ATTACHMENT D

CONTROL DATA CORPORATION

ENGINEERING SPECIFICATION

NO	88786800
REV	01
DATE	6/13/73
PAGE	1 of

SCDD

MP17

ENGINEERING SPECIFICATION

Originator:

W A Armstrong

REVISION RECORD

REV.	E.C.O.	PAGE	PARAGRAPH	DESCRIPTION	APVD.	DATE
01				MP17 Engineering Specification		

TABLE OF CONTENTS

TABLE OF CONTENTS

1.0 SCOPE

This specification describes a small scale stored program parallel mode digital computer. This computer has the capability of being configured in two ways.

a. The classical single level processor with a one-to-one relationship of instruction read from storage and a hardware decode and execution takes place.

b. A multilevel execution. In this atmosphere a macro-instruction program is stored in what may be called the main memory system and a micro-instruction program is stored in what may be called a micro-memory system. These memory systems may be one and the same or may be different. In this machine they are different; the micro-memory is smaller in capacity and faster in both access and cycle time. Macro-instructions are read from main memory serially by a portion of the micro-program. The macro instruction is then decoded by a combination hardware/software technique. The hardware portion of the macro-instruction decode is called a transform. The transform process causes the micro-program to form program branches, sets parameters and performs arithmetic or logical operations all in parallel. A transform is performed as if it was a standard stored micro-instruction. Numerous different types of transforms are executed in the process of executing a macro-instruction. Macro-instructions may be decoded either with or without the use of the transform feature, however, use of transforms increase the performance and reduces the number of micro-instructions many fold.

2.0 APPLICABLE DOCUMENTS

2.1 Standards and Specifications

NCR Standard	SK 077-9900005	605 Computer Standards
CDC Standard	1.30.011	Computer & Peripheral Equipment Design Requirements
CDC Manual	60165800	1700 I/O
CDC Specification	11825600	1700 I/O
CDC Specification	52429300	1714 Processor
CDC Specification	11828900	1700 DSA
CDC Manual	14232000	Micro-Programmable Processor
CDC Specification	11820600	1704 Processor
CDC Specification	52308600	1774 Processor
CDC Specification	89626200	CR17 Processor
CDC Standard	1.30.022	EMC Performance Requirements
CDC Standard	1.01.000	Hardware Configuration Management Standards
CDC Standard	1.20.007	Acoustical Noise Standard

3.0 REQUIREMENTS

3.1 System Definition

The processor is parallel mode,stored program, digital computer made up of a minimum of three of the six blocks shown in Figure 1 - The Basic System Block diagram. The minimum blocks are:

1. Micro-processor
2. Maintenance interface/maintenance panel
3. Micro-memory

This minimum configuration could be utilized in a peripheral controller or I/O processor type application. The complete configuration shown in Figure 1 is more typical of two-level computer with the microprocessor emulating program stored in the main memory. Figure 1 is the basic processor block diagram defining the MP17 emulation described in Appendix 1.

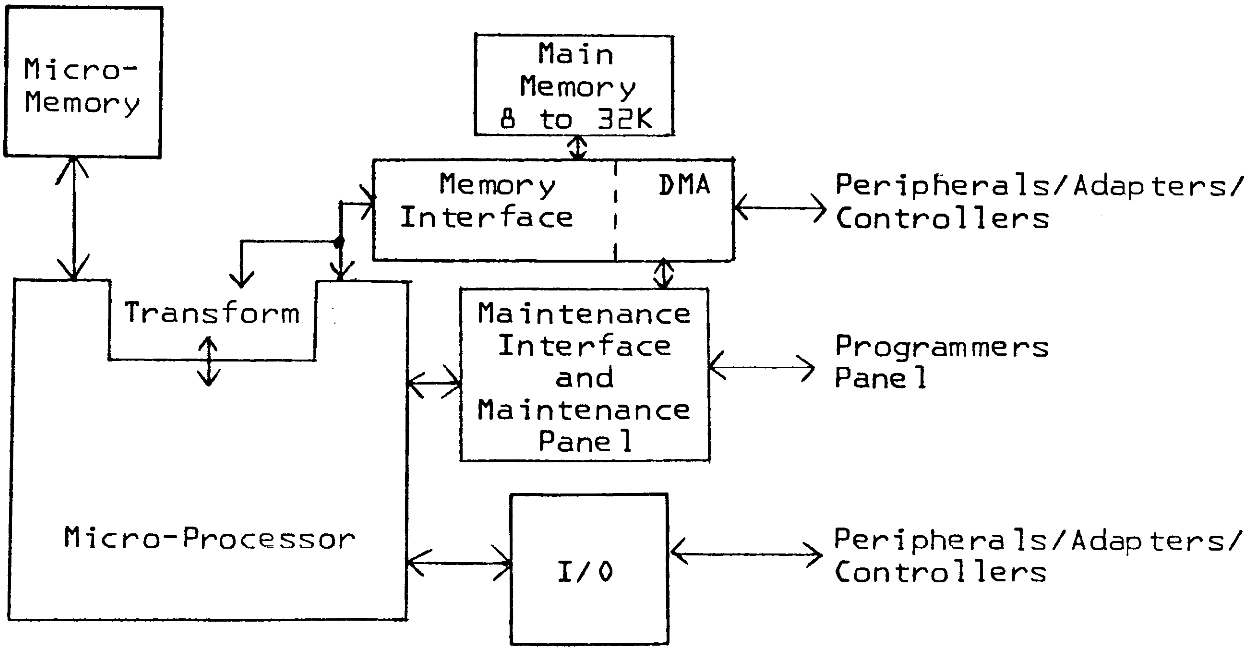

Micro-Memory
Main Memory 8 to 32K
Memory Interface
DMA
Peripherals/Adapters/Controllers
Transform
Maintenance Interface and Maintenance Panel
Programmers Panel
Micro-Processor
I/O
Peripherals/Adapters/Controllers

