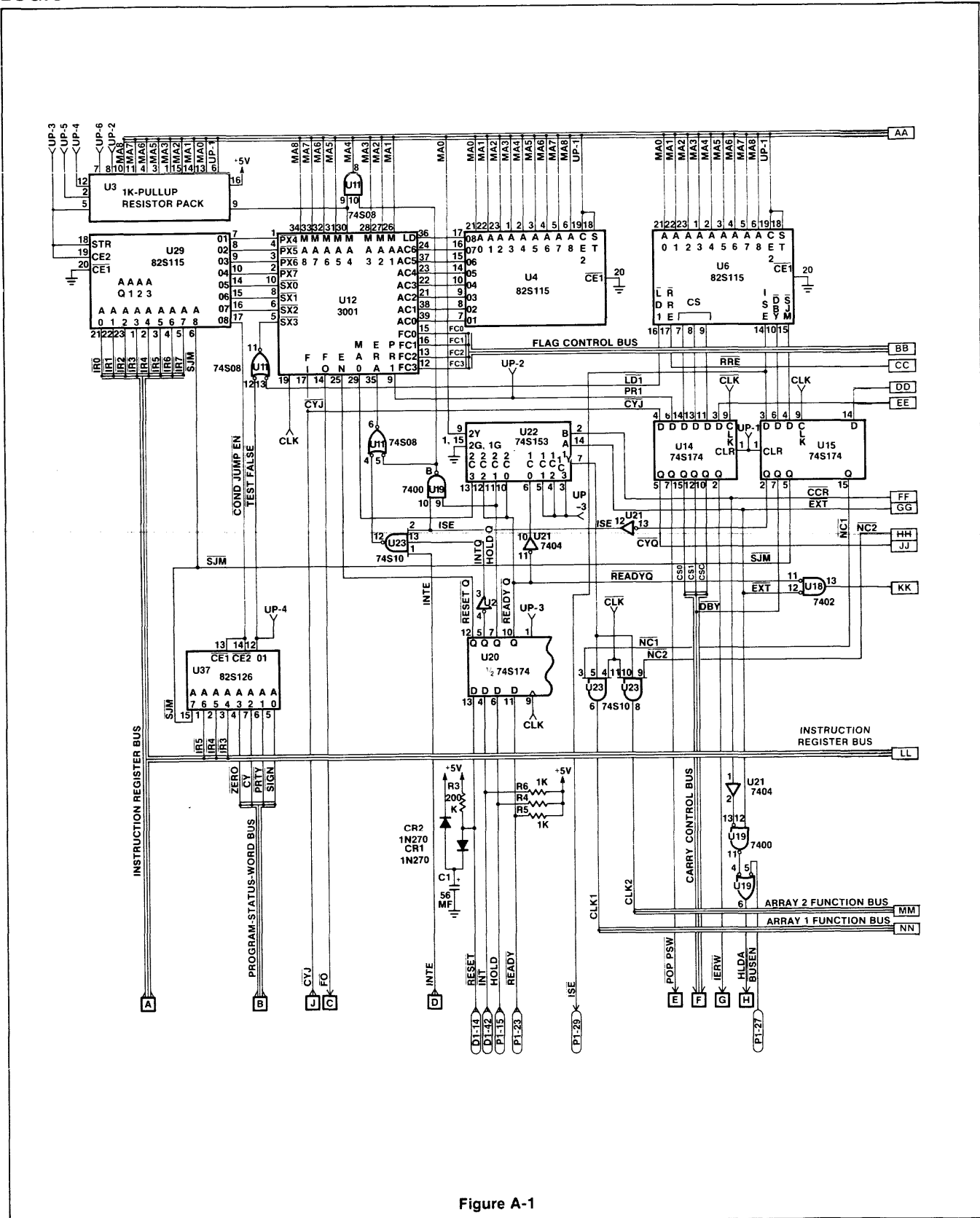


Figure A-1
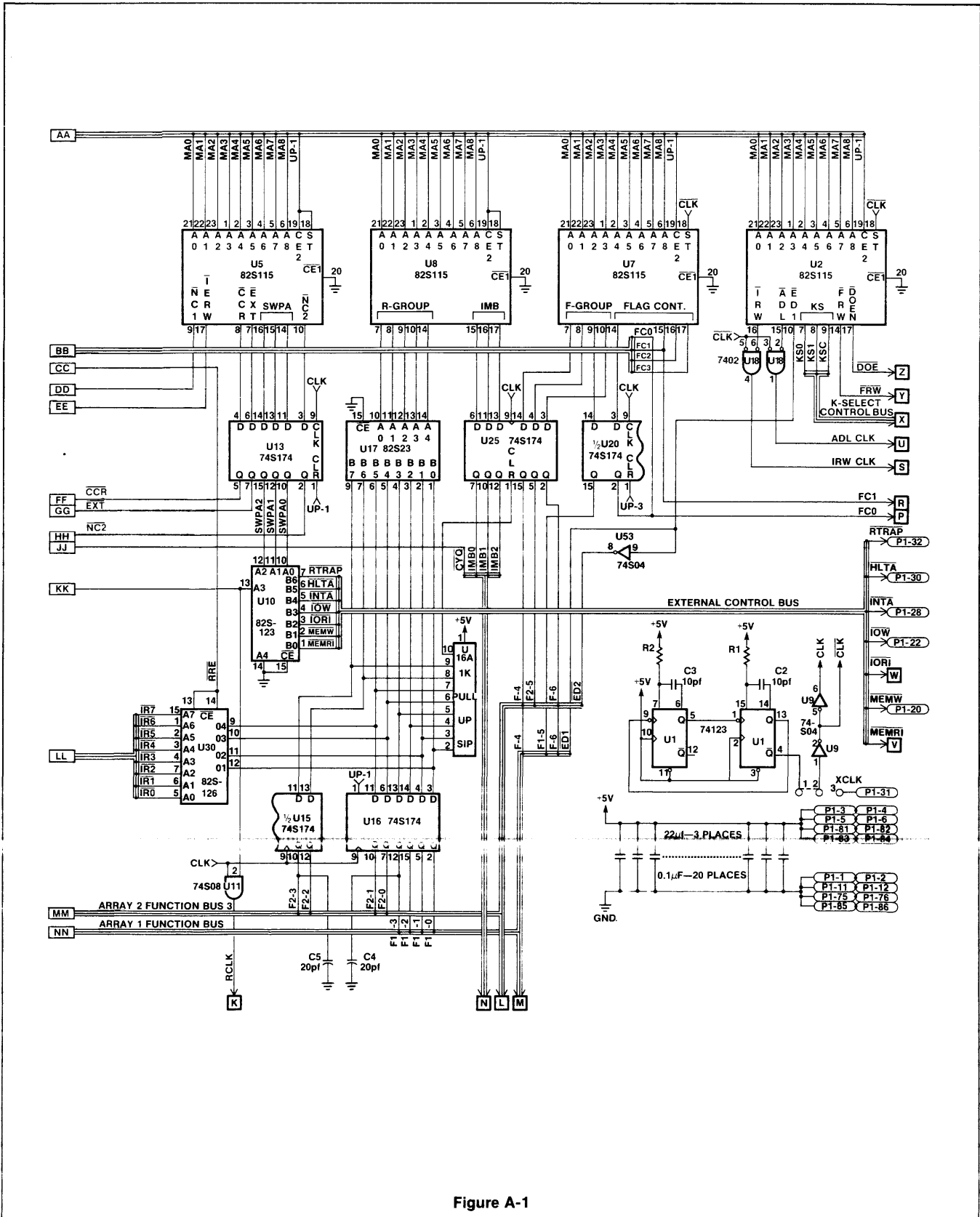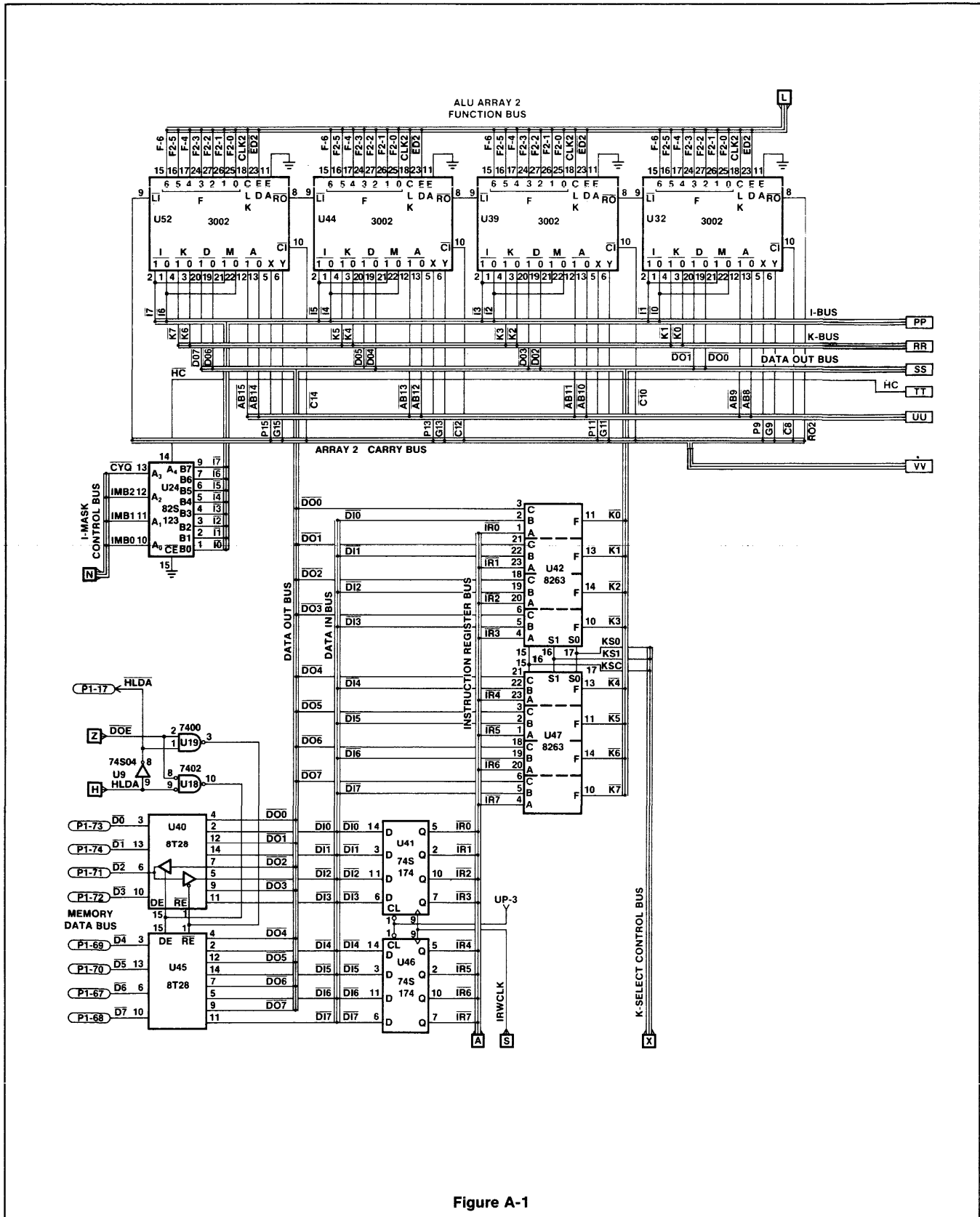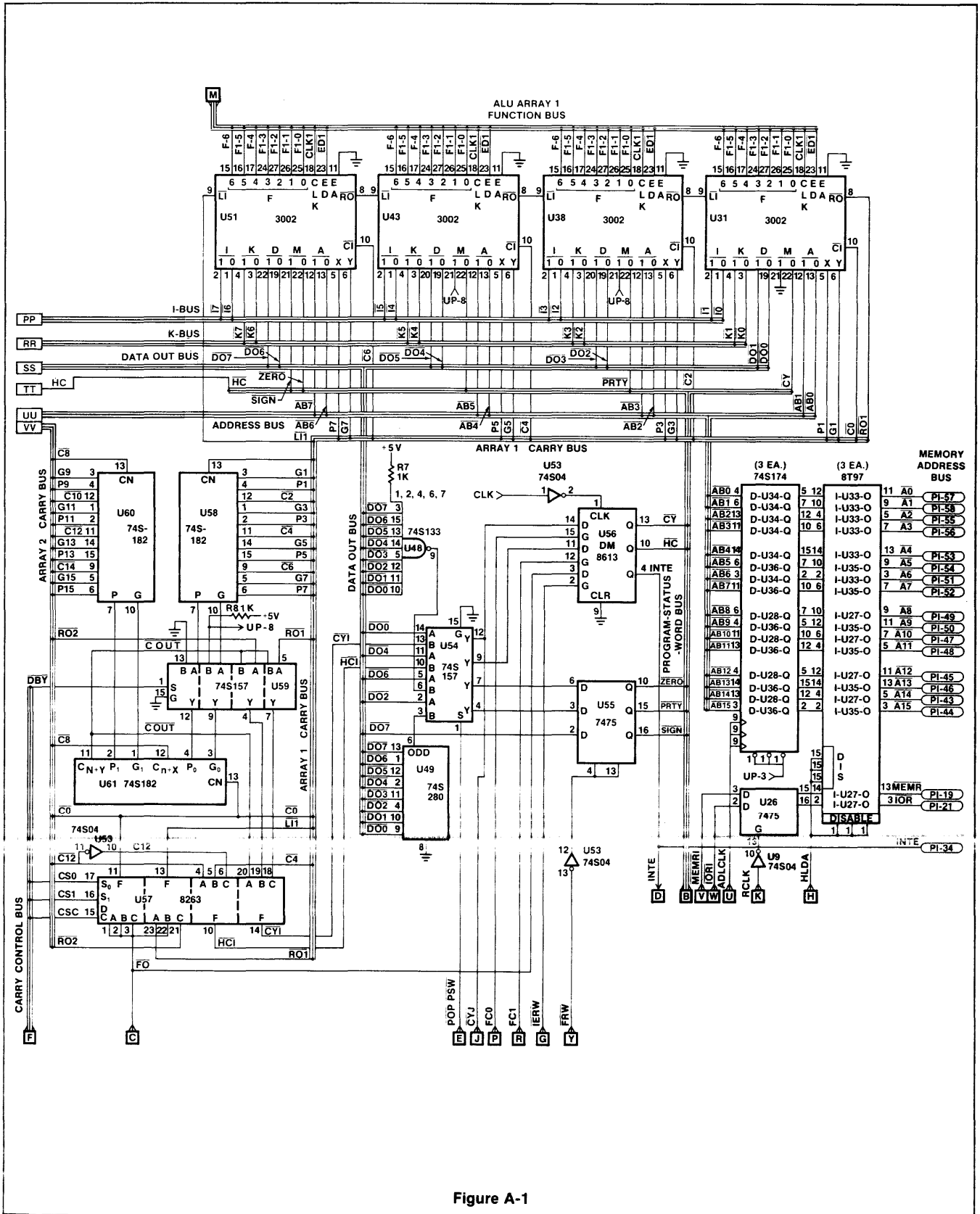
Figure A-1

**Figure A-1**

**Figure A-1**

## PARTS LIST

| QNT | PART NUMBER | DESCRIPTION | ITEM NO. |
|---|---|---|---|
| 5 | ICN-163-S3 | Socket, IC, 16-Pin (Robinson Nugent) ☑ | 51 |
| 7 | ICN-246-S4 | Socket, IC, 24-Pin (Robinson Nugent) ☑ | 50 |
| 2 | CM04ED200J03 | CAP, FXD, MICA, 500V, 5%, 20pF (Sprague) | 49 |
| 1 | | RES, FXD, CMPSN, ¼W, 10%, 200KΩ | 48 |
| 1 | D566S2B15M | CAP, FXD, TANT EL, 15V, 10%, 56µF (Dickson) | 47 |
| 3 | DI0GS2B15M | ▲ ▲ , TANT EL, 15V, 20%, 22µF (Dickson) | 46 |
| 20 | 5021ES50RD104M | | , CER, 50V, $^{+80}_{-20}$ %, 0.1µF, (Emcon) | 45 |
| 1 | CM05CD030D03 | ▼ ▼ , MICA, 500V, ±½pF, 3pF (Sprague) | 44 |
| 1 | CM05CD030D03 | CAP, FXD, MICA, 500V, ±½pF, 3pF (Sprague) | 43 |
| 2 | 1N270 | Diode, Germanium | 42 |
| 1 | Selected | RES, FXD ☒ | 41 |
| 1 | Selected | RES, FXD ☒ | 40 |
| 5 | | RES, FXD, CMPSN, ¼W, 10%, 1000Ω | 39 |
| 1 | CDP-16-02-102K | Resistor Network (DIP) 1KΩ (Dale) | 38 |
| 1 | CSP-10E-01-102K | Resistor Network (SIP) 1KΩ (Dale) | 37 |
| 1 | Spare | Integrated Circuit | 36 |
| 1 | 82S126-U37 | ▲ ▲ | 35 |
| 1 | 82S126-U30 | | | 34 |
| 1 | 82S123-U24 | | | 33 |
| 1 | 82S123-U10 | | | 32 |
| 1 | 82S115-U29 | | | 31 |
| 1 | ▲ -U8 | | | 30 |
| 1 | | -U7 | | | 29 |
| 1 | | -U6 | | | 28 |
| 1 | | -U5 | | | 27 |
| 1 | ▼ -U4 | | | 26 |
| 1 | 82S115-U2 | | | 25 |
| 3 | 8T97 | | | 24 |
| 2 | 8T28 | | | 23 |
| 1 | DM8613 | | | 22 |
| 3 | 8263 | | | 21 |
| 1 | 82S23-U17 | | | 20 |
| 1 | N74S280A | | | 19 |
| 3 | N74S182B | | | 18 |
| 11 | N74S174B | | | 17 |
| 2 | N74S157B | | | 16 |
| 1 | N74S153B | | | 15 |
| 1 | N74S133B | | | 14 |
| 1 | N74123AB | | | 13 |
| 2 | N7475B | | | 12 |
| 1 | N74S10A | | | 11 |
| 1 | N74S08A | | | 10 |
| 3 | N74S04A | | | 9 |
| 1 | N74S02A | | | 8 |
| 1 | N7400A | | | 7 |
| 8 | N3002XL | ▼ ▼ | 6 |
| 1 | N3001I | Integrated Circuit | 5 |
| 8 | ICN-286-S4 | Socket, IC, 28-Pin (Robinson Nugent) | 4 |
| 1 | ICN-406-S4 | Socket, IC, 40-Pin (Robinson Nugent) | 3 |
| REF | | User Manual ☑ | 2 |
| 1 | | Printed Wiring Board ☑ | 1 |

NOTES (See references to notes in Figure A-2)

☒① For internal clock, Jumper #1 to #2. For internal clock, Jumper #2 to #3.
☒② Use sockets as necessary for PROMs and LSI parts: items number 5, 6, 20, 25 through 35.
☒③ Resistor value selected for appropriate timing.

| ITEM 40 (R1) | ITEM 41 (R2) | |
|---|---|---|
| 3.1KΩ | 4.3KΩ | |
| | | |

# PC BOARD—COMPONENT SIDE

**Figure A-2**

○ Indicates item no. from parts list
▢ Indicates note no. from parts list
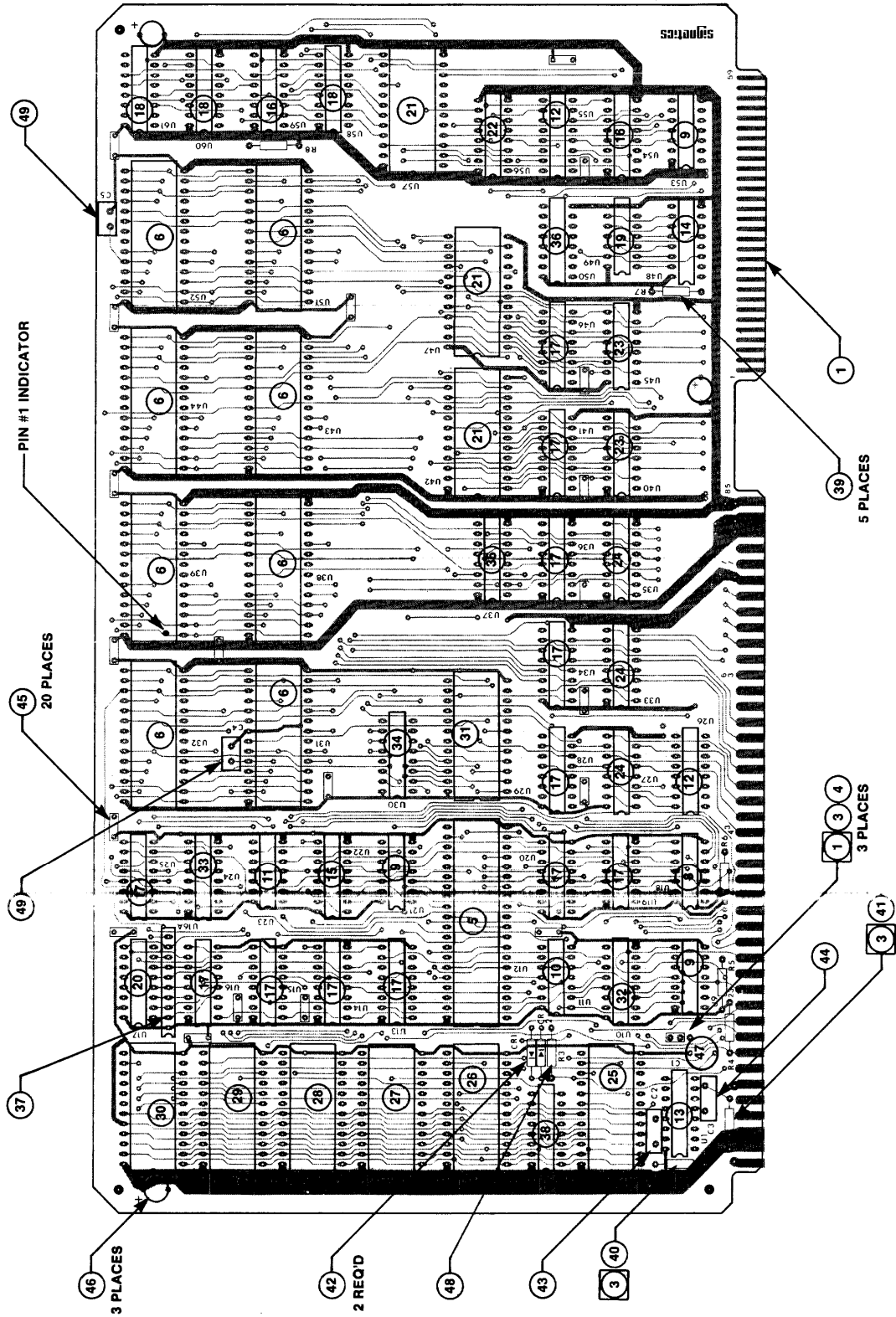
# PC BOARD—WIRING SIDE
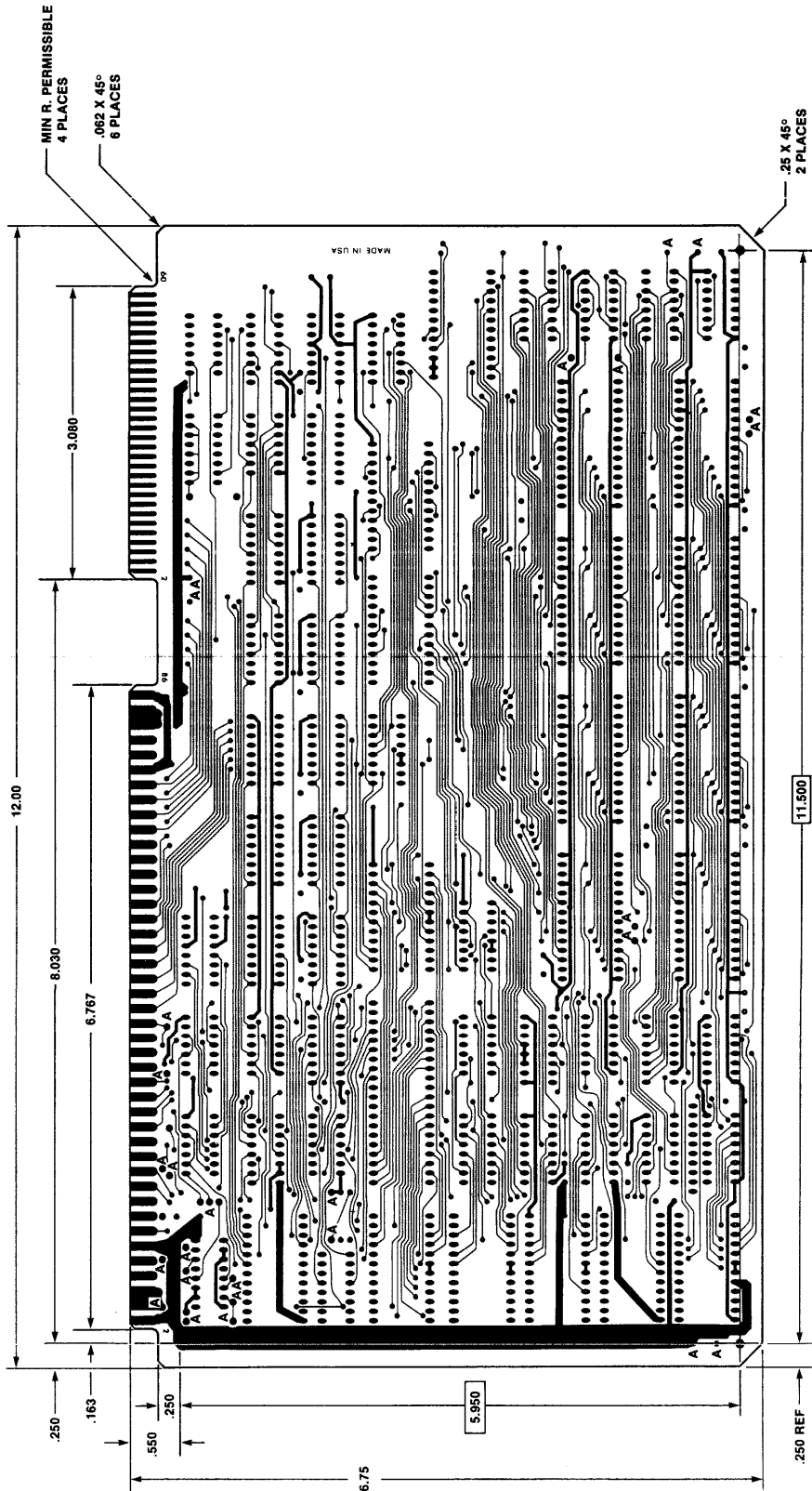


**Figure A-2**

## PC BOARD PIN-OUT AND SIGNAL DESCRIPTIONS

### Mating Edge Connectors

The 8080 Emulator communicates with other system modules via an 86-pin double-sided edge connector (P1). (See Table A-1) This edge connector will accept any of the following mating connectors:

1. CDC VPBO1E43A000A1
2. Microplastics MP-0156-43-BW-4 or
3. ARCO AE 443WP1.

### Signal Description

$\overline{A}_{(15-0)}$ output  **ADDRESS BUS**
The Address Bus provides addressability of up to 65K of memory. $\overline{A}_{(7-0)}$ are used to access I/O PORT. The Address Bus is driven by tri-state bus drivers. ($\overline{A}_0$ = LSB)

$\overline{D}_{(7-0)}$ bidi-rectional  **DATA BUS**
The Data Bus is an 8-bit bidirectional bus used to transmit/receive information to/from memory or an I/O PORT. ($\overline{D}_0$ = LSB)

$\overline{READY}$ input  **READY**
Ready is returned to the CPU by the memory or I/O port to indicate that requested data is valid on the Data Bus. Ready is used to synchronize the 8080 Emulator with slower memory and I/O devices. During a fetch cycle, the CPU idles in a dynamic wait loop until Ready is returned.

HOLD input  **HOLD**
Hold is a request for external control of the 8080 Emulator's Address and Data Buses. When Hold is activated, the CPU finishes the current instruction, fetches the next instruction, and then enters the Hold state. During the Hold state, the 8080 Emulator's Address and Data Buses are placed in the high impedance state and interrupt requests are ignored. Hold is recognized when the CPU is in the Halt state. See Figure A-3.

| | | COMPONENT SIDE | | | CIRCUIT SIDE | |
| --- | --- | --- | --- | --- | --- | --- |
| | PIN | MNEMONIC | DESCRIPTION | PIN | MNEMONIC | DESCRIPTION |
| **POWER SUPPLIES** | 1 | GND | Signal GND | 2 | GND | Signal GND |
| | 3 | VCC | +5VDC | 4 | VCC | +5VDC |
| | 5 | VCC | +5VDC | 6 | VCC | +5VDC |
| | 7 | * | | 8 | * | |
| | 9 | * | | 10 | * | |
| | 11 | GND | Signal GND | 12 | GND | Signal GND |
| **BUS CONTROLS** | 13 | * | | 14 | $\overline{RESET}$ | Initialize |
| | 15 | HOLD | | 16 | * | |
| | 17 | $\overline{HLDA}$ | Hold Ack. | 18 | * | |
| | 19 | $\overline{MEMR}$ | Mem Read Cmd. | 20 | $\overline{MEMW}$ | Mem Write Cmd |
| | 21 | $\overline{IOR}$ | I/O Read Cmd. | 22 | $\overline{IOW}$ | I/O Write Cmd. |
| | 23 | READY | XFER Ack. | 24 | * | |
| | 25 | * | | 26 | | Spare |
| | 27 | $\overline{BUSEN}$ | Bus Enable | 28 | $\overline{INTA}$ | Interrupt Ack. |
| | 29 | $\overline{IST}$ | Interrupt Strobe | 30 | $\overline{HLTA}$ | Halt Ack. |
| | 31 | CLK | Clock | 32 | $\overline{RTRAP}$ | Illegal Opcode Sig. |
| **INTERRUPTS** | 33 | * | | 34 | INTE | Interrupt Enable |
| | 35 | * | | 36 | * | |
| | 37 | * | | 38 | * | |
| | 39 | * | | 40 | * | |
| | 41 | * | | 42 | $\overline{INT}$ | Interrupt Request |
| **ADDRESS** | 43 | $\overline{A14}$ | | 44 | $\overline{A15}$ | |
| | 45 | $\overline{A12}$ | | 46 | $\overline{A13}$ | |
| | 47 | $\overline{A10}$ | | 48 | $\overline{A11}$ | Address Bus |
| | 49 | $\overline{A8}$ | Address Bus | 50 | $\overline{A9}$ | |
| | 51 | $\overline{A6}$ | | 52 | $\overline{A7}$ | |
| | 53 | $\overline{A4}$ | | 54 | $\overline{A5}$ | |
| | 55 | $\overline{A2}$ | | 56 | $\overline{A3}$ | |
| | 57 | $\overline{A0}$ | | 58 | $\overline{A1}$ | |
| **DATA** | 59 | * | | 60 | * | |
| | 61 | * | | 62 | * | |
| | 63 | * | | 64 | * | |
| | 65 | * | | 66 | * | |
| | 67 | $\overline{D6}$ | | 68 | $\overline{D7}$ | |
| | 69 | $\overline{D4}$ | Data Bus | 70 | $\overline{D5}$ | Data Bus |
| | 71 | $\overline{D2}$ | | 72 | $\overline{D3}$ | |
| | 73 | $\overline{D0}$ | | 74 | $\overline{D1}$ | |
| **POWER SUPPLIES** | 75 | GND | Signal GND | 76 | GND | Signal GND |
| | 77 | * | | 78 | * | |
| | 79 | * | | 80 | * | |
| | 81 | VCC | +5VDC | 82 | VCC | +5VDC |
| | 83 | VCC | +5VDC | 84 | VCC | +5VDC |
| | 85 | GND | Signal GND | 86 | GND | Signal GND |

Table A-1  PIN ASSIGNMENTS FOR CONNECTOR P1

*Used by Intel MDS System.

**IOR output** — **INPUT/OUTPUT READ**
IOR designates a CPU request for data from an I/O device. IOR indicates that the low-order eight bits of the Address Bus are valid and that the Data Bus is in an input mode. See Figure A-4.

**IOW output** — **INPUT/OUTPUT WRITE**
IOW signifies that the CPU wishes to write data to an I/O port. IOW indicates that the low-order eight bits of the Address Bus ($A_{(7-0)}$) are valid and that the Data Bus is in an output mode. See Figure A-5.

**INT input** — **INTERRUPT**
INT is a system interrupt request. It is recognized at the end of the instruction cycle when IST is active. INT is ignored if the CPU is in the Hold state or if the Interrupt Enable (INTE) flip-flop is reset. See Figure A-6.

**INTE output** — **INTERRUPT ENABLE**
INTE reflects the current status of the INTE flip-flop. The INTE flip-flop may be set and reset by the E1 and D1 instructions, respectively. The INTE flip-flop is reset by an interrupt request or a system reset. See Figure A-6.

**INTA output** — **INTERRUPT ACKNOWLEDGE**
INTA indicates CPU acknowledgment of an interrupt request. INTA is used to gate a Restart instruction onto the Data Bus. See Figure A-6.

**IST output** — **INTERRUPT STROBE**
IST indicates that the last microcycle of the current instruction is being executed, and that the CPU will recognize interrupt requests (providing the INTE flip-flop is set). See Figure A-6.

**HLTA output** — **HALT ACKNOWLEDGE**
HLTA indicates that the CPU has entered the Halt state. See Figure A-7.

**BUSEN input** — **BUS ENABLE**
When active, both the Address Bus and Data Bus are enabled; when deactivated, both buses are placed in a high-impedance state.

**HLDA output** — **HOLD ACKNOWLEDGE**
HLDA indicates that the 8080 Emulator has entered the Hold state.



**Figure A-3**



**Figure A-4**



**Figure A-5**

**signetics**

## TIMING DIAGRAM—INTERRUPT SEQUENCE

CLK

IST

INT

INTE

INTA

READY

D7-D0

Figure A-6

## TIMING DIAGRAM—HALT OPERATION

| MA 007₁₆ | MA 006₁₆ | MA 057₁₆ | MA 056₁₆ | MA 001₁₆ |

CLK

MEMR

READY

I R WCLK

D7-D0    HLT (OP = 76)

HLTA
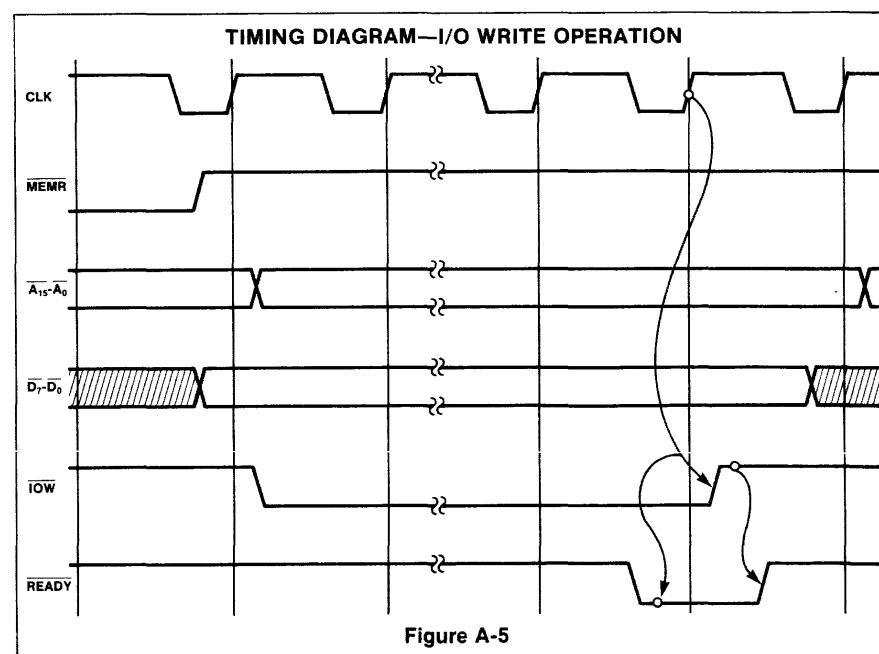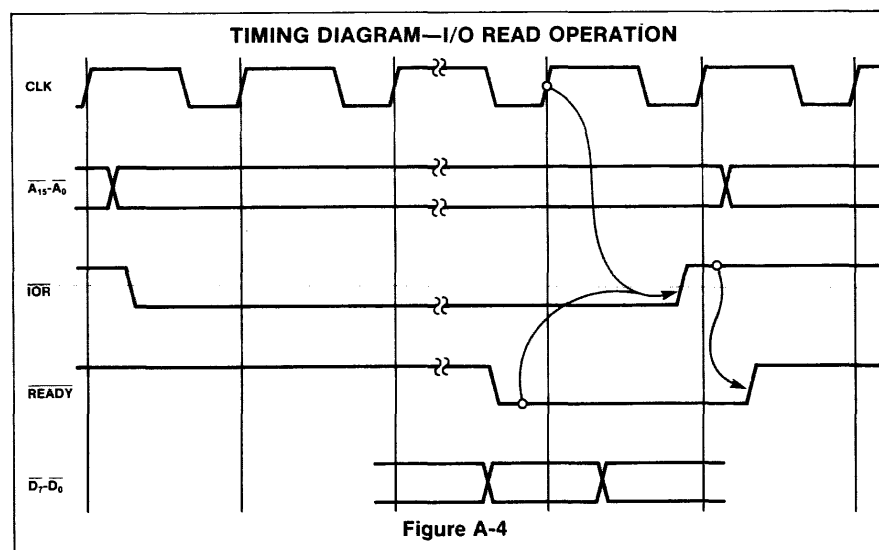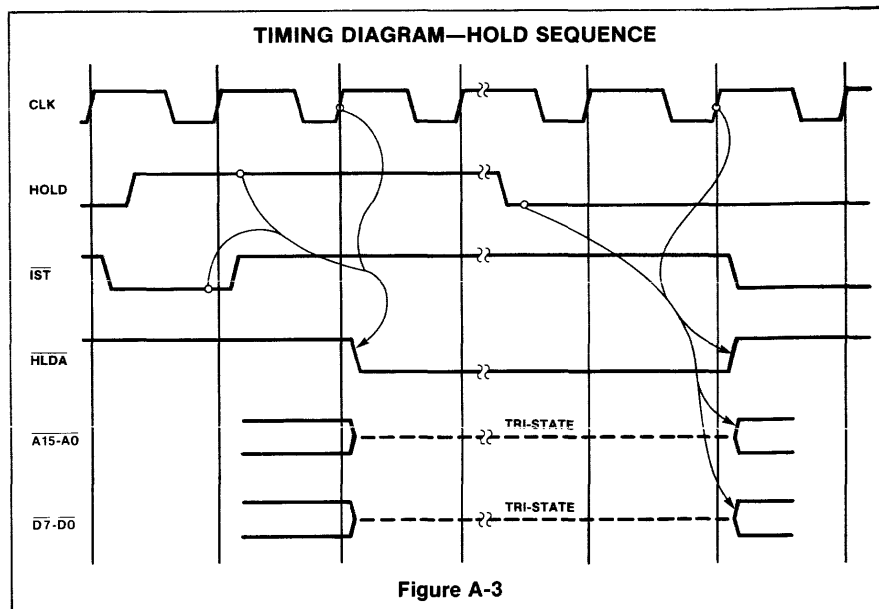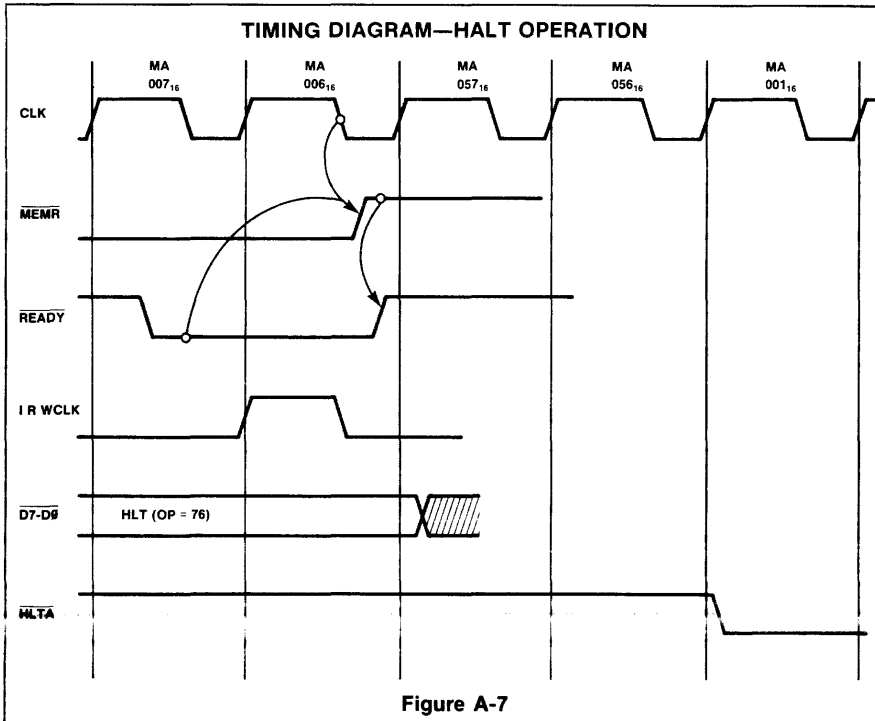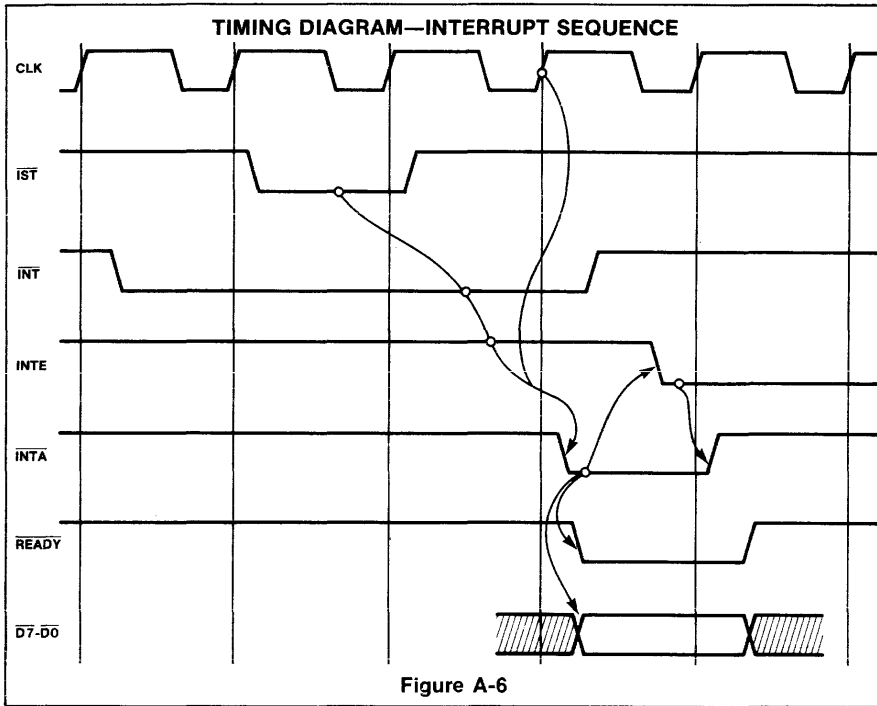
Figure A-7

**RESET input**  **RESET**
RESET clears the program counter and resets both the INTE and $\overline{HLDA}$ flip-flops. Reset must be active for at least one clock period to insure CPU acknowledgment. See Figure A-8.

**$\overline{MEMR}$ output**  **MEMORY READ**
$\overline{MEMR}$ designates a CPU request for memory data. $\overline{MEMR}$ indicates that the Address Bus is valid and that the Data Bus is in an input mode. See Figure A-9.

**$\overline{MEMW}$ output**  **MEMORY WRITE**
$\overline{MEMW}$ signifies that the CPU wishes to write data into memory. $\overline{MEMW}$ indicates that the Address Bus is valid and that the Data Bus is in an output mode. See Figure A-10.

**$\overline{RTRAP}$ output**  **$\overline{RTRAP}$**
$\overline{RTRAP}$ indicates that an illegal op code has been received. When $\overline{RTRAP}$ is detected, the CPU will enter the Halt state.

## 8080 EMULATOR SYSTEM TIMING

The 8080 Emulator is a completely synchronous logic system. All signals input to and output from the Emulator are referenced to the system clock. The system clock is a simple single phase clock. The frequency of the clock determines the execution speeds of the various instructions (providing the CPU doesn't have to wait for slow memory or I/O). As long as the minimum time requirements for the positive and negative portions of the clock are met, there are no restrictions on frequency or duty cycle.

Figures A-3 through A-10 detail the relationship between the system clock and system interface signals for each of the basic machine operations.

## ELECTRICAL SPECIFICATIONS

### Electrical Characteristics

• **Power Supply Requirement**
User provided power supply should have the following ratings:

$V_{CC}$ = 5V ± 5%; 5 Amps.

• **Clock Frequency**

6.6MHz (max)

|       | MIN   | MAX      |
|-------|-------|----------|
| tPWH  | 100ns | $\infty$ |
| tPWL  | 50ns  | $\infty$ |



**TIMING DIAGRAM—RESET OPERATION**

**Figure A-8**



**TIMING DIAGRAM—MEMORY READ OPERATION**
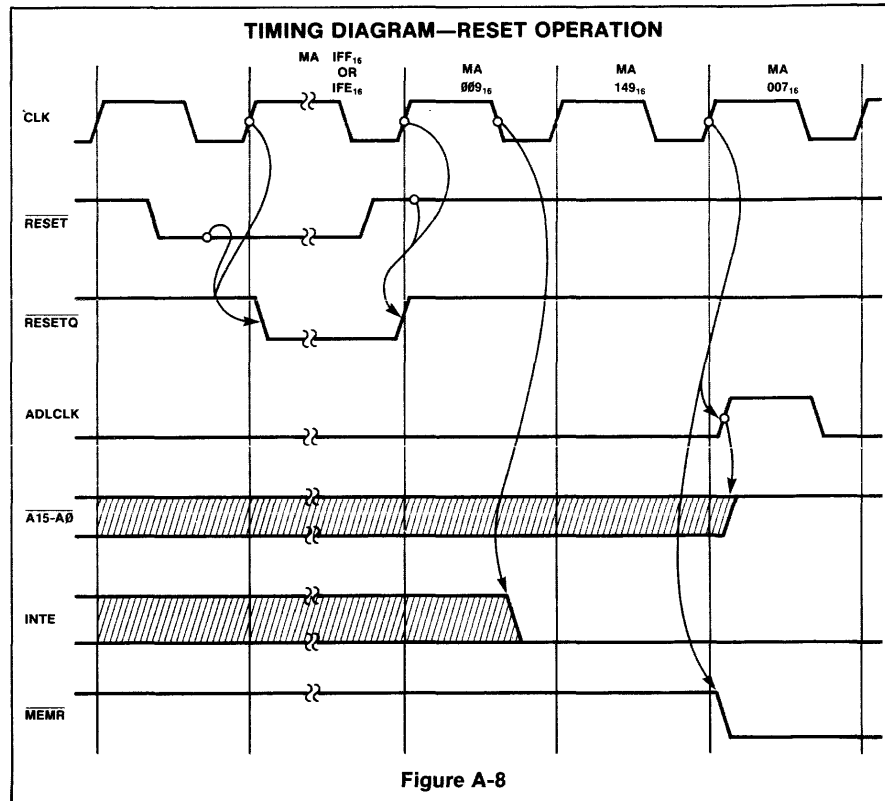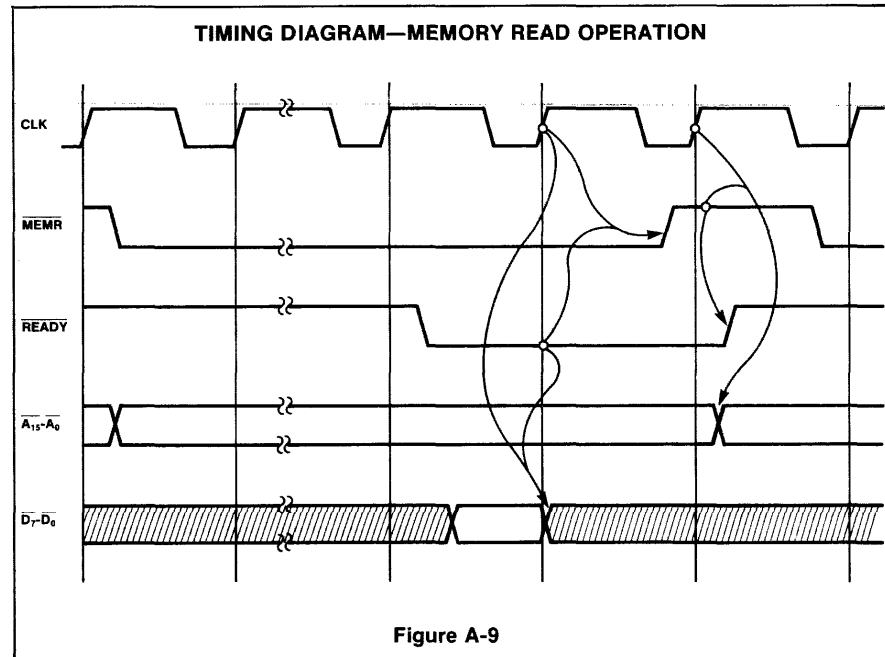
**Figure A-9**

• **Electrical Characteristics for Input and Output Signals**
Table A-2 shows where to obtain electrical characteristics data for output drivers and input receivers.

## TIMING DIAGRAM—MEMORY WRITE OPERATION



**Figure A-10**

**Table A-2  ELECTRICAL CHARACTERISTICS FOR INPUT AND OUTPUT SIGNALS**
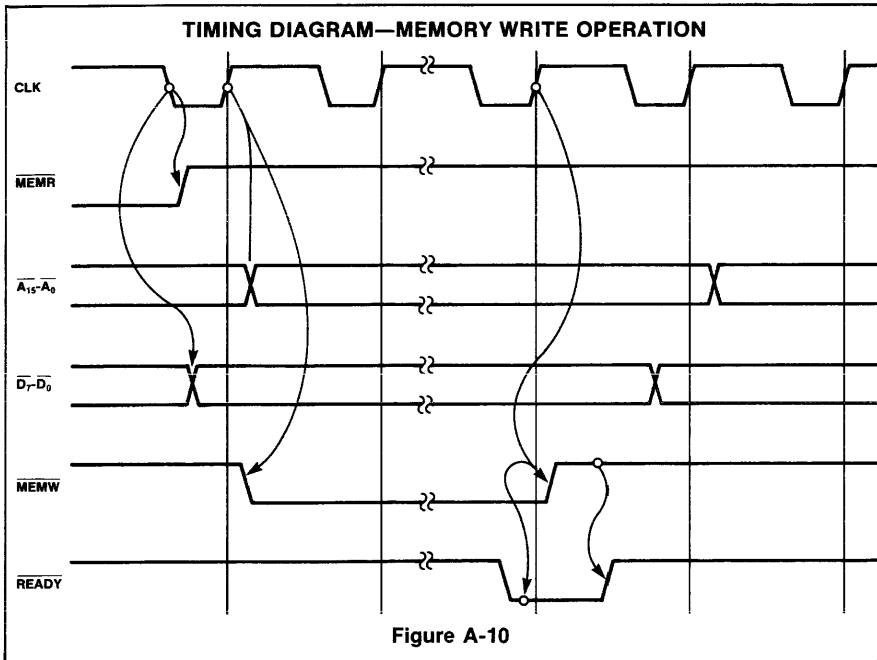
| SIGNAL NAMES | INPUT OR OUTPUT | DEVICE TYPE | REFERENCE (BY VENDOR NAME) |
|---|---|---|---|
| BUSEN | Input | 7400 | Signetics Data Manual |
| XCLK* | Input | 74S04 | Signetics Data Manual |
| XCLK* | Output | 74123 | Signetics Data Manual |
| RTRAP | Output | 82S123 | Signetics Data Manual |
| HLTA | Output | 82S123 | Signetics Data Manual |
| INTA | Output | 82S123 | Signetics Data Manual |
| IOW | Output | 82S123 | Signetics Data Manual |
| MEMW | Output | 82S123 | Signetics Data Manual |
| RESET | Input | 74S174 | Signetics Data Manual |
| INT | Input | 74S174 | Signetics Data Manual |
| HOLD | Input | 74S174 | Signetics Data Manual |
| READY | Input | 74S174 | Signetics Data Manual |
| D7-D0 | Input and Output | 8T28 | Signetics Data Manual |
| A15-A0 | Output | 8T97 | Signetics Data Manual |
| MEMR | Output | 8T97 | Signetics Data Manual |
| IOR | Output | 8T97 | Signetics Data Manual |
| INTE | Output | DM8613 | National Semiconductor Digital Manual |

*NOTE

XCLK is a clock signal which can be provided by the user (input) or generated internally (output) via
jumper options as shown in assembly drawing. (Figure A-2)

The 8080 Emulator makes extensive use of PROM based design techniques. The 8080 op codes are translated into a starting address for microroutines by a PROM (address mapping); fields of the microinstruction address control PROMs (control field expansion); jump decisions based on status conditions are made by a PROM (random logic decode); and the microprogram itself resides in PROM (program storage).

While PROM design techniques greatly simplify a system's schematic diagram, they add another element to its documentation package. This element is the PROM truth table. Appendix B presents the truth tables for each of the 8080 Emulator's PROMs.

| ADDRESS | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F | Lower order 4-bit address $(0_{16}\text{-}F_{16})$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 3 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 5 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 256 preprogrammed 4-bit data patterns represented in hex characters. |
| 6 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | |
| 7 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | |
| 8 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | |
| 9 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | |
| A | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | |
| B | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | |
| C | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | |
| D | 0 | 0 | 0 | 0 | 1 | (1) | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | |
| E | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| F | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | |

Higher order 4-bit address $(0_{16}\text{-}F_{16})$

For example: Address = D5

**Table B-1  CONDITIONAL JUMP CONTROL PROM**

LOCATION  U37  DEVICE TYPE  Signetics 82S126  (256 words X 4-bit PROM)

| ADDRESS | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F | Lower order 4-bit address $(0_{16}\text{-}F_{16})$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | A | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 1 | 0 | 0 | 0 | 0 | A | 0 | A | 0 | 0 | 0 | A | 0 | A | 0 | A | 0 | |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 0 | 0 | 5 | 0 | |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 4 | 0 | 0 | 2 | 8 | 1 | 4 | 0 | 0 | 0 | 0 | 2 | 8 | 1 | 4 | 0 | 0 | |
| 5 | 0 | 0 | 2 | 8 | 1 | 4 | 0 | 0 | 0 | 0 | 2 | 8 | 1 | 4 | 0 | 0 | |
| 6 | 0 | 0 | 2 | 8 | 1 | 4 | 0 | 0 | 0 | 0 | 2 | 8 | 1 | 4 | 0 | 0 | |
| 7 | 0 | 0 | 2 | 8 | 1 | 4 | 0 | 0 | 0 | 0 | 2 | 8 | 1 | 4 | 0 | 0 | 256 preprogrammed 4-bit data patterns represented in hex characters. |
| 8 | 0 | A | 2 | 8 | 1 | 4 | 0 | 0 | A | 0 | 2 | 8 | 1 | 4 | 0 | 0 | |
| 9 | 2 | 2 | 0 | A | 1 | 6 | 0 | 2 | 8 | 8 | A | 0 | 9 | 4 | 8 | 0 | |
| A | 1 | 1 | 2 | 9 | 0 | 5 | 0 | 1 | (4) | 4 | 6 | 8 | 5 | 0 | 4 | 0 | |
| B | 0 | 0 | 2 | 8 | 1 | 4 | 0 | 0 | 0 | 0 | 2 | 8 | 1 | 4 | 0 | 0 | |
| C | 0 | 0 | 0 | 0 | F | 0 | F | 0 | 0 | 0 | 0 | 0 | F | 0 | F | 0 | |
| D | 0 | 2 | 2 | 2 | A | A | A | 0 | 0 | 8 | 8 | 8 | A | A | A | 0 | |
| E | 0 | 1 | 1 | 1 | 5 | 5 | 5 | 0 | 0 | 4 | 4 | 4 | 5 | 5 | 5 | 0 | |
| F | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

Higher order 4-bit address $(0_{16}\text{-}F_{16})$

For example: Address = A8
Data = 4

**Table B-2  OP CODE REGISTER CONTROL PROM**

LOCATION  U30  DEVICE TYPE  Signetics 82S126  (256 words X 4 bits PROM)

| ADDRESS | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F | Lower order 4-bit address ($0_{16}$-$F_{16}$) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 00 | BF | 00 | 99 | 99 | C7 | FF | 00 | 00 | BF | FF | F9 | 99 | C7 | FF | 00 | 32 preprogrammed 8-bit data patterns represented in hex characters. |
| 1 | 00 | BF | 00 | 9F | 99 | C7 | FF | 00 | 00 | (BF) | FF | FF | 99 | C7 | FF | 00 | |

Higher order 1-bit address

For example: Address = 19
Data = BF

**Table B-3   I-BUS MASK PROM**

LOCATION U24   DEVICE TYPE   82S123   (32 words X 8 bits)

| ADDRESS | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | FF | 9F | DF | EF | F7 | FB | FD | FE | FF | FF | FF | FF | FF | FF | FF | FF |
| 1 | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF |

**Table B-4   MEMORY AND I/O CONTROL PROM**

LOCATION   U10   DEVICE TYPE   Signetics 82S123   (32 words X 4 bits)

| ADDRESS | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 00 | 11 | 22 | 33 | 44 | 55 | 66 | 77 | 88 | 99 | AA | BB | CC | DD | EE | FF |
| 1 | 3C | C3 | CB | CF | AC | 3D | D3 | 00 | 00 | 00 | 00 | DC | CD | FE | EF | FF |

**Table B-5   MICROCODE REGISTER CONTROL PROM**

LOCATION U17   DEVICE TYPE Signetics 82S23

|  | PROM U7 | PROM U8 |  | PROM U2 |  | PROM U5 |  | PROM U6 |  | PROM U4 |
|---|---|---|---|---|---|---|---|---|---|---|
|  | 8-7,6-5,4-1 | 8-6, 5-1 | 8, 7, 6, 5, 4, 3-1 | 8, 7-5, 4, 3, 2, 1 | 8, 7, 6, 5, 4, 3-1 | 8, 7-1 |

| FO | FI | FGP | IMB | RGP | DOE | IRW | ADL | FRW | ED1 | KS | IER | SW/PA | NC2 | NC1 | CCR | EXT | RRE | LD2 | SJM | IST | DBY | CS | LD | AC |

Lower order 4-bit address ($0_{16}$ – $F_{16}$)

| ADDRESS | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 000 | 16 | 31 | 26 | 26 | 26 | 26 | F8 | 36 | 26 | 14 | FA | 3A | 00 | 00 | 00 | 00 |
| 001 | 24 | 25 | 0C | 27 | 26 | 26 | 26 | 26 | 26 | 26 | 27 | 26 | 26 | 13 | 12 | 0C |
| 002 | 22 | 22 | 22 | 22 | 22 | 22 | 22 | 17 | 23 | 23 | 23 | 23 | 23 | 23 | 23 | 24 |
| 003 | 7A | 7A | 24 | 25 | FF | 26 | 24 | 24 | 13 | 0F | FF | 0A | 13 | 14 | FF | 3E |
| 004 | 10 | 11 | 31 | 32 | 32 | 27 | FF | 11 | 26 | 0A | 10 | FF | 12 | 10 | 13 | 13 |
| 005 | 0A | 15 | 79 | 0F | 09 | 34 | 21 | 36 | FF | 0F | 15 | 0A | 09 | 27 | 0C | 26 |
| 006 | 28 | 30 | 28 | 32 | 28 | 34 | 28 | 36 | 28 | 38 | 28 | 3A | 28 | 3C | 28 | 3E |
| 007 | 00 | 17 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 08 | 00 | 00 | 26 | 26 | 26 | 26 |
| 008 | 27 | 00 | 00 | 00 | 26 | 26 | 26 | 00 | 27 | 78 | 65 | 3A | 00 | 00 | 00 | 00 |
| 009 | 35 | 14 | 04 | 39 | 66 | 33 | 10 | 36 | 3F | 38 | 37 | 00 | 7B | 12 | 10 | 3E |
| 00A | 0B | 80 | 13 | 32 | 60 | 34 | 26 | 72 | 3F | 38 | 3D | 3A | 7A | 3C | 39 | 03 |
| 00B | 31 | 72 | 6A | 32 | 33 | 34 | 27 | 36 | 3A | 38 | 37 | 3D | 27 | 3C | 3B | 3E |
| 00C | 13 | 13 | 26 | 00 | 26 | 00 | 39 | 12 | 37 | 38 | 65 | 3A | 38 | 3C | 78 | 11 |
| 00D | 26 | 30 | 26 | 32 | 26 | 34 | 26 | 36 | 26 | 38 | 26 | 3A | 3E | 3C | 6F | 3E |
| 00E | 26 | 30 | 26 | 32 | 26 | 34 | 26 | 36 | 26 | 38 | 26 | 3A | 26 | 3C | 26 | 3E |
| 00F | 26 | 30 | 31 | 32 | 27 | 34 | 27 | 36 | 3B | 38 | 7F | 3A | 35 | 3C | 27 | 3E |
| 010 | 33 | 3F | 31 | 32 | 41 | 34 | 3C | 36 | 35 | 38 | 37 | 39 | 3B | 26 | 11 | 3E |
| 011 | 33 | 30 | 13 | 32 | 13 | 00 | 3B | 36 | 34 | 13 | 27 | 3A | 3B | 3C | 3C | 39 |
| 012 | 08 | 30 | 35 | 32 | 37 | 34 | 31 | 36 | 08 | 38 | 39 | 3A | 7A | 6B | 31 | 3E |
| 013 | 32 | 30 | 26 | 12 | 14 | 30 | 00 | 00 | 26 | 26 | 63 | 3A | 7B | 30 | F8 | 3E |
| 014 | 33 | 30 | 13 | 32 | 35 | 36 | 37 | 38 | 3A | 27 | 26 | 00 | 00 | 26 | 00 | 00 |
| 015 | 00 | 27 | 26 | 32 | 00 | 00 | 00 | 00 | 11 | 38 | 3B | 3D | 3E | 3C | 3F | 16 |
| 016 | 21 | 30 | 47 | 34 | 55 | 00 | 00 | 00 | 00 | 15 | 39 | 3A | 3B | 3C | 3D | 3E |
| 017 | 34 | 30 | 35 | 36 | 32 | 5E | 34 | 24 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 018 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 019 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 01A | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 01B | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 01C | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 01D | 00 | 00 | 00 | 00 | 00 | 00 | 26 | (36) | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 01E | 21 | 1E | 00 | 00 | 00 | 00 | 1D | 1E | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 01F | 00 | 2B | 00 | 00 | 00 | 00 | 2B | 1F | 00 | 00 | 00 | 00 | 00 | 00 | 29 | 29 |

512 preprogrammed 8-bit data patterns represented in hex characters.

Higher order 5-bit address ($00_{16}$ – $1F_{16}$)

For example:   Address = $1D7_{16}$
Data = $36_{16}$

**Table B-6   PARTIAL MICROCODE**

LOCATION  U4  DEVICE TYPE  Signetics 82S115 (512 words X 8-bit PROM)

Table B-7   PARTIAL MICROCODE

| ADDRESS | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 000 | FF | EF | EB | EB | EF | EF | D3 | EF | E3 | F3 | BF | DF | 00 | 00 | 00 | 00 |
| 001 | 7B | 7A | FB | EB | EB | EB | EB | EB | EB | EB | E3 | 67 | E3 | F3 | 7F | F3 |
| 002 | FB | FB | FD | FD | FC | FC | FC | FD | FB | FB | FD | FD | FC | FC | FC | FD |
| 003 | 7F | 7F | 7B | 7A | FF | 63 | FB | FB | 7F | FF | 7F | FB | 5F | F2 | F3 | DB |
| 004 | FF | FB | F3 | FF | F3 | E2 | B7 | FF | EF | 77 | F7 | FF | DF | 7F | FF | F8 |
| 005 | FF | FF | F7 | 7F | F3 | FF | EF | FF | BB | 7F | FF | 7F | D7 | E8 | D7 | EF |
| 006 | FF | FB | FF | FB | FF | FD | FF | FD | FF | FC | FF | FC | FF | FC | FD | FF |
| 007 | 00 | FB | 00 | 00 | 00 | 00 | 00 | 00 | 00 | D7 | 00 | 00 | EB | 6B | 6B | EB |
| 008 | E3 | 00 | 00 | 00 | E3 | E3 | E3 | 00 | E3 | FF | F7 | FF | 00 | 00 | 00 | 00 |
| 009 | F7 | FB | F3 | FF | FF | F7 | F7 | FF | F7 | FF | F7 | 00 | F3 | FF | FF | FF |
| 00A | FB | FB | F3 | FB | FF | FB | EF | FB | F3 | 7F | FF | FB | F3 | DF | F3 | DB |
| 00B | F8 | DB | F3 | FB | F3 | FF | F3 | F3 | FF | FF | F7 | F3 | F3 | FB | F7 | FF |
| 00C | FF | FF | EB | 00 | EF | 00 | 7F | FF | F7 | FF | F3 | FF | FF | FF | FF | FF |
| 00D | EB | FB | EB | FB | EB | FB | EB | FB | EB | FB | EB | FB | FB | FB | FF | FB |
| 00E | EB | FB | EB | FB | EB | FD | EB | FC | EB | FD | EB | FC | EB | FC | EB | FD |
| 00F | E3 | FB | F7 | FF | F3 | F3 | F3 | F3 | FF | FB | F3 | FF | FF | FF | F3 | F3 |
| 010 | FF | FF | F7 | FF | FF | FC | F7 | F7 | FB | FB | FF | FF | F7 | 7F | F7 | FF |
| 011 | FF | FF | F3 | FF | F1 | 00 | F7 | FB | FB | FF | F3 | F3 | FF | FF | F7 | F3 |
| 012 | F3 | F7 | F7 | FF | FF | FF | F7 | FF | F3 | F7 | F7 | FF | 7F | F3 | F7 | FF |
| 013 | F3 | FF | E3 | F7 | FB | FF | 00 | 00 | 6F | E3 | F7 | FF | F3 | FF | D3 | FF |
| 014 | FF | FF | F3 | FF | F1 | FB | FF | FB | FF | F3 | EB | 00 | 00 | E2 | 00 | 00 |
| 015 | 00 | E6 | EB | FB | 00 | 00 | 00 | 00 | F1 | FB | FF | FB | FB | F1 | F1 | FB |
| 016 | FF | FF | F1 | FB | FF | 00 | 00 | 00 | 00 | F1 | FB | F1 | FB | F1 | FB | F1 |
| 017 | FB | FB | F3 | FB | FF | FB | FB | FD | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 018 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 019 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 01A | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 01B | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 01C | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 01D | 00 | 00 | 00 | 00 | 00 | 00 | EF | FF | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 01E | EF | FF | 00 | 00 | 00 | 00 | FF | FF | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 01F | 00 | FF | 00 | 00 | 00 | 00 | FF | FF | 00 | 00 | 00 | 00 | 00 | 00 | FB | FB |

For example:  Address = $12D_{16}$
Data = $F3_{16}$

Higher order 5-bit address ($00_{16}$ – $1F_{16}$)

Lower order 4-bit address ($0_{16}$ – $F_{16}$)

512 preprogrammed 8-bit data patterns represented in hex characters.

LOCATION U6   DEVICE TYPE  Signetics 82S115 (512 words X 8-bit PROM)

| | PROM U7 | PROM U8 | | PROM U2 | | | | | | PROM U5 | | | | | PROM U6 | | | | | PROM U4 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

bit fields: 8-7,6-5,4-1 | 8-6, 5-1 | 8, 7, 6, 5, 4, 3-1 | 8, 7-5, 4, 3, 2, 1 | 8, 7, 6, 5, 4, 3-1 | 8, 7-1

signals: FC FI FGP IMB RGP DOE IRW ADL FRW ED1 KS IER | SW/PA NC2 NC1 CCR EXT | RRE LD2 SJM IST DBY CS LD AC

| ADDRESS | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 000 | FE | AF | FE | FE | FE | FE | FF | FE | FE | 0F | 8F | BE | 00 | 00 | 00 | 00 |
| 001 | FB | F7 | FF | FE | FA | FA | FA | F6 | F6 | F6 | FE | FE | FE | FF | FE | FF |
| 002 | FF | FF | FF | FF | FF | FF | FF | FB | FF | FF | FF | FF | FF | FF | FF | FF |
| 003 | FF | FF | FB | F7 | FF | FE | FF | FF | FF | FF | FF | FE | FF | F3 | FF | FE |
| 004 | FF | FE | FF | FE | FE | FE | FF | FF | FF | FE | FF | FF | FF | FF | 7E | 7E |
| 005 | FF | FF | FF | FE | FF | FE | AF | FE | FF | FF | FF | FF | FF | FE | FF | FE |
| 006 | FF | FC | FF | FC | FF | FC | FF | FC | FF | FC | FF | FC | FF | FC | FF | FE |
| 007 | 00 | F7 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | FF | 00 | 00 | FE | F6 | FA | FE |
| 008 | FF | 00 | 00 | 00 | 8F | 8F | 8F | 00 | FF | FE | FF | FE | 00 | 00 | 00 | 00 |
| 009 | 8F | BF | 8F | BF | FF | 8F | 8F | 8F | BF | B4 | 8F | 00 | FE | 8F | 8F | B8 |
| 00A | FB | F7 | FF | FE | 8F | FA | 8F | FE | FF | F4 | FF | FE | FF | FA | FF | FE |
| 00B | F7 | FB | FF | FE | FF | FE | FF | CC | 8F | FC | CF | DF | FF | DE | DF | FC |
| 00C | FE | FE | FE | 00 | FE | 00 | FF | EE | EF | FE | FF | FE | 8F | FE | FE | FF |
| 00D | 8B | FE | 8B | FE | 8B | FE | 87 | FE | 87 | FE | 87 | FE | FF | FE | EF | FE |
| 00E | 8F | FC | 8F | FC | 8F | FC | 8F | FC | 8F | FC | 8F | FC | 8F | FC | 8F | FC |
| 00F | 8F | FE | FF | FE | FF | EC | FF | EE | F7 | FE | EF | F8 | EF | EE | FF | EC |
| 010 | FF | FF | FF | F4 | 87 | EE | 8F | FE | EF | E8 | FF | EF | EF | FA | EF | F8 |
| 011 | F7 | FE | FF | F8 | 8F | 00 | EF | FE | 8B | FF | FF | EC | EF | EE | EE | FF |
| 012 | 8F | EC | FF | F4 | EF | F8 | EF | EE | 8F | EC | EF | FE | FE | 8F | EF | FE |
| 013 | FF | FE | FE | FF | 8B | 8F | 00 | 00 | F6 | FE | FF | FE | FE | FE | FF | FE |
| 014 | BF | B4 | FF | B8 | 8F | 8F | 8F | 8F | 8F | FF | 8F | 00 | 00 | FE | 00 | 00 |
| 015 | 00 | FE | 8F | 8B | 00 | 00 | 00 | 00 | 8F | 8B | F7 | FB | 8B | FC | 8F | 8B |
| 016 | 9F | FE | 87 | 87 | 8F | 00 | 00 | 00 | 00 | 8F | 8B | 8F | 8B | 8F | 8B | 8F |
| 017 | 8B | FA | 8F | 8F | 8F | 83 | 8B | FF | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 018 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 019 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 01A | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 01B | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 01C | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 01D | 00 | 00 | 00 | 00 | 00 | 00 | 8F | 8D | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 01E | AF | AD | 00 | 00 | 00 | 00 | 8D | FE | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 01F | 00 | BF | 00 | 00 | 00 | 00 | 3F | FE | 00 | 00 | 00 | 00 | 00 | 00 | 8F | 8F |

Lower order 4-bit address ($0_{16}$ – $F_{16}$)

For example: Address = $07E_{16}$, Data = $FA_{16}$

Higher order 5-bit address ($00_{16}$ – $1F_{16}$)

512 preprogrammed 8-bit data patterns represented in hex characters.

**Table B-8  PARTIAL MICROCODE**

LOCATION U5  DEVICE TYPE  Signetics 82S115 (512 words X 8-bit PROM)

| | PROM U7 | PROM U8 | | PROM U2 | | PROM U5 | | PROM U6 | | PROM U4 | |
|---|---|---|---|---|---|---|---|---|---|---|---|

```
        PROM U7    PROM U8    /   PROM U2              PROM U5            PROM U6      PROM U4

       V-----V  V----V V--------------V V--------------V V--------------V V V-----V
       8-7.6-5.4-1  8-6. 5-1  8.  7.  6.  5.  4.  3-1 8.  7-5.  4.  3.  2.  1 8.  7.  6.  5.  4. 3-1 8.     7-1


       FO  FI  FGP IMB RGP  DOE IRW ADL FRW ED1 KS   IER SW  NC2 NC1 CCR EXT RRE LD2 SJM IST DBY CS  LD  AC
                                                         PA
```

| ADDRESS | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 000 | FC | FC | A0 | A8 | AC | A4 | DC | BC | AC | FC | FC | BC | 00 | 00 | 00 | 00 |
| 001 | F8 | F8 | F8 | B8 | B8 | B8 | B8 | B8 | B8 | B8 | B8 | B8 | B8 | F8 | BC | F8 |
| 002 | F9 | F9 | FD | FD | F9 | FD | F9 | FC | F1 | F1 | F5 | F5 | F1 | F5 | F1 | F5 |
| 003 | FC | FC | FC | FC | FC | BC | FC | F8 | FC | FC | FC | FA | FC | FC | DC | FA |
| 004 | FC | BC | DC | FC | FC | BC | F8 | FC | BC | FC | F8 | FC | FC | FC | BC | BC |
| 005 | FC | FC | F8 | BC | DC | BC | FC | FC | FC | FC | FC | FC | F8 | BC | F8 | BC |
| 006 | DC | FA | DC | FA | DC | FE | DC | FE | DC | FA | DC | FE | DC | FA | DF | BC |
| 007 | 00 | F8 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | F8 | 00 | 00 | B1 | B8 | B8 | B9 |
| 008 | DC | 00 | 00 | 00 | F8 | F8 | F8 | 00 | DC | BC | D8 | BC | 00 | 00 | 00 | 00 |
| 009 | F8 | FC | FC | FC | FC | F8 | F8 | FC | F8 | BB | F8 | 00 | BC | FC | FC | BB |
| 00A | F8 | F8 | DC | FA | FC | FA | FC | EA | DC | FA | DC | FA | DC | FA | FC | FA |
| 008 | F8 | F4 | DC | FA | DC | BC | DC | 7C | FC | FA | 58 | FC | DC | FA | D8 | FA |
| 00C | FC | FC | BC | 00 | BC | 00 | FC | 7C | 58 | BC | DC | BC | FC | BC | BC | FC |
| 00D | F8 | FA | F8 | FA | F8 | FA | F8 | FA | F8 | FA | F8 | FA | 0C | FA | 6C | FA |
| 00E | E8 | FA | E8 | FA | E8 | FE | E8 | FA | E8 | FE | E8 | FE | E8 | FA | EC | FE |
| 00F | FC | FA | D8 | BC | DC | 7C | DC | 7C | DB | BC | 54 | FA | 5C | 74 | DC | 74 |
| 010 | FC | FC | D8 | FA | F9 | 74 | F8 | F8 | 54 | 7B | FC | 7C | 58 | B1 | 58 | FA |
| 011 | DB | BC | DC | BB | FC | 00 | 50 | BB | F9 | FC | DC | 74 | 54 | 7C | 7C | F8 |
| 012 | FC | 70 | D8 | FA | 7C | FA | 50 | 7C | FC | 78 | 58 | BC | BC | F8 | 50 | BC |
| 013 | DC | FC | BC | FC | F9 | FC | 00 | 00 | B9 | B8 | DC | BC | BC | FC | DC | BC |
| 014 | FC | BB | DC | BB | FC | F8 | FC | F8 | FC | DC | F1 | 00 | 00 | BC | 00 | 00 |
| 015 | 00 | BC | FD | F8 | 00 | 00 | 00 | 00 | FC | F9 | FC | FC | F9 | BC | FC | F9 |
| 016 | FC | FC | FC | F8 | FC | 00 | 00 | 00 | 00 | FC | F9 | FC | F9 | FC | F9 | FC |
| 017 | FC | BC | F8 | FC | FC | F8 | F8 | F1 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 018 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 019 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 01A | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 01B | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 01C | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 01D | 00 | 00 | 00 | 00 | 00 | 00 | FC | FC | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 01E | FC | FC | 00 | 00 | 00 | 00 | FC | BC | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 01F | 00 | FC | 00 | 00 | 00 | 00 | FC | FC | 00 | 00 | 00 | 00 | 00 | 00 | FC | FC |

Lower order 4-bit address ($0_{16}$–$F_{16}$)

For example:
Address = $OCE_{16}$
Data = $BC_{16}$

Higher order 5-bit address ($00_{16}$–$1F_{16}$)

512 preprogrammed 8-bit data patterns represented in hex characters.

**Table B-9  PARTIAL MICROCODE**

LOCATION  U2   DEVICE TYPE   Signetics 82S115 (512 words X 8-bit PROM)

| | PROM U7 | PROM U8 | PROM U2 | PROM U5 | PROM U6 | PROM U4 |
|---|---|---|---|---|---|---|
| | V-----V | V----V | V----------------V | V--------------V | V--------------V | V-----V |
| | 8-7,6-5,4-1 | 8-6, 5-1 | 8, 7, 6, 5, 4, 3-1 | 8, 7-5, 4, 3, 2, 1 | 8, 7, 6, 5, 4, 3-1 | 8, 7-1 |

FO FI FGP  IMB RGP  DOE IRW ADL FRW ED1 KS  IER  SW/PA  NC2 NC1 CCR EXT RRE LD2 SJM IST DBY CS LD AC

Lower order 4-bit address ($0_{16}$-$F_{16}$)

| ADDRESS | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 000 | FF | FF | FB | FC | FF | FF | E4 | FF | E4 | E4 | FF | FF | 00 | 00 | 00 | 00 |
| 001 | E3 | E3 | EC | EC | E0 | E1 | E2 | E0 | E1 | E2 | E2 | E3 | E3 | E4 | E3 | E9 |
| 002 | FE | FE | FE | FE | FE | FE | FF | FF | FD | FD | FD | FD | FD | FD | FD | FF |
| 003 | F0 | F1 | E3 | E3 | FF | E3 | EC | EC | E3 | EC | E3 | FF | E3 | EE | E4 | FF |
| 004 | E4 | E4 | E3 | FF | E3 | EE | E3 | E2 | FF | E3 | E3 | EC | E2 | E3 | FF | FF |
| 005 | F2 | FF | E4 | E3 | E4 | E2 | FF | FF | E9 | E3 | E0 | E3 | E4 | FF | E4 | EE |
| 006 | EC | EE | EC | EE | EC | EE | EC | EE | EC | EE | EC | EE | EC | EE | FF | EC |
| 007 | 00 | 2E | 00 | 00 | 00 | 00 | 00 | 00 | 00 | E4 | 00 | 00 | EE | E3 | E3 | EE |
| 008 | E4 | 00 | 00 | 00 | E0 | E1 | E2 | 00 | E4 | E2 | E4 | FF | 00 | 00 | 00 | 00 |
| 009 | FF | E4 | E3 | E4 | EC | E4 | FF | E3 | E3 | E4 | E3 | 00 | E3 | E2 | E3 | E4 |
| 00A | 9F | 6F | E4 | EE | FF | FF | FF | EE | E4 | E3 | ED | FF | ED | CB | ED | FF |
| 008 | EB | EC | E4 | FF | E3 | FF | E4 | E4 | EC | E9 | E4 | E4 | E4 | EE | E4 | E9 |
| 00C | E4 | E4 | EC | 00 | FF | 00 | E3 | E3 | E3 | E3 | E4 | FF | E3 | F2 | E2 | E1 |
| 00D | E0 | FF | E1 | FF | E2 | FF | E0 | FF | E1 | FF | E2 | FF | ED | EF | FF | FF |
| 00E | EC | FF | EC | FF | EC | FF | EC | FF | EC | FF | EC | FF | EC | FF | E0 | FF |
| 00F | E4 | EE | E4 | FF | E4 | E4 | E4 | E4 | E9 | E9 | E9 | E9 | FF | E9 | E4 | E4 |
| 010 | E4 | E3 | E3 | E4 | E4 | E4 | E4 | FF | E4 | BF | E3 | E3 | E3 | E3 | E3 | E4 |
| 011 | E4 | FF | E4 | E4 | EE | 00 | E4 | FF | 4E | E9 | E4 | E4 | FF | FF | E3 | E2 |
| 012 | E4 | E4 | E3 | E2 | E3 | E2 | E4 | FF | E4 | E4 | E4 | FF | E3 | E4 | E4 | FF |
| 013 | E4 | FF | E4 | E2 | 4E | E4 | 00 | 00 | E3 | E1 | E3 | FF | E3 | E4 | E4 | FF |
| 014 | FF | E4 | E4 | E4 | EE | E9 | EC | E0 | E9 | E4 | EE | 00 | 00 | EE | 00 | 00 |
| 015 | 00 | FF | EE | E0 | 00 | 00 | 00 | 00 | EE | 4E | EC | EC | 4E | EE | EE | 4E |
| 016 | FF | FF | EE | E0 | EC | 00 | 00 | 00 | 00 | EE | 4E | EE | 4E | EE | (4E) | EE |
| 017 | EC | E0 | E0 | EC | E0 | EC | E0 | FF | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 018 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 019 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 01A | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 01B | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 01C | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 01D | 00 | 00 | 00 | 00 | 00 | 00 | FF | FF | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 01E | FF | FF | 00 | 00 | 00 | 00 | FF | FF | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 01F | 00 | FF | 00 | 00 | 00 | 00 | FF | FF | 00 | 00 | 00 | 00 | 00 | 00 | E4 | E4 |

For example:
Address = $16E_{16}$
Data = $4E_{16}$

Higher order 5-bit address ($00_{16}$-$1F_{16}$)

512 preprogrammed 8-bit data patterns represented in hex characters.

**Table B-10   PARTIAL MICROCODE**

LOCATION  U8  DEVICE TYPE  Signetics 82S115  (512 words X 8-bit PROM)

| | PROM U7 | PROM U8 | PROM U2 | PROM U5 | PROM U6 | PROM U4 |
|---|---|---|---|---|---|---|
| | 8-7,6-5,4-1 | 8-6, 5-1 8. 7. 6. 5. 4. 3-1 | 8. 7-5. 4. 3. 2. 1 | 8. 7. 6. 5. 4. 3-1 | 8. 7-1 |

| | FO | FI | FGP | IMB | RGP | DOE | IRW | ADL | FRW | ED1 | KS | IER | SW PA | NC2 | NC1 | CCR | EXT | RRE | LD2 | SJM | IST | DBY | CS | LD | AC |

| ADDRESS | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 000 | FE | FE | F6 | F6 | FE | FE | F1 | FE | F1 | F1 | FE | FE | 00 | 00 | 00 | 00 |
| 001 | 10 | 10 | 20 | 60 | F6 | F6 | F6 | F6 | F6 | F6 | 20 | F0 | F6 | F6 | F1 | F6 |
| 002 | 07 | 47 | 07 | 47 | 08 | 0F | 0E | 3F | 07 | 47 | 07 | 47 | 08 | 0F | 0E | 07 |
| 003 | F0 | F0 | D0 | D0 | F6 | F0 | D0 | 10 | F4 | F0 | F0 | F6 | F0 | E0 | F1 | F6 |
| 004 | F0 | F6 | F1 | FE | F1 | 60 | F1 | F1 | FE | F6 | F1 | F0 | F1 | F2 | 3E | FE |
| 005 | F0 | 76 | F1 | F1 | F1 | F1 | FE | FE | F6 | F0 | F0 | F6 | F1 | EE | F1 | FF |
| 006 | F0 | 07 | F0 | 47 | F0 | 07 | F0 | 47 | F0 | 08 | F0 | 0F | F0 | 0E | 07 | F0 |
| 007 | 00 | F6 | 00 | 00 | 00 | 00 | 00 | 00 | F1 | 00 | 00 | F6 | F6 | F6 | F6 | F6 |
| 008 | F1 | 00 | 00 | 00 | F6 | F6 | F6 | 00 | F1 | F1 | F1 | FE | 00 | 00 | 00 | 00 |
| 009 | F6 | F6 | F1 | F6 | F0 | F0 | F6 | F1 | F1 | F9 | F1 | 00 | F1 | F0 | F1 | F9 |
| 00A | 07 | F6 | F1 | F6 | FE | F6 | FE | C6 | F1 | F9 | F1 | F6 | F1 | F1 | F1 | F6 |
| 00B | 3E | 00 | F1 | F6 | F1 | FE | F1 | F1 | F0 | F1 | F1 | F1 | F1 | F6 | F1 | F1 |
| 00C | F1 | F1 | 70 | 00 | FE | 00 | F0 | F1 | F1 | F1 | F1 | FE | F1 | F0 | F1 | F0 |
| 00D | F6 | F6 | F6 | F6 | F6 | F6 | F6 | F6 | F6 | F6 | F6 | F6 | D0 | F6 | FE | 16 |
| 00E | F6 | 07 | F6 | 47 | F6 | 07 | F6 | 09 | F6 | 47 | F6 | 0F | F6 | 0E | FE | 07 |
| 00F | F1 | F6 | F1 | FE | F1 | F1 | F1 | F1 | F1 | F6 | F1 | F1 | FE | F1 | F1 | F1 |
| 010 | F6 | F1 | F1 | F9 | F1 | F1 | F0 | F6 | F6 | F6 | F1 | F1 | F1 | F9 | F1 | F9 |
| 011 | F1 | FE | F1 | F1 | 50 | 00 | F1 | F6 | 27 | F0 | F1 | F1 | FE | FE | F1 | F6 |
| 012 | F1 | F1 | F1 | F9 | F1 | F9 | F1 | FE | F1 | F1 | F1 | FE | F0 | F6 | F1 | FE |
| 013 | F1 | FE | F1 | F6 | 27 | F1 | 00 | 00 | F9 | F6 | F1 | FE | F1 | F1 | F1 | FE |
| 014 | FE | F1 | F1 | F1 | 50 | F6 | F0 | F6 | F0 | F1 | F6 | 00 | 00 | 70 | 00 | 00 |
| 015 | 00 | E0 | F6 | F7 | 00 | 00 | 00 | 00 | 50 | 27 | F0 | F6 | 27 | 20 | 50 | 27 |
| 016 | FE | FE | 10 | 70 | F0 | 00 | 00 | 00 | 00 | 50 | 27 | 50 | 27 | 50 | 27 | 50 |
| 017 | 0F | F6 | 70 | 30 | F0 | E0 | F7 | 07 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 018 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 019 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 01A | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 01B | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 01C | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 01D | 00 | 00 | 00 | 00 | 00 | 00 | FE | FE | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 01E | FE | FE | 00 | 00 | 00 | 00 | FE | FE | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 01F | 00 | FE | 00 | 00 | 00 | 00 | FE | FE | 00 | 00 | 00 | 00 | 00 | 00 | F6 | F6 |

Lower order 4-bit address ($0_{16}$-$F_{16}$)

For example:
Address = $00B_{16}$
Data = $FE_{16}$

Higher order 5-bit address ($00_{16}$-$1F_{16}$)

512 preprogrammed 8-bit data patterns represented in hex characters.

**Table B-11  PARTIAL MICROCODE**

LOCATION  U7  DEVICE TYPE  Signetics 82S115  (512 words X 8-bit PROM)

**signetics**

| ADDRESS | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 000 | E5 | 00 | 00 | 00 | 00 | 00 | E3 | 00 | E5 | 00 | B2 | 00 | 00 | 00 | 3B | 00 |
| 001 | E5 | 00 | 00 | 00 | E0 | 00 | E2 | 00 | E5 | 00 | B9 | 00 | C4 | 00 | 1B | 00 |
| 002 | E5 | 00 | 00 | 00 | C0 | 00 | 00 | 00 | E5 | 00 | B9 | 00 | C6 | 00 | 2B | 00 |
| 003 | E5 | 00 | EF | 00 | 00 | 00 | ED | 00 | E5 | 00 | B9 | 00 | EE | 00 | 3B | 00 |
| 004 | D8 | 0A | D0 | D8 | D0 | D8 | D0 | D8 | D9 | 2A | D1 | D9 | D1 | D9 | D1 | D9 |
| 005 | DA | 4A | D2 | DA | D2 | DA | D2 | DA | DB | 0A | D3 | DB | D3 | DB | D3 | DB |
| 006 | DC | 6A | D4 | DC | D4 | DC | D4 | DC | DD | 2A | D5 | DD | D5 | DD | D5 | DD |
| 007 | DE | 4A | D6 | DE | D6 | DE | D6 | DE | DF | 6A | D7 | DF | D7 | DF | D7 | DF |
| 008 | 00 | 00 | 83 | 80 | 83 | 80 | 83 | 80 | E1 | 00 | F1 | F5 | F1 | F5 | F1 | F5 |
| 009 | E6 | 45 | 00 | 00 | E6 | 00 | E6 | 00 | E9 | 25 | 00 | 00 | E9 | 00 | E9 | |
| 00A | E7 | 65 | E7 | 00 | 00 | 00 | E7 | 00 | EA | 45 | 00 | EA | 00 | 00 | 00 | EA |
| 00B | E8 | 05 | E8 | 00 | E8 | 00 | 00 | 00 | EB | 65 | 00 | EB | 00 | EB | 00 | 00 |
| 00C | 00 | 80 | C8 | 00 | E4 | FC | E5 | 00 | 00 | 00 | 05 | 25 | 00 | 00 | 81 | 00 |
| 00D | 00 | 82 | EE | 00 | E4 | 81 | E5 | 00 | D7 | 81 | EF | 00 | 00 | 00 | 81 | 00 |
| 00E | 00 | 82 | EE | 00 | E4 | 00 | E5 | 00 | EC | 81 | EF | 00 | 00 | E1 | 81 | 00 |
| 00F | 00 | 82 | EE | 00 | E4 | 00 | E5 | 00 | ED | 81 | EF | 00 | 00 | E1 | 81 | 00 |
| 010 | B5 | 90 | 8E | 3F | B1 | 3E | C5 | 3B | B5 | 92 | B9 | 3F | B0 | 3E | A3 | 3B |
| 011 | B5 | 94 | A5 | 3F | C5 | 3E | C5 | 3B | B5 | 96 | B9 | 3F | C3 | 3E | A3 | 3B |
| 012 | B5 | 98 | FF | 3F | A7 | 3E | FF | 3B | B5 | 9A | B9 | 3F | A7 | 3E | A3 | 3B |
| 013 | B5 | 9C | BF | 3F | FF | 3E | BB | 3B | B5 | 9E | B9 | 3F | BE | 3E | A3 | 3B |
| 014 | B4 | AD | CE | CF | CE | CF | CE | CF | B4 | AD | CE | CF | CE | CF | CE | CF |
| 015 | B4 | AD | CE | CF | CE | CF | CE | CF | B4 | AD | CE | CF | CE | CF | CE | CF |
| 016 | B4 | AD | CE | CF | CE | CF | CE | CF | B4 | AD | CE | CF | CE | CF | CE | CF |
| 017 | B4 | AD | CE | CF | CE | CF | CE | CF | B4 | AD | CE | CF | CE | CF | CE | CF |
| 018 | B7 | AC | C5 | C5 | C5 | C5 | C5 | C5 | B4 | A8 | B3 | B3 | B3 | B3 | B3 | B3 |
| 019 | B4 | A1 | B7 | C7 | C5 | C7 | C5 | C7 | B4 | A1 | B2 | B7 | B2 | C5 | B2 | C5 |
| 01A | B4 | A1 | C5 | C7 | B7 | C7 | C5 | C7 | B4 | A1 | B2 | C5 | B2 | B7 | B2 | C5 |
| 01B | B4 | A1 | C5 | C7 | C5 | C7 | B7 | C7 | B4 | A1 | B2 | C5 | B2 | C5 | B2 | B7 |
| 01C | AE | C0 | CB | C9 | CB | C4 | C5 | FF | A2 | B8 | 86 | 86 | CA | C6 | B6 | FF |
| 01D | A0 | C0 | CB | CC | CB | A4 | C5 | FF | AF | C0 | CB | CD | CA | A6 | B6 | FF |
| 01E | BA | C0 | CB | CC | CB | AC | C5 | FF | B4 | C0 | CB | CD | CA | B4 | B6 | FF |
| 01F | C2 | C0 | CB | CC | CB | AC | C5 | FF | B4 | C0 | CB | CD | CA | B4 | B6 | B7 |

Lower order 4-bit address ($0_{16}$-$F_{16}$)

For example:
Address = $10F_{16}$
Data = $3B_{16}$

Higher order 5-bit address ($00_{16}$-$1F_{16}$)

512 preprogrammed 8-bit data patterns represented in hex characters.

**Table B-12   INSTRUCTION DECODE PROM (PRIMARY AND SECONDARY DECODES)**

LOCATION U29  DEVICE TYPE  Signetics 82S115  (512 words X 8-bit PROM)

The 8080 instruction set includes five different types of instructions:

- Data Transfer Group—move data between registers or between memory and registers
- Arithmetic Group—add, subtract, increment or decrement data in registers or in memory
- Logical Group—AND, OR, EXCLUSIVE-OR, compare, rotate or complement data in registers or in memory
- Branch Group—conditional and unconditional jump instructions, subroutine call instructions and return instructions
- Stack, I/O and Machine Control Group—includes I/O instructions, as well as instructions for maintaining the stack and internal control flags.

## Instruction and Data Formats

Memory for the 8080 is organized into 8-bit quantities, called Bytes. Each byte has a unique 16-bit binary address corresponding to its sequential position in memory. The 8080 can directly address up to 65,536 bytes of memory, which may consist of both read-only memory (ROM) elements and random-access memory (RAM) elements (read/write memory).

Data in the 8080 is stored in the form of 8-bit binary integers:

**DATA WORD**

| $D_7$ | $D_6$ | $D_5$ | $D_4$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ |
|---|---|---|---|---|---|---|---|

**MSB**                                      **LSB**
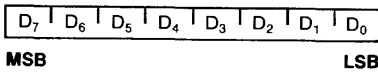
When a register or data word contains a binary number, it is necessary to establish the order in which the bits of the number are written. In the Intel 8080, BIT 0 is referred to as the Least Significant Bit (LSB), and BIT 7 (of an 8 bit number) is referred to as the Most Significant Bit (MSB).

The 8080 program instructions may be one, two or three bytes in length. Multiple byte instructions must be stored in successive memory locations; the address of the first byte is always used as the address of the instructions. The exact instruction format will depend on the particular operation to be executed.

### Single Byte Instructions

| $D_7$ | | | | | | | $D_0$ | Op Code |

### Two-Byte Instructions

Byte One | $D_7$ | | | | | | | $D_0$ | Op Code

Byte Two | $D_7$ | | | | | | | $D_0$ | Data or Address

### Three-Byte Instructions

Byte One | $D_7$ | | | | | | | $D_0$ | Op Code

Byte Two | $D_7$ | | | | | | | $D_0$ | Data
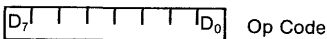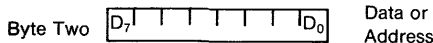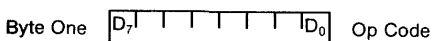
Byte Three | $D_7$ | | | | | | | $D_0$ | or Address
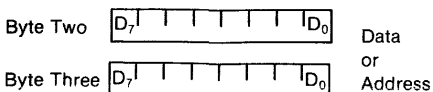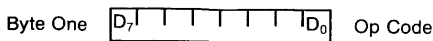
## Addressing Modes

Often the data that is to be operated on is stored in memory. When multi-byte numeric data is used, the data, like instructions, is stored in successive memory locations, with the least significant byte first, followed by increasingly significant bytes. The 8080 has four different modes for addressing data stored in memory or in registers:

- Direct — Bytes 2 and 3 of the instruction contain the exact memory address of the data item (the low-order bits of the address are in byte 2, the high-order bits in byte 3).
- Register — The instruction specifies the register or register-pair in which the data is located.
- Register Indirect — The instruction specifies a register-pair which contains the memory address where the data is located (the high-order bits of the address are in the first register of the pair, the low-order bits in the second).
- Immediate — The instruction contains the data itself. This is either an 8-bit quantity or a 16-bit quantity (least significant byte first, most significant byte second).

Unless directed by an interrupt or branch instruction, the execution of instructions proceeds through consecutively increasing memory locations. A branch instruction can specify the address of the next instruction to be executed in one of two ways:

- Direct — The branch instruction contains the address of the next instruction to be executed. (Except for the "RST" instruction, byte 2 contains the low-order address and byte 3 the high-order address.)
- Register indirect — The branch instruction indicates a register-pair which contains the address of the next instruction to be excuted. (The high-order bits of the address are in the first register of the pair, the low-order bits in the second.)

The RST instruction is a special one-byte call instruction (usually used during interrupt sequences). RST includes a three-bit field; program control is transferred to the instruction whose address is eight times the contents of this three-bit field.

## Condition Flags

There are five condition flags associated with the execution of instructions on the 8080. They are Zero, Sign, Parity, Carry, and Auxiliary Carry, and are each represented by a 1-bit register in the CPU. A flag is "set" by forcing the bit to 1; "reset" by forcing the bit to 0.

Unless indicated otherwise, when an instruction affects a flag, if affects it in the following manner:

Zero: If the result of an instruction has the value 0, this flag is set; otherwise it is reset.

Sign: If the most significant bit of the result of the operation has the value 1, this flag is set; otherwise it is reset.

Parity: If the modulo 2 sum of the bits of the result of the operation is 0, (that is, if the result has even parity), this flag is set; otherwise it is reset (that is, if the result has odd parity).

Carry: If the instruction resulted in a carry (from addition), or a borrow (from subtraction or a comparison) out of the high-order bit, this flag is set; otherwise it is reset.

Auxiliary
Carry: If the instruction caused a carry out of bit 3 and into bit 4 of the resulting value, the auxiliary carry is set; otherwise it is reset. This flag is affected by single precision additions, subtractions, increments, decrements, comparisons, and logical operations, but is principally used with additions and increments preceding a DAA (Decimal Adjust Accumulator) instruction.

## Symbols and Abbreviations

The following symbols and abbreviations are used in the subsequent description of the 8080 instructions:

| SYMBOLS | MEANING |
|---|---|
| accumulator | Register A |
| addr | 16-bit address quantity |
| data | 8-bit data quantity |
| data 16 | 16-bit data quantity |
| byte 2 | The second byte of the instruction |
| byte 3 | The third byte of the instruction |
| port | 8-bit address of an I/O device |
| r,r1,r2 | One of the registers A,B,C,D,E,H,L |
| DDD,SSS | The bit pattern designating one of the registers A,B,C,D,E,H,L (DDD = destination, SSS = source): |

| DD or SSS | REGISTER NAME |
|---|---|
| 111 | A |
| 000 | B |
| 001 | C |
| 010 | D |
| 011 | E |
| 100 | H |
| 101 | L |

rp — One of the register pairs:
B represents the B,C pair with B as the high-order register and C as the low-order register;
D represents the D,E, pair with D as the high-order register and E as the low-order register;
H represents the H,L pair with H as the high-order register and L as the

RP low-order register;
SP represents the 16-bit stack pointer register.

RP  The bit pattern designating one of the register pairs B,D,H,SP:

| RP | REGISTER PAIR |
|----|---------------|
| 00 | B-C |
| 01 | D-E |
| 10 | H-L |
| 11 | SP |

rh  The first (high-order) register of a designated register pair.

rl  The second (low-order) register of a designated register pair.

PC  16-bit program counter register (PCH and PCL are used to refer to the high-order and low-order 8 bits respectively).

SP  16-bit stack pointer register (SPH and SPL are used to refer to the high-order and low-order 8 bits respectively).

rm  Bit m of the register r (bits are number 7 through 0 from left to right).

Z,S,P,CY,AC  The condition flags:
    Zero,
    Sign,
    Parity,
    Carry,
    and Auxiliary Carry, respectively.

( )  The contents of the memory location or registers enclosed in the parentheses.
$\leftarrow$  "Is transferred to"
$\wedge$  Logical AND
$\forall$  Exclusive OR
V  Inclusive OR
+  Addition
−  Two's complement subtraction
*  Multiplication
$\leftrightarrow$  "Is exchanged with"
—  The one's complement (e.g., (A))
n  The restart number 0 through 7
NNN  The binary representation 000 through 111 for restart number 0 through 7 respectively.

## Description Format
The following pages provide a detailed description of the instruction set of the 8080. Each instruction is described in the following manner:

1. The MAC 80 assembler format, consisting of the instruction mnemonic and operand fields, is printed in BOLDFACE on the left side of the first line.
2. The name of the instruction is enclosed in parenthesis on the right side of the first line.
3. The next line(s) contain a symbolic description of the operation of the instruction.
4. This is followed by a narative description of the operation of the instruction.
5. The following line(s) contain the binary fields and patterns that comprise the machine instruction.
6. The last four lines contain incidental information about the execution of the instruction. The number of machine cycles and states required to execute the instruction are listed first. If the instruction has two possible execution times, as in a Conditional Jump, both times will be listed, separated by a slash. Next, any significant data addressing modes (see Page 4-2) are listed. The last line lists any of the five Flags that are affected by the execution of the instruction.

## Data Transfer Group
This group of instructions transfers data to and from registers and memory. Condition flags are not affected by any instruction in this group.

### MOV r1, r2 (Move Register)
(r1) $\leftarrow$ (r2)
The content of register r2 is moved to register r1

| 0 | 1 | D | D | D | S | S | S |
|---|---|---|---|---|---|---|---|
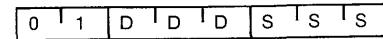
Addressing: register
Flags: none

### MOV r, M (Move from memory)
(r) $\leftarrow$ ((H) (L))
The content of the memory location, whose address is in registers H and L, is moved to register r.

| 0 | 1 | D | D | D | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|
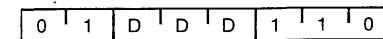
Addressing: reg. indirect
Flags: none

### MOV M, r (Move to memory)
((H) (L)) $\leftarrow$ (r)
The content of register r is moved to the memory location whose address is in registers H and L.

| 0 | 1 | 1 | 1 | 0 | S | S | S |
|---|---|---|---|---|---|---|---|
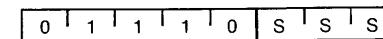
Addressing: reg. indirect
Flags: none

### MVI r, data (Move Immediate)
(r) $\leftarrow$ (byte 2)
The content of byte 2 of the instruction is moved to register r.

| 0 | 0 | D | D | D | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| data |||||||
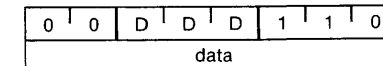
Addressing: immediate
Flags: none

### MVI M, data (Move to memory immediate)
((H) (L)) $\leftarrow$ (byte 2)
The content of byte 2 of the instruction is moved to the memory location whose address is in registers H and L.

| 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| data |||||||
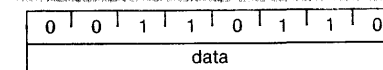
Addressing: immed./reg. indirect
Flags: none

### LXI rp, data 16 (Load register pair immediate)
(rh) $\leftarrow$ (byte 3),
(rl) $\leftarrow$ (byte 2)
Byte 3 of the instruction is moved into the high-order register (rh) of the register pair rp. Byte 2 of the instruction is moved into the low-order register (rl) of the register pair rp.

| 0 | 0 | R | P | 0 | 0 | 0 | 1 |
|---|---|---|---|---|---|---|---|
| low-order data |||||||
| high-order data |||||||
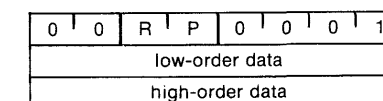
Addressing: immediate
Flags: none

### LDA addr (Load Accumulator direct)
(A) $\leftarrow$ ((byte 3) (byte 2))
The content of the memory location, whose address is specified in byte 2 and byte 3 of the instruction, is moved to register A.

| 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| low-order addr |||||||
| high-order addr |||||||
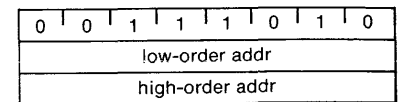
Addressing: direct
Flags: none

### STA addr (Store Accumulator direct)
((byte 3) (byte 2)) $\leftarrow$ (A)
The content of the accumulator is moved to the memory location whose address is specified in byte 2 and byte 3 of the instruction.

| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| low-order addr |||||||
| high-order addr |||||||
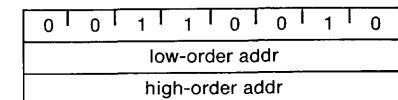
Addressing: direct
Flags: none

### LHLD addr (Load H and L direct)
(L) $\leftarrow$ ((byte 3) (byte 2))
(H) $\leftarrow$ ((byte 3) (byte 2) + 1)
The content of the memory location, whose address is specified in byte 2 and byte 3 of the instruction, is moved to register L. The content of the memory location at the succeeding address is moved to register H.

| 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| low-order addr |||||||
| high-order addr |||||||
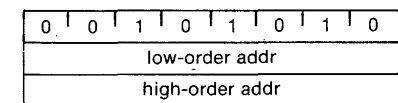
Addressing: direct
Flags: none

### SHLD addr (Store H and L direct)
((byte 3) (byte 2)) $\leftarrow$ (L)
((byte 3) (byte 2) + 1) $\leftarrow$ (H)
The content of register L is moved to the memory location whose address is specified in byte 2 and byte 3. The content of register H is moved to the succeeding memory location.

| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| low-order addr |||||||
| high-order addr |||||||
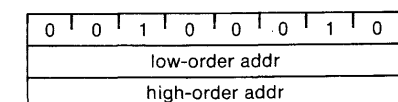
Addressing: direct
Flags: none

### LDAX rp (Load accumulator indirect)
(A) $\leftarrow$ ((rp))
The content of the memory location, whose address is in the register pair rp, is moved to register A. Note: only register pairs rp=B (registers B and C) or rp=D (registers D and E) may be specified.

| 0 | 0 | R | P | 1 | 0 | 1 | 0 |
|---|---|---|---|---|---|---|---|
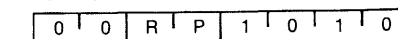
Addressing: reg. indirect
Flags: none

## STAX rp (Store accumulator indirect)

((r)) ← (A)

The content of register A is moved to the memory location whose address is in the register pair rp. Note: only register pairs rp=B (registers B and C) or rp=D (registers D and E) may be specified.

| 0 | 0 | R | P | 0 | 0 | 1 | 0 |
|---|---|---|---|---|---|---|---|
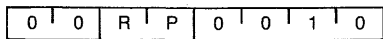
Addressing: reg. indirect
Flags: none

## XCHG (Exchange H and L with D and E)

(H)     (D)
(L)     (E)

The contents of registers H and L are exchanged with the contents of registers D and E.

| 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|
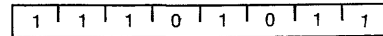
Addressing: register
Flags: none

# Arithmetic Group

This group of instructions performs arithmetic operations on data in registers and memory.

Unless indicated otherwise, all instructions in this group affect the Zero, Sign, Parity, Carry, and Auxiliary Carry flags according to the standard rules.

All subtraction operations are performed via two's complement arithmetic and set the carry flag to one to indicate a borrow and clear it to indicate no borrow.

## ADD r (Add Register)

(A) ← (A) + (r)

The content of register r is added to the content of the accumulator. The result is placed in the accumulator.

| 1 | 0 | 0 | 0 | 0 | S | S | S |
|---|---|---|---|---|---|---|---|
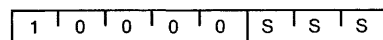
Addressing: register
Flags: Z,S,P,CY,AC

## ADD M (Add memory)

(A) ← (A) + ((H) (L))

The content of the memory location whose address is contained in the H and L registers is added to the content of the accumulator. The result is placed in the accumulator.

| 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|
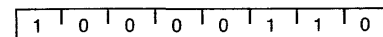
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

## ADI data (Add immediate)

(A) ← (A) + (byte 2)

The content of the second byte of the instruction is added to the content of the accumulator. The result is placed in the accumulator.

| 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| data |||||||
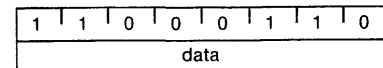
Addressing: immediate
Flags: Z,S,P,CY,AC

## ADC r (Add Register with carry)

(A) ← (A) + (r) + (CY)

The content of register r and the content of the carry bit are added to the content of the accumulator. The result is placed in the accumulator.

| 1 | 0 | 0 | 0 | 1 | S | S | S |
|---|---|---|---|---|---|---|---|
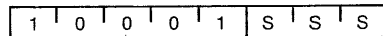
Addressing: register
Flags: Z,S,P,CY,AC

## ADC M (Add memory with carry)

(A) ← (A) + ((H) (L)) + (CY)

The content of the memory location whose address is contained in the H and L registers and the content of the CY flag are added to the accumulator. The result is placed in the accumulator.

| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|
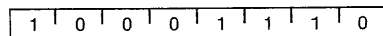
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

## ACI data (Add immediate with carry)

(A) ← (A) + (byte 2) + (CY)

The content of the second byte of the instruction and the content of the CY flag are added to the contents of the accumulator. The result is placed in the accumulator.

| 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| data |||||||
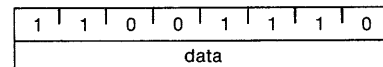
Addressing: immediate
Flags: Z,S,P,CY,AC

## SUB r (Subtract Register)

(A) ← (A) – (r)

The content of register r is subtracted from the content of the accumulator. The result is placed in the accumulator.

| 1 | 0 | 0 | 1 | 0 | S | S | S |
|---|---|---|---|---|---|---|---|
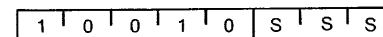
Addressing: register
Flags: Z,S,P,CY,AC

## SUB M (Subtract memory)

The content of the memory location whose address is contained in the H and L registers is subtracted from the content of the accumulator. The result is placed in the accumulator.

| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|
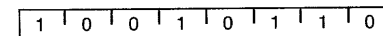
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

## SUI data (Subtract immediate)

(A) ← (A) – (byte 2)

The content of the second byte of the instruction is subtracted from the content of the accumulator. The result is placed in the accumulator.

| 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| data |||||||
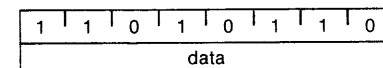
Addressing: immediate
Flags: Z,S,P,CY,AC

## SBB r (Subtract Register with borrow)

(A) ← (A) – (r) – (CY)

The content of register r and the content of the CY flag are both subtracted from the accumulator. The result is placed in the accumulator.

| 1 | 0 | 0 | 1 | 1 | S | S | S |
|---|---|---|---|---|---|---|---|
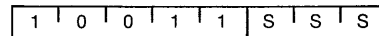
Addressing: register
Flags: Z,S,P,CY,AC

## SBB M (Subtract memory with borrow)

(A) ← (A) – ((H) (L)) – (CY)

The content of the memory location whose address is contained in the H and L registers and the content of the CY flag are both subtracted from the accumulator. The result is placed in the accumulator.

| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|
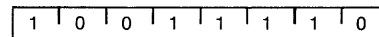
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

## SBI data (Subtract immediate with borrow)

(A) ← (A) – (byte 2) – (CY)

The contents of the second byte of the instruction and the contents of the CY flag are both subtracted from the accumulator. The result is placed in the accumulator.

| 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| data |||||||
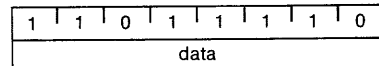
Addressing: immediate
Flags: Z,S,P,CY,AC

## INR r (Increment Register)

(r) ← (r) + 1

The content of register r is incremented by one. Note: All condition flags except CY are affected.

| 0 | 0 | D | D | D | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|
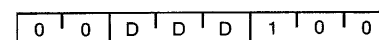
Addressing: register
Flags: Z,S,P,AC

## INR M (Increment memory)

((H) (L)) ← ((H) (L)) + 1

The content of the memory location whose address is contained in the H and L registers is incremented by one. Note: All condition flags except CY are affected.

| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|
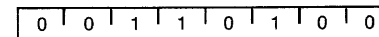
Addressing: reg. indirect
Flags: Z,S,P,AC

## DCR r (Decrement Register)

(R) ← (r) – 1

The content of register r is decremented by one. Note: All condition flags except CY are affected.

| 0 | 0 | D | D | D | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|
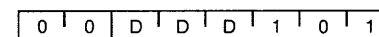
Addressing: register
Flags: Z,S,P,AC

## DCR M (Decrement memory)

$((H)(L)) \leftarrow ((H)(L)) - 1$

The content of the memory location whose address is contained in the H and L registers is decremented by one. Note: All condition flags except CY are affected.

| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|
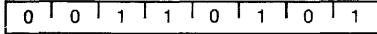
Addressing: reg. indirect
Flags: Z,S,P,AC

## INX rp (Increment register pair)

$(rh)(rl) \leftarrow (rh)(rl) + 1$

The content of the register pair rp is incremented by one. Note: No condition flags are affected.

| 0 | 0 | R | P | 0 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|
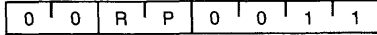
Addressing: register
Flags: none

## DCX rp (Decrement register pair)

$(rh)(rl) \leftarrow (rh)(rl) - 1$

The content of the register pair rp is decremented by one. Note: No condition flags are affected.

| 0 | 0 | R | P | 1 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|
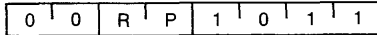
Addressing: register
Flags: none

## DAD rp (Add register pair to H and L)

$(H)(L) \leftarrow (H)(L) + (rh)(rl)$

The content of the register pair rp is added to the content of the register pair H and L. The result is placed in the register pair H and L. Note: Only the CY flag is affected. It is set if there is a carry out of the double precision add; otherwise it is reset.

| 0 | 0 | R | P | 1 | 0 | 0 | 1 |
|---|---|---|---|---|---|---|---|
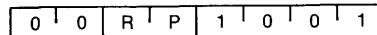
Addressing: register
Flags: CY

## DAA (Decimal Adjust Accumulator)

The 8-bit number in the accumulator is adjusted to form two 4-bit Binary-Coded-Decimal digits by the following process:

1. If the value of the least significant 4 bits of the accumulator is greater than 9 or if the AC flag is set, 6 is added to the accumulator.
2. If the value of the most significant 4 bits of the accumulator is now greater than 9, or if the CY flag is set, 6 is added to the most significant 4 bits of the accumulator.

Note: All flags are affected.

| 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|
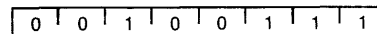
Flags: Z,S,P,CY,AC

## Logical Group

This group of instructions performs logical (Boolean) operations on data in registers and memory and on condition flags.

---

Unless indicated otherwise, all instructions in this group affect the Zero, Sign, Parity, Auxiliary Carry, and Carry flags according to the standard rules.

## ANA r (AND Register)

$(A) \leftarrow (A) \wedge (r)$

The content of register r is logically anded with the content of the accumulator. The result is placed in the accumulator. The CY flag is cleared.

| 1 | 0 | 1 | 0 | 0 | S | S | S |
|---|---|---|---|---|---|---|---|
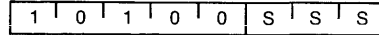
Addressing: register
Flags: Z,S,P,CY,AC

## ANA M (AND memory)

$(A) \leftarrow (A) \wedge ((H)(L))$

The contents of the memory location whose address is contained in the H and L registers is logically anded with the content of the accumulator. The result is placed in the accumulator. The CY flag is cleared.

| 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|
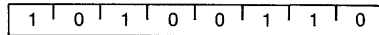
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

## ANI data (AND immediate)

$(A) \leftarrow (A) \wedge (byte\ 2)$

The content of the second byte of the instruction is logically anded with the contents of the accumulator. The result is placed in the accumulator. The CY and AC flags are cleared.

| 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| data | | | | | | | |
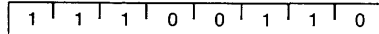
Addressing: immediate
Flags: Z,S,P,CY,AC

## XRA r (Exclusive-OR Register)

$(A) \leftarrow (A) \veebar (r)$

The content of register r is exclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. The CY and AC flags are cleared.

| 1 | 0 | 1 | 0 | 1 | S | S | S |
|---|---|---|---|---|---|---|---|
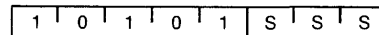
Addressing: register
Flags: Z,S,P,CY,AC

## XRA M (Exclusive-OR Memory)

$(A) \leftarrow (A) \veebar ((H)(L))$

The content of the memory location whose address is contained in the H and L registers is exclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. The CY and AC flags are cleared.

| 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|
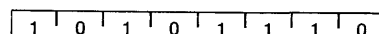
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

---

## XRI data (Exclusive-OR immediate)

$(A) \leftarrow (A) \veebar (byte\ 2)$

The content of the second byte of the instruction is exclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. The CY and AC flags are cleared.

| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| data | | | | | | | |
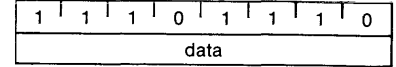
Addressing: immediate
Flags: Z,S,P,CY,AC

## ORA r (OR Register)

$(A) \leftarrow (A) \vee (r)$

The content of register r is inclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. The CY and AC flags are cleared.

| 1 | 0 | 1 | 1 | 0 | S | S | S |
|---|---|---|---|---|---|---|---|
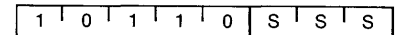
Addressing: register
Flags: Z,S,P,CY,AC

## ORA M (OR memory)

$(A) \leftarrow (A) \vee ((H)(L))$

The content of the memory location whose address is contained in the H and L registers is inclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. The CY and AC flags are cleared.

| 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|
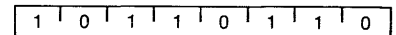
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

## ORI data (OR immediate)

$(A) \leftarrow (A) \vee (byte\ 2)$

The content of the second byte of the instruction is inclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. The CY and AC flags are cleared.

| 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| data | | | | | | | |
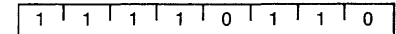
Addressing: immediate
Flags: Z,S,P,CY,AC

## CMP r (Compare Register)

$(A) - (r)$

The content of register r is subtracted from the accumulator. The accumulator remains unchanged. The condition flags are set as a result of the subtraction. The Z flag is set to 1 if $(A) = (r)$. The CY flag is set to 1 if $(A) < (r)$.

| 1 | 0 | 1 | 1 | 1 | S | S | S |
|---|---|---|---|---|---|---|---|
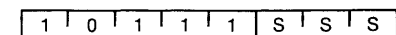
Addressing: register
Flags: Z,S,P,CY,AC

## CMP M (Compare memory)

$(A) - ((H)(L))$

The content of the memory location whose address is contained in the H and L registers is subtracted from the accumulator. The accumulator remains unchanged. The con-

dition flags are set as a result of the subtraction. The Z flag is set to 1 if (A) = ((H) (L)). The CY flag is set to 1 if (A) < ((H) (L)).

| 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
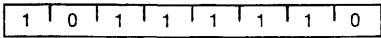
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

## CPI data (Compare immediate)

(A) – (byte 2)
The content of the second byte of the instruction is subtracted from the accumulator. The condition flags are set by the result of the subtraction. The Z flag is set to 1 if (A) = (byte 2). The CY flag is set to 1 if (A) < (byte 2).

| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| data | | | | | | | | | |
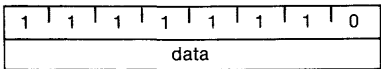
Addressing: immediate
Flags: Z,S,P,CY,AC

## RLC (Rotate left)

$(A_{n+1}) \leftarrow (A_n)$; $(A_0) \leftarrow (A_7)$
$(C) \leftarrow (A_7)$
The content of the accumulator is rotated left one position. The low order bit and the CY flag are both set to the value shifted out of the high order bit position. Only the CY flag is affected.

| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|---|
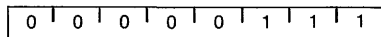
Flags: CY

## RRC (Rotate right)

$(A_n) \leftarrow (A_{n-1})$; $(A_7) \leftarrow (A_0)$
$(CY) \leftarrow (A_0)$
The content of the accumulator is rotated right one position. The high order bit and the CY flag are both set to the value shifted out of the low order bit position. Only the CY flag is affected.

| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|---|
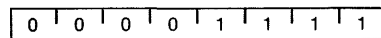
Flags: CY

## RAL (Rotate left through carry)

$(A_{n+1}) \leftarrow (A_n)$; $(CY) \leftarrow (A_7)$
$(A_0) \leftarrow (CY)$
The content of the accumulator is rotated left one position through the CY flag. The low order bit is set equal to the CY flag and the CY flag is set to the value shifted out of the high order bit. Only the CY flag is affected.

| 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|---|
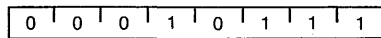
Flags: CY

## RAR (Rotate right through carry)

$(A_n) \leftarrow (A_{n+1})$; $(CY) \leftarrow (A_0)$
$(A_7) \leftarrow (CY)$
The content of the accumulator is rotated right one position through the CY flag. The high order bit is set to the CY flag and the CY flag is set to the value shifted out of the low order bit. Only the CY flag is affected.

| 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|---|
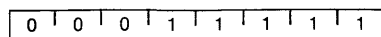
Flags: CY

## CMA (Complement accumulator)

$(A) \leftarrow (\overline{A})$
The contents of the accumulator are complemented (zero bits become 1, one bits become 0). No flags are affected.

| 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|
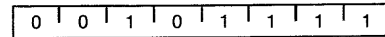
Flags: none

## CMC (Complement carry)

$(CY) \leftarrow (\overline{CY})$
The CY flag is complemented. No other flags are affected.

| 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|---|
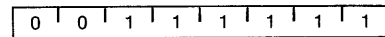
Flags: CY

## STC (Set carry)

$(Cy) \leftarrow 1$
The CY flag is set to 1. No other flags are affected.

| 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|---|
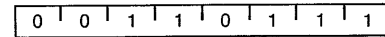
Flags: CY

# Branch Group

This group of instructions alters normal sequential program flow.

Condition flags are not affected by any instruction in this group.

The two types of branch instructions are unconditional and conditional. Unconditional transfers simply perform the specified operation on register PC (the program counter). Conditional transfers examine the status of one of the four processor flags to determine if the specified branch is to be executed. The conditions that may be specified are as follows:

| CONDITION | CCC |
|---|---|
| NZ — not zero (Z = 0) | 000 |
| Z — zero (Z = 1) | 001 |
| NC — no carry (CY = 0) | 010 |
| C — carry (CY = 1) | 011 |
| PO — parity odd (P = 0) | 100 |
| PE — parity even (P = 1) | 101 |
| P — plus (S = 0) | 110 |
| M — minus (S = 1) | 111 |

## JMP addr (Jump)

$(PC) \leftarrow$ (byte 3) (byte 2)
Control is transferred to the instruction whose address is specified in byte 3 and byte 2 of the current instruction.

| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|---|
| low-order addr | | | | | | | | | |
| high-order addr | | | | | | | | | |
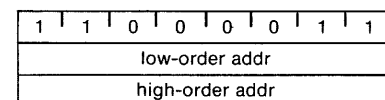
Addressing: immediate
Flags: none

## Jcondition addr (Conditional jump)

If (CCC),
$(PC) \leftarrow$ (byte 3) (byte 2)
If the specified condition is true, control is transferred to the instruction whose address is specified in byte 3 and byte 2 of the current instruction; otherwise, control continues sequentially.

| 1 | 1 | C | C | C | 0 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| low-order addr | | | | | | | |
| high-order addr | | | | | | | |
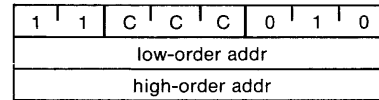
Addressing: immediate
Flags: none

## CALL addr (Call)

$((SP) - 1) \leftarrow (PCH)$
$((SP) - 2) \leftarrow (PCL)$
$(SP) \leftarrow (SP) - 2$
$(PC) \leftarrow$ (byte 3) (byte 2)
The high-order eight bits of the next instruction address are moved to the memory location whose address is one less than the content of register SP. The low-order eight bits of the next instruction address are moved to the memory location whose address is two less than the content of register SP. The content of register SP is decremented by 2. Control is transferred to the instruction whose address is specified in byte 3 and byte 2 of the current instruction.

| 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|
| low-order addr | | | | | | | |
| high-order addr | | | | | | | |
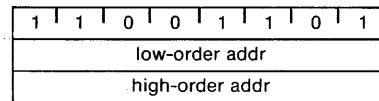
Addressing: immediate/reg. indirect
Flags: none

## Ccondition addr (Condition call)

If (CCC),
$((SP) - 1) \leftarrow (PCH)$
$((SP) - 2) \leftarrow (PCL)$
$(SP) \leftarrow (SP) - 2$
$(PC) \leftarrow$ (byte 3) (byte 2)
If the specified condition is true, the actions specified in the CALL instruction (see above) are performed; otherwise, control continues sequentially.

| 1 | 1 | C | C | C | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|
| low-order addr | | | | | | | |
| high-order addr | | | | | | | |
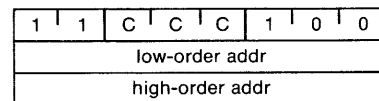
Addressing: immediate/reg. indirect
Flags: none

## RET (Return)

$(PCL) \leftarrow ((SP))$;
$(PCH) \leftarrow ((SP) + 1)$;
$(SP) \leftarrow (SP) + 2$;
The content of the memory location whose address is specified in register SP is moved to the low-order eight bits of register PC. The content of the memory location whose

address is one more than the content of register SP is moved to the high-order eight bits of register PC. The content of register SP is incremented by 2.

| 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
|---|---|---|---|---|---|---|---|
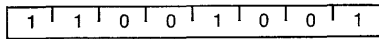
Addressing: reg. indirect
Flags: none

### Rcondition (Conditional return)
If (CCC),
(PCL) ← ((SP))
(PCH) ← ((SP) + 1)
(SP) ← (SP) + 2
If the specified condition is true, the actions specified in the RET instruction (see above) are performed; otherwise, control continues sequentially.

| 1 | 1 | C | C | C | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|
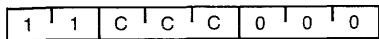
Addressing: reg. indirect
Flags: none

### RST n (Restart)
((SP) – 1) ← (PCH)
((SP) – 2) ← (PCL)
(SP) ← (SP) – 2
(PC) ← 8 • (NNN)
The high-order eight bits of the next instruction address are moved to the memory location whose address is one less than the content of register SP. The low-order eight bits of the next instruction address are moved to the memory location whose address is two less than the content of register SP. The content of register SP is decremented by two. Control is transferred to the instruction whose address is eight times the content of NNN.

| 1 | 1 | N | N | N | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|

Addressing: reg. indirect
Flags: none

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | N | N | N | 0 | 0 | 0 |
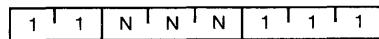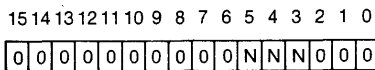
Program Counter After Restart

### PCHL (Jump H and L indirect—move H and L to PC)
(PCH) ← (H)
(PCL) ← (L)
The content of register H is moved to the high-order eight bits of register PC. The content of register L is moved to the low-order eight bits of register PC.

| 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
|---|---|---|---|---|---|---|---|
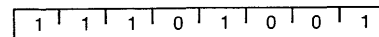
Addressing: register
Flags: none

## Stack, I/O, and Machine Control Group
This group of instructions performs I/O, manipulates the Stack, and alters internal control flags.

Unless otherwise specified, condition flags are not affected by any instructions in this group.

### PUSH rp (Push)
((SP) – 1) ← (rh)
((SP) – 2) ← (rl)
(SP) ← (SP) – 2

The content of the high-order register of register pair rp is moved to the memory location whose address is one less than the content of register SP. The content of the low-order register of register pair rp is moved to the memory location whose address is two less than the content of register SP. The content of register SP is decremented by 2. Note: Register pair rp=SP may not be specified.

| 1 | 1 | R | P | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|
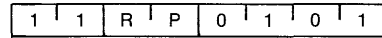
Addressing: reg. indirect
Flags: none

### PUSH PSW (Push processor status word)
((SP) – 1) ← (A)
((SP) – 2)$_0$ ← (CY), ((SP) – 2)$_1$ ← 1
((SP) – 2)$_2$ ← (P), ((SP) – 2)$_3$ ← 0
((SP) – 2)$_4$ ← (AC), ((SP) – 2)$_5$ ← 0
((SP) – 2)$_6$ ← (Z), ((SP) – 2)$_7$ ← (S)
(SP) ← (SP) – 2
The content of register A is moved to the memory location whose address is one less than register SP. The contents of the condition flags are assembled into a processor status word and the word is moved to the memory location whose address is two less than the content of register SP. The content of register SP is decremented by two.

| 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|
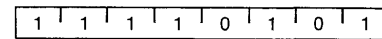
Addressing: reg. indirect
Flags: none

FLAG WORD

| $D_7$ | $D_6$ | $D_5$ | $D_4$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ |
|-------|-------|-------|-------|-------|-------|-------|-------|
| S | Z | 0 | AC | 0 | P | 1 | CY |
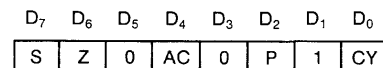
### POP rp (Pop)
(rl) ← ((SP))
(rh) ← ((SP) + 1)
(SP) ← (SP) + 2

The content of the memory location, whose address is specified by the content of register SP, is moved to the low-order register of register pair rp. The content of the memory location, whose address is one more than the content of register SP, is moved to the high-order register of register pair rp. The content of register SP is incremented by 2. Note: Register pair rp = SP may not be specified.

| 1 | 1 | R | P | 0 | 0 | 0 | 1 |
|---|---|---|---|---|---|---|---|
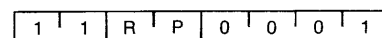
Addressing: reg. indirect
Flags: none

### POP PSW (Pop processor status word)
(CY) ← ((SP))$_0$
(P) ← ((SP))$_2$
(AC) ← ((SP))$_4$
(Z) ← ((SP))$_6$
(S) ← ((SP))$_7$
(A) ← ((SP) + 1)
(SP) ← (SP) + 2

The content of the memory location whose address is specified by the content of register SP is used to restore the condition flags. The content of the memory location whose address is one more than the content of register SP is moved to register A. The content of register SP is incremented by 2.

| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
|---|---|---|---|---|---|---|---|
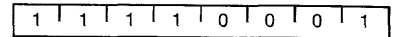
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

### XTHL (Exchange stack top with H and L)
(L) ↔ ((SP))
(H) ↔ ((SP) + 1)
The content of the L register is exchanged with the content of the memory location whose address is specified by the content of register SP. The content of the H register is exchanged with the content of the memory location whose address is one more than the content of register SP.

| 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|
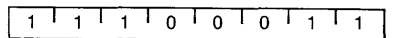
Addressing: reg. indirect
Flags: none

### SPHL (Move HL to SP)
(SP) ← (H) (L)
The contents of registers H and L (16 bits) are moved to register SP.

| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
|---|---|---|---|---|---|---|---|
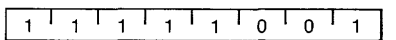
Addressing: register
Flags: none

### IN port (Input)
(A) ← (data)
The data placed on the eight bit bidirectional data bus by the specified port is moved to register A.

| 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|
| port |
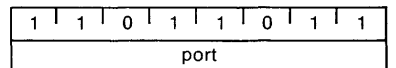
Addressing: direct
Flags: none

### OUT port (Output)
(data) ← (A)
The content of register A is placed on the eight bit bidirectional data bus for transmission to the specified port.

| 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|
| port |
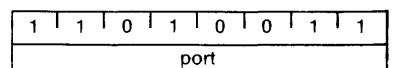
Addressing: direct
Flags: none

## EI (Enable interrupts)
The interrupt system is enabled following the execution of the next instruction.

| 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
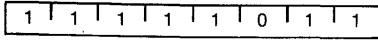
Flags:     none

## DI (Disable interrupts)
The interrupt system is disabled immediately following the execution of the DI instruction.

| 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
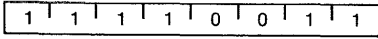
Flags:     none

## HLT (Halt)
The processor is stopped. The registers and flags are unaffected.

| 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 |
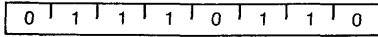
Flags:     none

## NOP (No op)
No operation is performed. The registers and flags are unaffected.

| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
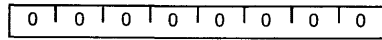
Flags:     none

## MUL (multiply)
*Setup Conditions*
Multiplier in A Register
Multiplicand in B Register

*Resultant Conditions*
16-bit result in B and C Registers (LSB in C)
Carry Flag (CY) contains MSB of result
Half Carry Flag (HC) is indeterminate

| 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
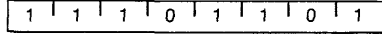
Flags:     CY = MSB of result
           HC = Indeterminate

## DIV (divide)
*Setup Conditions*

Divisor in A Register
Dividend in C Register

*Resultant Conditions*
Quotient in C Register
Remainder in B Register
Divisor in A Register (Unchanged)
Carry Flag (CY) contains LSB of quotient
Half Carry Flag (HC) contains 1

| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
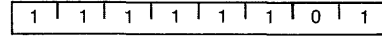
Flags:     CY = LSB of quotient
           HC = 1

# Table C-1 INSTRUCTION SET SUMMARY OF PROCESSOR INSTRUCTIONS

| MNEMONIC | DESCRIPTION | $D_7$ | $D_6$ | $D_5$ | $D_4$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ |
|---|---|---|---|---|---|---|---|---|---|
| MOV r1 | Move register to register | 0 | 1 | D | D | D | S | S | S |
| MOV M,r | Move register to memory | 0 | 1 | 1 | 1 | 0 | S | S | S |
| MOV r,M | Move memory to register | 0 | 1 | D | D | D | 1 | 1 | 0 |
| MOV r,M | Move memory to register | 0 | 1 | D | D | D | 1 | 1 | 0 |
| HLT | Halt | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 |
| MVI r | Move immediate register | 0 | 0 | D | D | D | 1 | 1 | 0 |
| MVI M | Move immediate memory | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| INR r | Increment register | 0 | 0 | D | D | D | 1 | 0 | 0 |
| DCR r | Decrement register | 0 | 0 | D | D | D | 1 | 0 | 1 |
| INR M | Increment memory | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| DCR M | Decrement memory | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| ADD r | Add register to A | 1 | 0 | 0 | 0 | 0 | S | S | S |
| ADC r | Add register to A with carry | 1 | 0 | 0 | 0 | 1 | S | S | S |
| SUB r | Subtract register from A | 1 | 0 | 0 | 1 | 0 | S | S | S |
| SBB r | Subtract register from A with borrow | 1 | 0 | 0 | 1 | 1 | S | S | S |
| ANA r | And register with A | 1 | 0 | 1 | 0 | 0 | S | S | S |
| XRA r | Exclusive Or register with A | 1 | 0 | 1 | 0 | 1 | S | S | S |
| ORA r | Or register with A | 1 | 0 | 1 | 1 | 0 | S | S | S |
| CMP r | Compare register with A | 1 | 0 | 1 | 1 | 1 | S | S | S |
| ADD M | Add memory to A | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| ADC M | Add memory to A with carry | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| SUB M | Subtract memory from A | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| SBB M | Subtract memory from A with borrow | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| ANA M | And memory with A | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| XRA M | Exclusive Or memory with A | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 |
| ORA M | Or memory with A | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| CMP M | Compare memory with A | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| ADI | Add immediate to A | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| ACI | Add immediate to A with carry | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 |
| SUI | Subtract immediate from A | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| SBI | Subtract immediate from A with borrow | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |
| ANI | And immediate with A | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| XRI | Exclusive OR immediate with A | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |
| ORI | Or immediate with A | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 |
| CPI | Compare immediate with A | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| RLC | Rotate A left | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| RRC | Rotate A right | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| RAL | Rotate A left through carry | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| RAR | Rotate A right through carry | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| JMP | Jump unconditional | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| JC | Jump on carry | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 |
| JNC | Jump on no carry | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| JZ | Jump on zero | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| JNZ | Jump on no zero | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| JP | Jump on positive | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 |
| JM | Jump on minus | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 |
| JPE | Jump on parity even | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| JPO | Jump on parity odd | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| CALL | Call unconditional | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 |
| CC | Call on carry | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 |
| CNC | Call on no carry | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| CZ | Call on zero | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| CNZ | Call on no zero | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| CP | Call on positive | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
| CM | Call on minus | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| CPE | Call on parity even | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| CPO | Call on parity odd | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |

| MNEMONIC | DESCRIPTION | $D_7$ | $D_6$ | $D_5$ | $D_4$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ |
|---|---|---|---|---|---|---|---|---|---|
| RET | Return | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| RC | Return on carry | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| RNC | Return on no carry | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| RZ | Return on zero | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| RNZ | Return on no zero | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| RP | Return on positive | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| RM | Return on minus | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| RPE | Return on parity even | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| RPO | Return on parity odd | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| RST | Restart | 1 | 1 | A | A | A | 1 | 1 | 1 |
| IN | Input | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| OUT | Output | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 |
| LXI B | Load immediate register Pair B & C | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| LXI D | Load immediate register Pair D & E | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| LXI H | Load immediate register Pair H & L | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| LXI SP | Load immediate stack pointer | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| PUSH B | Push register Pair B & C on stack | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| PUSH D | Push register Pair D & E on stack | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| PUSH H | Push register Pair H & L on stack | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 |
| PUSH PSW | Push A and Flags on stack | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |
| POP B | Pop register Pair B & C off stack | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| POP D | Pop register Pair D & E off stack | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| POP H | Pop register Pair H & L off stack | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| POP PSW | Pop A and Flags off stack | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| STA | Store A direct | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| LDA | Load A direct | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 |
| XCHG | Exchange D & E, H & L Registers | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 |
| XTHL | Exchange top of stack, H & L | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| SPHL | H & L to stack pointer | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| PCHL | H & L to program counter | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| DAD B | Add B & C to H & L | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| DAD D | Add D & E to H & L | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| DAD H | Add H & L to H & L | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| DAD SP | Add stack pointer to H & L | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| STAX B | Store A indirect | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| STAX D | Store A indirect | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| LDAX B | Load A indirect | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| LDAX D | Load A indirect | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| INX B | Increment B & C registers | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| INX D | Increment D & E registers | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |
| INX H | Increment H & L registers | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| INX SP | Increment stack pointer | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| DCX B | Decrement B & C | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| DCX D | Decrement D & E | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| DCX H | Decrement H & L | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| DCX SP | Decrement stack pointer | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |
| CMA | Complement A | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| STC | Set carry | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 |
| CMC | Complement carry | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| DAA | Decimal adjust A | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 |
| SHLD | Store H & L direct | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| LHLD | Load H & L direct | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| EI | Enable Interrupts | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| DI | Disable interrupts | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| NOP | No operation | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| MUL | Multiply | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |
| DIV | Divide | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |

*NOTE

DDD or SSS – 000 B – 001 C – 010 D – 011 E – 100 H – 101 L – 110 Memory – 111 A.

**Table C-1 INSTRUCTION SET SUMMARY OF PROCESSOR INSTRUCTIONS**

## MICROPROGRAM CONTROL UNIT                                    S/N3001

N3001-I

## DESCRIPTION

The N3001 MCU is 1 element of a bipolar microcomputer set. When used with the S/N3002, 54/74S182, ROM or PROM memory, a powerful microprogrammed computer can be implemented.

The 3001 MCU controls the fetch sequence of microinstructions from the microprogram memory. Functions performed by the 3001 include:

- Maintenance of microprogram address register
- Selection of next microinstruction address
- Decoding and testing of data supplied via several input buses
- Saving and testing of carry output data from the central processing (CP) array
- Control of carry/shift input data to the CP array
- Control of microprogram interrupts

## FEATURES

- **Schottky TTL process**
- **45ns cycle time (typ.)**
- **Direct addressing of standard bipolar PROM or ROM**
- **512 microinstruction addressability**
- **Advanced organization:**
  - **9-bit microprogram address register and bus organized to address memory by row and column**
  - **4-bit program latch**
  - **2-flag registers**
- **11 address control functions:**
  - **3 jump and test latch function**
  - **16 way jump and test instruction**
- **8 flag control functions:**
  - **4 flag input functions**
  - **4 flag output functions**

## PIN CONFIGURATION

### I PACKAGE

| | | | |
|---|---|---|---|
| $\overline{PX_4}$ | 1 | 40 | $V_{CC}$ |
| $\overline{PX_7}$ | 2 | 39 | $AC_0$ |
| $\overline{PX_6}$ | 3 | 38 | $AC_1$ |
| $\overline{PX_5}$ | 4 | 37 | $AC_5$ |
| $\overline{SX_3}$ | 5 | 36 | LD |
| $\overline{SX_2}$ | 6 | 35 | ERA |
| $PR_2$ | 7 | 34 | $MA_8$ |
| $\overline{SX_1}$ | 8 | 33 | $MA_7$ |
| $PR_1$ | 9 | 32 | $MA_6$ |
| $\overline{SX_0}$ | 10 | 31 | $MA_5$ |
| $PR_0$ | 11 | 30 | $MA_4$ |
| $FC_3$ | 12 | 29 | $MA_0$ |
| $FC_2$ | 13 | 28 | $MA_3$ |
| $\overline{FO}$ | 14 | 27 | $MA_2$ |
| $FC_0$ | 15 | 26 | $MA_1$ |
| $FC_1$ | 16 | 25 | EN |
| $\overline{FI}$ | 17 | 24 | $AC_6$ |
| ISE | 18 | 23 | $AC_4$ |
| CLK | 19 | 22 | $AC_3$ |
| GND | 20 | 21 | $AC_2$ |

## BLOCK DIAGRAM

## PIN DESCRIPTION

| PIN | SYMBOL | NAME AND FUNCTION | TYPE |
|---|---|---|---|
| 1-4 | $\overline{PX_4}$-$\overline{PX_7}$ | Primary Instruction Bus Inputs<br>Data on the primary instruction bus is tested by the JPX function to determine the next microprogram address. | Active low |
| 5,6,8,10 | $\overline{SX_0}$-$\overline{SX_3}$ | Secondary Instruction Bus Inputs<br>Data on the secondary instruction bus is synchronously loaded into the PR-latch while the data on the PX-bus is being tested (JPX). During a subsequent cycle, the contents of the PR-latch may be tested by the JPR, JLL, or JRL functions to determine the next microprogram address. | Active low |
| 7,9,11 | $PR_0$-$PR_2$ | PR-Latch Outputs<br>The PR-latch outputs ($SX_0$-$SX_2$) are synchronously enabled by the JCE function. They can be used to modify microinstructions at the outputs of the microprogram memory or to provide additional control lines. | Open Collector |
| 12,13<br>15,16 | $FC_0$-$FC_3$ | Flag Logic Control Inputs<br>The flat logic control inputs are used to cross-switch the flags (C and Z) with the flag logic input (FI) and the flag logic output (FO). | Active high |
| 14 | $\overline{FO}$ | Flag Logic Output<br>The outputs of the flags (C and Z) are multiplexed internally to form the common flag logic output. The output may also be forced to a logical 0 or logical 1. | Active low<br>Three-state |
| 17 | $\overline{FI}$ | Flag Logic Input<br>The flag logic input is demultiplexed internally and applied to the inputs of the flags (C and Z). Note: The flag input data is saved in the F-latch when the clock input (CLK) is low. | Active low |
| 18 | ISE | Interrupt Strobe Enable Output<br>The interrupt strobe enable output goes to logical 1 when one of the JZR functions are selected (see Functional Description). It can be used to provide the strobe signal required by interrupt circuits. | Active high |
| 19 | CLK | Clock Input | |
| 20 | GND | Ground | |
| 21-24<br>37-39 | $AC_0$-$AC_6$ | Next Address Control Function Inputs<br>All jump functions are selected by these control lines. | Active high |
| 25 | EN | Enable Input<br>When in the high state, the enable input enables the microprogram address, PR-latch and flag outputs. | |
| 26-29 | $MA_0$-$MA_3$ | Microprogram Column Address Outputs | Three-state |
| 30-34 | $MA_4$-$MA_8$ | Microprogram Row Address Outputs | Three-state |
| 35 | ERA | Enable Row Address Input<br>When in the low state, the enable row address input independently disables the microprogram row address outputs. It can be used to facilitate the implementation of priority interrupt systems. | Active high |
| 36 | LD | Microprogram Address Load Input<br>When the active high state, the microprogram address load input inhibits all jump functions and synchronously loads the data on the instruction buses into the microprogram address register. However, it does not inhibit the operation of the PR-latch or the generation of the interrupt strobe enable. | Active high |
| 40 | $V_{CC}$ | +5 Volt supply | |

## THEORY OF OPERATION

The MCU controls the sequence of microinstructions in the microprogram memory. The MCU simultaneously controls 2 flip-flops (C, Z) which are interactive with the carry-in and carry-out logic of an array of CPEs.

The functional control of the MCU provides both unconditional jumps to new memory locations and jumps which are dependent on the state of MCU flags or the state of the "PR" latch. Each instruction has a "jump set" associated with it. This "jump set" is the total group of memory locations which can be addressed by that instruction.

The MCU utilizes a two-dimensional addressing scheme in the microprogram memory. Microprogram memory is organized as 32 rows and 16 columns for a total of 512 words. Word length is variable according to application. Address is accomplished by a 9-bit address organized as a 5-bit row and 4-bit column address.

## ADDRESSING ORGANIZATION



MA4–MA8

32 ROWS

|← 16 COLUMNS →|

MA0–MA3

## FUNCTIONAL DESCRIPTION

The following is a description of each of the eleven address control functions. The symbols shown below are used to specify row and column addresses.

| SYMBOL | MEANING |
|--------|---------|
| row$_n$ | 5-bit next row address where n is the decimal row address. |
| col$_n$ | 4-bit next column address where n is the decimal column address. |

## Unconditional Address Control (Jump) Functions

The jump functions use the current microprogram address (i.e., the contents of the microprogram address register prior to the rising edge of the clock) and several bits from the address control inputs (AC0-AC6) to generate the next microprogram address.

## Flag Conditional Address Control (Jump Test) Functions

The jump/test flag functions use the current microprogram address, the contents of the selected flag or latch, and several bits from the address control function to generate the next microprogram address.

## JUMP FUNCTION TABLE

| MNEMONIC | FUNCTION DESCRIPTION |
|----------|----------------------|
| JCC | Jump in current column. $AC_0$-$AC_4$ are used to select 1 of 32 row addresses in the current column, specified by $MA_0$-$MA_3$, as the next address. |
| JZR | Jump to zero row. $AC_0$-$AC_3$ are used to select 1 of 16 column addresses in row$_0$, as the next address. |
| JCR | Jump in current row. $AC_0$-$AC_3$ are used to select 1 of 16 addresses in the current row, specified by $MA_4$-$MA_8$, as the next address. |
| JCE | Jump in current column/row group and enable PR-latch outputs. $AC_0$-$AC_2$ are used to select 1 of 8 row addresses in the current row group, specified by $MA_7$-$MA_8$, as the next row address. The current column is specified by $MA_0$-$MA_3$. The PR-latch outputs are asynchronously enabled. |

## JUMP/TEST FUNCTION TABLE

| MNEMONIC | FUNCTION DESCRIPTION |
|----------|----------------------|
| JFL | Jump/test F-latch. $AC_0$-$AC_3$ are used to select 1 of 16 row addresses in the current row group, specified by $MA_8$, as the next row address. If the current column group, specified by $MA_3$, is col$_0$-col$_7$, the F-latch is used to select col$_2$ or col$_3$ as the next column address. If $MA_3$ specifies column group col$_8$-col$_{15}$, the F-latch is used to select col$_{10}$ or col$_{11}$ as the next column address. |
| JCF | Jump/test C-flag. $AC_0$-$AC_2$ are used to select 1 of 8 row addresses in the current row group, specified by $MA_7$ and $MA_8$, as the next row address. If the current column group specified by $MA_3$ is col$_0$-col$_7$, the C-flag is used to select col$_2$ or col$_3$ as the next column address. If $MA_3$ specifies column group col$_8$-col$_{15}$, the C-flag is used to select col$_{10}$ or col$_{11}$ as the next column address. |
| JZF | Jump/test Z-flag. Identical to the JCF function described above, except that the Z-flag, rather than the C-flag, is used to select the next column address. |
| JPR | Jump/test PR-latch. $AC_0$-$AC_2$ are used to select 1 of 8 row addresses in the current row group, specified by $MA_7$ and $MA_8$, as the next row address. The 4 PR-latch bits are used to select 1 of 16 possible column addresses as the next column address. |
| JLL | Jump/test leftmost PR-latch bits. $AC_0$-$AC_2$ are used to select 1 of 8 row addresses in the current row group, specified by $MA_7$ and $MA_8$, as the next row address. $PR_2$ and $PR_3$ are used to select 1 of 4 column addresses in col$_4$ through col$_7$ as the next column address. |
| JRL | Jump/test rightmost PR-latch bits. $AC_0$ and $AC_1$ are used to select 1 of 4 high-order row addresses in the current row group, specified by $MA_7$ and $MA_8$, as the next row address. $PR_0$ and $PR_1$ are used to select 1 of 4 possible column addresses in col$_{12}$ through col$_{16}$ as the next column address. |
| JPX | Jump/test PX-bus and load PR-latch. $AC_0$ and $AC_1$ are used to select 1 of 4 row addresses in the current row group, specified by $MA_6$-$MA_8$, as the next row address. $PX_4$-$PX_7$ are used to select 1 of 16 possible column addresses as the next column address. $SX_0$-$SX_3$ data is locked in the PR-latch at the rising edge of the clock. |

## PX-Bus and PR-Latch Conditional Address Control (Jump/Test) Functions

The PX-bus jump/test function uses the data on the primary instruction bus ($PX_4$-$PX_7$), the current microprogram address, and several selection bits from the address control function to generate the next microprogram address. The PR-latch jump/test functions use the data held in the PR-latch, the current microprogram address, and several selection bits from the address control function to generate the next microprogram address.

## Flag Control Functions

The flag control functions of the MCU are selected by the 4 input lines designated $FC_0$-$FC_3$. Function code formats are given in "Flag Control Function summary."

The following is a detailed description of each of the 8 flag control functions.

## Flag Input Control Functions

The flag input control functions select which flag or flags will be set to the current value of the flag input ($\overline{FI}$) line.

Data on $\overline{FI}$ is stored in the F-latch when the clock is low. The content of the F-latch is loaded into the C and/or Z flag on the rising edge of the clock.

## Flag Output Control Functions

The flag output control functions select the value to which the flag output ($\overline{FO}$) line will be forced.

## FLAG CONTROL FUNCTION TABLE

| MNEMONIC | FUNCTION DESCRIPTION |
|---|---|
| SCZ | Set C-flag and Z-flag to FI. The C-flag and the Z-flag are both set to the value of FI. |
| STZ | Set Z-flag to FI. The Z-flag is set to the value of FI. The C-flag is unaffected. |
| STC | Set C-flag to FI. The C-flag is set to the value of FI. The Z-flag is unaffected. |
| HCZ | Hold C-flag and Z-flag. The values in the C-flag and Z-flag are unaffected. |

## FLAG OUTPUT CONTROL FUNCTION TABLE

| MNEMONIC | FUNCTION DESCRIPTION |
|---|---|
| FF0 | Force FO to 0. FO is forced to the value of logical 0. |
| FFC | Force FO to C. FO is forced to the value of the C-flag. |
| FFZ | Force FO to Z. FO is forced to the value of the Z-flag. |
| FF1 | Force FO to 1. FO is forced to the value of logical 1. |

## FLAG CONTROL FUNCTION SUMMARY

| TYPE | MNEMONIC | DESCRIPTION | $FC_1$ | 0 |
|---|---|---|---|---|
| Flag Input | SCZ | Set C-flag and Z-flag to f | 0 | 0 |
| | STZ | Set Z-flag to f | 0 | 1 |
| | STC | Set C-flag to f | 1 | 0 |
| | HCZ | Hold C-flag and Z-flag | 1 | 1 |

| TYPE | MNEMONIC | DESCRIPTION | $FC_3$ | 2 |
|---|---|---|---|---|
| Flag Output | FF0 | Force FO to 0 | 0 | 0 |
| | FFC | Force FO to C-flag | 1 | 0 |
| | FFZ | Force FO to Z-flag | 0 | 1 |
| | FF1 | Force FO to 1 | 1 | 1 |

| LOAD FUNCTION | NEXT ROW | | | | NEXT COL | | | |
|---|---|---|---|---|---|---|---|---|
| LD | $MA_8$ | 7 | 6 | 5 | 4 | $MA_3$ | 2 | 1 | 0 |
| 0 | See Address Control Function Summary | | | | | | | |
| 1 | 0 | $X_3$ | $X_2$ | $X_1$ | $X_0$ | $X_7$ | $X_6$ | $X_5$ | $X_4$ |

NOTE
f = Contents of the F-latch     x n = Data on PX- or SX-bus line n (active low)

## ADDRESS CONTROL FUNCTION SUMMARY

| MNEMONIC | DESCRIPTION | FUNCTION | | | | | | | NEXT ROW | | | | | NEXT COL | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | $AC_6$ | 5 | 4 | 3 | 2 | 1 | 0 | $MA_8$ | 7 | 6 | 5 | 4 | $MA_3$ | 2 | 1 | 0 |
| JCC | Jump in current column | 0 | 0 | $d_4$ | $d_3$ | $d_2$ | $d_1$ | $d_0$ | $d_4$ | $d_3$ | $d_2$ | $d_1$ | $d_0$ | $m_3$ | $m_2$ | $m_1$ | $m_0$ |
| JZR | Jump to zero row | 0 | 1 | 0 | $d_3$ | $d_2$ | $d_1$ | $d_0$ | 0 | 0 | 0 | 0 | 0 | $d_3$ | $d_2$ | $d_1$ | $d_0$ |
| JCR | Jump in current row | 0 | 1 | 1 | $d_3$ | $d_2$ | $d_1$ | $d_0$ | $m_8$ | $m_7$ | $m_6$ | $m_5$ | $m_4$ | $d_3$ | $d_2$ | $d_1$ | $d_0$ |
| JCE | Jump in column/enable | 1 | I | 1 | 0 | $d_2$ | $d_1$ | $d_0$ | $m_8$ | $m_7$ | $d_2$ | $d_1$ | $d_0$ | $m_3$ | $m_2$ | $m_1$ | $m_0$ |
| JFL | Jump/test F-latch | 1 | 0 | 0 | $d_3$ | $d_2$ | $d_1$ | $d_0$ | $m_8$ | $d_3$ | $d_2$ | $d_1$ | $d_0$ | $m_3$ | 0 | 1 | f |
| JCF | Jump/test Z-flag | 1 | 0 | 1 | 1 | $d_2$ | $d_1$ | $d_0$ | $m_8$ | $m_7$ | $d_2$ | $d_1$ | $d_0$ | $m_3$ | 0 | 1 | c |
| JPR | Jump/test PR-latch | 1 | 1 | 0 | 0 | $d_2$ | $d_1$ | $d_0$ | $m_8$ | $m_7$ | $d_2$ | $d_1$ | $d_0$ | $m_3$ | 0 | 1 | z |
| JLL | Jump/test left PR bits | 1 | 1 | 0 | 1 | $d_2$ | $d_1$ | $d_0$ | $m_8$ | $m_7$ | $d_2$ | $d_1$ | $d_0$ | $p_3$ | $p_2$ | $p_1$ | $p_0$ |
| JRL | Jump/test right PR bits | 1 | 1 | 1 | 1 | 1 | $d_1$ | $d_0$ | $m_8$ | $m_7$ | 1 | $d_1$ | $d_0$ | 0 | 1 | $p_3$ | $p_2$ |
| JPX | Jump/test PX-bus | 1 | 1 | 1 | 1 | 0 | $d_1$ | $d_0$ | $m_8$ | $m_7$ | $m_6$ | $d_1$ | $d_0$ | $x_7$ | $x_6$ | $x_5$ | $x_4$ |

NOTE

dn = Data on address control line n     Pn = Data in PR-latch bit n
mn = Data in microprogram address register bit n     xn = Data on PX-bus line n (active low)
    f,c,z = Contents of F-latch, C-flag, or Z-flag, respectively

## STROBE FUNCTIONS

The load function of the MCU is controlled by the input line designated LD. If the LD line is active high at the rising edge of the clock, the data on the primary and secondary instruction buses, $PX_4$-$PX_7$ and $SX_0$-$SX_3$, is loaded into the microprogram address register. $PX_4$-$PX_7$ are loaded into $MA_0$-$MA_7$ and $SX_0$-$SX_3$ are loaded into $MA_4$-$MA_7$. The high-order bit of the microprogram address register $MA_8$ is set to a logical 0. The bits from the primary instruction bus select 1 of 16 possible column addresses. Likewise, the bits from the secondary instruction bus select 1 of the first 16 row addresses.

The MCU generates an interrupt strobe enable on the output line designated ISE. The line is placed in the active high state whenever a JZR to $col_{15}$ is selected as the address control function. Generally, the start of a macroinstruction fetch sequence is situated at $row_0$ and $col_{15}$ so the interrupt control may be enabled at the beginning of the fetch/execute cycle. The interrupt control responds to the interrupt by pulling the enable row address (ERA) input line low to override the selected next row address from the MCU. Then by gating an alternative next row address on to the row address lines of the microprogram memory, the microprogram may be forced to enter an interrupt handling routine. The alternative row address placed on the microprogram memory address lines does not alter the contents of the microprogram address register. Therefore, subsequent jump functions will utilize the row address in the register, and not the alternative row address, to determine the next microprogram address.

Note, the load function always overrides the address control function on $AC_0$-$AC_6$. It does not, however, override the latch enable or load sub-functions of the JCE or JPX instruction, respectively. In addition, it does not inhibit the interrupt strobe enable or any of the flag control functions.

## JUMP SET DIAGRAMS

The following 10 diagrams illustrate the jump set for each of the 11 jump and jump/test functions of the MCU. Location 341 indicated by the circled square, represents 1 current row ($row_{21}$) and current column ($col_5$) address. The dark boxes indicate the microprogram locations that may be selected by the particular function as the next address.

## JUMP SET DIAGRAMS



JCC
JUMP IN CURRENT COLUMN

JZR
JUMP TO ZERO ROW

JPR
JUMP/TEST PR-LATCH

JLL
JUMP/TEST LEFT LATCH

**JUMP SET DIAGRAMS** (Cont'd)

### JRL
### JUMP/TEST RIGHT LATCH

CURRENT
ROW
GROUP
$M_{8\,7\,6}$
$1\,0\,1$

$COL_{12}$  $COL_{15}$

### JPX
### JUMP/TEST PX-BUS

CURRENT
ROW
GROUP
$M_{8\,7\,6}$
$1\,0\,1$

### JCR
### JUMP IN CURRENT ROW

$ROW_0$

CURRENT
ROW

$ROW_{31}$

$COL_0$  $COL_{15}$

### JCE
### JUMP COLUMN/ENABLE

CURRENT
ROW
GROUP
$M_{8\,7}$
$1\,0$

CURRENT COLUMN

### JFL
### JUMP/TEST F-LATCH

CURRENT
COLUMN
GROUP
$M_3 = 0$

CURRENT
ROW
GROUP
$M_8$
$1$

$COL_3$ (f = 1)
$COL_2$ (f = 0)

### JCF, JZF
### JUMP/TEST C-FLAG
### JUMP/TEST Z-FLAG

CURRENT
COLUMN
GROUP
$M_3 = 0$

CURRENT
ROW
GROUP
$M_{8\,7}$
$1\,0$

$COL_3$ (c,z = 1)
$COL_2$ (c,z = 0)

N3001 $T_A$ = 0°C to +70°C, $V_{CC}$ = 5.0V, ± 5%                N3001-I

## AC ELECTRICAL CHARACTERISTICS S3001 $T_A$ = -55°C to +125°C, $V_{CC}$ = 5.0V ± 10%

| PARAMETER | N3001 | | | S3001 | | | UNIT |
|---|---|---|---|---|---|---|---|
| | Min | Typ[1] | Max | Min | Typ[1] | Max | |
| $t_{CY}$ Cycle Time[2] | 60 | 45 | | 95 | 45 | | ns |
| $t_{PW}$ Clock Pulse Width | 17 | 10 | | 40 | 10 | | ns |
| Control and Data Input Set-Up Times: | | | | | | | |
| $t_{SF}$ LD, $AC_0$-$AC_6$ (Set to "1"/"0") | 20 | 3/14 | | 20 | 3/14 | | ns |
| $t_{SK}$ $FC_0$, $FC_1$ | 7 | 5 | | 10 | 5 | | ns |
| $t_{SX}$ $PX_4$-$PX_7$ (Set to "1"/"0") | 28 | 4/13 | | 35 | 4/13 | | ns |
| $t_{SI}$ FI (Set to "1"/"0") | 12 | –6/0 | | 15 | –6/10 | | ns |
| $t_{SX}$ $SX_0$-$SX_3$ | 15 | 5 | | 35 | 5 | | ns |
| Control and Data Input Hold Times: | | | | | | | |
| $t_{HF}$ LD, $AC_0$-$AC_6$ (Hold to "1"/"0") | 4 | –3/–14 | | 5 | –3/–14 | | ns |
| $t_{HK}$ $FC_0$, $FC_1$ | 4 | –5 | | 10 | –5 | | ns |
| $t_{HX}$ $PX_4$-$PX_7$ (Hold to "1"/"0") | 0 | –4/–13 | | 25 | –4/–13 | | ns |
| $t_{HI}$ FI (Hold to "1"/"0") | 16 | 6.5/0 | | 22 | 6.5/0 | | ns |
| $t_{HX}$ $SX_0$-$SX_3$ | 0 | –5 | | 25 | –5 | | ns |
| $t_{CO}$ Propagation Delay from Clock Input (CLK) to Outputs ($mA_0$-$mA_8$, FO) ($t_{PHL}$/$t_{PLH}$) | | 17/24 | 36 | 10 | 17/24 | 45 | ns |
| $t_{KO}$ Propagation Delay from Control Inputs $FC_2$ and $FC_3$ to Flag Out (FO) | | 13 | 24 | | 13 | 50 | ns |
| $t_{FO}$ Propagation Delay from Control Inputs $AC_0$-$AC_6$ to Latch Outputs ($PR_0$-$PR_2$) | | 21 | 32 | | 21 | 50 | ns |
| $t_{EO}$ Propagation Delay from Enable Inputs EN and ERA to Outputs ($mA_0$-$mA_8$, FO, $PR_0$-$PR_2$) | | 17 | 26 | | 17 | 35 | ns |
| $t_{FI}$ Propagation Delay from Control Inputs $AC_0$-$AC_6$ to Interrupt Strobe Enable Output (ISE) | | 20 | 32 | | 20 | 40 | ns |

NOTE

1. Typical values are for $T_A$ = 25°C and 5.0 supply voltage.
2. S3001: $t_{CY}$ = $t_{WP}$ + $t_{SF}$ + $t_{CO}$

**VOLTAGE WAVEFORMS**

CLK
CLOCK INPUT

$t_{CY}$    $t_{PW}$

EN, ERA
ENABLE INPUTS

$t_{CO}$
$t_{EO}$

$MA_0-MA_8$
CONTROL MEMORY
ADDRESS OUTPUTS

$t_{HF}$    $t_{SF}$

$AC_0-AC_6$, LD
ADDRESS CONTROL
INPUTS

$t_{EO}$
$t_{FO}$

$PR_0-PR_2$
"PR" LATCH OUTPUTS

$t_{HK}$    $t_{SK}$

$FC_0-FC_3$
FLAG CONTROL
INPUTS

$t_{HI}$    $t_{SI}$

FI
FLAG INPUT

$t_{KO}$
$t_{EO}$
$t_{CO}$

FO
FLAG OUTPUT

$t_{FI}$

ISE
INTERRUPT STROBE
ENABLE OUTPUT

$t_{HX}$    $t_{SX}$

$SX_0-PX_7$
INSTRUCTION
BUS INPUTS
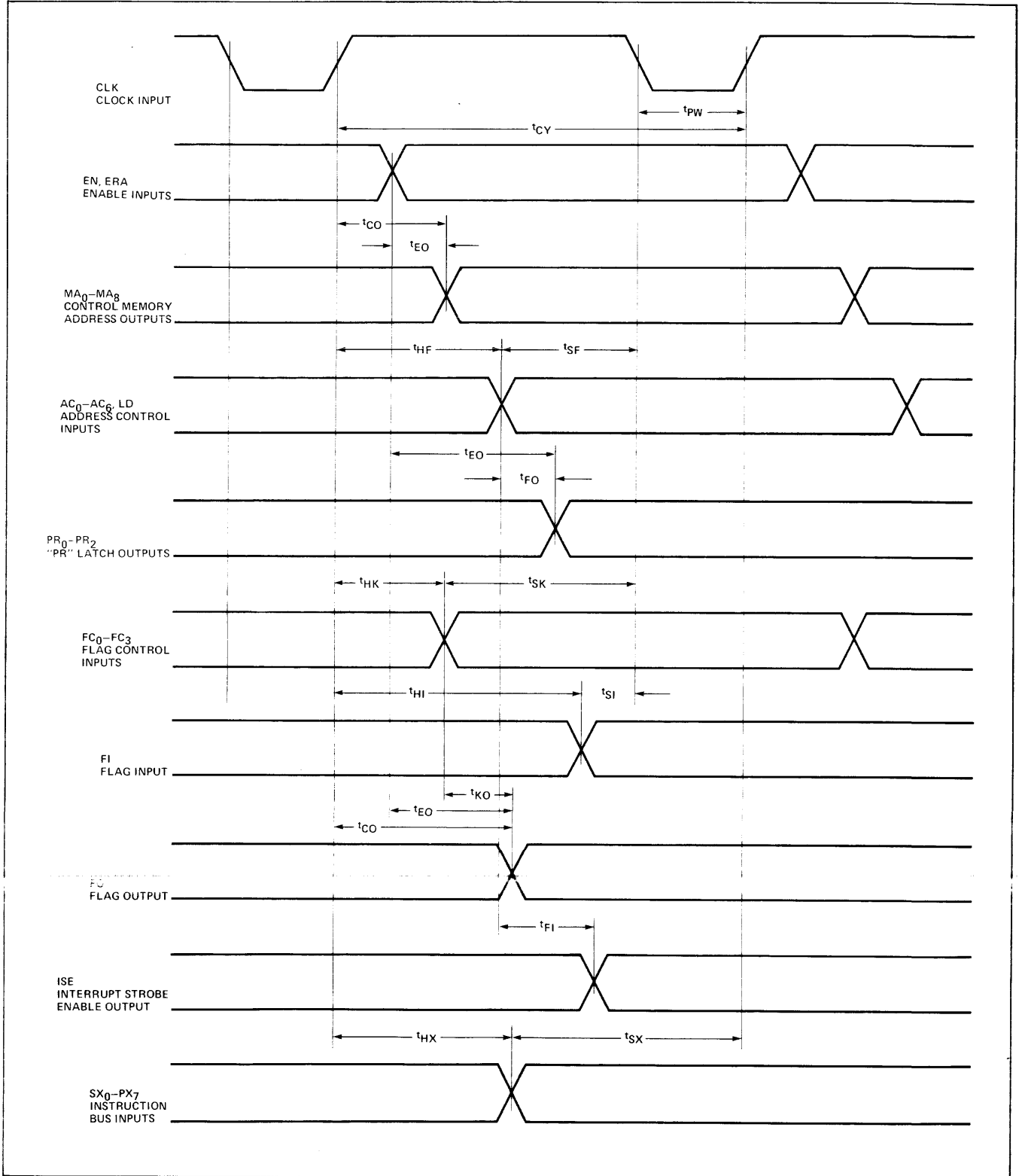
## DESCRIPTION

The N3002 Central Processing Element (CPE) is one part of a bipolar microcomputer set. The N3002 is organized as a 2-bit slice and performs the logical and arithmetic functions required by microinstructions. A system with any number of bits in a data word can be implemented by using multiple N3002s, the N3001 microcomputer control unit, the N74S182 carry look-ahead unit and ROM or PROM memory.

## FEATURES

- **45ns cycle time (typ)**
- **Easy expansion to multiple of 2 bits**
- **11 general purpose registers**
- **Full function accumulator**
- **Useful functions include:**
  - **2's complement arithmetic**
  - **Logical AND, OR, NOT, exclusive-NOR**
  - **Increment, decrement**
  - **Shift left/shift right**
  - **Bit testing and zero detection**
  - **Carry look-ahead generation**
  - **Masking via K-bus**
  - **Conditioned clocking allowing non-destructive testing of data in accumulator and scratchpad**
- **3 input buses**
- **2 output buses**
- **Control bus**

## FUNCTION TRUTH TABLE

| FUNCTION GROUP | $F_6$ | $F_5$ | $F_4$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 |
| 2 | 0 | 1 | 0 |
| 3 | 0 | 1 | 1 |
| 4 | 1 | 0 | 0 |
| 5 | 1 | 0 | 1 |
| 6 | 1 | 1 | 0 |
| 7 | 1 | 1 | 1 |

| REGISTER GROUP | REGISTER | $F_3$ | $F_2$ | $F_1$ | $F_0$ |
|---|---|---|---|---|---|
| I | $R_0$ | 0 | 0 | 0 | 0 |
| | $R_1$ | 0 | 0 | 0 | 1 |
| | $R_2$ | 0 | 0 | 1 | 0 |
| | $R_3$ | 0 | 0 | 1 | 1 |
| | $R_4$ | 0 | 1 | 0 | 0 |
| | $R_5$ | 0 | 1 | 0 | 1 |
| | $R_6$ | 0 | 1 | 1 | 0 |
| | $R_7$ | 0 | 1 | 1 | 1 |
| | $R_8$ | 1 | 0 | 0 | 0 |
| | $R_9$ | 1 | 0 | 0 | 1 |
| | T | 1 | 1 | 0 | 0 |
| | AC | 1 | 1 | 0 | 1 |
| II | T | 1 | 0 | 1 | 0 |
| | AC | 1 | 0 | 1 | 1 |
| III | T | 1 | 1 | 1 | 0 |
| | AC | 1 | 1 | 1 | 1 |

## PIN CONFIGURATION

**XL,I PACKAGE**

| | | | |
|---|---|---|---|
| $\overline{I_0}$ | 1 | 28 | $V_{CC}$ |
| $\overline{I_1}$ | 2 | 27 | $F_2$ |
| $\overline{K_0}$ | 3 | 26 | $F_1$ |
| $\overline{K_1}$ | 4 | 25 | $F_0$ |
| X | 5 | 24 | $F_3$ |
| Y | 6 | 23 | $\overline{ED}$ |
| $\overline{CO}$ | 7 | 22 | $\overline{M_0}$ |
| $\overline{RO}$ | 8 | 21 | $\overline{M_1}$ |
| $\overline{LI}$ | 9 | 20 | $\overline{D_1}$ |
| $\overline{CI}$ | 10 | 19 | $\overline{D_0}$ |
| $\overline{EA}$ | 11 | 18 | $\overline{CLK}$ |
| $\overline{A_1}$ | 12 | 17 | $F_4$ |
| $\overline{A_0}$ | 13 | 16 | $F_5$ |
| GND | 14 | 15 | $F_6$ |
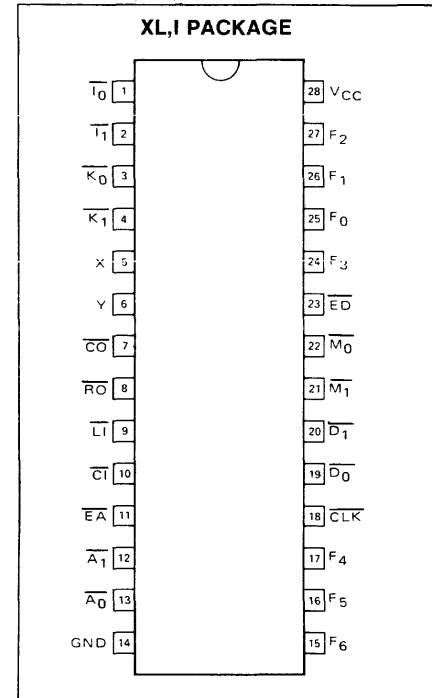
## BLOCK DIAGRAM
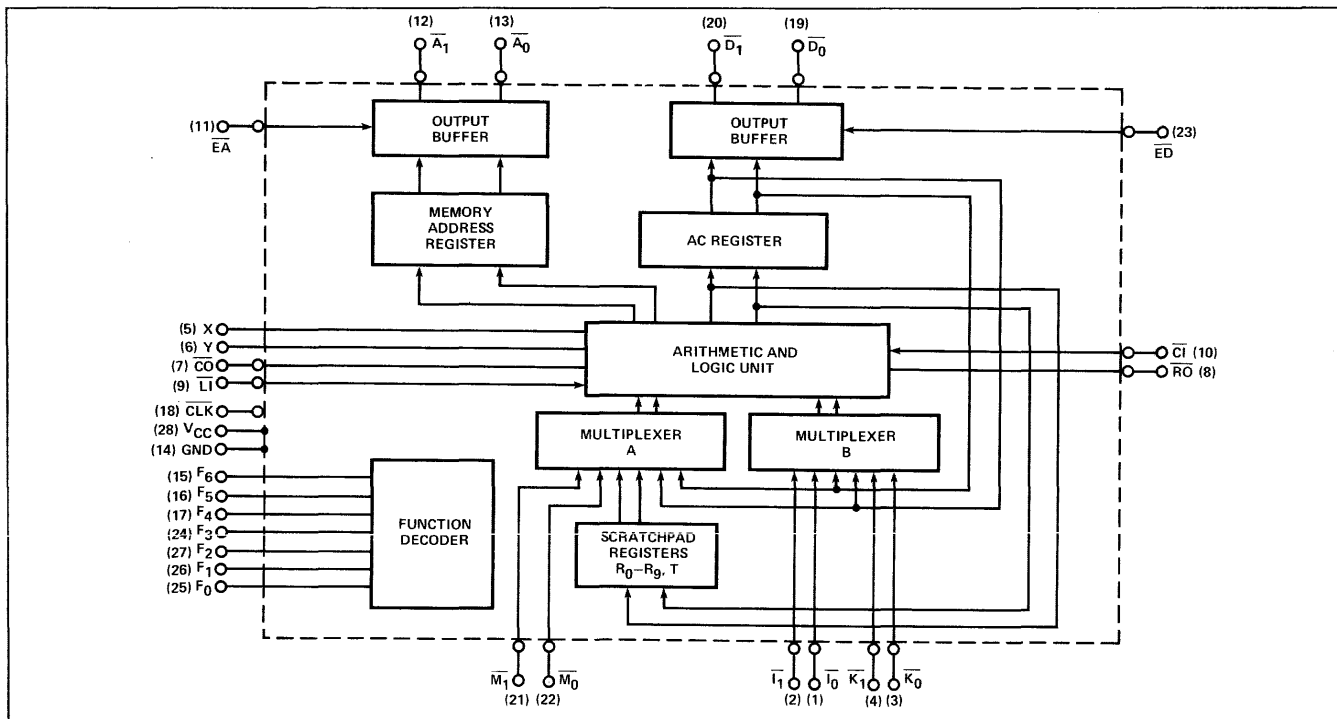
## PIN DESCRIPTION

| PIN | SYMBOL | NAME AND FUNCTION | TYPE |
|---|---|---|---|
| 1, 2 | $\overline{I_0}$-$\overline{I_1}$ | External Bus Inputs<br>The external bus inputs provide a separate input port for external input devices. | Active low |
| 3, 4 | $\overline{K_0}$-$\overline{K_1}$ | Mask Bus Inputs<br>The mask bus inputs provide a separate input port from the microprogram memory, to allow mask or constant entry. | Active low |
| 5, 6 | X, Y | Standard Carry Look-Ahead Cascade Outputs<br>The cascade outputs allow high speed arithmetic operations to be performed when they are used in conjunction with the 74S182 Look-Ahead Carry Generator | Active high |
| 7 | $\overline{CO}$ | Ripple Carry Out<br>The ripple carry output is only disabled during shift right operations. | Active low<br>Three-state |
| 8 | $\overline{RO}$ | Shift Right Output<br>The shift right output is only enabled during shift right operations. | Active low<br>Three-state |
| 9 | $\overline{LI}$ | Shift Right Input | Active low |
| 10 | $\overline{CI}$ | Carry Input | Active low |
| 11 | $\overline{EA}$ | Memory Address Enable Input<br>When in the low state, the memory address enable input enables the memory address outputs ($A_0$-$A_1$). | Active low |
| 12-13 | $\overline{A_0}$-$\overline{A_1}$ | Memory Address Bus Outputs<br>The memory address bus outputs are the buffered outputs of the memory address register (MAR). | Active low<br>Three-state |
| 14 | GND | Ground | |
| 14-17,<br>24-27 | $F_0$-$F_6$ | Micro-Function Bus Inputs<br>The micro-function bus inputs control ALU function and register selection. | Active high |
| 18 | $\overline{CLK}$ | Clock Input | |
| 19-20 | $\overline{D_0}$-$\overline{D_1}$ | Memory Data Bus Outputs<br>The memory data bus outputs are the buffered outputs of the full function accumulator register (AC). | Active low<br>Three-state |
| 21-22 | $\overline{M_0}$-$\overline{M_1}$ | Memory Data Bus Inputs<br>The memory data bus inputs provide a separate input port for memory data. | Active low |
| 23 | $\overline{ED}$ | Memory Data Enable Input<br>When in the low state, the memory data enable input enables the memory data outputs ($D_0$-$D_1$). | Active low |
| 28 | $V_{CC}$ | +5 Volt Supply | |

## SYSTEM DESCRIPTION

### Microfunction Decoder and K-Bus

Basic microfunctions are controlled by a 7-bit bus ($F_0$-$F_6$) which is organized into 2 groups. The higher 3 bits ($F_4$-$F_6$) are designated as F-Group and the lower 4 bits ($F_0$-$F_3$) are designated as the R-Group. The F-Group specifies the type of operation to be performed and the R-Group specifies the registers involved.

The F-Bus instructs the microfunction decoder to:
• Select ALU functions to be performed
• Generate scratchpad register address
• Control A and B multiplexer

The resulting microfunction action can be:
• Data transfer
• Shift operations
• Increment and decrement
• Initialize stack
• Test for zero conditions
• 2's complement addition and subtraction
• Bit masking
• Maintain program counter

### A and B Multiplexers

A and B multiplexers select the proper 2 operands to the ALU.

A multiplexer selects inputs from one of the following:
• M-bus (data from main memory)
• Scratchpad registers
• Accumulator

B multiplexer selects inputs from one of the following:
• I-bus (data from external I/O devices)
• Accumulator
• K-bus (literal or masking information from micro-program memory)

### Scratchpad Registers

• Contains 11 registers ($R_0$-$R_9$, T)
• Scratchpad register outputs are multiplexed to the ALU via the A multiplexer
• Used to store intermediate results from arithmetic/logic operations
• Can be used as program counter

### Arithmetic/Logic Unit (ALU)

The ALU performs the arithmetic and logic operations of the CPE.

Arithmetic operations are:
• 2's complement addition
• Incrementing
• Decrementing
• Shift left
• Shift right

Logical operations are:
• Transfer
• AND
• Inclusive-OR
• Exclusive-NOR
• Logic complement

ALU operation results are then stored in the accumulator and/or scratchpad registers. For easy expansion to larger arrays, carry look-ahead outputs (X and Y) and cascading shift inputs (LI, RO) are provided.

## Accumulator

- Stores results from ALU operations
- The output of accumulator is multiplexed into ALU via the A and B multiplexer as one of the operands

## Input Buses

M-bus: Data bus from main memory
- Accepts 2 bits of data from main memory into CPE
- Is multiplexed into the ALU via the A multiplexer

I-bus: Data bus from input/output devices

- Accepts 2 bits of data from external input/output devices into CPE
- Is multiplexed into the ALU via the B multiplexer

K-bus: A special feature of the N3002 CPE
- During arithmetic operations, the K-bus can be used to **mask** portions of the field being operated on
- Select or remove accumulator from operation by placing K-bus in all "1" or all "0" state respectively
- During non-arithmetic operation, the carry circuit can be used in conjunction with the K-bus for word-wise-OR operation for bit testing
- Supply literal or constant data to CPE

## Output Buses

A-bus and Memory Address Register
- Main memory address is stored in the memory address register (MAR)
- Main memory is addressed via the A-bus
- MAR and A-bus may also be used to generate device address when executing I/OO instruc- instructions
- A-bus has Tri-State outputs

D-bus: Data bus from CPE to main memory or to I/O devices
- Sends buffered accumulator outputs to main memory or the external I/O devices
- D-bus has Tri-State outputs

## FUNCTION DESCRIPTION

| F GROUP | R GROUP | K BUS | NAME | EQUATION | DESCRIPTION |
|---|---|---|---|---|---|
| 0 | I | XX | — | $R_n + (AC \wedge K) + CI \rightarrow R_n, AC$ | **Logically AND AC with the K-bus.** Add the result to $R_n$ and carry input (CI). Deposit the sum in AC and $R_n$. |
| | | OO | ILR | $R_n + CI \rightarrow R , AC$ | Conditionally increment $R_n$ and load the result in AC. Used to load AC from $R_n$ or to increment $R_n$ and load a copy of the result in AC. |
| | | 11 | ALR | $AC + R_n + CI \rightarrow R_n, AC$ | Add AC and CI to $R_n$ and load the result in AC. Used to add AC to a register. If $R_n$ is AC, then AC is shifted left one bit position. |
| 0 | II | XX | — | $M + (AC \wedge K) + CI \rightarrow AT$ | **Logically AND AC with the K-bus.** Add the result to CI and the M-bus. Deposit the sum in AC or T. |
| | | OO | ACM | $M + CI \rightarrow AT$ | Add CI to M-bus. Load the result in AC or T, as specified. Used to load memory data in the specified register, or to load incremented memory data in the specified register. |
| | | 11 | AMA | $M + AC + CI \rightarrow AT$ | Add the M-bus to AC and CI, and load the result in AC or T, as specified. Used to add memory data or incremented memory data to AC and store the sum in the specified register. |
| 0 | III | XX | — | $AT_L \wedge (\overline{I_L \wedge K_L}) \rightarrow RO$ <br> $LI \vee [((I_H \wedge K_H) \wedge AT_H] \rightarrow AT_H$ <br> $[AT_L \wedge (I_L \wedge K_L)]$ <br> $[AT_H \vee (I_H \wedge K_H)] \rightarrow AT_L$ | None |
| | | OO | SRA | $AT_L \rightarrow RO$ <br> $AT_H \rightarrow AT_L$ <br> $L_I \rightarrow AT_H$ | Shift AC or T, as specified, right one bit position. Place the previous low order bit value on RO and fill the high order bit from the data on LI. Used to shift or rotate AC or T right one bit. |
| 1 | I | XX | — | $K \vee R_n \rightarrow MAR$ <br> $R_n + K + CI \rightarrow R_n$ | **Logically OR $R_n$ with the K-bus.** Deposit the result in MAR. Add the K-bus to Rn and CI. Deposit the result in $R_n$. |
| | | OO | LMI | $Rn \rightarrow MAR, Rn + CI \rightarrow Rn$ | Load MAR from $R_n$. Conditionally increment $R_n$. Used to maintain a macro-instruction program counter. |
| | | 11 | DSM | $11 \rightarrow MAR, Rn - 1 + CI \rightarrow Rn$ | Set MAR to all ones. Conditionally decrement $R_n$ by one. Used to force MAR to its highest address and to decrement Rn. |
| 1 | II | XX | — | $KVM \rightarrow MAR$ <br> $M + K + CI \rightarrow AT$ | **Logically OR the M-bus with the K-Bus.** Deposit the result in MAR. Add the K-bus to the M-bus and CI. Deposit the sum in AC or T. |
| | | OO | LMM | $M \rightarrow MAR, M + CI \rightarrow AT$ | Load MAR from the M-bus. Add CI to the M-bus. Deposit the result in AC or T. Used to load the address register with memory data for macro-instructions using indirect addressing. |
| | | 11 | LDM | $11 \rightarrow MAR$ <br> $M - 1 + CI \rightarrow AT$ | Set MAR to all ones. Subtract one from the M-bus. Add CI to the difference and deposit the result in AC or T, as specified. Used to load decremented memory data in AC or T. |

## FUNCTION DESCRIPTION (Cont'd)

| F GROUP | R GROUP | K BUS | NAME | EQUATION | DESCRIPTION |
|---|---|---|---|---|---|
| 1 | III | XX | — | $(\overline{AT} \lor K) + (AT \land K) + CI \rightarrow AT$ | **Logically OR the K-bus with the complement of AC or T, as specified.** Add the result to the logical AND of specified register with the K-bus. Add the sum to CI. Deposit the result in the specified register. |
| | | OO | CIA | $\overline{AT} + CI \rightarrow AT$ | Add CI to the complement of AC or T, as specified. Deposit the result in the specified register. Used to form the 1's or 2's complement of AC or T. |
| | | 11 | DCA | $\overline{AT} - 1 + CI \rightarrow AT$ | Subtract one from AC or T, as specified. Add CI to the difference and deposit the sum in the specified register. Used to decrement AC or T. |
| 2 | I | XX | — | $(AC \land K) - 1 + CI \rightarrow R_n$ | **Logically AND the K-bus with AC.** Subtract one from the result and add the difference to CI. Deposit the sum in $R_n$. |
| | | OO | CSR | $CI - 1 \rightarrow R_n$ (See Note 1) | Subtract one from CI and deposit the difference in Rn. Used to conditionally clear or set $R_n$ to all 0's or 1's, respectively. |
| | | 11 | SDR | $AC - 1 + CI \rightarrow R_n$ (See Note 1) | Subtract one from AC and add the difference to CI. Deposit the sum in $R_n$. Used to store AC in $R_n$ or to store the decremented value of AC in $R_n$. |
| 2 | II | XX | — | $(AC \land K) - 1 + CI \rightarrow AT$ (See Note 1) | **Logically AND the K-bus with AC.** Subtract one from the result and add the difference to CI. Deposit the sum in AC or T, as specified. |
| | | OO | CSA | $CI - 1 \rightarrow AT$ (See Note 1) | Subtract one from CI and deposit the difference in AC or T. Used to conditionally clear or set AC or T. |
| | | 11 | SDA | $AC - 1 + CI \rightarrow AT$ (See Note 1) | Subtract one from AC and add the difference to CI. Deposit the sum in AC or T. Used to store AC in T, or decrement AC, or store the decremented value of AC in T. |
| 2 | III | XX | — | $(I \land K) - 1 + CI \rightarrow AT$ (See Note 1) | **Logically AND the data of the K-bus with the data on the I-bus.** Subtract one from the result and add the difference to CI. Deposit the sum in AC or T, as specified. |
| | | OO | CSA | $CI - 1 \rightarrow AT$ | Subtract one from CI and deposit the difference in AC or T. Used to conditionally clear or set AC or T. |
| | | 11 | LDI | $I - 1 + CI \rightarrow AT$ | Subtract one from the data on the I-bus and add the difference to CI. Deposit the sum in AC or T, as specified. Used to load input bus data or decremented input bus data in the specified register. |
| 3 | I | XX | — | $R_n + (AC \land K) + CI \rightarrow R_n$ | **Logically AND AC with the K-bus.** Add $R_n$ and CI to the result. Deposit the sum in $R_n$. |
| | | OO | INR | $R_n + CI \rightarrow R_n$ | Add CI to $R_n$ and deposit the sum in $R_n$. Used to increment $R_n$. |
| | | 11 | ADR | $AC + R_n + CI \rightarrow R_n$ | Add AC to $R_n$. Add the result to CI and deposit the sum in $R_n$. Used to add the accumulator to a register or to add the incremented value of the accumulator to a register. |
| 3 | II | XX | — | $M + (AC \land K) + CI \rightarrow AT$ | **Logically AND AC with the K-bus.** Add the result to CI and the M-bus. Deposit the sum in AC or T. |
| | | OO | ACM | $M + CI \rightarrow AT$ | Add CI to M-bus. Load the result in AC or T, as specified. Used to load memory data in the specified register, or to load incremented memory data in the specified register. |
| | | 11 | AMA | $M + AC + CI \rightarrow AT$ | Add the M-bus to AC and CI, and load the result in AC or T, as specified. Used to add memory data or incremented memory data to AC and store the sum in the specified register. |

## FUNCTION DESCRIPTION (Cont'd)

| F GROUP | R GROUP | K BUS | NAME | EQUATION | DESCRIPTION |
|---------|---------|-------|------|----------|-------------|
| 3 | III | XX | — | $AT + (I \wedge K) + CI \rightarrow AT$ | **Logically AND the K-bus with the I-bus.** Add CI and the contents of AC or T, as specified, to the result. Deposit the sum in the specified register. |
|   |     | OO | INA | $AT + CI \rightarrow AT$ | Conditionally increment AC or T. Used to increment AC or T. |
|   |     | 11 | AIA | $I + AT + CI \rightarrow AT$ | Add the I-bus to AC or T. Add CI to the result and deposit the sum in the specified register. Used to add input data or incremented input data to the specified register. |
| 4 | I | XX | — | $CI \vee (R_n \wedge AC \wedge K) \rightarrow CO$ <br> $R_n \wedge (AC \wedge K) \rightarrow R_n$ | **Logically AND the K-bus with AC.** Logically AND the result with the contents of $R_n$. Deposit the final result in $R_n$. Logically OR the value of CI with the word-wise OR of the bits of the final result. Place the value of the carry OR on the carry output (CO) line. |
|   |   | OO | CLR | $CI \rightarrow CO, \quad 0 \rightarrow R_n$ | Clear $R_n$ to all O's. Force CO to CI. Used to clear a register and force CO to CI. |
|   |   | 11 | ANR | $CI \vee (R_n \wedge AC) \rightarrow CO$ <br> $R_n \wedge AC \rightarrow R_n$ | Logically AND AC with $R_n$. Deposit the result in $R_n$. Force CO to one if the result is non-zero. Used to AND the accumulator with a register and test for a zero result. |
| 4 | II | XX | — | $CI \vee (M \wedge AC \wedge K) \rightarrow CO$ <br> $M \wedge (AC \wedge K) \rightarrow AT$ | **Logically AND the K-bus with AC.** Logically AND the result with the M-bus. Deposit the final result in AC or T. Logically OR the value of CI with the word-wise OR of the bits of the final result. Place the value of the carry OR on CO. |
|   |   | OO | CLA | $CI \rightarrow CO, \quad 0 \rightarrow AT$ | Clear AC or T, as specified, to all O's. Force CO to CI. Used to clear the specified register and force CO to CI. |
|   |   | 11 | ANM | $CI \vee (M \wedge AC) \rightarrow CO$ <br> $M \wedge AC \rightarrow AT$ | Logically AND the M-bus with AC. Deposit the result in AC or T. Force CO to one if the result is non-zero. Used to AND M-bus data to the accumulator and test for a zero result. |
| 4 | III | XX | — | $CI \vee (AT \wedge 1 \wedge K) \rightarrow CO$ <br> $AT \wedge (I \wedge K) \rightarrow AT$ | **Logically AND the I-bus with the K-bus.** Logically AND the result with AC or T. Deposit the final result in the specified register. Logically OR CI with the word-wise OR of the final result. Place the value of the carry OR on CO. |
|   |   | OO | CLA | $CI \rightarrow CO, \quad 0 \rightarrow AT$ | Clear AC or T, as specified, to all O's. Force CO to CI. Used to clear the specified register and force CO to CI. |
|   |   | 11 | ANI | $CI \vee (AT \wedge I) \rightarrow CO$ <br> $AT \wedge 1 \rightarrow AT$ | Logically AND the I-bus with AC or T, as specified. Deposit the result in the specified register. Force CO to one if the result is non-zero. Used to AND the I-bus to the accumulator and test for a zero result. |
| 5 | I | XX | — | $CI \vee (R_n \wedge K) \rightarrow CO$ <br> $K \wedge R_n \rightarrow R_n$ | **Logically AND the K-bus with $R_n$.** Deposit the result in $R_n$. Logically OR CI with the word-wise OR of the result. Place the value of the carry OR on CO. |
|   |   | OO | CLR | $CI \rightarrow CO, \quad 0 \rightarrow R_n$ | Clear Rn to all O's. Force CO to CI. Used to clear a register and force CO to CI. |
|   |   | 11 | TZR | $CI \vee R_n \rightarrow CO$ <br> $R_n \rightarrow R_n$ | Force CO to one if Rn is non-zero. Used to test a register for zero. Also used to AND K-bus data with a register for masking and, optionally, testing for a zero result. |
| 5 | II | XX | — | $CI \vee (M \wedge K) \rightarrow CO$ <br> $K \wedge M \rightarrow AT$ | **Logically AND the K-bus with the M-bus.** Deposit the result in AC or T, as specified. Logically OR CI with the word-wise OR of the result. Place the value of the carry OR on CO. |
|   |   | OO | CLA | $CI \rightarrow CO, \quad 0 \rightarrow AT$ | Clear AC or T, as specified, to all O's. Force CO to CI. Used to clear the specified register and force CO to CI. |
|   |   | 11 | LTM | $CI \vee M \rightarrow CO$ <br> $M \rightarrow AT$ | Load AC or T, as specified, from the M-bus. Force CO to one if the result is non-zero. Used to load the specified register from memory and test for a zero result. Also used to AND the K-bus with the M-bus for masking and, optionally, testing for a zero result. |

## FUNCTION DESCRIPTION (Cont'd)

| F GROUP | R GROUP | K BUS | NAME | EQUATION | DESCRIPTION |
|---|---|---|---|---|---|
| 5 | III | XX | — | $CI \vee (AT \wedge K) \rightarrow CO$ <br> $K \wedge AT \rightarrow AT$ | Logically AND the K-bus with AC or T, as specified. Deposit the result in the specified register. Logically OR CI with the word-wise OR of the result. Place the value of the carry OR on CO. |
|  |  | OO | CLA | $CI \rightarrow CO, \; O \rightarrow AT$ | Clear AC or T, as specified, to all O's. Force CO to CI. Used to clear the specified register and force CO to CI. |
|  |  | 11 | TZA | $CI \vee AT \rightarrow CO$ <br> $AT \rightarrow AT$ | Force CO to one if AC or T, as specified, is non-zero. Used to test the specified register for zero. Also used to AND the K-bus to the specified register for masking and, optionally, testing for a zero result. |
| 6 | I | XX | — | $CI \vee (AC \wedge K) \rightarrow CO$ <br> $R_n \vee (AC \wedge K) \rightarrow R_n$ | Logically OR CI with the word-wise OR of the logical AND of AC and the K-bus. Place the result of the carry OR on CO. Logically OR $R_n$ with the logical AND of AC and the K-bus. Deposit the result in $R_n$. |
|  |  | OO | NOP | $CI \rightarrow CO, \; R_n \rightarrow R_n$ | Force CO to CI. Used as a null operation or to force CO to CI. |
|  |  | 11 | ORR | $CI \vee AC \rightarrow CO$ <br> $R_n \vee AC \rightarrow R_n$ | Force CO to one if AC is non-zero. Logically OR AC with $R_n$. Deposit the result in $R_n$. Used to OR the accumulator to a register and, optionally, test the previous accumulator value for zero. |
| 6 | II | XX | — | $CI \vee (AC \wedge K) \rightarrow CO$ <br> $M \vee (AC \wedge K) \rightarrow AT$ | Logically OR CI with the word-wise OR of the logical AND of AC and the K-bus. Place the carry OR on CO. Logically OR the M-bus, with the logical AND of AC and the K-bus. Deposit the final result in AC or T. |
|  |  | OO | LMF | $CI \rightarrow CO, \; M \rightarrow AT$ | Load AC or T, as specified, from the M-bus. Force CO to CI. Used to load the specified register with memory data and force CO to CI. |
|  |  | 11 | ORM | $CI \vee AC \rightarrow CO$ <br> $M \vee AC \rightarrow AT$ | Force CO to one if AC is non-zero. Logically OR the M-bus with AC. Deposit the result in AC or T, as specified. Used to OR M-bus with the AC and, optionally, test the previous value of AC for zero. |
| 6 | III | XX | — | $CI \vee (I \wedge K) \rightarrow CO$ <br> $AT \vee (I \wedge I) \rightarrow AT$ | Logical OR CI with the word-wise OR of the logical AND of the I-bus and the K-bus. Place the carry OR on CO. Logically AND the K-bus with the I-bus. Logically OR the result with AC or T, as specified. Deposit the final result in the specified register. |
|  |  | OO | NOP | $CI \rightarrow CO, \; AT \rightarrow AT$ | Force CO to CI. Used as a null operation or to force CO to CI. |
|  |  | 11 | ORI | $CI \vee I \rightarrow CO$ <br> $I \vee AT \rightarrow AT$ | Force CO to one if the data on the I-bus is non-zero. Logically OR the I-bus to AC or T, as specified. Deposit the result in the specified register. Used to OR I-bus data with the specified register and, optionally, test the I-bus data for zero. |
| 7 | I | XX | — | $CI \vee (R_n \wedge AC \wedge K) \rightarrow CO$ <br> $R_n \oplus (AC \wedge K) \rightarrow R_n$ | Logically OR CI with the word-wise OR of the logical AND of $R_n$ and AC and the K-bus. Place the carry OR on CO. Logically AND the K-bus with AC. Exclusive-NOR the result with $R_n$. Deposit the final result in $R_n$. |
|  |  | OO | CMR | $CI \rightarrow CO, \; Rn \rightarrow Rn$ | Complement the contents of $R_n$. Force CO to CI. |
|  |  | 11 | XNR | $CI \vee (R_n \wedge AC) \rightarrow CO$ <br> $R_n \oplus AC \rightarrow R_n$ | Force CO to one if the logical AND of AC and $R_n$ is non-zero. Exclusive-NOR AC with $R_n$. Deposit the result in $R_n$. Used to exclusive-NOR the accumulator with a register. |

Signetics

## FUNCTION DESCRIPTION (Cont'd)

| F GROUP | R GROUP | K BUS | NAME | EQUATION | DESCRIPTION |
|---------|---------|-------|------|----------|-------------|
| 7 | II | XX | — | $CI \vee (M \wedge AC \wedge K) \rightarrow CO$<br>$M \oplus (AC \wedge K) \rightarrow AT$ | Logically OR CI with the word-wise OR of the logical AND of AC and the K-bus and M-bus. Place the carry OR on CO. Logically AND the K-bus with AC. Exclusive NOR the result with the M-bus. Deposit the final result in AC or T. |
| | | OO | LCM | $CI \rightarrow CO, \ \overline{M} \rightarrow AT$ | Load the complement of the M-bus into AC or T, as specified. Force CO to CI. |
| | | 11 | XNM | $CI \vee (M \wedge AC) \rightarrow CO$<br>$M \oplus AC \rightarrow AT$ | Force CO to one if the logical AND of AC and the M-bus is non-zero. Exclusive-NOR AC with the M-bus. Deposit the result in AC or T, as specified. Used to exclusive-NOR memory data with the accumulator. |
| 7 | III | XX | — | $CI \vee (AT \wedge I \wedge K) \rightarrow CO$<br>$AT \oplus (I \quad K) \rightarrow AT$ | Logically OR CI with the word-wise OR of the logical AND of the specified register and the I-bus and K-bus. Place the carry OR on CO. Logically AND the K-bus with the I-bus. Exclusive-NOR the result with AC or T, as specified. Deposit the final result in the specified register. |
| | | OO | CMA | $CI \rightarrow CO \quad \overline{AT} \rightarrow AT$ | Complement AC or T, as specified. Force CO to CI. |
| | | 11 | XNI | $CI \vee (AT \wedge I) \rightarrow CO$<br>$I \oplus AT \rightarrow AT$ | Force CO to one if the logical AND of the specified register and the I-bus is non-zero. Exclusive-NOR AC with the I-bus. Deposit the result in AC or T, as specified. Used to exclusive-NOR input data with the accumulator. |

## FUNCTION DESCRIPTION KEY

| SYMBOL | MEANING |
|--------|---------|
| I,K,M | Data on the I, K, and M buses, respectively |
| CI,LI | Data on the carry input and left input, respectively |
| CO,RO | Data on the carry output and right output, respectively |
| Rn | Contents of register n including T and AC (R-Group I) |
| AC | Contents of the accumulator |
| AT | Contents of AC or T, as specified |
| MAR | Contents of the memory address register |
| L,H | As subscripts, designate low and high order bit, respectively |
| + | 2's complement addition |
| – | 2's complement subtraction |
| ∧ | Logical AND |
| ∨ | Logical OR |
| ⊕ | Exclusive-NOR |
| → | Deposit into |

NOTE

1. 2's complement arithmetic adds 111 . . . 11 to perform subtraction of 000 . . . 01.

## AC ELECTRICAL CHARACTERISTICS N3001 = $T_A$ = 0°C to +70°C, $V_{CC}$ = 5V ± 5%
### S3001 = $T_A$ = -55°C to +125°C, $V_{CC}$ = 5V ± 10%

| PARAMETER | N3002 | | | S3002 | | | UNIT |
|---|---|---|---|---|---|---|---|
| | Min | Typ[1] | Max | Min | Typ[1] | Max | |
| tCY   Clock Cycle Time | 70 | 45 | | 120 | 45 | | ns |
| tWP   Clock Pulse Width | 17 | 10 | | 42 | 10 | | ns |
| tFS   Function Input Set-Up Time ($F_0$ through $F_6$) | 48 | -23 → 35 | | 70 | -23 → 35 | | ns |
| Data Set-Up Time: | | | | | | | |
| tDS   $I_0$, $I_1$, $M_0$, $M_1$, $K_0$, $K_1$ | 40 | 12 → 29 | | 60 | 12 → 29 | | ns |
| tSS   LI, CI | 21 | 0 → 7 | | 30 | 0 → 7 | | ns |
| Data and Function Hold Time: | | | | | | | |
| tFH   $F_0$ through $F_6$ | 4 | 0 | | 5 | 0 | | ns |
| tDH   $I_0$, $I_1$, $M_0$, $M_1$, $K_0$, $K_1$ | 4 | -28 → -11 | | 5 | -28 → -11 | | ns |
| tSH   LI, CI | 12 | -7 → 0 | | 15 | -7 → 0 | | ns |
| Propagation Delay to X, Y, RO from: | | | | | | | |
| tXF   Any Function Input | | 28 | 52 | | 28 | 65 | ns |
| tXD   Any Data Input | | 16 → 20 | 33 | | 16 → 20 | 65 | ns |
| tXT   Trailing Edge of CLK | | 33 | 48 | | 33 | 75 | ns |
| tXL   Leading Edge of CLK | 13 | 18 → 40 | 70 | 13 | 18 → 40 | 90 | ns |
| Propagation Delay to CO from: | | | | | | | |
| tCL   Leading Edge of CLK | 16 | 24 → 44 | 70 | | 24 → 44 | 90 | ns |
| tCT   Trailing Edge of CLK | | 30 → 40 | 56 | | 30 → 40 | 100 | ns |
| tCF   Any Function Input | | 25 → 35 | 52 | | 25 → 35 | 75 | ns |
| tCD   Any Data Input | | 17 → 23 | 55 | | 17 → 23 | 65 | ns |
| tCC   CI (Ripple Carry) | | 9 → 13 | 20 | | 9 → 13 | 30 | ns |
| Propagation Delay to $A_0$, $A_1$, $D_0$, $D_1$ from: | | | | | | | |
| tDL   Leading Edge of CLK | | 17 → 25 | 40 | | 17 → 25 | 75 | ns |
| tDE   Enable Input ED, EA | | 10 → 12 | 20 | | 10 → 12 | 35 | ns |

NOTE

1. Typical values are for $T_A$ = 25°C and typical supply voltage.

## CARRY LOOK-AHEAD CONFIGURATION
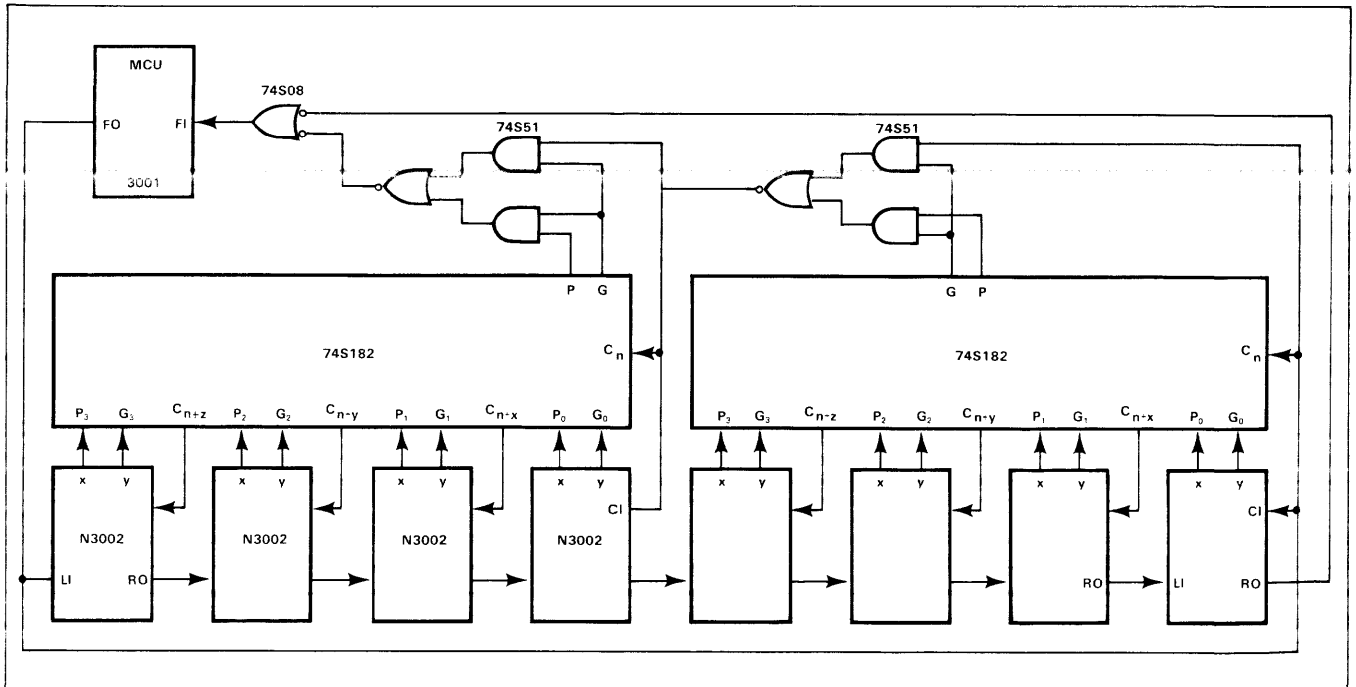
## TYPICAL CONFIGURATIONS



Ripple-Carry Configuration
(2N Bit Array)
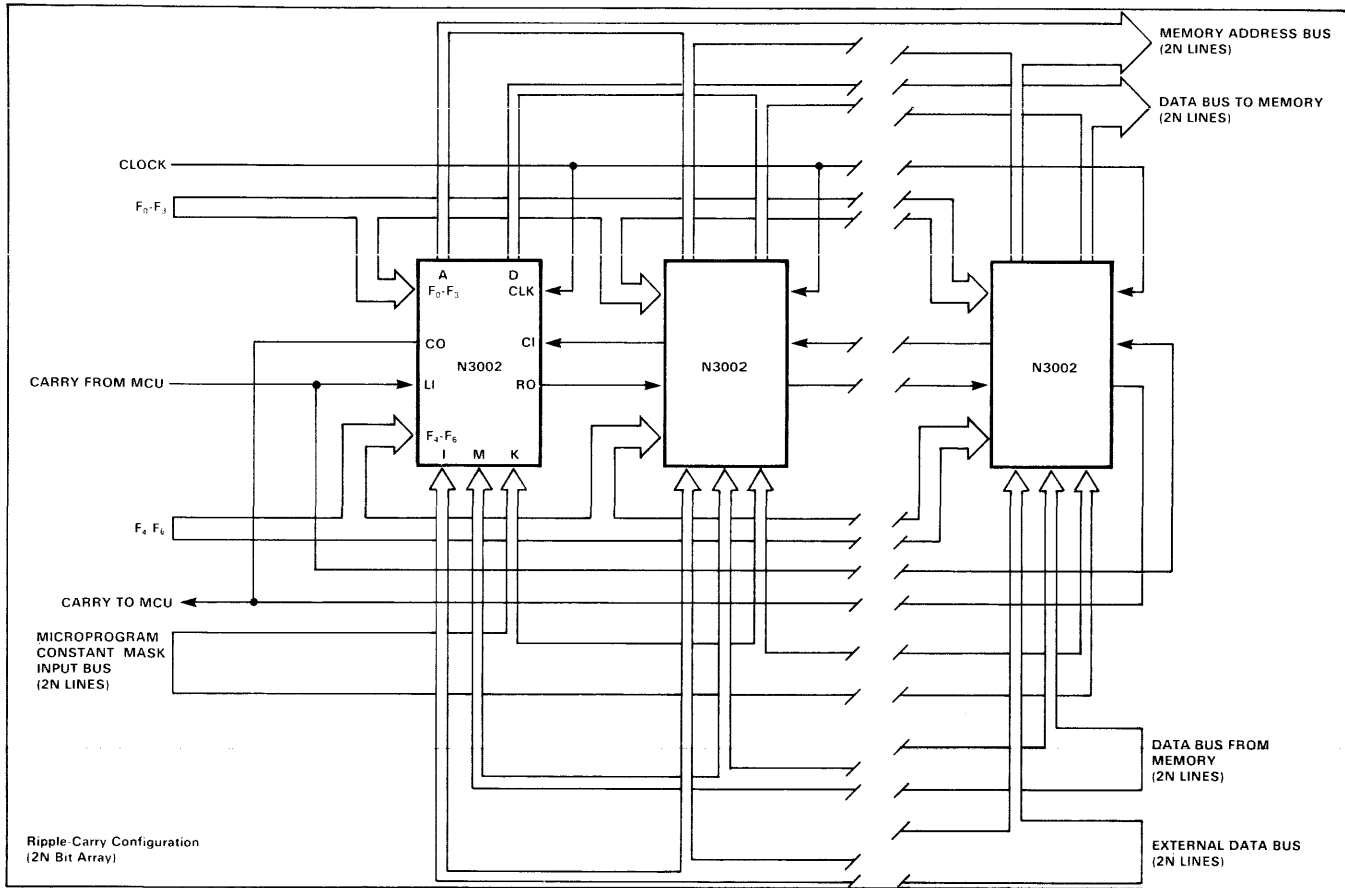
## PARAMETER MEASUREMENT INFORMATION
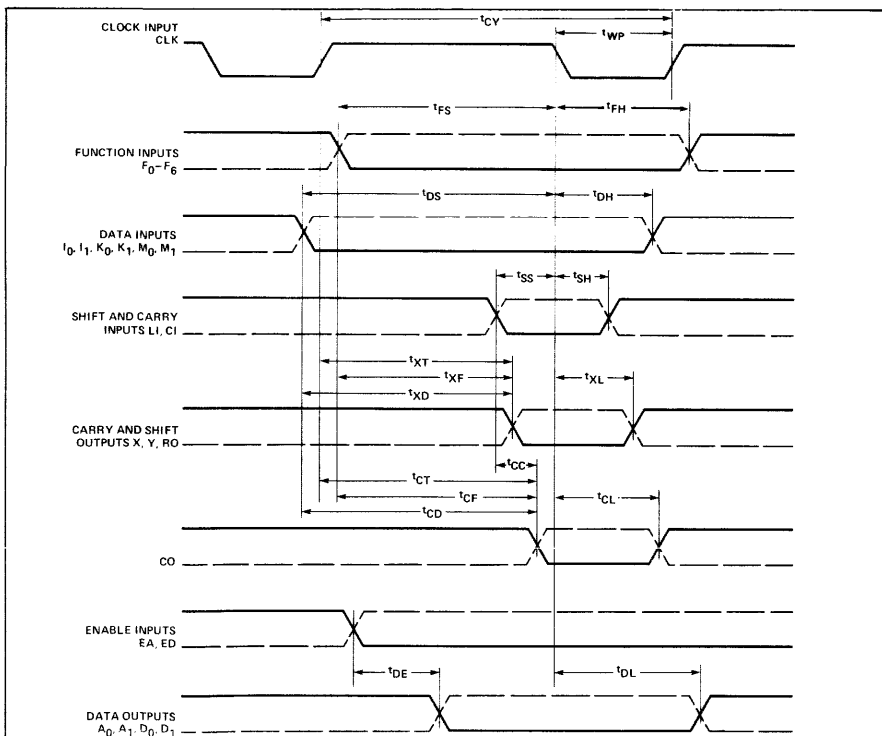
# APPENDIX E
## MICROCODE LISTING

```
*
* FIELD DEFINITIONS:
*
*                    --- --- --- --- ---        ---      --- --- --- --- --- --- --- --- ---
* ADDR FO  FI  FGP IMB RGP  DOE IRW ADL FRW ED1 KS    IER SWPA  NC2 NC1 CCR EXT RRE LD2 SJM IST DBY  CS  LD     AC    ; COMMENTS
*----- --- --- --- --- --- ---- --- --- --- --- --- ---- ----- --- --- --- --- --- --- --- --- ---- --- -- --------- --------
* :     :   :   :   :   :   :    :   :   :   :   :    :   :     :   :   :   :   :   :   :   :   :    :   :  :       V
* :     :   :   :   :   :   :    :   :   :   :   :    :   :     :   :   :   :   :   :   :   :   :    :   :  :       ADDRESS CONTROL,
* :     :   :   :   :   :   :    :   :   :   :   :    :   :     :   :   :   :   :   :   :   :   :    :   :  V    FOR 3001 MCU.
* :     :   :   :   :   :   :    :   :   :   :   :    :   :     :   :   :   :   :   :   :   :   :    :   :  LOAD, FOR OP-CODE
* :     :   :   :   :   :   :    :   :   :   :   :    :   :     :   :   :   :   :   :   :   :   :    :   V  DECODE INTO 3001.
* :     :   :   :   :   :   :    :   :   :   :   :    :   :     :   :   :   :   :   :   :   :   :    :   CARRY SELECT, CONTROLS MUX
* :     :   :   :   :   :   :    :   :   :   :   :    :   :     :   :   :   :   :   :   :   :   :    V   FOR ALU CARRY IN AND OUT.
* :     :   :   :   :   :   :    :   :   :   :   :    :   :     :   :   :   :   :   :   :   :   :    DOUBLE BYTE, STES UP ALU
* :     :   :   :   :   :   :    :   :   :   :   :    :   :     :   :   :   :   :   :   :   :   V    FOR 16-BIT OPERATION.
* :     :   :   :   :   :   :    :   :   :   :   :    :   :     :   :   :   :   :   :   :   :   INTERRUPT STROBE, CLOCKS INTERRUPT
* :     :   :   :   :   :   :    :   :   :   :   :    :   :     :   :   :   :   :   :   :   V    WHEN ENABLED, AND HOLD INPUTS.
* :     :   :   :   :   :   :    :   :   :   :   :    :   :     :   :   :   :   :   :   :   SECONDARY JUMP, ACCESSES BOTTOM HALF
* :     :   :   :   :   :   :    :   :   :   :   :    :   :     :   :   :   :   :   :   V    OF OP-DECODE PROM FOR ANOTHER DECODE
* :     :   :   :   :   :   :    :   :   :   :   :    :   :     :   :   :   :   :   :   LOAD 2, USED WITH LD, FORCES AN OP-DECODE
* :     :   :   :   :   :   :    :   :   :   :   :    :   :     :   :   :   :   :   V    JUMP INTO 2ND QUADRANT (080 TO 0FF).
* :     :   :   :   :   :   :    :   :   :   :   :    :   :     :   :   :   :   :   REGISTER ROM ENABLE, ALLOWS REGISTER
* :     :   :   :   :   :   :    :   :   :   :   :    :   :     :   :   :   :   V    SELECTION ON THE BASIS OF THE INST. REGISTER.
* :     :   :   :   :   :   :    :   :   :   :   :    :   :     :   :   :   :   EXTERNAL TEST, CONDITIONS MICRO JUMP ON
* :     :   :   :   :   :   :    :   :   :   :   :    :   :     :   :   :   V    READY SIGNAL. (READY = EVEN ADDR, NOT READY = ODD)
* :     :   :   :   :   :   :    :   :   :   :   :    :   :     :   :   :   CONDITIONAL CLOCK READY, DISABLES BOTH ARRAYS 'TIL MEM
* :     :   :   :   :   :   :    :   :   :   :   :    :   :     :   :   V    READY (WHEN EXT OFF, CONDITIONS MICRO JUMP ON HOLD)
* :     :   :   :   :   :   :    :   :   :   :   :    :   :     :   :   NO CLOCK ARRAY 1, DISABLES ARRAY 1 CLOCK UNTIL
* :     :   :   :   :   :   :    :   :   :   :   :    :   :     :   V    MEMORY OR I/O INDICATES READY.
* :     :   :   :   :   :   :    :   :   :   :   :    :   :     :   NO CLOCK ARRAY 2, DISABLES ARRAY 2 CLOCK UNTIL
* :     :   :   :   :   :   :    :   :   :   :   :    :   :     V    MEMORY OR I/O INDICATES READY.
* :     :   :   :   :   :   :    :   :   :   :   :    :   :     STATUS WORD PROM ADDRESS, OUTPUTS STATUS:
* :     :   :   :   :   :   :    :   :   :   :   :    :   V     MEMR, MEMW, IOR, IOW, INTA, HLTA, OR RTRAP.
* :     :   :   :   :   :   :    :   :   :   :   :    :   INTERRUPT-ENABLE-F/F CONTROL, USED DURING
* :     :   :   :   :   :   :    :   :   :   :   :    V   EI, DI, INT & RESET TO CLEAR OR SET INTE F/F.
* :     :   :   :   :   :   :    :   :   :   :   :    K-BUS SELECT, INPUTS TO ALU: ALU DATA OUT, MEMORY DATA IN,
* :     :   :   :   :   :   :    :   :   :   :   V    INSTRUCTION REGISTER, ALL 0'S OR THE COMPLIMENT OF ANY OF THESE.
* :     :   :   :   :   :   :    :   :   :   :   ENABLE DATA ARRAY 1, ENABLES ARRAY 1 ACCUMULATOR ONTO THE
* :     :   :   :   :   :   :    :   :   :   V    ALU DATA OUT BUS; OTHERWISE ARRAY 2 ACCUMULATOR.
* :     :   :   :   :   :   :    :   :   :   FLAG REGISTER WRITE, GATES NEW VALUES FOR PARITY, SIGN &
* :     :   :   :   :   :   :    :   :   V    ZERO FLAGS INTO PSW LATCH.
* :     :   :   :   :   :   :    :   :   ADDRESS LOAD, LOAD CONTENTS OF INTERNAL (3002) ADDRESS REGISTER
* :     :   :   :   :   :   :    :   V    INTO THE EXTERNAL MEMORY ADDRESS REGISTER.
* :     :   :   :   :   :   :    :   INSTRUCTION REGISTER WRITE, LOADS INSTRUCTION OR DATA RETURNED
* :     :   :   :   :   :   :    V    FROM MEMORY, INTO THE INSTRUCTION REGISTER.
* :     :   :   :   :   :   :    DATA OUT ENABLE, REVERSES DIRECTION OF THE BI-DIRECTIONAL DATA BUS
* :     :   :   :   :   :   V    FOR WRITING OUT TO MEMORY OR I/O.
* :     :   :   :   :   :   REGISTER GROUP SELECTION, ADDRESSES PROM U-17 TO PROVIDE REGISTER SELECTION FOR ARRAY 1 AND ARRAY 2.
* :     :   :   :   :   V   (USES 0011 AS A MASK  WHEN RRE IS ON BECAUSE OF 'WIRE OR' OF U-17 & U-30)
* :     :   :   :   :   I-MASK BUS CONTROL, WITH CY & HC, ADDRESSES PROM U-24 TO PROVIDE 1 OF 32 MASKS
* :     :   :   :   V   FOR RST, DAA, MUL AND DIV INSTRUCTIONS.
* :     :   :   :   FUNCTION GROUP SELECTION, PROVIDES A 3-BIT F-GROUP CONTROL FOR EACH OF ARRAY1 & ARRAY 2.
* :     :   :   V   (F-6 AND F-4 TO BOTH ARRAYS, F1-5 TO ARRAY 1, AND F2-5 TO ARRAY 2).
* :     :   :   FLAG INPUT CONTROL, ROUTES FI INPUT TO 3001 TO C-FLAG AND/OR Z-FLAG, INTERNAL TO THE 3002 MCU,
* :     :   V   AND ALSO EXTERNALLY ROUTES FI TO THE CY F/F, AND HCI TO THE HC F/F.
* :     :   FLAG OUTPUT CONTROL, SETS FO OUTPUT ON 3001 TO C-FLAG, Z-FLAG, 1, OR 0.
* :     V
* ADDRESS TO MICRO CONTROL STROE, 000 TO 1FF (HEX).
```

| | FO | FI | FGP | IMB | RGP | DOE | IRW | ADL | FRW | ED1 | KS | IER | SWPA | NC2 | NC1 | CCR | EXT | RRE | LD2 | SJM | IST | DBY | CS | LD | AC | ; COMMENTS |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *00000 | FF0 | SCZ | | I00 | R00 | DOE | IRW | ADL | FRW | ED1 | K0 | IER | NSTRT | NC2 | NC1 | CCR | EXT | RRE | LD2 | SJM | IST | DBY | CS0 | (D) | | 00000 |
| *00001 | FFC | STC | | IBF | R11 | (D) | (D) | (D) | (D) | (D) | KIR | (D) | RTRAP | (D) | (D) | (D) | (D) | (D) | (D) | (D) | (D) | (D) | CN | LD | | 00001 |
| *00010 | FFZ | STC | | - | R22 | | | | | | KM | | HLTA | | | | | | | | | | BN | | | 00010 |
| *00011 | FF1 | HCZ | | I99 | R33 | | | | | | KD | | INTA | | | | | | | | | | AN | | | 00011 |
| *00100 | | | | I99 | R44 | | | | | | K1 | | IOW | | | | | | | | | | CS1 | | | 00100 |
| *00101 | | | | IC7 | R55 | | | | | | KNIR | | IOR | | | | | | | | | | NCN | | | 00101 |
| *00110 | | | | IFF | R66 | | | | | | KNM | | MEMW | | | | | | | | | | NBN | | | 00110 |
| *00111 | | | | I00 | R77 | | | | | | KND | | (MEMR) | | | | | | | | | | NAN | | | 00111 |
| *01000 | | | | | R88 | | | | | | | | | | | | | | | | | | | | | 01000 |
| *01001 | | | | | R99 | | | | | | | | | | | | | | | | | | | | | 01001 |
| *01010 | | | | | RAA | | | | | | | | | | | | | | | | | | | | | 01010 |
| *01011 | | | | | RBB | | | | | | | | | | | | | | | | | | | | | 01011 |
| *01100 | | | | | RCC | | | | | | | | | | | | | | | | | | | | | 01100 |
| *01101 | | | | | RDD | | | | | | | | | | | | | | | | | | | | | 01101 |
| *01110 | | | | | REE | | | | | | | | | | | | | | | | | | | | | 01110 |
| *01111 | | | | | RFF | | | | | | | | | | | | | | | | | | | | | 01111 |
| *10000 | | | | | R3C | | | | | | | | | | | | | | | | | | | | | 10000 |
| *10001 | | | | | RC3 | | | | | | | | | | | | | | | | | | | | | 10001 |
| *10010 | | | | | RCB | | | | | | | | | | | | | | | | | | | | | 10010 |
| *10011 | | | | | RCF | | | | | | | | | | | | | | | | | | | | | 10011 |
| *10100 | | | | | RAC | | | | | | | | | | | | | | | | | | | | | 10100 |
| *10101 | | | | | R3D | | | | | | | | | | | | | | | | | | | | | 10101 |
| *10110 | | | | | - | | | | | | | | | | | | | | | | | | | | | 10110 |
| *10111 | | | | | - | | | | | | | | | | | | | | | | | | | | | 10111 |
| *11000 | | | | | - | | | | | | | | | | | | | | | | | | | | | 11000 |
| *11001 | | | | | - | | | | | | | | | | | | | | | | | | | | | 11001 |
| *11010 | | | | | - | | | | | | | | | | | | | | | | | | | | | 11010 |
| *11011 | | | | | RDC | | | | | | | | | | | | | | | | | | | | | 11011 |
| *11100 | | | | | RCD | | | | | | | | | | | | | | | | | | | | | 11100 |
| *11101 | | | | | RFE | | | | | | | | | | | | | | | | | | | | | 11101 |
| *11110 | | | | | REF | | | | | | | | | | | | | | | | | | | | | 11110 |
| *11111 | | | | | RFF | | | | | | | | | | | | | | | | | | | | | 11111 |

```
*
* UTILITY GROUP: FETCH, INTERRUPT, HOLD, RTRAP, RESET, HALT
*
*       PROM U7    PROM U8      PROM U2                      PROM U5                      PROM U6            PROM U4
*      V--------V V-----V V--------------------V V---------------------V V---------------------V V---------V
*      8-7,6-5,4-1 8-6,5-1 8,  7,  6,  5,  4, 3-1 8,  7--5,  4,  3,  2, 1 8,  7,  6,  5,  4, 3-1 8,  7---1
*
*
* ADDR FO  FI  FGP IMB RGP  DOE IRW ADL FRW ED1 KS   IER SWPA NC2 NC1 CCR EXT RRE LD2 SJM IST DBY  CS  LD   AC      ; COMMENTS
*---- --- --- --- --- ---  --- --- --- --- --- ---  --- --- --- --- --- --- --- --- --- --- ---  --  -- ---------- ---------
(007H):     NOP     RFF  IRW             K1                        EXT             IST     NAN    JCR(006H); FETCH
(006H):FF1  LMI     R44          ADL     K1                                    SJM     DBY AN  LD JPX     ; FETCH

(1FFH):FF1  CSR     R44                  K1   NSTAT                                         AN     JZR(009H); RESET
(1FEH):FF1  CSR     R44                  K1   NSTAT                                         AN     JZR(009H); RESET
(009H):FF1  LMI     R44                  K1   IER NSTAT                              DBY AN     JCC(149H); RESET
(149H):FF1  LMI     R44          ADL     K1                                     IST DBY AN     JZR07;

(1E7H):     NOP     RFF  IRW             K1                        EXT                     NAN    JCC(1E7H); HOLD
(1E6H):     NOP     RFF                  K1   NSTAT         CCR                            NAN    JCC(1D6H);
(1D7H):     NOP     RFF                  K1   NSTAT         CCR                            NAN    JCR(1D6H);
(1D6H):     NOP     RFF                  K1   NSTAT                                 IST     NAN    JZR06;

(001H):     NOP     RFF                  K1   HLTA                                  IST     NAN    JCR(001H); HALT
(1E1H):     NOP     RFF                  K1   HLTA          CCR                            NAN    JCC(1E1H);
(1E0H):     NOP     RFF                  K1   HLTA                                  IST     NAN    JZR(001H);

(1F1H):     NOP     RFF                  K1   INTA                                         NAN    JZR(00BH);

(000H):     NOP     RFF                  K1                        EXT                     NAN    JCC(161H); RTRAP
(160H):     NOP     RFF                  K1   RTRAP                                        NAN    JZR(001H);

(1F7H):     NOP     RFF                  K1                        EXT                     NAN    JCC(1F7H); INT
(1F6H):FF1  NOP     RFF                  K1   IER INTA                                     AN     JZR(00BH);
(00BH):     NOP     RFF  IRW             K1   INTA                     EXT             SJM     NAN    JCR(00AH);
(00AH):     NOP     RFF                  K1   NSTAT                        LD2             NAN LD JPX      ;
*                                                                                          (TO 09A, 090, 092, 091, OR 09D)

(09AH):FF1  DSM     R33                  K0   NSTAT                                      DBY NAN    JCR(097H); INT: RST
(097H):FF1  LMI     R33                  K1   NSTAT                                         NAN    JCR(096H);
(096H):FF1  LDI     RFF                  K0   NSTAT                                      DBY NAN    JCC(106H);(@ GP 1A)

(090H):FF1  LDI     RFF                  K0   NSTAT                                      DBY NAN    JCR(095H); INT: CALL
(095H):FF1  ALR     R44                  K0   NSTAT                                      DBY NAN    JCR(093H);
(093H):FF1  CSR     R44                  K1   INTA                                         NAN    JCR(099H);
(099H):     F5      R44  IRW             KIR  INTA NC2         CCR EXT                      NAN    JCR(098H);
(098H):FF1  DSM     R33                  K0   INTA                                       DBY NAN    JCR(09FH);
(09FH):     F5      R44  IRW             KIR  INTA         NC1 CCR EXT                      NAN    JCR(09EH);
(09EH):FF1  LMI     R33                  K1   NSTAT                                         NAN    JCC(10EH);(@ GP 10)

(092H):FF1  LMI     R33                  K1   NSTAT                                      DBY AN     JCC(042H); INT: RET
*                                                                                          (TO 042 @ GP 14)
(091H):FF1  CSR     R44                  K1   INTA                                         AN     JCC(141H); INT: JMP
(141H):FF1  LMI     R44  IRW             KIR  INTA NC2         CCR EXT                      NAN    JCR(140H);
(140H):     NOP     RFF                  K1   INTA                                         NAN    JCR(143H);
(143H):FF1  LMI     R44  IRW             KIR  INTA         NC1 CCR EXT                      NAN    JCR(142H);
(142H):FF1  LMI     R44          ADL     K1                                             DBY AN     JCC(132H);(@ GP 0A)

(09DH):FF1  ILR     R22                  K1   NSTAT                                         NAN    JCC(12DH); INT: PCHL
(12DH):FF1  SDR     R44                  K0   NSTAT                                      DBY AN     JLL(135H);
(135H):FF1  LMI     R44                  K1   NSTAT                                         NAN    JCR(130H);(@ GP 0A)
*---- --- --- --- --- ---  --- --- --- --- --- ----- --- --- --- --- --- --- --- --- --- --- ---  --  -- ---------- ---------
*     FF0 HC2  -  IFF  -   1   1   1   1   ED2  -   1 MEMR  1   1   1   1   1   1   1   1   1   1   -   0  1'S   ; DEFAULTS
*-------------------------------------------------------------------------------------------------------------------
```

```
*
* GROUP 00   ARITH R;  R = B, D, H    (ADD R, ADC R, SUB R, SBB R, ANA R, XRA R, ORA R, CMP R)
*
*      PROM U7   PROM U8      PROM U2                    PROM U5                   PROM U6            PROM U4
*      V---------V V-----V V---------------------V V------------------------V V-------------------V V---------V
*      8-7,6-5,4-1 8-6,5-1 8,   7,   6,   5,   4, 3-1 8,  7--5,  4,   3,   2,  1  8,  7,   6,   5,   4, 3-1 8,   7---1
*                          ---  ---  ---  --- ---    ---    ---  --- --- --- ---  --- --- --- --- --- ---  --- ---
* ADDR FO  FI  FGP IMB RGP DOE IRW ADL FRW ED1 KS   IER SWPA NC2 NC1 CCR EXT RRE LD2 SJM IST DBY CS  LD    AC    ; COMMENTS
*----- --- --- --- --- --- ---- --- --- --- --- --- ---- ----- --- --- --- --- --- --- --- --- --- -- --------- --------
(030H):FF1     ILR     R3C                     K1                          RRE                   NAN    JPX02  ;R=B,D,H
*
(020H):FF0 SCZ AIA     REF                     KD                                                AN     JZR02  ;ADD B,D,H
(021H):FFC SCZ AIA     REF                     KD                                                AN     JZR02  ;ADC B,D,H
(022H):FF0 SCZ AIA     REF                     KND                                               NCN    JZR02  ;SUB B,D,H
(023H):FFC SCZ AIA     REF                     KND                                               NCN    JZR02  ;SBB B,D,H
(024H):FF0 SCZ ANI     REF                     KD                                                CS1    JZR02  ;ANA B,D,H
(025H):FF0 SCZ XNI     REF                     KND                                               CS1    JZR02  ;XRA B,D,H
(026H):FF0 SCZ ORI     REF                     KD                                                CS1    JZR02  ;ORA B,D,H
(027H):FF0 SCZ AIA     RFF                     KND                                               NCN    JZR02  ;CMP B,D,H
*----- --- --- --- --- --- ---- --- --- --- --- --- ---- ----- --- --- --- --- --- --- --- --- --- -- --------- --------
*      FF0 HCZ  -  IFF  -   1   1   1   1  ED2  -   1  MEMR   1   1   1   1   1   1   1   1   1   -   0  1'S   ; DEFAULTS
*-------------------------------------------------------------------------------------------------------------------------
```

```
*
* GROUP 01 : ARITH R;  R = C, E, L    (ADD R, ADC R, SUB R, SBB R, ANA R, XRA R, ORA R, CMP R)
*
*      PROM U7   PROM U8      PROM U2                    PROM U5                   PROM U6            PROM U4
*      V---------V V-----V V---------------------V V------------------------V V-------------------V V---------V
*      8-7,6-5,4-1 8-6,5-1 8,   7,   6,   5,   4, 3-1 8,  7--5,  4,   3,   2,  1  8,  7,   6,   5,   4, 3-1 8,   7---1
*                          ---  ---  ---  --- ---    ---    ---  --- --- --- ---  --- --- --- --- --- ---  --- ---
* ADDR FO  FI  FGP IMB RGP DOE IRW ADL FRW ED1 KS   IER SWPA NC2 NC1 CCR EXT RRE LD2 SJM IST DBY CS  LD    AC    ; COMMENTS
*----- --- --- --- --- --- ---- --- --- --- --- --- ---- ----- --- --- --- --- --- --- --- --- --- -- --------- --------
(031H):FF1     ILR     RC3                     K1                          RRE                   NAN    JPX02  ;R=C,E,L
*
(028H):FF0 SCZ AIA     RFE                     KD                                                AN     JZR03  ;ADD C,E,L
(029H):FFC SCZ AIA     RFE                     KD                                                AN     JZR03  ;ADC C,E,L
(02AH):FF0 SCZ AIA     RFE                 ED1 KND                                               NCN    JZR03  ;SUB C,E,L
(02BH):FFC SCZ AIA     RFE                 ED1 KND                                               NCN    JZR03  ;SBB C,E,L
(02CH):FF0 SCZ ANI     RFE                 ED1 KD                                                CS1    JZR03  ;ANA C,E,L
(02DH):FF0 SCZ XNI     RFE                 ED1 KND                                               CS1    JZR03  ;XRA C,E,L
(02EH):FF0 SCZ ORI     RFE                 ED1 KD                                                CS1    JZR03  ;ORA C,E,L
(02FH):FF0 SCZ AIA     RFF                 ED1 KND                                               NCN    JZR04  ;CMP C,E,L
*----- --- --- --- --- --- ---- --- --- --- --- --- ---- ----- --- --- --- --- --- --- --- --- --- -- --------- --------
*      FF0 HCZ  -  IFF  -   1   1   1   1  ED2  -   1  MEMR   1   1   1   1   1   1   1   1   1   -   0  1'S   ; DEFAULTS
*-------------------------------------------------------------------------------------------------------------------------
```

```
*
* GROUP 02 : INR B,   INR D,   INR H
*
*      PROM U7   PROM U8      PROM U2                    PROM U5                   PROM U6            PROM U4
*      V---------V V-----V V---------------------V V------------------------V V-------------------V V---------V
*      8-7,6-5,4-1 8-6,5-1 8,   7,   6,   5,   4, 3-1 8,  7--5,  4,   3,   2,  1  8,  7,   6,   5,   4, 3-1 8,   7---1
*                          ---  ---  ---  --- ---    ---    ---  --- --- --- ---  --- --- --- --- --- ---  --- ---
* ADDR FO  FI  FGP IMB RGP DOE IRW ADL FRW ED1 KS   IER SWPA NC2 NC1 CCR EXT RRE LD2 SJM IST DBY CS  LD    AC    ; COMMENTS
*----- --- --- --- --- --- ---- --- --- --- --- --- ---- ----- --- --- --- --- --- --- --- --- --- -- --------- --------
(032H):FF1 STZ ILR     R33                     K1                  NC1     RRE                   AN     JZR04  ;INR B,D,H
*----- --- --- --- --- --- ---- --- --- --- --- --- ---- ----- --- --- --- --- --- --- --- --- --- -- --------- --------
*      FF0 HCZ  -  IFF  -   1   1   1   1  ED2  -   1  MEMR   1   1   1   1   1   1   1   1   1   -   0  1'S   ; DEFAULTS
*-------------------------------------------------------------------------------------------------------------------------
```

```
EJECT;
*
* GROUP 03 :  INR C,   INR E,   INR L
*
*      PROM U7   PROM U8        PROM U2                   PROM U5                      PROM U6              PROM U4
*      V---------V V-----V V----------------------V V--------------------------V V----------------------V V---------V
*      8-7,6-5,4-1 8-6,5-1 8,  7,  6,  5,  4, 3-1 8, 7--5,  4,  3,  2,  1  8,  7,  6,  5,  4, 3-1 8,  7---1
*                          --- --- --- --- ---     ---    --- --- --- --- --- --- --- --- --- ---
* ADDR FO FI  FGP IMB RGP DOE IRW ADL FRW ED1 KS   IER SWPA NC2 NC1 CCR EXT RRE LD2 SJM IST DBY CS LD    AC   ; COMMENTS
*----- --- --- --- --- --- --- --- --- --- --- ---- ---- ---- --- --- --- --- --- --- --- --- --- -- -------- ---------
(033H):FF1 STZ ILR    R33                     K1            NC2             RRE                  BN     JZR05  ; INR C,E,L
*----- --- --- --- --- --- --- --- --- --- --- ---- ---- ---- --- --- --- --- --- --- --- --- --- -- -------- ---------
*      FF0 HCZ  - IFF  -   1   1   1   1 ED2  -   1 MEMR    1   1   1   1   1   1   1   1   1   1  -   0   1'S  ; DEFAULTS
*----------------------------------------------------------------------------------------------------------------------
*
*
* GROUP 04 :  DCR B,  DCR D,  DCR H,  DCR C,  DCR E,  DCR L,  DCR A,  DCX B,  DCX D,  DCX H
*
*      PROM U7   PROM U8        PROM U2                   PROM U5                      PROM U6              PROM U4
*      V---------V V-----V V----------------------V V--------------------------V V----------------------V V---------V
*      8-7,6-5,4-1 8-6,5-1 8,  7,  6,  5,  4, 3-1 8, 7--5,  4,  3,  2,  1  8,  7,  6,  5,  4, 3-1 8,  7---1
*                          --- --- --- --- ---     ---    --- --- --- --- --- --- --- --- --- ---
* ADDR FO  FI  FGP IMB RGP DOE IRW ADL FRW ED1 KS   IER SWPA NC2 NC1 CCR EXT RRE LD2 SJM IST DBY CS LD    AC   ; COMMENTS
*----- --- --- --- --- --- --- --- --- --- --- ---- ---- ---- --- --- --- --- --- --- --- --- --- -- -------- ---------
(034H):FF1     CAS  RFF                     K1                                                       NAN LD     ; ACC = -1
*
(010H):FF0 STZ ALR    R33                     K0                NC1             RRE                  AN     JZR04  ; DCR B,D,H
(011H):FF0 STZ ALR    R33                     K0            NC2             RRE                      BN     JZR05  ; DCR C,E,L
(037H):FF0 STZ ALR    R33                     K0                                                     AN     JZR04  ; DCR A
(01BH):FF1     ALR    R33      IRW            K0                            EXT RRE         IST DBY NAN    JZR06  ; DCX B,D,H
*----- --- --- --- --- --- --- --- --- --- --- ---- ---- ---- --- --- --- --- --- --- --- --- --- -- -------- ---------
*      FF0 HCZ  - IFF  -   1   1   1   1 ED2  -   1 MEMR    1   1   1   1   1   1   1   1   1   1  -   0   1'S  ; DEFAULTS
*----------------------------------------------------------------------------------------------------------------------
*
*
* GROUP 05 :  INX B,  INX D,  INX H
*
*      PROM U7   PROM U8        PROM U2                   PROM U5                      PROM U6              PROM U4
*      V---------V V-----V V----------------------V V--------------------------V V----------------------V V---------V
*      8-7,6-5,4-1 8-6,5-1 8,  7,  6,  5,  4, 3-1 8, 7--5,  4,  3,  2,  1  8,  7,  6,  5,  4, 3-1 8,  7---1
*                          --- --- --- --- ---     ---    --- --- --- --- --- --- --- --- --- ---
* ADDR FO  FI  FGP IMB RGP DOE IRW ADL FRW ED1 KS   IER SWPA NC2 NC1 CCR EXT RRE LD2 SJM IST DBY CS LD    AC   ; COMMENTS
*----- --- --- --- --- --- --- --- --- --- --- ---- ---- ---- --- --- --- --- --- --- --- --- --- -- -------- ---------
(035H):FF1     ILR    R33      IRW            K1                            EXT RRE         IST DBY AN     JZR06  ; INX B,D,H
*----- --- --- --- --- --- --- --- --- --- --- ---- ---- ---- --- --- --- --- --- --- --- --- --- -- -------- ---------
*      FF0 HCZ  - IFF  -   1   1   1   1 ED2  -   1 MEMR    1   1   1   1   1   1   1   1   1   1  -   0   1'S  ; DEFAULTS
*----------------------------------------------------------------------------------------------------------------------
*
*
* GROUP 06 :  INR A
*
*      PROM U7   PROM U8        PROM U2                   PROM U5                      PROM U6              PROM U4
*      V---------V V-----V V----------------------V V--------------------------V V----------------------V V---------V
*      8-7,6-5,4-1 8-6,5-1 8,  7,  6,  5,  4, 3-1 8, 7--5,  4,  3,  2,  1  8,  7,  6,  5,  4, 3-1 8,  7---1
*                          --- --- --- --- ---     ---    --- --- --- --- --- --- --- --- --- ---
* ADDR FO  FI  FGP IMB RGP DOE IRW ADL FRW ED1 KS   IER SWPA NC2 NC1 CCR EXT RRE LD2 SJM IST DBY CS LD    AC   ; COMMENTS
*----- --- --- --- --- --- --- --- --- --- --- ---- ---- ---- --- --- --- --- --- --- --- --- --- -- -------- ---------
(036H):FF1 STZ ILR    RCC                     K1                                                     AN     JZR04  ; INR A
*----- --- --- --- --- --- --- --- --- --- --- ---- ---- ---- --- --- --- --- --- --- --- --- --- -- -------- ---------
*      FF0 HCZ  - IFF  -   1   1   1   1 ED2  -   1 MEMR    1   1   1   1   1   1   1   1   1   1  -   0   1'S  ; DEFAULTS
*----------------------------------------------------------------------------------------------------------------------
*
```

```
*
*  GROUP 08:  MOV R1,R2;       R1 = C, E, L;    R2 = B, D, H
*
*       PROM U7   PROM U8         PROM U2                      PROM U5                        PROM U6             PROM U4
*      V---------V V-----V V-----------------------V V-----------------------------V V-------------------------V V---------V
*      8-7,6-5,4-1 8-6,5-1 8,  7,  6,  5,  4, 3-1 8,  7--5,  4,  3,  2,  1  8,  7,  6,  5,  4, 3-1 8,    7---1
*
*
*                          --- --- --- --- ---     ---     --- --- --- --- --- --- --- --- ---
* ADDR FO  FI  FGP IMB RGP DOE IRW ADL FRW ED1 KS   IER SWPA NC2 NC1 CCR EXT RRE LD2 SJM IST DBY  CS  LD     AC    ; COMMENTS
*----- --- --- --- --- --- --- --- --- --- --- ---  --- ---- --- --- --- --- --- --- --- --- --- --  -- --------- ----------
(038H):FF0    F0_2 R33                             K1                              RRE             NAN    JCC(138H);
(138H):       TZR  R33         IRW                 KD               NC2         EXT RRE         IST NAN    JZR06    ;MOV R1,R2
*                                                                                                                  ;R1=C,E,L
*                                                                                                                  ;R2=B,D,H
*----- --- --- --- --- --- --- --- --- --- --- ---  --- ---- --- --- --- --- --- --- --- --- --- --  -- --------- ----------
*      FF0 HCZ  -  IFF  -   1   1   1   1  ED2  -    1  MEMR  1   1   1   1   1   1   1   1   1  -   0   1'S    ; DEFAULTS
*-------------------------------------------------------------------------------------------------------------------------
*
*
*
*
*
*
*
*
*
*  GROUP 09:  STA
*
*       PROM U7   PROM U8         PROM U2                      PROM U5                        PROM U6             PROM U4
*      V---------V V-----V V-----------------------V V-----------------------------V V-------------------------V V---------V
*      8-7,6-5,4-1 8-6,5-1 8,  7,  6,  5,  4, 3-1 8,  7--5,  4,  3,  2,  1  8,  7,  6,  5,  4, 3-1 8,    7---1
*
*
*                          --- --- --- --- ---     ---     --- --- --- --- --- --- --- --- ---
* ADDR FO  FI  FGP IMB RGP DOE IRW ADL FRW ED1 KS   IER SWPA NC2 NC1 CCR EXT RRE LD2 SJM IST DBY  CS  LD     AC    ; COMMENTS
*----- --- --- --- --- --- --- --- --- --- --- ---  --- ---- --- --- --- --- --- --- --- --- --- --  -- --------- ----------
(039H):FF1    ILR  RCC                             K1                                              NAN    JCC(03FH); STA
*----- --- --- --- --- --- --- --- --- --- --- ---  --- ---- --- --- --- --- --- --- --- --- --- --  -- --------- ----------
*      FF0 HCZ  -  IFF  -   1   1   1   1  ED2  -    1  MEMR  1   1   1   1   1   1   1   1   1  -   0   1'S    ; DEFAULTS
*-------------------------------------------------------------------------------------------------------------------------


EJECT;
*
*  GROUP 0A:  DAD B,D,H;   SPHL;   PCHL;   XCHG;   MOV(B,D,H),(B,D,H);   MOV(C,E,L),(C,E,L)
*
*       PROM U7   PROM U8         PROM U2                      PROM U5                        PROM U6             PROM U4
*      V---------V V-----V V-----------------------V V-----------------------------V V-------------------------V V---------V
*      8-7,6-5,4-1 8-6,5-1 8,  7,  6,  5,  4, 3-1 8,  7--5,  4,  3,  2,  1  8,  7,  6,  5,  4, 3-1 8,    7---1
*
*
*                          --- --- --- --- ---     ---     --- --- --- --- --- --- --- --- ---
* ADDR FO  FI  FGP IMB RGP DOE IRW ADL FRW ED1 KS   IER SWPA NC2 NC1 CCR EXT RRE LD2 SJM IST DBY  CS  LD     AC    ; COMMENTS
*----- --- --- --- --- --- --- --- --- --- --- ---  --- ---- --- --- --- --- --- --- --- --- --- --  -- --------- ----------
(03AH):FF1    ILR  R33                             K1                          RRE             NAN LD            ;
*
(01AH):FF0 STC ALR  R22         IRW               K0                          EXT         IST DBY BN    JZR07  ;DAD B,D,H
*
(01CH):FF1    SDR  R33          IRW               K0                          EXT         IST DBY AN    JZR06  ; SPHL
*
(01DH):FF1    SDR  R44                            K0                                          DBY AN    JCC(13DH); PCHL
(13DH):FF0    LMI  R44                            K1                          EXT             NAN    JCR(131H);
(131H):       NOP  RFF                            K1                          EXT             NAN    JCR(130H);
(130H):FF1    LMI  R44              ADL           K1                              DBY AN    JCR(132H);
(132H):FF1    LMI  R44          IRW               K1                          EXT         IST DBY AN    JZR06    ;
```

```
*
(01FH):FF1  SDR  R99           K0                                    DBY AN   JCC(0CFH); XCHG
(0CFH):FF0  ILR  R11           K1                                        NAN  JCC(11FH);
(11FH):FF1  SDR  R22           K0                                    DBY AN   JCR(119H);
(119H):FF0  ILR  R99           K1                                        NAN  JCC(139H);
(139H):FF1  SDR  R11  IRW      K0                        EXT    IST DBY AN   JZR06;
*
(014H):FF1  SDR  R00  IRW      K0                NC1     EXT    IST     AN   JZR06;  MOV B,(D,H)
*
(015H):FF1  SDR  R11  IRW      K0                NC1     EXT    IST     AN   JZR06;  MOV D,(B,H)
*
(016H):FF1  SDR  R11  IRW      K0                NC1     EXT    IST     AN   JZR06;  MOV H,(B,D)
*
(017H):FF1  SDR  R00  IRW      K0          NC2           EXT    IST     AN   JZR06;  MOV C,(E,L)
*
(018H):FF1  SDR  R11  IRW      K0          NC2           EXT    IST     AN   JZR06;  MOV E,(C,L)
*
(019H):FF1  SDR  R22  IRW      K0          NC2           EXT    IST     AN   JZR06;  MOV L,(C,E)
*
(07CH):FF1  LDI  REE  IRW  ED1 KD                        EXT    IST     AN   JZR06;MOV A,(C,E,L)
*
(07FH):FF1  LDI  REE  IRW      KD                        EXT    IST     AN   JZR06;MOV A,(B,D,H)
*----- --- --- --- --- --- ---- --- --- --- --- --- --- --- --- --- --- --- --- --- --- -- -------- --------
*      FF0 HCZ -  IFF -  1  1  1  1 ED2 -  1 MEMR 1  1  1  1  1  1  1  1  1  -  0   1'S   ; DEFAULTS
*--------------------------------------------------------------------------------------------------------
*
*
* GROUP 0B:  LDA
*
*       PROM U7    PROM U8      PROM U2                    PROM U5              PROM U6          PROM U4
*       V--------V V-----V V-------------------V V--------------------V V------------------V V---------V
*       8-7,6-5,4-1 8-6,5-1 8,  7,  6,  5,  4, 3-1 8,  7--5,  4,  3,  2,  1  8,  7,  6,  5,  4, 3-1 8,  7---1
*
*
*                         --- --- --- --- ---    ---      --- --- --- --- --- --- --- --- ---
* ADDR FO  FI  FGP IMB RGP DOE IRW ADL FRW ED1 KS  IER SWPA NC2 NC1 CCR EXT RRE LD2 SJM IST DBY CS LD   AC   ; COMMENTS
*----- --- --- --- --- --- ---- --- --- --- --- ---   --- --- --- --- --- --- --- --- --- --- -- -------- --------
(03BH):FF1  LDI  RFF           KM                        EXT                  AN   JCC(03AH);  LDA
*----- --- --- --- --- --- ---- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- -- -------- --------
*      FF0 HCZ -  IFF -  1  1  1  1 ED2 -  1 MEMR 1  1  1  1  1  1  1  1  1  -  0   1'S   ; DEFAULTS
*--------------------------------------------------------------------------------------------------------
*
*
* GROUP 0C:  XTHL
*
*       PROM U7    PROM U8      PROM U2                    PROM U5              PROM U6          PROM U4
*       V--------V V-----V V-------------------V V--------------------V V------------------V V---------V
*       8-7,6-5,4-1 8-6,5-1 8,  7,  6,  5,  4, 3-1 8,  7--5,  4,  3,  2,  1  8,  7,  6,  5,  4, 3-1 8,  7---1
*
*
*                         --- --- --- --- ---    ---      --- --- --- --- --- --- --- --- ---
* ADDR FO  FI  FGP IMB RGP DOE IRW ADL FRW ED1 KS  IER SWPA NC2 NC1 CCR EXT RRE LD2 SJM IST DBY CS LD   AC   * COMMENTS
*----- --- --- --- --- --- ---- --- --- --- --- ---   --- --- --- --- --- --- --- --- --- --- -- -------- --------
(03CH):FF0  ILR  R33           K1                              RRE    SJM         NAN  JCC(13CH); XTHL
(13CH):FF1  LMI  R33  IRW      K1                        EXT                  DBY AN   JPX(130H); ROW=13
*
(13BH):     NOP  RFF  IRW      K1                        EXT                      NAN  JCR(13AH);
(13AH):FF0  LMI  R33      ADL  K1                                             DBY NAN  JPR(130H); ROW=13
(133H):FF0  CSR  R22           K1                                             DBY AN   JCC(123H);
(123H):     TZR  R22           KM            NC2     CCR EXT                      NAN  JCR(122H);
(122H):FF0  DSM  R33      ADL  K0                                             DBY NAN  JCR(125H);
(125H):     TZR  R22           KM                NC1 CCR EXT                      NAN  JCR(124H);
(124H):FF0  LMI  R33 DOE       K1    MEMW                                         NAN  JCR(127H);
(127H):     NOP  RFF DOE       K1    MEMW               EXT                       NAN  JCR(126H);
(126H):FF1  DSM  R44 DOE  ADL ED1 K0 MEMW                                     DBY NAN  JCR(121H);
(121H):FF1  DSM  R44 DOE      ED1 K0 MEMW           CCR EXT                    DBY AN   JCR(120H);
(120H):FF1  LMI  R44           K1    NOP                                      DBY AN   JCC(080H);
(080H):FF1  LMI  R44      ADL  K1                                         IST DBY AN   JZR07;
```

```
*----- --- --- --- --- -----                                                            ___  _____
*     FF0 HCZ -  IFF  -   1   1   1   1 ED2 -   1 MEMR  1   1   1   1   1   1   1   1   1  -  0   1'S   ; DEFAULTS
*-------------------------------------------------------------------------------------------------------------
*
*
*
*
*
*
*
* GROUP 00:  RRC
*
*         PROM U7   PROM U8       PROM U2                   PROM U5                  PROM U6              PROM U4
*       V---------V V-----V V----------------------V V--------------------------V V------------------V V---------V
*       8-7,6-5,4-1 8-6,5-1  8,   7,   6,   5,   4, 3-1  8,  7--5,  4,  3,  2,  1   8,  7,  6,  5,  4, 3-1  8,   7---1
*
*                           --- --- --- --- ---     ---   --- --- --- --- --- --- --- --- ---
* ADDR FO  FI  FGP IMB RGP  DOE IRW ADL FRW ED1 KS   IER SWPA NC2 NC1 CCR EXT RRE LD2 SJM IST DBY  CS  LD    AC    * COMMENTS
*----- --- --- --- --- -----                                                            ___  _____
(03DH):FF1 STC SRA     REE                  K1       NC2 NC1                                      CN    JCC(14DH); RRC
(14DH):FFC     SRA     REE      IRW          K1              EXT                             ISE  BN    JZR06     ;
*----- --- --- --- --- -----                                                            ___  _____
*     FF0 HCZ -  IFF  -   1   1   1   1 ED2 -   1 MEMR  1   1   1   1   1   1   1   1   1  -  0   1'S   ; DEFAULTS
*-------------------------------------------------------------------------------------------------------------
*
* GROUP 0F:  MVI R;   R = B, C, D, E, H, L
*
*         PROM U7   PROM U8       PROM U2                   PROM U5                  PROM U6              PROM U4
*       V---------V V-----V V----------------------V V--------------------------V V------------------V V---------V
*       8-7,6-5,4-1 8-6,5-1  8,   7,   6,   5,   4, 3-1  8,  7--5,  4,  3,  2,  1   8,  7,  6,  5,  4, 3-1  8,   7---1
*
*                           --- --- --- --- ---     ---   --- --- --- --- --- --- --- --- ---
* ADDR FO  FI  FGP IMB RGP  DOE IRW ADL FRW ED1 KS   IER SWPA NC2 NC1 CCR EXT RRE LD2 SJM IST DBY  CS  LD    AC    * COMMENTS
*----- --- --- --- --- -----
(03FH):FF1 LDI     RFF                       KM                          EXT         SJM        AN    JCR(03EH); MVI
(03EH):FF1 LMI     R44           ADL         K1                                           DBY  AN  LD ;(TO 07E OR 07D)
*
(07EH):FF1 SDR     R33      IRW              K0              NC1 EXT RRE         IST      AN    JZR06  ;MVI B,D,H
(07DH):FF1 SDR     R33      IRW              K0          NC2     EXT RRE         IST      AN    JZR06  ;MVI C,E,L
*----- --- --- --- --- -----
*     FF0 HCZ -  IFF  -   1   1   1   1 ED2 -   1 MEMR  1   1   1   1   1   1   1   1   1  -  0   1'S   ; DEFAULTS
*-------------------------------------------------------------------------------------------------------------
*
*
*
*
*
*
* GROUP 10:  CALL
*
*         PROM U7   PROM U8       PROM U2                   PROM U5                  PROM U6              PROM U4
*       V---------V V-----V V----------------------V V--------------------------V V------------------V V---------V
*       8-7,6-5,4-1 8-6,5-1  8,   7,   6,   5,   4, 3-1  8,  7--5,  4,  3,  2,  1   8,  7,  6,  5,  4, 3-1  8,   7---1
*
*                           --- --- --- --- ---     ---   --- --- --- --- --- --- --- --- ---
* ADDR FO  FI  FGP IMB RGP  DOE IRW ADL FRW ED1 KS   IER SWPA NC2 NC1 CCR EXT RRE LD2 SJM IST DBY  CS  LD    AC    * COMMENTS
*----- --- --- --- --- -----
(040H):FF0 ILR     R44                       K1                                                NAN   JCC(100H); CALL
(100H):FF0 CSR     R44                       K1                                                NAN   JCR(103H);
(103H):     F5     R44                       KM              NC2     CCR EXT                    NAN   JCR(102H);
(102H):FF0 DSM     R33           ADL         K0                                           DBY  NAN   JCR(101H);
(101H):FF0 LMI     R33                       K1                                                NAN   JCR(10FH);
(10FH):     F5     R44                       KM                  NC1 CCR EXT                    NAN   JCR(11EH);
(10EH):FF0 DSM     R33 DOE       ADL         K0      MEMW                                  DBY  NAN   JCC(11EH);
(11EH):FF0 LMI     R33 DOE                    K1      MEMW             EXT                       NAN   JCR(11CH);
(11DH):     NOP    RFF DOE                    K1      MEMW             EXT                       NAN   JCR(11CH);
(11CH):     NOP    RFF DOE       ADL     ED1 K1      MEMW                                        NAN   JCR(11BH);
(11BH):FF1 LMI     R44 DOE               ED1 K1      MEMW         CCR EXT                   DBY  AN    JCR(11AH);
```

```
(11AH):FF1   LMI   R44              ADL           K1                                              DBY AN    JZR07   ;
*
(0C0H):FF0   LMI   R44                            K1              EXT                       NAN    JCC(130H); NON-CALL
*      THE ABOVE INSTRUCTION IS EXECUTED FOR A CONDITIONAL CALL IN WHICH THE CONDITION IS NOT MET.         (TO 130 @ GP 0A)
*----- --- --- --- --- --- ---    --- --- --- --- --- --- --- --- --- --- --- --- --- -- --------- ---------
*      FF0 HCZ - IFF -   1  1  1   1 ED2 -   1 MEMR  1  1  1  1  1  1  1  1  1  -  0    1'S    ; DEFAULTS
*------------------------------------------------------------------------------------------------------------
```

EJECT;
*
* GROUP 11:  JMP
*
```
*        PROM U7   PROM U8       PROM U2                      PROM U5                    PROM U6          PROM U4
*        V---------V V-----V V-----------------------V V-------------------------V V---------------------V V---------V
*        8-7,6-5,4-1 8-6,5-1 8,   7,  6,  5,  4, 3-1 8,  7--5,  4,  3,  2,  1  8,  7,  6,  5,  4, 3-1 8,   7---1
*
*                            --- --- --- --- ---       ---     --- --- --- --- --- --- --- --- ---
* ADDR FO  FI  FGP IMB RGP  DOE IRW ADL FRW ED1 KS   IER SWPA  NC2 NC1 CCR EXT RRE LD2 SJM IST DBY  CS  LD     AC     * COMMENTS
*----- --- --- --- --- ---  --- --- --- --- --- ---  --- ----  --- --- --- --- --- --- --- --- ---  --  ---  --------- ---------
(041H):FF1   CSR   R44      IRW           K1                              EXT                       AN     JCC(111H); JMP
(111H):      NOP   RFF      IRW           K1                              EXT                       NAN    JCR(110H);
(110H):FF0   LMI   R44          ADL       KIR           NC2                                         NAN    JCR(113H);
(113H):FF0   LMI   R44      IRW           KIR                NC1 CCR EXT                             NAN    JCR(112H);
(112H):FF1   LMI   R44          ADL       K1                                              DBY AN    JCC(132H);
(132H):FF1   LMI   R44      IRW           K1                              EXT             IST DBY AN    JZR06   ;
*
(0C1H):FF0   LMI   R44                    K1                              EXT                       NAN    JCC(131H); NON-JMP
*      THE ABOVE INSTRUCTION IS EXECUTED FOR A CONDITIONAL JUMP IN WHICH THE CONDITION IS NOT MET.         (TO 131 @ GP 0A)
*----- --- --- --- --- ---  --- --- --- --- --- ---  --- ----  --- --- --- --- --- --- --- --- ---  --  ---  --------- ---------
*      FF0 HCZ - IFF -   1  1  1   1 ED2 -   1 MEMR  1  1  1  1  1  1  1  1  1  -  0    1'S    ; DEFAULTS
*------------------------------------------------------------------------------------------------------------
```
*
*
*
*
*
*
* GROUP 14:  RET
*
```
*        PROM U7   PROM U8       PROM U2                      PROM U5                    PROM U6          PROM U4
*        V---------V V-----V V-----------------------V V-------------------------V V---------------------V V---------V
*        8-7,6-5,4-1 8-6,5-1 8,   7,  6,  5,  4, 3-1 8,  7--5,  4,  3,  2,  1  8,  7,  6,  5,  4, 3-1 8,   7---1
*
*                            --- --- --- --- ---       ---     --- --- --- --- --- --- --- --- ---
* ADDR FO  FI  FGP IMB RGP  DOE IRW ADL FRW ED1 KS   IER SWPA  NC2 NC1 CCR EXT RRE LD2 SJM IST DBY  CS  LD     AC     * COMMENTS
*----- --- --- --- --- ---  --- --- --- --- --- ---  --- ----  --- --- --- --- --- --- --- --- ---  --  ---  --------- ---------
(044H):FF1   LMI   R33                    K1                              EXT                 DBY AN    JCR(042H); RET
(043H):      NOP   RFF                    K1                              EXT                     NAN    JCR(042H);
(042H):FF1   LMI   R33          ADL       K1                                              DBY AN    JCR(041H);(@ GP 11)
*
(0C4H):      NOP   RFF      IRW           K1                              EXT             IST     NAN    JZR06   ; NON-RET
*      THE ABOVE INSTRUCTION IS EXECUTED FOR A CONDITIONAL RETURN IN WHICH THE CONDITION IS NOT MET.
*----- --- --- --- --- ---  --- --- --- --- --- ---  --- ----  --- --- --- --- --- --- --- --- ---  --  ---  --------- ---------
*      FF0 HCZ - IFF -   1  1  1   1 ED2 -   1 MEMR  1  1  1  1  1  1  1  1  1  -  0    1'S    ; DEFAULTS
*------------------------------------------------------------------------------------------------------------
```
*
*
*
*
*
*

```
* GROUP 15:  RAR
*
*        PROM U7   PROM U8        PROM U2                   PROM U5                    PROM U6           PROM U4
*        V--------V V-----V V--------------------V V-------------------------V V-------------------V V---------V
*        8-7,6-5,4-1 8-6,5-1 8,  7,  6,  5,  4, 3-1 8,  7--5,  4,  3,  2,  1  8,  7,  6,  5,  4, 3-1 8,  7---1
*
*
*                     ---  --- --- --- ---       ---    --- --- --- --- --- --- --- --- --- ---
* ADDR FO  FI  FGP IMB RGP  DOE IRW ADL FRW ED1 KS   IER SWPA NC2 NC1 CCR EXT RRE LD2 SJM IST DBY CS  LD    AC       * COMMENTS
*----- --- --- --- --- ---  --- --- --- --- --- --   --- ---- --- --- --- --- --- --- --- --- --- --  --- --------- ---------
(045H):FFC STC SRA     REE      IRW             K1                             EXT             IST     BN    JZR07   ; RAR
*----- --- --- --- --- ---  --- --- --- --- --- --   --- ---- --- --- --- --- --- --- --- --- --- --  --- --------- ---------
*      FF0 HCZ  -  IFF  -    1   1   1   1  ED2  -    1  MEMR  1   1   1   1   1   1   1   1   1   -   0   1'S    ; DEFAULTS
*------------------------------------------------------------------------------------------------------------------
```

```
* GROUP 16:  PUSH B, D, H;   PUSH PSW
*
*        PROM U7   PROM U8        PROM U2                   PROM U5                    PROM U6           PROM U4
*        V--------V V-----V V--------------------V V-------------------------V V-------------------V V---------V
*        8-7,6-5,4-1 8-6,5-1 8,  7,  6,  5,  4, 3-1 8,  7--5,  4,  3,  2,  1  8,  7,  6,  5,  4, 3-1 8,  7---1
*
*
*                     ---  --- --- --- ---       ---    --- --- --- --- --- --- --- --- --- ---
* ADDR FO  FI  FGP IMB RGP  DOE IRW ADL FRW ED1 KS   IER SWPA NC2 NC1 CCR EXT RRE LD2 SJM IST DBY CS  LD    AC       * COMMENTS
*----- --- --- --- --- ---  --- --- --- --- --- --   --- ---- --- --- --- --- --- --- --- --- --- --  --- --------- ---------
(046H):FF0     DSM     R33                      K0                                 LD2            DBY NAN LD   ;(T0 0C6 OR 0CD)
*
(0C6H):FF0     ILR     R33                      K1                         RRE                    NAN      JCR(0C9H); * PUSH
(0C9H):FF0     LMI     R33      IRW             K1                     EXT                         NAN      JCR(0C8H);   B,D,H
(0C8H):FF0     DSM     R33 DOE      ADL         K0      MEMW                                   DBY NAN      JCR(0C7H);
(0C7H):FF0     LMI     R33 DOE                  K1      MEMW           EXT                         NAN      JCC(127H);(@ GP 0C)
*
(0CDH):FF1     ACM     RCB      IRW             K1                     EXT                         NAN      JCR(0CCH); PUSH PSW
(0CCH):FF0     LMI     R33                      K1     NOP                                         NAN      JCR(0C8H);
*----- --- --- --- --- ---  --- --- --- --- --- --   --- ---- --- --- --- --- --- --- --- --- --- --  --- --------- ---------
*      FF0 HCZ  -  IFF  -    1   1   1   1  ED2  -    1  MEMR  1   1   1   1   1   1   1   1   1   -   0   1'S    ; DEFAULTS
*------------------------------------------------------------------------------------------------------------------
```

```
* GROUP 17:  MVI M
*
*        PROM U7   PROM U8        PROM U2                   PROM U5                    PROM U6           PROM U4
*        V--------V V-----V V--------------------V V-------------------------V V-------------------V V---------V
*        8-7,6-5,4-1 8-6,5-1 8,  7,  6,  5,  4, 3-1 8,  7--5,  4,  3,  2,  1  8,  7,  6,  5,  4, 3-1 8,  7---1
*
*
*                     ---  --- --- --- ---       ---    --- --- --- --- --- --- --- --- --- ---
* ADDR FO  FI  FGP IMB RGP  DOE IRW ADL FRW ED1 KS   IER SWPA NC2 NC1 CCR EXT RRE LD2 SJM IST DBY CS  LD    AC       * COMMENTS
*----- --- --- --- --- ---  --- --- --- --- --- --   --- ---- --- --- --- --- --- --- --- --- --- --  --- --------- ---------
(047H):FF0     LMI     R22                      K1                                                NAN      JCC(117H); MVI M
(117H):FF1     LDI     RFF      IRW             KIR                    EXT                         AN      JCR(116H);
(116H):FF0     DSM     R44 DOE      ADL     ED1 K0      MEMW                                   DBY NAN      JCR(11BH);(@ GP 10)
*----- --- --- --- --- ---  --- --- --- --- --- --   --- ---- --- --- --- --- --- --- --- --- --- --  --- --------- ---------
*      FF0 HCZ  -  IFF  -    1   1   1   1  ED2  -    1  MEMR  1   1   1   1   1   1   1   1   1   -   0   1'S    ; DEFAULTS
*------------------------------------------------------------------------------------------------------------------
```

```
EJECT;
*
*  GROUP 18:  NOP
*
*        PROM U7   PROM U8       PROM U2              PROM U5                  PROM U6              PROM U4
*        V--------V V-----V V---------------------V V-----------------------V V-------------------V V---------V
*        8-7,6-5,4-1 8-6,5-1 8,  7,  6,  5,  4, 3-1 8,  7--5,  4,  3,  2,  1  8,  7,  6,  5,  4, 3-1 8,  7---1
*
*                            --- --- --- --- ---        --- --- --- --- --- --- --- --- ---
* ADDR FO  FI  FGP IMB RGP  DOE IRW ADL FRW ED1 KS   IER SWPA NC2 NC1 CCR EXT RRE LD2 SJM IST DBY  CS  LD    AC      * COMMENTS
*----- --- --- --- --- ---  --- --- --- --- --- ---  --- ---- --- --- --- --- --- --- --- --- --- --- --- --------
(048H):         NOP         RFF         IRW         K1                       EXT                 IST     NAN  JZR06    ; NOP
*----- --- --- --- --- ---  --- --- --- --- --- ---  --- ---- --- --- --- --- --- --- --- --- --- --- --- --------
*     FF0 HCZ  -  IFF  -    1   1   1   1  ED2  -    1  MEMR  1   1   1   1   1   1   1   1   1   -   0   1'S   ; DEFAULTS
*------------------------------------------------------------------------------------------------------------------
*
*
*
*
*
*
*
*  GROUP 19:  LXI B, D, H, SP
*
*        PROM U7   PROM U8       PROM U2              PROM U5                  PROM U6              PROM U4
*        V--------V V-----V V---------------------V V-----------------------V V-------------------V V---------V
*        8-7,6-5,4-1 8-6,5-1 8,  7,  6,  5,  4, 3-1 8,  7--5,  4,  3,  2,  1  8,  7,  6,  5,  4, 3-1 8,  7---1
*
*                            --- --- --- --- ---        --- --- --- --- --- --- --- --- ---
* ADDR FO  FI  FGP IMB RGP  DOE IRW ADL FRW ED1 KS   IER SWPA NC2 NC1 CCR EXT RRE LD2 SJM IST DBY  CS  LD    AC      * COMMENTS
*----- --- --- --- --- ---  --- --- --- --- --- ---  --- ---- --- --- --- --- --- --- --- --- --- --- --- --------
(049H):FF0     CSR         R33                 K1                           RRE             DBY NAN  JCC(0A9H);LXI B, D,
*                                                                                               (TO 0A9 @ GP 28)   H, SP
*----- --- --- --- --- ---  --- --- --- --- --- ---  --- ---- --- --- --- --- --- --- --- --- --- --- --- --------
*     FF0 HCZ  -  IFF  -    1   1   1   1  ED2  -    1  MEMR  1   1   1   1   1   1   1   1   1   -   0   1'S   ; DEFAULTS
*------------------------------------------------------------------------------------------------------------------




EJECT;
*
*
*
*
*
*  GROUP 1A:  RST
*
*        PROM U7   PROM U8       PROM U2              PROM U5                  PROM U6              PROM U4
*        V--------V V-----V V---------------------V V-----------------------V V-------------------V V---------V
*        8-7,6-5,4-1 8-6,5-1 8,  7,  6,  5,  4, 3-1 8,  7--5,  4,  3,  2,  1  8,  7,  6,  5,  4, 3-1 8,  7---1
*
*                            --- --- --- --- ---        --- --- --- --- --- --- --- --- ---
* ADDR FO  FI  FGP IMB RGP  DOE IRW ADL FRW ED1 KS   IER SWPA NC2 NC1 CCR EXT RRE LD2 SJM IST DBY  CS  LD    AC      * COMMENTS
*----- --- --- --- --- ---  --- --- --- --- --- ---  --- ---- --- --- --- --- --- --- --- --- --- --- --- --------
(04AH):FF0     DSM         R33                 K0                                           DBY NAN  JCC(10AH); RST
(10AH):FF0     LMI         R33                 K1                                               NAN  JCR(107H);
(107H):FF0     LDI         RFF                 KD                           EXT               DBY NAN  JCR(106H);
(106H):FF0     ALR         R44                 K0   NSTAT                                     DBY NAN  JCR(10CH);
(10CH):FF0     DSM         R33 DOE     ADL     KD   MEMW                                      DBY NAN  JCR(10BH);
(10BH):FF0     LMI         R33 DOE             K1   MEMW                                          NAN  JCR(109H);
(109H):FF1     LDI IC7 RFF DOE             KIR  MEMW          NC1 CCR EXT                      AN   JCR(108H);
(108H):FF1     CSR         R44 DOE     ADL ED1 K1   MEMW                                       AN   JCR(105H);
(105H):FF0     LMI         R44 DOE         ED1 K1   MEMW                  EXT                  CS1  JCR(104H);
```

```
(104H):FF0    LMI    R44                    KD       NSTAT NC2                                    NAN     JFL(112H);(@ GP 11)
*------ --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- -- -- -------- ----------
*    FF0 HCZ - IFF -  1  1  1  1 ED2 -  1 MEMR  1  1  1  1  1  1  1  1  -  0   1'S   ; DEFAULTS
*------------------------------------------------------------------------------------------------------------------




*
*
*
* GROUP 1B: MOV R,A; RLC; RAL; STAX; MOV M,A; ADI; ACI; SUI; SBI; ANI; XRI; ORI; CPI; MVI A
*
*    PROM U7  PROM U8    PROM U2              PROM U5                  PROM U6          PROM U4
*    V-------V V-----V V--------------------V V------------------------V V----------------V V--------V
*    8-7,6-5,4-1 8-6,5-1 8,  7,  6,  5,  4, 3-1 8,  7--5,  4,  3,  2,  1 8,  7,  6,  5,  4, 3-1 8,  7---1
*
*
* ADDR FO FI FGP IMB RGP DOE IRW ADL FRW ED1 KS  IER SWPA NC2 NC1 CCR EXT RRE LD2 SJM IST DBY CS  LD    AC    * COMMENTS
*----- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- -- -- -------- ----------
(04BH):FF1     ILR    RCC              K1                                                  NAN  LD  ;MOV R,A; ARITH I
*
(012H):FF0 STC ALR    RCC              K0                                                  AN   JCC(0C2H); RLC
(0C2H):FFC     ILR    RCC    IRW       K1                       EXT        IST             AN   JZR06   ;
*
(013H):FFC STC ALR    RCC    IRW       K0                       EXT        IST             AN   .JZR07  ; RAL
*
(01EH):FF0     LMI    R33    IRW       K1                       EXT RRE                     NAN  JCC(12EH); STAX,
*                                                                                                  MOV M,A
(061H):FF0 SCZ AIA    REE              KM               CCR EXT                             AN   JCR(060H); ADI
(060H):FF0     ILR    RCC        ADL   K1                                                  NAN  JZR08;(BELOW)
*
(063H):FFC SCZ AIA    REE              KM               CCR EXT                             AN   JCR(062H); ACI
(062H):FF0     ILR    RCC        ADL   K1                                                  NAN  JZR08;(BELOW)
*
(065H):FF0 SCZ AIA    REE              KNM              CCR EXT                             NCN  JCR(064H); SUI
(064H):FF0     ILR    RCC        ADL   K1                                                  NAN  JZR08;(BELOW)
*
(067H):FFC SCZ AIA    REE              KNM              CCR EXT                             NCN  JCR(066H); SBI
(066H):FF0     ILR    RCC        ADL   K1                                                  NAN  JZR08;(BELOW)
*
(069H):FF0 SCZ ANI    REE              KM               CCR EXT                             CS1  JCR(068H); ANI
(068H):FF0     ILR    RCC        ADL   K1                                                  NAN  JZR08;(BELOW)
*
(06BH):FF0 SCZ XNI    REE              KNM              CCR EXT                             CS1  JCR(06AH); XRI
(06AH):FF0     ILR    RCC        ADL   K1                                                  NAN  JZR08;(BELOW)
*
(06DH):FF0 SCZ ORI    REE              KM               CCR EXT                             CS1  JCR(06CH); ORI
(06CH):FF0     ILR    RCC        ADL   K1                                                  NAN  JZR08;(BELOW)
*
(06FH):FF0     ILR    RCC    IRW       K1                       EXT                         NAN  JCR(06EH); CPI
(06EH):FF1 SCZ AIA    RFF        ADL   KNIR                                                 NCN  JZR08;(BELOW)
*
(008H):FF1     LMI    R44    IRW       KD                       EXT        IST DBY AN            JZR06;
*
(07FH):FF1     LDM    REE    IRW       KD                       EXT        IST     AN            JZR06   ; MVI A
*----- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- -- -- -------- ----------
*    FF0 HCZ - IFF -  1  1  1  1 ED2 -  1 MEMR  1  1  1  1  1  1  1  1  -  0   1'S   ; DEFAULTS
*------------------------------------------------------------------------------------------------------------------
```

**Signetics**

```
EJECT;
*
*
*
*
*
*
*
*
*
* GROUP 1C:  MOV M,R;  WHERE R = A, B, C, D, E, H, L.
*
*      PROM U7    PROM U8       PROM U2                    PROM U5                      PROM U6            PROM U4
*    V--------V V-----V V--------------------V V------------------------V V------------------V V--------V
*    8-7,6-5,4-1 8-6,5-1 8,  7,  6,  5,  4, 3-1 8,  7--5, 4,  3,  2,  1 8,  7,  6,  5,  4, 3-1 8,  7---1
*
*                        --- --- --- --- ---     ---     --- --- --- --- --- --- --- --- ---
* ADDR FO  FI  FGP IMB RGP DOE IRW ADL FRW ED1 KS  IER SWPA NC2 NC1 CCR EXT RRE LD2 SJM IST DBY CS LD   AC    * COMMENTS
*----- --- --- --- --- --- --- --- --- --- --- --- --- ---- --- --- --- --- --- --- --- --- --- -- --- -------- --------
(04CH):FF0   LMI   R22                   K1                                          SJM        NAN   JCC(12CH); MOV M,R
(12CH):FF0   ILR   R33       IRW         K1                      EXT RRE                        NAN   JPX(120H);

(12FH):      NOP   RFF   IRW                                     EXT                            NAN   JCR(12EH);R=C,E,L,A
(12EH):FF1   DSM   R44 DOE   ADL     ED1 K0       MEMW                              DBY NAN   JCR(121H);(@ GP 0C)

(12BH):      NOP   RFF   IRW             K1                      EXT                            NAN   JCR(12AH); R=B,D,H
(12AH):FF1   DSM   R44 DOE   ADL         K0       MEMW                          DBY NAN   JCR(129H);
(129H):FF1   DSM   R44 DOE               K0       MEMW          CCR EXT         DBY AN   JCR(128H);
(128H):FF1   LMI   R44                   K1       NSTAT                         DBY AN   JCC(088H);
(088H):FF1   LMI   R44       ADL         K1                                 IST DBY AN   JZR07;
*----- --- --- --- --- --- --- --- --- --- --- --- --- ---- --- --- --- --- --- --- --- --- --- -- --- -------- --------
*     FF0 HCZ  -  IFF  -   1   1   1   1 ED2  -   1 MEMR  1   1   1   1   1   1   1   1   1   -   0  1'S   ; DEFAULTS
*-----------------------------------------------------------------------------------------------------------------
*
*
*
*
*
*
*
* GROUP 1D:  MOV R1,R2;  WHERE R1 = B, D, H;  AND R2 = C, E, L.
*
*      PROM U7    PROM U8       PROM U2                    PROM U5                      PROM U6            PROM U4
*    V--------V V-----V V--------------------V V------------------------V V------------------V V--------V
*    8-7,6-5,4-1 8-6,5-1 8,  7,  6,  5,  4, 3-1 8,  7--5, 4,  3,  2,  1 8,  7,  6,  5,  4, 3-1 8,  7---1
*
*                        --- --- --- --- ---     ---     --- --- --- --- --- --- --- --- ---
* ADDR FO  FI  FGP IMB RGP DOE IRW ADL FRW ED1 KS  IER SWPA NC2 NC1 CCR EXT RRE LD2 SJM IST DBY CS LD   AC    * COMMENTS
*----- --- --- --- --- --- --- --- --- --- --- --- --- ---- --- --- --- --- --- --- --- --- --- -- --- -------- --------
(04DH):FF0   CSRILR R33                  K1                          RRE                        NAN   JCC(10DH);
(10DH):      F5    R33       IRW         ED1 KD              NC1 EXT RRE              IST        NAN   JZR06   ;
*----- --- --- --- --- --- --- --- --- --- --- --- --- ---- --- --- --- --- --- --- --- --- --- -- --- -------- --------
*     FF0 HCZ  -  IFF  -   1   1   1   1 ED2  -   1 MEMR  1   1   1   1   1   1   1   1   1   -   0  1'S   ; DEFAULTS
*-----------------------------------------------------------------------------------------------------------------
```

```
EJECT;
*
*
*
*
*
*
*
*  GROUP 1E:  EI (ENABLE INTERRUPTS)
*
*        PROM U7    PROM U8        PROM U2                        PROM U5                        PROM U6              PROM U4
*       V--------V V----V V-------------------V V----------------------V V------------------V V--------V
*       8-7,6-5,4-1 8-6,5-1 8,  7,  6,  5,  4, 3-1 8,  7--5,  4,  3,  2,  1  8,  7,  6,  5,  4, 3-1 8,  7---1
*
*                          --- --- --- --- ---    ---      --- --- --- --- --- --- --- --- ---
* ADDR FO  FI  FGP IMB RGP DOE IRW ADL FRW ED1 KS     IER SWPA  NC2 NC1 CCR EXT RRE LD2 SJM IST DBY CS  LD    AC    * COMMENTS
*---- --- --- --- --- --- --- --- --- --- --- ---    --- ----  --- --- --- --- --- --- --- --- --- --  -- --------- ---------
(04EH):FF0     NOP     RFF     IRW             K1 IER                    EXT                         NAN    JCC(13EH);(@ GP 1F)
*---- --- --- --- --- --- --- --- --- --- --- ---    --- ----  --- --- --- --- --- --- --- --- --- --  -- --------- ---------
*      FF0 HCZ  -  IFF  -   1   1   1   1 ED2  -     1 MEMR   1   1   1   1   1   1   1   1   1   -   0   1'S   ; DEFAULTS
*-------------------------------------------------------------------------------------------------------------------------
*
*
*
*
*
*
*  GROUP 1F:  DI (DISABLE INTERRUPTS)
*
*        PROM U7    PROM U8        PROM U2                        PROM U5                        PROM U6              PROM U4
*       V--------V V----V V-------------------V V----------------------V V------------------V V--------V
*       8-7,6-5,4-1 8-6,5-1 8,  7,  6,  5,  4, 3-1 8,  7--5,  4,  3,  2,  1  8,  7,  6,  5,  4, 3-1 8,  7---1
*
*                          --- --- --- --- ---    ---      --- --- --- --- --- --- --- --- ---
* ADDR FO  FI  FGP IMB RGP DOE IRW ADL FRW ED1 KS     IER SWPA  NC2 NC1 CCR EXT RRE LD2 SJM IST DBY CS  LD    AC    * COMMENTS
*---- --- --- --- --- --- --- --- --- --- --- ---    --- ----  --- --- --- --- --- --- --- --- --- --  -- --------- ---------
(04FH):FF1     NOP     RFF     IRW             K1 IER                    EXT                         CS0    JCC(13FH); DI
(13FH):        NOP     RFF     IRW             K1                        EXT                         NAN    JCR(13EH);
(13EH):FF1     LMI     R44         ADL         K1                                            SJM     DBY AN LD JPX       ;
* PREVIOUS 2 INST. ARE A FETCH IDENTICAL TO 007 AND 006, EXCEPT FOR THE ABSENCE OF "IST" TO PREVENT INTERRUPTS.
*---- --- --- --- --- --- --- --- --- --- --- ---    --- ----  --- --- --- --- --- --- --- --- --- --  -- --------- ---------
*      FF0 HCZ  -  IFF  -   1   1   1   1 ED2  -     1 MEMR   1   1   1   1   1   1   1   1   1   -   0   1'S   ; DEFAULTS
*-------------------------------------------------------------------------------------------------------------------------




*
*  GROUP 20:  DAA (DECIMAL ADJUST ACCUMULATOR)
*
*        PROM U7    PROM U8        PROM U2                        PROM U5                        PROM U6              PROM U4
*       V--------V V----V V-------------------V V----------------------V V------------------V V--------V
*       8-7,6-5,4-1 8-6,5-1 8,  7,  6,  5,  4, 3-1 8,  7--5,  4,  3,  2,  1  8,  7,  6,  5,  4, 3-1 8,  7---1
*
*                          --- --- --- --- ---    ---      --- --- --- --- --- --- --- --- ---
* ADDR FO  FI  FGP IMB RGP DOE IRW ADL FRW ED1 KS     IER SWPA  NC2 NC1 CCR EXT RRE LD2 SJM IST DBY CS  LD    AC    * COMMENTS
*---- --- --- --- --- --- --- --- --- --- --- ---    --- ----  --- --- --- --- --- --- --- --- --- --  -- --------- ---------
(050H):FF1     ILRACM  RCB             K1                                                            NAN    JCC(0A0H); DAA
(0A0H):FF0 SCZ AIA I99 RFF            K0                   NC1                                        AN     JCC(0B0H);
(0B0H):FF0     ORM     RBB            K0              NC2                                             CS0    JCR(0B1H);
(0B1H):FF0 SCZ ILR     RCC         ED1 K1                  NC1                     SJM                AN     JCE(0A1H);
(0A1H):FF1     LDI IBB RFF            K0              NC2                                             AN  LD ;(TO 028 @ GP 01)
*---- --- --- --- --- --- --- --- --- --- --- ---    --- ----  --- --- --- --- --- --- --- --- --- --  -- --------- ---------
*      FF0 HCZ  -  IFF  -   1   1   1   1 ED2  -     1 MEMR   1   1   1   1   1   1   1   1   1   -   0   1'S   ; DEFAULTS
*-------------------------------------------------------------------------------------------------------------------------
```

```
*
*
*
*
*
*
*
* GROUP 21: CMC (COMPLIMENT CARRY)
*
*       PROM U7   PROM U8      PROM U2              PROM U5                PROM U6          PROM U4
*      V--------V V-----V V-------------------V V-------------------------V V-------------------V V---------V
*      8-7,6-5,4-1 8-6,5-1 8,  7,  6,  5,  4, 3-1 8,  7--5,  4,  3,  2,  1  8,  7,  6,  5,  4, 3-1 8,  7---1
*
*                         --- --- --- --- ---     ---   --- --- --- --- --- --- --- --- --- --- ---
* ADDR FO  FI  FGP IMB RGP DOE IRW ADL FRW ED1 KS  IER SWPA NC2 NC1 CCR EXT RRE LD2 SJM IST DBY CS  LD    AC   * COMMENTS
*----- --- --- --- --- --- --- --- --- --- --- --- ----- --- --- --- --- --- --- --- --- --- --- -- -------- --------
(051H):FFC     CAS     RFF                 K1                                                  NAN     JCC(151H); CMC
(151H):    STC SRA     RFF     IRW         K1                         EXT             IST       NAN     JZR07    ;
*----- --- --- --- --- --- --- --- --- --- --- --- ----- --- --- --- --- --- --- --- --- --- --- -- -------- --------
*      FF0 HCZ  -  IFF  -   1   1   1   1 ED2  -   1  MEMR  1   1   1   1   1   1   1   1   1  -   0   1'S   ; DEFAULTS
*----------------------------------------------------------------------------------------------------------------------


EJECT;
*
*
* GROUP 22:  ADD M,  ADC M,  SUB M,  SBB M,  ANA M,  XRA M,  ORA M,  CMP M.
*
*       PROM U7   PROM U8      PROM U2                  PROM U5                PROM U6          PROM U4
*      V--------V V-----V V-------------------V V-------------------------V V-------------------V V---------V
*      8-7,6-5,4-1 8-6,5-1 8,  7,  6,  5,  4, 3-1 8,  7--5,  4,  3,  2,  1  8,  7,  6,  5,  4, 3-1 8,  7---1
*
*                         --- --- --- --- ---     ---   --- --- --- --- --- --- --- --- --- --- ---
* ADDR FO  FI  FGP IMB RGP DOE IRW ADL FRW ED1 KS  IER SWPA NC2 NC1 CCR EXT RRE LD2 SJM IST DBY CS  LD    AC   * COMMENTS
*----- --- --- --- --- --- --- --- --- --- --- --- ----- --- --- --- --- --- --- --- --- --- --- -- -------- --------
(052H);FF0     DSM     R44                 K0                                              DBY NAN     JPX(055H); ARITH M
(055H):FF0     LMI     R22     IRW         K1                         EXT                      NAN     JCR(054H);
(054H):FF1     LMI     R44         ADL     K1                                              DBY NAN     JCC(094H);
(094H):FF0     ILR     RCC                 K1                                                  NAN     JPR(0E0H);(ROW = E)

(0E1H);FF0 SCZ AIA     RFF                 KM              CCR EXT                           AN     JCR(0E0H); ADD M
(0E0H):FF1     SDR     RCC             FRW K0  NSTAT                                IST       AN     JZR06;

(0E3H):FFC SCZ AIA     RFF                 KM              CCR EXT                           AN     JCR(0E2H); ADC M
(0E2H):FF1     SDR     RCC             FRW K0  NSTAT                                IST       AN     JZR06;

(0E5H):FF0 SCZ AIA     RFF                 KNM             CCR EXT                          NCN     JCR(0E4H); SUB M
(0E4H):FF1     SDR     RCC             FRW K0                                       IST       AN     JZR06;

(0E9H):FFC SCZ AIA     RFF                 KNM             CCR EXT                          NCN     JCR(0E8H); SBB M
(0E8H):FF1     SDR     RCC             FRW K0  NSTAT                                IST       AN     JZR06;

(0E7H):FF0 SCZ F5      RFF                 KM              CCR EXT                          CS1     JCR(0E6H); ANA M
(0E6H):FF1     SDR     RCC             FRW K0  NSTAT                                IST       AN     JZR06;

(0EBH):FF0 SCZ XNI     RFF                 KNM             CCR EXT                          CS1     JCR(0EAH); XRA M
(0EAH):FF1     SDR     RCC             FRW K0  NSTAT                                IST       AN     JZR06;

(0EDH):FF0 SCZ F6      RFF                 KM              CCR EXT                          CS1     JCR(0ECH); ORA M
(0ECH):FF1     SDR     RCC             FRW K0  NSTAT                                IST       AN     JZR06;

(0EFH):FF0 SCZ AIA     RFF                 KNM             CCR EXT                          NCN     JCR(0EEH); CMP M
(0EEH):FF1     NOP     R00             FRW K1  NSTAT                                IST       AN     JZR06;
```

```
*----- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- -- --------- ---------
*    FF0 HCZ  -  IFF  -   1  1  1  1 ED2  -   1 MEMR   1  1  1  1  1  1  1  1  1  -  0   1'S   ; DEFAULTS
*---------------------------------------------------------------------------------------------------
```

EJECT.
*
* GROUP 23: LDAX, MOV A M.
*
*        PROM U7    PROM U8        PROM U2                    PROM U5                PROM U6        · PROM U4
*       V--------V V-----V V---------------------V V---------------------V V-------------------V V----------V
*       8-7,6-5,4-1 8-6,5-1  8,   7,  6,  5,  4, 3-1 8,  7--5,  4,  3,  2,  1  8,  7,  6,  5,  4, 3-1 8,  7---1
*
*
* ADDR FO  FI  FGP IMB RGP  DOE IRW ADL FRW ED1 KS   IER SWPA  NC2 NC1 CCR EXT RRE LD2 SJM IST DBY  CS  LD     AC     * COMMENTS
*----- --- --- --- --- --- --- --- --- --- --- ---  --- ----  --- --- --- --- --- --- --- --- ---  --- -- --------- ---------
(053H):FF0  LMI     R33     IRW         K1                          EXT RRE             NAN     JCC(0F3H);
(0F3H):     NOP     RFF     IRW         K1                          EXT                 NAN     JCR(0F2H);
(0F2H):FF0  DSM     R44         ADL     K0                                          DBY NAN     JCR(0F1H);
(0F1H):FF1  LDI     REE                 KM                          EXT                 AN      JCR(0F0H);
(0F0H):FF1  LMI     R44                 K1    NSTAT                              IST DBY AN      JZR06;
*----- --- --- --- --- --- --- --- --- --- --- ---  --- ----  --- --- --- --- --- --- --- --- ---  --- -- --------- ---------
*    FF0 HCZ  -  IFF  -   1  1  1  1 ED2  -   1 MEMR   1  1  1  1  1  1  1  1  1  -  0   1'S   ; DEFAULTS
*---------------------------------------------------------------------------------------------------
*
*
*
*
*
*
* GROUP 27: HLT (HALT)
*
*        PROM U7    PROM U8        PROM U2                    PROM U5                PROM U6        PROM U4
*       V--------V V-----V V---------------------V V---------------------V V-------------------V V----------V
*       8-7,6-5,4-1 8-6,5-1  8,   7,  6,  5,  4, 3-1 8,  7--5,  4,  3,  2,  1  8,  7,  6,  5,  4, 3-1 8,  7---1
*
*
* ADDR FO  FI  FGP IMB RGP  DOE IRW ADL FRW ED1 KS   IER SWPA  NC2 NC1 CCR EXT RRE LD2 SJM IST DBY  CS  LD     AC     * COMMENTS
*----- --- --- --- --- --- --- --- --- --- --- ---  --- ----  --- --- --- --- --- --- --- --- ---  --- -- --------- ---------
(057H):     NOP     RFF                 K1                          EXT                 NAN     JCR(056H); HLT
(056H):     NOP     RFF                 K1    HLTA                              IST     NAN     JZR01;
*----- --- --- --- --- --- --- --- --- --- --- ---  --- ----  --- --- --- --- --- --- --- --- ---  --- -- --------- ---------
*    FF0 HCZ  -  IFF  -   1  1  1  1 ED2  -   1 MEMR   1  1  1  1  1  1  1  1  1  -  0   1'S   ; DEFAULTS
*---------------------------------------------------------------------------------------------------
*
*
*
* GROUP 28: IN, OUT.
*
*        PROM U7    PROM U8        PROM U2                    PROM U5                PROM U6        · PROM U4
*       V--------V V-----V V---------------------V V---------------------V V-------------------V V----------V
*       8-7,6-5,4-1 8-6,5-1  8,   7,  6,  5,  4, 3-1 8,  7--5,  4,  3,  2,  1  8,  7,  6,  5,  4, 3-1 8,  7---1
*
*
* ADDR FO  FI  FGP IMB RGP  DOE IRW ADL FRW ED1 KS   IER SWPA  NC2 NC1 CCR EXT RRE LD2 SJM IST DBY  CS  LD     AC     * COMMENTS
*----- --- --- --- --- --- --- --- --- --- --- ---  --- ----  --- --- --- --- --- --- --- --- ---  --- -- --------- ---------
(058H):FF1  CSR     R99                 K1                              LD2             AN  LD     ; IN OUT
*                                                                                                  (T0 0BF OR 0B9)
```

```
(0BFH):FF0    LMI    R99                    KM                  CCR EXT                      NAN   JCR(0BEH); IN
(0BEH):FF0    DSM    R44           ADL      K0    IOR                                    DBY NAN   JCR(0BBH);
(0BBH):FF1    LMI    R44                    K1    IOR                                    DBY AN    JCR(0BDH);
(0BDH):FF1    LDI    REE                    KM    IOR              EXT                       AN    JCR(0BCH);
(0BCH):FF1    LMI    R44           ADL      K1                                           DBY AN    JZR07;

(0B9H):FF0    LMI    R99                    KM                  CCR EXT                      NAN   JCR(0B8H); OUT
(0B8H):FF0    ILR    RCC                    K1    NSTAT                                       NAN   JCR(0BAH);
(0BAH):FF0    DSM    R44  DOE      ADL      K0    IOW                                    DBY NAN   JCR(0B7H);
(0B7H):FF1    LMI    R44  DOE               K1    IOW              CCR EXT               DBY AN    JCR(0B6H);
(0B6H):FF1    LMI    R44           ADL      K1                                           DBY AN    JZR07;
*----- --- --- --- --- --- ---- --- --- --- --- --- ----- --- --- --- ---- --- --- --- --- --- ---- -- --------- ----------
*     FF0 HCZ -  IFF -   1   1   1   1  ED2  -    1  MEMR   1   1   1   1   1   1   1   1   1   -   0  1'S   ; DEFAULTS
*-----------------------------------------------------------------------------------------------------------------------
*
*
*
*
*
*
*
* GROUP 29:  SHLD (STORE H & L DIRECT),  STA (STORE A DIRECT)
*
*       PROM U7   PROM U8        PROM U2                      PROM U5                   PROM U6            PROM U4
*     V--------V V-----V V----------------------V V----------------------------V V---------------------V V---------V
*     8-7,6-5,4-1 8-6,5-1 8,  7,  6,  5,  4, 3-1 8,  7--5,  4,  3,  2,  1 8,  7,  6,  5,  4, 3-1 8,  7---1
*
*
*                              --- --- --- --- ---        --- --- --- --- --- --- --- --- ---
* ADDR FO  FI  FGP IMB RGP DOE IRW ADL FRW ED1 KS    IER SWPA NC2 NC1 CCR EXT RRE LD2 SJM IST DBY CS  LD     AC     * COMMENTS
*----- --- --- --- --- --- ---- --- --- --- --- --- ----- --- --- --- ---- --- --- --- --- --- ---- -- --------- ----------
(059H):FF0    ILR    R33                    K1                                  RRE              NAN   JCC(0F9H);SHLD, STA
(0F9H):FF1    CSR    R99  IRW               K1                       EXT                         AN    JCR(0F8H);
(0F8H):FF0    LMI    R99           ADL      KIR            NC2                                    NAN   JCR(0FBH);
(0FBH):FF0    LMI    R99                    KM                   NC1 CCR EXT                      NAN   JCR(0FAH);
(0FAH):FF1    LMI    R99  DOE      ADL  ED1 K1    MEMW                                    DBY AN    JRL(0FCH);(ROW = F)
*                                                                                                       (TO 0FD OR 0FF)
(0FDH):FF0    LMI    R99  DOE          ED1 K1     MEMW                EXT                       NAN   JCR(0FCH); SHLD
(0FCH):      NOP    RFF  DOE      ADL      K1     MEMW                                          NAN   JCR(0F5H);
(0F5H):FF1    LMI    R44  DOE               K1    MEMW                CCR EXT               DBY NAN   JCR(0F4H);
(0F4H):FF1    LMI    R44           ADL                                                     DBY AN    JZR07;

(0FFH):FF1    LMI    R44  DOE          ED1 K1                         CCR EXT               DBY AN    JCR(0FEH); STA
(0FEH):FF1    LMI    R44           ADL      K1                                             DBY AN    JZR07;
*----- --- --- --- --- --- ---- --- --- --- --- --- ----- --- --- --- ---- --- --- --- --- --- ---- -- --------- ----------
*     FF0 HCZ -  IFF -   1   1   1   1  ED2  -    1  MEMR   1   1   1   1   1   1   1   1   1   -   0  1'S   ; DEFAULTS
*-----------------------------------------------------------------------------------------------------------------------



EJECT;
*
*
* GROUP 2A:  MUL (MULTIPLY)  OP-CODE = ED (HEX)
*
*                      SETUP CONDITIONS                    RESULTANT CONDITIONS
*                      ----------------                    --------------------
*                      MULTIPLIER IN A-REG                 16-BIT RESULT IN B & C REGS (LSB IN C)
*                      MULTIPLICAND IN B-REG               CARRY FLAG (CY) CONTAINS MSB OF RESULT
*                                                          HALF CARRY FLAG (HC) IS INDETERMINATE
*
*       PROM U7   PROM U8        PROM U2                      PROM U5                   PROM U6            PROM U4
*     V--------V V-----V V----------------------V V----------------------------V V---------------------V V---------V
*     8-7,6-5,4-1 8-6,5-1 8,  7,  6,  5,  4, 3-1 8,  7--5,  4,  3,  2,  1 8,  7,  6,  5,  4, 3-1 8,  7---1
*
```

```
*
* ADDR FO   FI  FGP IMB RGP  DOE IRW ADL FRW ED1 KS    IER SWPA  NC2 NC1 CCR EXT RRE LD2 SJM IST DBY  CS  LD   AC      * COMMENTS
*----- --- --- --- --- ---  --- --- --- --- --- ---    --- ----- --- --- --- --- --- --- --- --- --- --- --  --------  ---------
(05AH):FF1  ILR     R00               K1*                                                           NAN      JCC(15AH); MUL
(15AH):FF1  ILR     RCC               K1                                                            NAN      JCR(15BH);
(15BH):FF1  CSR     RCC               K1              NC1                                            AN       JCR(15DH);
(15DH):FF0 STZ SRA  REE       IRW     K1                      CCR EXT                           DBY CN       JCR(15CH);
(15CH):FF0 STC AIA I02 REE            KD    NSTAT     NC1                                            AN.      JCR(15EH);
(15EH):FFC STC SRA  REE               K1    NSTAT                                               DBY CN       JCR(15FH);
(15FH):FF0 STC AIA I02 REE            KD    NSTAT     NC1                                            AN       JCC(16FH);
(16FH):FFC STC SRA  REE               K1    NSTAT                                               DBY CN       JCR(16EH);
(16EH):FF0 STC AIA I02 REE            KD    NSTAT     NC1                                            AN       JCR(16DH);
(16DH):FFC STC SRA  REE               K1    NSTAT                                               DBY CN       JCR(16CH);
(16CH):FF0 STC AIA I02 REE            KD    NSTAT     NC1                                            AN       JCR(16BH);
(16BH):FFC STC SRA  REE               K1    NSTAT                                               DBY CN       JCR(16AH);
(16AH):FF0 STC AIA I02 REE            KD    NSTAT     NC1                                            AN       JCR(169H);
(169H):FFC STC SRA  REE               K1    NSTAT                                               DBY CN       JCC(159H);
(159H):FF0 STC AIA I02 REE            KD    NSTAT     NC1                                            AN       JCR(158H);
(158H):FFC STC SRA  REE               K1    NSTAT                                               DBY CN       JCC(118H);
(118H):FF0 STC AIA I02 REE            KD    NSTAT     NC1                                            AN       JCR(114H);
(114H):FFC STC SRA  REE               K1    NSTAT                                               DBY CN       JCC(134H);
(134H):FF0 STC AIA I02 REE            KD    NSTAT     NC1                                            AN       JCC(144H);
(144H):FFC STC SRA  REE               K1    NSTAT                                               DBY CN       JCR(145H);
(145H):FF1  SDR     R99               K0    NSTAT                                                    AN       JCR(146H);
(146H):FF1  ILR     RCC               K1    NSTAT                                                   NAN      JCR(147H);
(147H):FF1  SDR     R00               K0    NSTAT                                                    AN       JCR(148H);
(148H):FF1  ILR     R99               K1    NSTAT                                                   NAN      JCR(14AH);
(14AH):FF1  LDI     REE           ED1 KD    NSTAT                                      IST         AN       JZR06;
*----- --- --- --- --- ---  ---- --- --- --- --- ---   --- ----- --- --- --- --- --- --- --- --- ---- -- --------  ---------
*       FF0 HCZ  -  IFF  -   1   1   1   1  ED2  -     1  MEMR   1   1   1   1   1   1   1   1   1   -   0   1'S    ; DEFAULTS
*-------------------------------------------------------------------------------------------------------------------------------
```

```
*
* GROUP 2B:  LHLD (LOAD H & L DIRECT),   LDA (LOAD A DIRECT)
*
*     PROM U7    PROM U8      PROM U2                PROM U5               PROM U6              PROM U4
*     V--------V V-----V V----------------------V V------------------------V V------------------------V V----------V
*     8-7,6-5,4-1 8-6,5-1 8,  7,  6,  5,  4, 3-1  8,  7--5,  4,  3,  2,  1  8,  7,  6,  5,  4, 3-1  8,  7---1
*
*
* ADDR FO   FI  FGP IMB RGP  DOE IRW ADL FRW ED1 KS    IER SWPA  NC2 NC1 CCR EXT RRE LD2 SJM IST DBY  CS  LD   AC      * COMMENTS
*----- --- --- --- --- ---  --- --- --- --- --- ---    --- ----- --- --- --- --- --- --- --- --- --- --- --  --------  ---------
(05BH):FF1  CSR     R33               K1                          RRE                            NAN      JCC(0ABH);LHLD, LDA
(0ABH):FF1  LDI     RFF               KM                      EXT                                 AN       JCR(0AAH);
(0AAH):FF1  LMI     RDD      ADL      K1                                                         NAN      JCR(0ADH);
(0ADH):FF1  LMM IFF RBB               KM               NC1     EXT              SJM              NAN      JCR(0ACH);
(0ACH):FF1  LMI     RDD      ADL      K1                                                     DBY AN       JPX(0A0H);(ROW = A)
*                                                                                                                  (TO 0AE OR 0A3)
(0AEH):FF1  LMI     RDD               K1                                                     DBY AN       JCR(0A9H); LHLD
(0A9H):     F5      R33               KM          NC2     CCR EXT RRE                            NAN      JCR(0A8H);
(0A8H):FF1  LMI     R44      ADL      K1                                                     DBY AN       JCR(0AFH);
(0AFH):FF1  LDI     RFF               KM                      EXT              SJM                 AN       JCC(03FH);(@ GP 0F)

(0A3H):FF1  LDI     REE               KM                      EXT                                 AN       JCR(0A2H); LDA
(0A2H):FF1  LMI     R44      ADL      K1                                                     DBY AN       JCC(132H);(@ GP 0A)
*----- --- --- --- --- ---  ---- --- --- --- --- ---   --- ----- --- --- --- --- --- --- --- --- ---- -- --------  ---------
*       FF0 HCZ  -  IFF  -   1   1   1   1  ED2  -     1  MEMR   1   1   1   1   1   1   1   1   1   -   0   1'S    ; DEFAULTS
*-------------------------------------------------------------------------------------------------------------------------------
*
*
*
*
*
*
```

100

**Signetics**

```
EJECT;
*
*  GROUP 2C:  POP B,  POP D,  POP H,  POP PSW
*
*        PROM U7    PROM U8        PROM U2              PROM U5                    PROM U6          PROM U4
*       V--------V V-----V V-----------------------V V-----------------------V V-----------------V V---------V
*       8-7,6-5,4-1 8-6,5-1 8,  7,  6,  5,  4, 3-1 8,  7--5,  4,  3,  2,  1 8,  7,  6,  5,  4, 3-1 8,  7---1
*
*
*                            --- --- --- --- ---    ---       --- --- --- --- --- --- --- --- ---
* ADDR FO  FI  FGP IMB RGP  DOE IRW ADL FRW ED1 KS  IER SWPA  NC2 NC1 CCR EXT RRE LD2 SJM IST DBY  CS  LD    AC     * COMMENTS
*----- --- --- --- --- ---  ---- --- --- --- --- --  ---- ----  --- --- --- --- --- --- --- --- --- --- --  ---------- ---------
(05CH):FF1     DSM     R44                      K0                                            SJM     DBY NAN     JCC(09CH); POP
(09CH):FF1     LMI     R33      IRW             K1                                EXT                 DBY AN      JPX(0B5H);(T0 0B5)
(0B5H):        NOP     RFF      IRW             K1                                EXT                     NAN     JCR(0B4H);
(0B4H):FF1     LMI     R33          ADL         K1                                                    DBY AN      JCR(0B3H);
(0B3H):FF1     LDI     RFF                      KM                                EXT                     AN      JCR(0B2H);
(0B2H):FF1     LMI     R44          ADL         K1                                                    DBY AN      JLL(0A4H);(ROW = A)
*                                                                                                                  (TO 0A5 OR 0A7)
(0A5H):FF1     LDI     RFF                      KM            NC1     EXT                                     AN      JCR(0A4H);POP B,D,H
(0A4H):        NOP     RFF                      K1  NSTAT                                                     NAN     JPR(080H);(ROW = 8)
*                                                                                                                  (TO 084, 085 OR 086)
(0B4H):FF1     SDR     R00                      K0  NSTAT                                         IST DBY AN      JZR06     ; POP B

(085H):FF1     SDR     R11                      K0  NSTAT                                         IST DBY AN      JZR06     ; POP D

(086H):FF1     SDR     R22                      K0  NSTAT                                         IST DBY AN      JZR06     ; POP H
*          NOTE:  THE ABOVE 3 INSTRUCTIONS COULD BE COMBINED INTO 1 BY USING "RRE" AND RGP = "R33", AND BY PROGRAMMING PROM U-30
*                 FOR POP B,  POP D,  AND POP H  OP-CODES, REGISTERS "R00", "R11", AND "R22" RESPECTIVELY:
*                      INST.    HEX.    U-30      U-30      ARRAY 1    ARRAY 2
*                      MNEMONIC OPCODE  ADDRESS   CONTENT   REGISTER   REGISTER
*
*                      -------  ------  -------   -------   --------   --------
*                      POP B    C1      3E        0         0 (B)      0 (C)
*                      POP D    D1      2E        5         1 (D)      1 (E)
*                      POP H    E1      1E        A         2 (H)      2 (L)
*
(0A7H):FF1 SCZ LDI     REE              FRW      KM                        EXT                             AN      JCE(0A7H); POP PSW
*                                                                                                            (JCE ENABLES PR1 FOR POP PSW.)
(0A6H):        NOP     RFF                      K1  NSTAT                                         IST         NAN     JZR06     ;
*----- --- --- --- --- ---  ---- --- --- --- --- --  ---- ----  --- --- --- --- --- --- --- --- --- --- --  ---------- ---------
*    FF0 HCZ -  IFF -   1   1   1   1 ED2 -   1 MEMR  1   1   1   1   1   1   1   1   1  -   0   1'S   ; DEFAULTS
*-----------------------------------------------------------------------------------------------------------------------------------
*
*
*
*
*
*
*
* GROUP 2D:  STC (SET CARRY)
*
*        PROM U7    PROM U8        PROM U2              PROM U5                    PROM U6          PROM U4
*       V--------V V-----V V-----------------------V V-----------------------V V-----------------V V---------V
*       8-7,6-5,4-1 8-6,5-1 8,  7,  6,  5,  4, 3-1 8,  7--5,  4,  3,  2,  1 8,  7,  6,  5,  4, 3-1 8,  7---1
*
*                            --- --- --- --- ---    ---       --- --- --- --- --- --- --- --- ---
* ADDR FO  FI  FGP IMB RGP  DOE IRW ADL FRW ED1 KS  IER SWPA  NC2 NC1 CCR EXT RRE LD2 SJM IST DBY  CS  LD    AC     * COMMENTS
*----- --- --- --- --- ---  ---- --- --- --- --- --  ---- ----  --- --- --- --- --- --- --- --- --- --- --  ---------- ---------
(05DH):    STC NOP     RFF      IRW             K1                                EXT                 IST     CS0     JZR07 ; STC
*----- --- --- --- --- ---  ---- --- --- --- --- --  ---- ----  --- --- --- --- --- --- --- --- --- --- --  ---------- ---------
*    FF0 HCZ -  IFF -   1   1   1   1 ED2 -   1 MEMR  1   1   1   1   1   1   1   1   1  -   0   1'S   ; DEFAULTS
*-----------------------------------------------------------------------------------------------------------------------------------
```

```
* GROUP 2E:  INR M;  DCR M;  MOV R M,  WHERE R = B, C, D, E, H, L
*
*      PROM U7    PROM U8        PROM U2              PROM U5                PROM U6           PROM U4
*      V--------V V----V V-------------------------V V-----------------------V V-----------------------V V----V
*      8-7,6-5,4-1 8-6,5-1 8,  7,  6,  5,  4,  3-1 8,  7--5,  4,  3,  2,  1 8,  7,  6,  5,  4,  3-1 8,  7---1
```

| ADDR | FO | FI | FGP | IMB | RGP | DOE | IRW | ADL | FRW | ED1 | KS | IER | SWPA | NC2 | NC1 | CCR | EXT | RRE | LD2 | SJM | IST | DBY | CS | LD | AC | COMMENTS |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| (05EH): | FF1 | DSM | | | R44 | | | | | | K0 | | | | | | | | | SJM | | DBY | NAN | | | JCC(0CEH); |
| (0CEH): | FF1 | LMI | | | R22 | | IRW | | | | K1 | | | | | | | EXT | | | | | | NAN | | | JPX(0C0H);(ROW = C) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | (TO 0CB, 0D1, 0D7, 0D3, 0D9, 0D5 OR 0DB) |
| (0CBH): | | NOP | | | RFF | | IRW | | | | K1 | | | | | | | EXT | | | | | | NAN | | | JCR(0CAH); INR/DCR M |
| (0CAH): | FF1 | LMI | | | R44 | | | ADL | | | K1 | | | | | | | | | | | | DBY | AN | | | JPR(0D0H);(ROW = D) |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | (TO 0DD OR 0DF) |
| (0DDH): | FF1 | LDI | | | RFF | | | | | | KM | | | | | | | EXT | | | | | | | AN | | JCR(0DCH); INR M |
| (0DCH): | FF1 | STZ | ILR | | R0D | | | | | | K1 | | | | | | | | | | | | | AN | | JCR(0DEH);(DOWN 2) |
| (0DFH): | FF0 | STZ | LDI | | RFF | | | | | | KM | | | | | | | EXT | | | | | | | AN | | JCR(0DEH); DCR M |
| (0DEH): | | NOP | | | RFF | DOE | | | FRW | | K1 | | MEMW | | | | | | | | | | | NAN | | JLL(0F4H);(TO 0F7) |
| (0F7H): | FF1 | LMI | | | R44 | DOE | | | | | K1 | | MEMW | | | | | EXT | | | | | DBY | AN | | | JCR(0F6H); |
| (0F6H): | FF1 | LMI | | | R44 | | | ADL | | | K1 | | | | | | | | | | | | DBY | AN | | | JZR07; |
| (0D1H): | FF1 | LDI | | | RFF | | | | | | KM | | | | | | | EXT | | | | | | | AN | | JCR(0D0H); MOV B M |
| (0D0H): | FF1 | SDR | | | R00 | | | | | | K0 | | NSTAT | | NC1 | | | | | | IST | | AN | | | JZR06; |
| (0D7H): | FF1 | LDI | | | RFF | | | | | | KM | | | | | | | EXT | | | | | | | AN | | JCR(0D6H); MOV C M |
| (0D6H): | FF1 | SDR | | | R00 | | | | | | K0 | | NSTAT | NC2 | | | | | | | IST | | AN | | | JZR06; |
| (0D3H): | FF1 | LDI | | | RFF | | | | | | KM | | | | | | | EXT | | | | | | | AN | | JCR(0D2H); MOV D M |
| (0D2H): | FF1 | SDR | | | R11 | | | | | | K0 | | NSTAT | | NC1 | | | | | | IST | | AN | | | JZR06; |
| (0D9H): | FF1 | LDI | | | RFF | | | | | | KM | | | | | | | EXT | | | | | | | AN | | JCR(0D8H); MOV E M |
| (0D8H): | FF1 | SDR | | | R11 | | | | | | K0 | | NSTAT | NC2 | | | | | | | IST | | AN | | | JZR06; |
| (0D5H): | FF1 | LDI | | | RFF | | | | | | KM | | | | | | | EXT | | | | | | | AN | | JCR(0D4H); MOV H M |
| (0D4H): | FF1 | SDR | | | R22 | | | | | | K0 | | NSTAT | | NC1 | | | | | | IST | | AN | | | JZR06; |
| (0DBH): | FF1 | LDI | | | RFF | | | | | | KM | | | | | | | EXT | | | | | | | AN | | JCR(0DAH); MOV L M |
| (0DAH): | FF1 | SDR | | | R22 | | | | | | K0 | | NSTAT | NC2 | | | | | | | IST | | AN | | | JZR06; |

NOTE: SEE NOTE FOR GROUP 2C.  IT ALSO APPLIES HERE IN TWO CASES.

| | FO | FI | FGP | IMB | RGP | DOE | IRW | ADL | FRW | ED1 | KS | IER | SWPA | NC2 | NC1 | CCR | EXT | RRE | LD2 | SJM | IST | DBY | CS | LD | AC | COMMENTS |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FF0 | HCZ | - | IFF | - | 1 | 1 | 1 | 1 | ED2 | - | 1 | MEMR | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | - | 0 | 1'S | ; DEFAULTS |

```
EJECT
*
*
*  GROUP 2F:  CMA (COMPLIMENT ACCUMULATOR)
*
*       PROM U7    PROM U8      PROM U2              PROM U5              PROM U6         PROM U4
*       V---------V V-----V V-----------------V V-------------------V V-------------------V V-------V
*       8-7,6-5,4-1 8-6,5-1 8,  7,  6,  5,  4, 3-1 8,  7--5,  4,  3,  2,  1  8,  7,  6,  5,  4, 3-1 8,  7---1
*
*                           --- --- --- --- ---      ---      --- --- --- --- --- --- --- --- --- --- ---
* ADDR FO  FI  FGP IMB RGP  DOE IRW ADL FRW ED1 KS   IER SWPA NC2 NC1 CCR EXT RRE LD2 SJM IST DBY CS  LD    AC      * COMMENTS
*----- --- --- --- --- ---  --- --- --- --- --- ---  --- ---- --- --- --- --- --- --- --- --- --- --- --   -------- ---------
(05FH):FF0     CMA     REE      IRW         K1                            EXT            IST     NAN    JZR06  ; CMA
*----- --- --- --- --- ---  --- --- --- --- --- ---  --- ---- --- --- --- --- --- --- --- --- --- --- --   -------- ---------
*      FF0 HCZ -   IFF -    1   1   1   1   ED2 -    1   MEMR 1   1   1   1   1   1   1   1   1   -   0   1'S    ; DEFAULTS
*==============================================================================================================================
*
*
*
*
*
*
*
*  GROUP 41:  DIV (DIVIDE)  OP-CODE = FD (HEX)
*
*
*                           SETUP CONDITIONS                      RESULTANT CONDITIONS
*                           ----------------                      --------------------
*
*                           DIVISOR IN A-REG                      QUOTIENT IN C-REG
*                           DIVIDEND IN C-REG                     REMAINDER IN B-REG
*                                                                 DIVISOR IN A-REG (UNCHANGED)
*                                                                 CARRY FLAG (CY) CONTAINS LSB OF QUOTIENT
*                                                                 HALF CARRY FLAG (HC) CONTAINS 1
*
*       PROM U7    PROM U8      PROM U2              PROM U5              PROM U6         PROM U4
*       V---------V V-----V V-----------------V V-------------------V V-------------------V V-------V
*       8-7,6-5,4-1 8-6,5-1 8,  7,  6,  5,  4, 3-1 8,  7--5,  4,  3,  2,  1  8,  7,  6,  5,  4, 3-1 8,  7---1
*
*                           --- --- --- --- ---      ---      --- --- --- --- --- --- --- --- --- --- ---
* ADDR FO  FI  FGP IMB RGP  DOE IRW ADL FRW ED1 KS   IER SWPA NC2 NC1 CCR EXT RRE LD2 SJM IST DBY CS  LD    AC      * COMMENTS
*----- --- --- --- --- ---  --- --- --- --- --- ---  --- ---- --- --- --- --- --- --- --- --- --- --- --   -------- ---------
(071H):FF1     LDI IBF REE              K0            NC2                                       AN    JCC(171H); DIV
(171H):FF1     CSR     R00      IRW     K1                    NC1     EXT                       AN    JCR(170H);
(170H):FF0 SCZ CMR     RCC              K1       NSTAT NC1                                      AN    JCR(174H);
(174H):FF0     ILR     R00              K1       NSTAT                                          NAN   JCR(172H);(SHIFT)
(172H):FFC     ALR     R00              K0       NSTAT                                 DBY BN   JCR(175H);
(175H):FF1 STC ALR     RCC              K0       NSTAT NC2 NC1                                  AN    JZF(162H);(ROW=16)
*                                                                                                        (TO 162 OR 163)
(162H):FF0 STZ SRA     REE              K1       NSTAT NC2                                      CN    JFL(172H);(ROW=17)
*                                                                                                        (TO 172 OR 173)
(173H):FF0     ILR     RCC              K1       NSTAT                                          AN    JCR(176H);
(176H):FF1     ADR     R00              K0       NSTAT     NC1                                  AN    JCR(174H);(TO 174)

(163H):FFC     ALR     R00              K0       NSTAT NC2                                      AN    JCR(164H);
(164H):FF0     ILR     RCC              K1       NSTAT                                          NAN   JCF(152H);(ROW=15)
*                                                                                                        (TO 152 OR 153)
(153H):FF1     ADR     R00              K0       NSTAT     NC1                                  AN    JCR(152H);
(152H):FF1     LDI     REE              KND      NSTAT                              IST         AN    JZR06;
*----- --- --- --- --- ---  --- --- --- --- --- ---  --- ---- --- --- --- --- --- --- --- --- --- --- --   -------- ---------
*      FF0 HCZ -   IFF -    1   1   1   1   ED2 -    1   MEMR 1   1   1   1   1   1   1   1   1   -   0   1'S    ; DEFAULTS
*==============================================================================================================================
```

**SIGNETICS HEADQUARTERS**

*811 East Arques Avenue*
*Sunnyvale, California 94086*
Phone: (408) 739-7700

**ALABAMA**
**Huntsville**
Phone: (205) 533-4540

**ARIZONA**
**Phoenix**
Phone: (602) 971-2517

**CALIFORNIA**
**Inglewood**
Phone: (213) 670-1101
**Irvine**
Phone: (714) 833-8980
(213) 924-1668
**San Diego**
Phone: (714) 560-0242
**Sunnyvale**
Phone: (408) 736-7565

**COLORADO**
**Parker**
Phone: (303) 841-3274

**FLORIDA**
**Pompano Beach**
Phone: (305) 782-8225

**ILLINOIS**
**Rolling Meadows**
Phone: (312) 259-8300

**INDIANA**
**Noblesville**
Phone: (317) 773-6770

**KANSAS**
**Wichita**
Phone: (316) 683-6035

**MARYLAND**
**Columbia**
Phone: (301) 730-8100

**MASSACHUSETTS**
**Woburn**
Phone: (617) 933-8450

**MINNESOTA**
**Edina**
Phone: (612) 835-7455

**NEW JERSEY**
**Cherry Hill**
Phone: (609) 665-5071
**Piscataway**
Phone: (201) 981-0123

**NEW YORK**
**Wappingers Falls**
Phone: (914) 297-4074
**Woodbury, L.I.**
Phone: (516) 364-9100

**OHIO**
**Worthington**
Phone: (614) 888-7143

**TEXAS**
**Dallas**
Phone: (214) 661-1296

**REPRESENTATIVES**

**ARIZONA**
**Phoenix**
Chaparral-Dorton
Phone: (602) 263-0414

**CALIFORNIA**
**San Diego**
Mesa Engineering
Phone: (714) 278-8021
**Sherman Oaks**
Astralonics
Phone: (213) 990-5903

**CANADA**
**Calgary, Alberta**
Philips Electronics
Industires Ltd.
Phone: (403) 243-7737

**Montreal, Quebec**
Philips Electronics
Industries Ltd.
Phone: (514) 342-9180
**Ottawa, Ontario**
Philips Electronics
Industries Ltd.
Phone: (613) 237-3131
**Scarborough, Ontario**
Philips Electronics
Industries Ltd.
Phone: (416) 292-5161
**Vancouver, B.C.**
Philips Electronics
Industries Ltd.
Phone: (604) 435-4411

**COLORADO**
**Denver**
Barnhill Five, Inc.
Phone: (303) 426-0222

**CONNECTICUT**
**Newtown**
Kanan Associates
Phone: (203) 426-8157

**FLORIDA**
**Altamonte Springs**
Semtronic Associates
Phone: (305) 831-8233
**Largo**
Semtronic Associates
Phone: (813) 586-1404

**ILLINOIS**
**Chicago**
L-Tec Inc.
Phone: (312) 286-1500

**KANSAS**
**Lenexa**
Buckman & Associates
Phone: (913) 492-8470

**MARYLAND**
**Glen Burni**
Microcomp, Inc.
Phone: (301) 761-4600

**MASSACHUSETTS**
**Reading**
Kanan Associates
Phone: (617) 944-8484

**MICHIGAN**
**Bloomfield Hills**
Enco Marketing
Phone: (313) 642-0203

**MINNESOTA**
**Edina**
Mel Foster Tech. Assoc.
Phone: (612) 835-2254

**MISSOURI**
**St. Charles**
Buckman & Associates
Phone: (314) 724-6690

**NEW JERSEY**
**Haddonfield**
Thomas Assoc., Inc.
Phone: (609) 854-3011

**NEW MEXICO**
**Albuquerque**
The Staley Company, Inc.
Phone: (505) 821-4310/11

**NEW YORK**
**Ithaca**
Bob Dean, Inc.
Phone: (607) 272-2187

**NORTH CAROLINA**
**Cary**
Montgomery Marketing
Phone: (919) 467-6319

**OHIO**
**Centerville**
Norm Case Associates
Phone: (513) 433 0966
**Fairview Park**
Norm Case Associates
Phone: (216) 333-4120

**OREGON**
**Portland**
Western Technical Sales
Phone: (503) 297-1711

**TEXAS**
**Dallas**
Cunningham Company
Phone: (214) 233-4303
**Houston**
Cunningham Company
Phone: (713) 461-4197

**UTAH**
**West Bountiful**
Barnhill Five, Inc.
Phone: (801) 292-8991

**WASHINGTON**
**Bellevue**
Western Technical Sales
Phone: (206) 641-3900

**WISCONSIN**
**Greenfuld**
L-Tec, Inc.
Phone: (414) 545-8900

**DISTRIBUTORS**

**ALABAMA**
**Huntsville**
Hamilton/Avnet Electronics
Phone (205) 533-1170

**ARIZONA**
**Phoenix**
Hamilton/Avnet Electronics
Phone: (602) 275-7851
Liberty Electronics
Phone: (602) 257-1272

**CALIFORNIA**
**Costa Mesa**
Schweber Electronics
Phone: (714) 556-3880-Culver City
Hamilton Electro Sales
Phone: (213) 558-2173
**El Segundo**
Liberty Electronics
Phone: (213) 322-8100
**Mountain View**
Elmar Electronics
Phone: (415) 961-3611
Hamilton/Avnet Electronics
Phone: (415) 961-7000
**San Diego**
Hamilton/Avnet Electronics
Phone: (714) 279-2421
Liberty Electronics
Phone(714) 565-9171
**Sunnyvale**
Intermark Electronics
Phone: (408) 738-1111

**CANADA**
**Downsview, Ontario**
Cesco Electronics
Phone: (416) 661-0220
**Mississauga, Ontario**
Hamilton/Avnet Electronics
Phone: (416) 677-7432
**Montreal, Quebec**
Cesco Electronics
Phone: (514) 735-5511
Zentronics Ltd.
Phone: (514) 735-5361
**Ottawa, Ontario**
Hamilton/Avnet Electronics
Phone: (613) 226-1700
Zentronics Ltd.
Phone: (613) 238-6411
**Toronto, Ontario**
Zentronics Ltd.
Phone: (416) 789-5111
**Vancouver, B.C.**
Bowtek Electronics Co., Ltd.
Phone: (604) 736-1141
**Ville St. Laurent, Quebec**
Hamilton/Avnet Electronics
Phone (514) 331-6443

**COLORADO**
**Commerce City**
Elmar Electronics
Phone: (303) 287-9611

**Denver**
Hamilton/Avnet Electronics
Phone: (303) 534-1212

**CONNECTICUT**
**Danbury**
Schweber Electronics
Phone: (203) 792-3500
**Georgetown**
Hamilton/Avnet Electronics
Phone: (203) 762-0361
**Hamden**
Arrow Electronics
Phone: (203) 248-3801

**FLORIDA**
**Ft. Lauderdale**
Arrow Electronics
Phone: (305) 776-7790
Hamilton/Avnet Electronics
Phone: (305) 971-2900
**Hollywood**
Schweber Electronics
Phone: (305) 922-4506
**Orlando**
Hammond Electronics
Phone: (305) 241-6601

**GEORGIA**
**Atlanta**
Schweber Electronics
Phone: (404) 449-9170
**Norcross**
Hamilton/Avnet Electronics
Phone: (404) 448-0800

**ILLINOIS**
**Elk Grove**
Schweber Electronics
Phone: (312) 593-2740
**Elmhurst**
Semiconductor Specialists
Phone: (312) 279-1000
**Schieler Park**
Hamilton/Avnet Electronics
Phone: (312) 671-6082

**INDIANA**
**Indianapolis**
Semiconductor Specialists
Phone: (317) 243-8271

**KANSAS**
**Lenexa**
Hamilton/Avnet Electronics
Phone: (913) 888-8900

**MARYLAND**
**Baltimore**
Arrow Electronics
Phone: (301) 247-5200
**Gaithersburg**
Pioneer Washington
Electronics
Phone: (301) 948-0710
**Hanover**
Hamilton/Avnet Electronics
Phone: (301) 796-5000
**Rockville**
Schweber Electronics
Phone: (301) 881-2970

**MASSACHUSETTS**
**Waltham**
Schweber Electronics
Phone: (617) 890-8484
**Woburn**
Arrow Electronics
Phone: (617) 933-8130
Hamilton/Avnet Electronics
Phone: (617) 933-8000

**MICHIGAN**
**Farmington**
Semiconductor Specialists
Phone: (313) 478-2700
**Livonia**
Hamilton/Avnet Electronics
Phone: (313) 522-4700
**Troy**
Schweber Electronic
Phone: (313) 583-9242

**MINNESOTA**
**Eden Prairie**
Schweber Electronics
Phone: (612) 941-5280

**Denver**
Hamilton/Avnet Electronics
Phone: (303) 534-1212

**Edina**
Hamilton/Avnet Electronics
Phone: (612) 941-3801
**Minneapolis**
Semiconductor Specialists
Phone: (612) 854-8841

**MISSOURI**
**Hazelwood**
Hamilton/Avnet Electronics
Phone: (314) 731-1144

**NEW MEXICO**
**Albuquerque**
Hamilton/Avnet Electronics
Phone: (505) 765-1500

**NEW YORK**
**Buffalo**
Summit Distributors
Phone: (716) 884-3450
**East Syracuse**
Hamilton/Avnet Electronics
Phone: (315) 437-2642
**Farmingdale, L.I.**
Arrow Electronics
Phone: (516) 694-6800
**Johnson City**
Wilshire Electronics
Phone: (607) 797-1236
**Rochester**
Hamilton/Avnet Electronics
Phone: (716) 442-7820
Schweber Electronics
Phone (716) 461-4000
**Westbury, L.I.**
Hamilton/Avnet Electronics
Phone: (516) 333-5800
Schweber Electronics
Phone: (516) 334-7474

**NORTHERN NEW JERSEY**
**Cedar Grove**
Hamilton/Avnet Electronics
Phone: (201) 239-0800
**Saddlebrook**
Arrow Electronics
Phone: (201) 797-5800

**SOUTHERN NEW JERSEY
AND PENNSYLVANIA**
**Cherry Hill, N.J.**
Milgray-Delaware Valley
Phone: (609) 424-1300
**Moorestown, N.J.**
Arrow/Angus Electronics
Phone: (609) 235-1900
**Mt. Laurel, N.J.**
Hamilton/Avnet Electronics
Phone: (609) 234-2133

**CENTRAL NEW JERSEY
AND PENNSYLVANIA**
**Somerest, N.J.**
Schweber Electronics
Phone: (201) 469-6008
**Horsham, PA**
Schweber Electronics
Phone: (215) 441-0600

**NORTH CAROLINA**
**Greensboro**
Hammond Electronics
Phone: (919) 275-6391
Pioneer Electronics
Phone: (919) 273-4441

**OHIO**
**Beechwood**
Schweber Electronics
Phone: (216) 464-2970
**Cleveland**
Arrow Electronics
Phone: (216) 464-2000
Hamilton/Avnet Electronics
Phone: (216) 461-1400
Pioneer Standard Electronics
Phone: (216) 587-3600
**Dayton**
Arrow Electronics
Phone: (513) 253-9176
Hamilton/Avnet Electronics
Phone: (513) 433-0610
Pioneer Standard Electronics
Phone: (513) 236-9900

**TEXAS**
**Dallas**
Component Specialists
Phone: (214) 357-6511
Hamilton/Avnet Electronics
Phone: (214) 661-8204
Schweber Electronics
Phone: (214) 661-5010
**Houston**
Component Specialists
Phone: (713) 771-7237
Hamilton/Avnet Electronics
Phone: (713) 780-1771
Schweber Electronics
Phone: (713) 784-3600

**UTAH**
**Salt Lake City**
Alta Electronics
Phone: (801) 486-7227
Hamilton/Avnet Electronics
Phone: (801) 262-8451

**WASHINGTON**
**Bellevue**
Hamilton/Avnet Electronics
Phone: (206) 746-8750
**Seattle**
Liberty Electronics
Phone: (206) 763-8200

**FOR SIGNETICS PRODUCTS WORLDWIDE**

**ARGENTINA**
Fapesa I.y.C.
Buenos-Aires
Phone: 652-7438/7478

**AUSTRIA**
Österreichische Philips
Wien
Phone: 93 26 11

**AUSTRALIA**
Philips Industries-ELCOMA
Lane-Cove N.S.W.
Phone: 421261

**BELGIUM**
M.B.L.E.
Brussels
Phone: 523 00 00

**BRAZIL**
Ibrape, S.A.
Sao Paulo
Phone: 287-7144

**CANADA**
Philips Electron Devices
Toronto
Phone: 425-5161

**CHILE**
Philips Chilena S.A.
Santiago
Phone: 39-4001

**DENMARK**
Miniwatt A/S
Kobenhavn
Phone: (01) 69 16 22

**FINLAND**
Oy Philips Ab
Helsinki
Phone: 1 72 71

**FRANCE**
R.T.C.
Paris
Phone: 355-44-99

**GERMANY**
Valvo
Hamburg
Phone (040) 3296-1

**HONG KONG**
Philips Hong Kong, Ltd.
Kwuntong
Phone: 3-427232

**INDIA**
Semiconductors, Ltd.
(REPRESENTATIVE ONLY)
Bombay
Phone: 293-667

**INDONESIA**
P.T. Philips Ralin Electronics
Jakarta
Phone: 581058

**IRAN**
Berkeh Company, Ltd.
Tehran
Phone: 831564

**ISRAEL**
Rapac Electronics, Ltd.
Tel Aviv
Phone: 477115-6-7

**ITALY**
Philips S.p.A.
Milano
Phone: 2-6994

**JAPAN**
Signetics Japan, Ltd.
Tokyo
Phone: (03) 230-1521

**KOREA**
Philips Electronics Korea, Ltd.
Seoul
Phone: 44-4202

**MEXICO**
Electronica S.A. de C.V.
Mexico D.F.
Phone: 533-1180

**NETHERLANDS**
Philips Nederland B.V.
Eindhoven
Phone: (040) 79 33 33

**NEW ZEALAND**
E.D.A.C. Ltd.
Wellington
Phone: 873-159

**NORWAY**
Electronica A.S.
Oslo
Phone: (02) 15 05 90

**PHILIPPINES**
Philips Industrial Dev., Inc.
Makata-Rizal
Phone: 868951-9

**SINGAPORE/MALAYSIA**
Philips Singapore Pte., Ltd.
Toa Payoh
Phone: 538811

**SOUTH AFRICA**
E.D.A.C. (PTY), Ltd.
Johannesburg
Phone: 24-6701-3

**SPAIN**
Copresa S.A.
Barcelona
Phone: 329 63 12

**SWEDEN**
Elcoma A.B.
Stockholm
Phone: 08/67 97 80

**SWITZERLAND**
Philips A.G.
Zurich
Phone: 01/44 22 11

**TAIWAN**
Philips Taiwan, Ltd.
Taipei
Phone: (02) 551-3101-5

**THAILAND/LAOS**
Saeng Thong Radio, Ltd.
Bangkok
Phone: 527195, 519763

**UNITED KINGDOM**
Mullard, Ltd.
London
Phone: 01-580 6633

**UNITED STATES**
Signetics International Corp.
Sunnyvale, California
Phone: (408) 739-7700

**VENEZUELA, PANAMA,
ARUBA, TRINIDAD**
Industrias Venezolanas Philips S.A.
Caracas
Phone: 360511/363011

March 1977

**Signetics**

**8080 Emulator Manual**
**Errata Sheet**

770421 - BMW
SHEET 1 OF 3


PAGE 47:   IN TABLE B-6:  THE LINE AT ADDRESS
-------                   003 SHOULD BE CHANGED TO READ:

                          003  7A 7A 24 25 FF 26 24 24 13 0F FF 1A 13 14 FF 3E
                                                                            -

                          ALSO THE LINE AT ADDRESS
                          01A SHOULD BE CHANGED TO READ:

                          01A  00 00 13 32 00 00 33 36 00 00 37 3A 00 00 00 00
                                     -- --       -- --       -- --


PAGE 48:   IN TABLE B-7:  THE LINE AT ADDRESS
-------                   01A SHOULD BE CHANGED TO READ:

                          01A  00 00 F3 FB 00 00 F3 FF 00 00 FF FB 00 00 00 00
                                     -- --       -- --       -- --


PAGE 49:   IN TABLE B-8:  THE LINE AT ADDRESS
-------                   01A SHOULD BE CHANGED TO READ:

                          01A  00 00 FF FE 00 00 FF FA 00 00 FF FE 00 00 00 00
                                     -- --       -- --       -- --


PAGE 50:   IN TABLE B-9:  THE LINE AT ADDRESS
-------                   01A SHOULD BE CHANGED TO READ:

                          01A  00 00 DC FA 00 00 DC FA 00 00 DC FA 00 00 00 00
                                     -- --       -- --       -- --


PAGE 51:   IN TABLE B-10: THE LINE AT ADDRESS
-------                   01A SHOULD BE CHANGED TO READ:

                          01A  00 00 E4 EE 00 00 E4 CB 00 00 ED FF 00 00 00 00
                                     -- --       -- --       -- --


PAGE 52:   IN TABLE B-11: THE LINE AT ADDRESS
-------                   01A SHOULD BE CHANGED TO READ:

                          01A  00 00 F1 F6 00 00 F1 F1 00 00 F1 F6 00 00 00 00
                                     -- --       -- --       -- --

PAGES 85-103:             THE DEFAULT VALUE FOR THE IMB FIELD OF MICRO CODE
                          SHOULD BE CHANGED TO I00 INSTEAD OF IFF.  THUS:

*     FF0 HCZ  -   I00  -    1    1    1    1 ED2  -    1 MEMR   1    1 ... ETC.
                  ---

PAGE 89:
--------
GROUP 0B, THE LDA INSTRUCTION, 13 LINES IN THE MIDDLE
OF THE PAGE, SHOULD BE REPLACED BY THE FOLLOWING
MICRO CODE:

```
*
*  GROUP 0B:  LDA
*
*       PROM U7    PROM U8      PROM U2                    PROM U5                    PROM U6           PROM U4
*       V--------V V----V V--------------------V V--------------------V V------------------V V--------V
*       8-7,6-5,4-1 8-6,5-1 8,  7,  6,  5,  4, 3-1 8,  7--5,  4,  3,  2,  1  8,  7,  6,  5,  4, 3-1 8,  7---1
*
*
* ADDR  FO   FI  FGP INB RGP  DOE IRW ADL FRW ED1 KS   IER SWPA   NC2 NC1 CCR EXT RRE LD2 SJM IST DBY  CS  LD    AC     ; COMMENTS
*----- --- --- --- --- --- ---- --- --- --- --- --   --- ----   --- --- --- --- --- --- --- --- ---  --- --  -------- --------
(03BH) FF1  LDI      RFF                      KM                          EXT                          AN   JCC(1ABH);  LDA
(1ABH) FF1  LDI      RFF                      KM                          EXT                          AN   JCR(1AAH);
(1AAH) FF1  LMI      RDD      ADL             K1                                                       NAN  JCR(1A7H);
(1A7H) FF1  LMM IFF  RBB                      KM                NC1       EXT         SJM               NAN  JCR(1A6H);
(1A6H) FF1  LMI      R44      ADL             K1                                                   DBY AN   JCR(1A3H);
(1A3H) FF1  LDI      REE                      KM                          EXT                          AN   JCR(1A2H);
(1A2H) FF1  LMI      R44      ADL             K1                                                   DBY AN   JCC(132H);(@ GP 0A)
*----- --- --- --- --- --- ---- --- --- --- --- --   --- ----   --- --- --- --- --- --- --- --- ---  --- --  -------- --------
*     FF0 HCZ  -  I00 -    1   1   1   1  ED2 -    1  MEMR   1   1   1   1   1   1   1   1   1   -   0   1'S   ; DEFAULTS
*----------------------------------------------------------------------------------------------------------
```

PAGE 91:
--------
IN GROUP 11, JMP, AT ADDRESSES (110H) AND (113H),
THE  FO  FIELD OF MICRO CODE SHOULD BE CHANGED FROM
FF0 TO FF1.  THUS:

```
(110H) FF1  LMI  R44        ADL        KIR        NC2                      NAN  JCR(113H);
       -
(113H) FF1  LMI  R44   IRW              KIR                NC1 CCR EXT      NAN  JCR(112H);
       -
```

PAGE 97:
--------
IN GROUP 22, ADD M ... ETC, AT ADDRESS (0E4H), THE
SWPA  FIELD OF MICRO CODE SHOULD BE CHANGED FROM DEFAULT
TO  K1.  THUS:

```
(0E4H) FF1  SDR  RCC        FRW   K0    NSTAT              IST  AN   JZR06;
                                        -----
```

PAGE 99:
--------
IN GROUP 29, SHLD, AT ADDRESS (0F4H), THE KS FIELD
OF MICRO CODE SHOULD BE CHANGED FROM DEFAULT TO K1.
THUS:

```
(0F4H) FF1  LMI  R44        ADL   K1                         DBY AN   JZR07;
                                  --
```

AND AT ADDRESS (0FFH), THE  SWPA  FIELD OF MICRO
CODE SHOULD BE CHANGED FROM DEFAULT TO  MEMW  THUS:

```
(0FFH) FF1  LMI  R44  DOE         ED1 K1    MEMW       CCR EXT         DBY AN   JCR(0FEH); STA
                                            -----
```

770421 - BMW
SHEET 3 OF 3

PAGE 100:                 GROUP 2B, LHLD, SHOULD BE REPLACED BY THE FOLLOWING
--------                  MICRO CODE:

```
*
*  GROUP 2B: LHLD (LOAD H & L DIRECT).
*
*       PROM U7   PROM U8         PROM U2                    PROM U5                      PROM U6            PROM U4
*       V--------V V-----V V--------------------V V--------------------------V V-------------------V V---------V
*       8-7,6-5,4-1 8-6,5-1 8,  7,  6,  5,  4, 3-1 8,  7--5,  4,  3,  2,  1  8,  7,  6,  5,  4, 3-1 8,  7---1
*
*                       --- --- --- --- ---      ---    --- --- --- --- --- --- --- --- --- ---
* ADDR FO  FI  FGP INB RGP  DOE IRW ADL FRW ED1 KS   IER SWPA  NC2 NC1 CCR EXT RRE LD2 SJM IST DBY  CS  LD    AC      * COMMENTS
*----- --- --- --- --- ---  ---- --- --- --- --- ---  ----- -----  --- --- --- --- --- --- --- --- ---  ---- --- --------  ---------
(05BH):FF1      CSR     R33                       K1                                    RRE                 NAN   JCC(0ABH);LHLD
(0ABH):FF1      LDI     RFF                       KM                            EXT                          AN   JCR(0AAH);
(0AAH):FF1      LMI     RDD         ADL           K1                                                        NAN   JCR(0ADH);
(0ADH):FF1      LMM IFF RBB                       KM             NC1      EXT          SJM                   NAN   JCR(0ACH);
(0ACH):FF1      LMI     RDD         ADL           K1                                              DBY  AN   JPX(0A0H);(ROW = A)
*                                                                                                                (TO 0AE)
(0AEH):FF1      LMI     RDD                       K1                                              DBY  AN   JCR(0A9H); LHLD
(0A9H):         F5      R33                       KM          NC2      CCR EXT RRE                          NAN   JCR(0A8H);
(0A8H):FF1      LMI     R44         ADL           K1                                              DBY  AN   JCR(0AFH);
(0AFH):FF1      LDI     RFF                       KM                            EXT          SJM             AN   JCC(03FH);(@ GP 0F)


*----- --- --- --- --- ---  ---- --- --- --- --- ---  ----- -----  --- --- --- --- --- --- --- --- ---  ---- --- --------  ---------
*      FF0 HCZ  -  I00  -    1   1   1   1  ED2  -    1 MEMR    1   1   1   1   1   1   1   1   1   -   0   1'S   ; DEFAULTS
*-----------------------------------------------------------------------------------------------------------------------
```

# signetics

a subsidiary of **U.S. Philips Corporation**