Figure 1. Basic System Block Diagram
Mini-Configuration

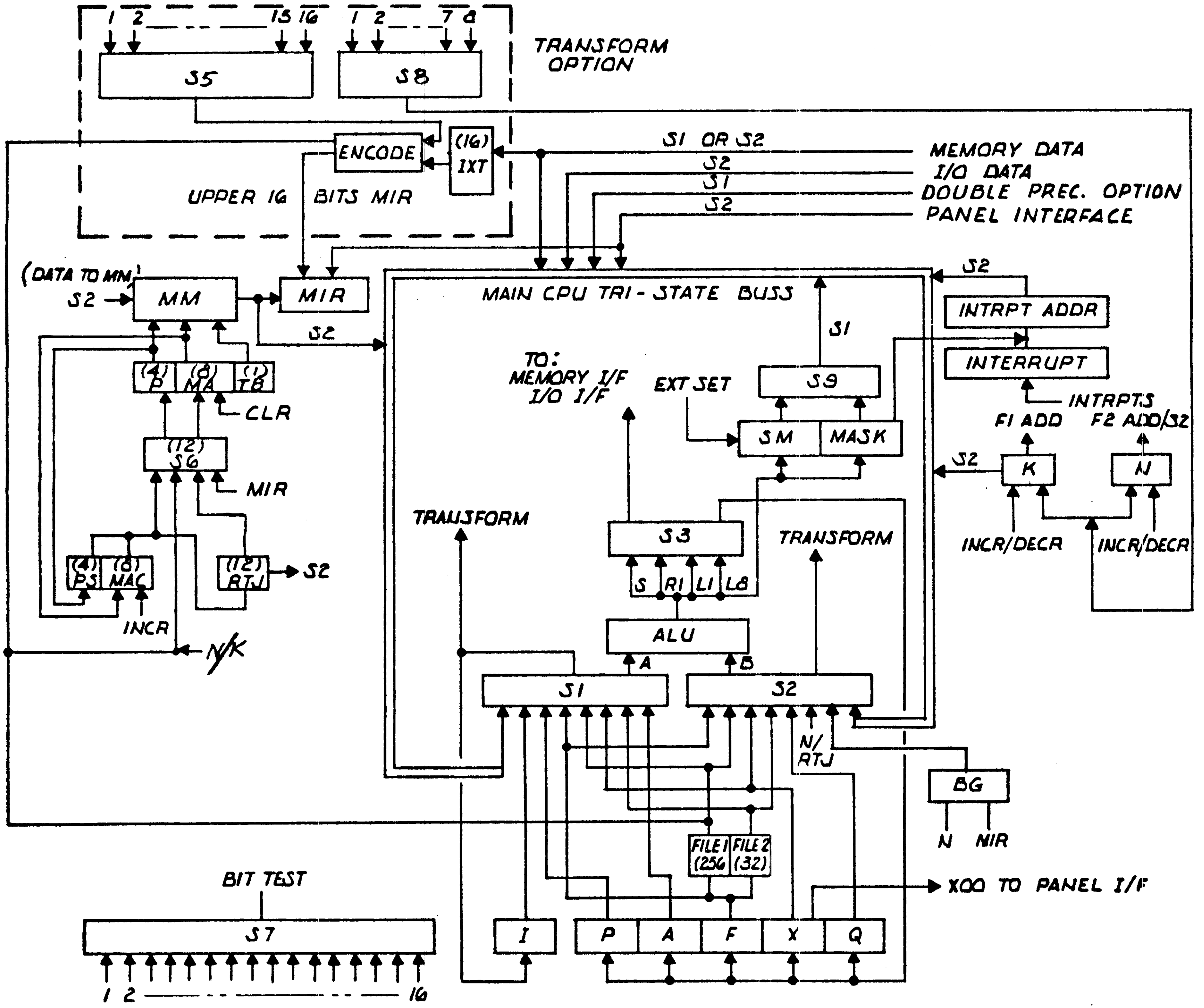


Figure 2 Detailed Micro-processor Block Diagram

3.3 General Description

3.3.1 Micro-Processor (MP)

The detailed organization of the MP is shown in Figure 2. This diagram shows that the registers of the MP are interconnected primarily by selectors. The selectors in the diagram are numbered from S1 to S3 and S5 to S9. A selector is a multiplexer that provides for transferring one of several inputs to an output. Selectors S1, S2, S3, and S9 are word-width selectors that provide for the transfer of a complete MP word to the output. Selectors S5, and S8 are 8-bit selectors and provide for the transfer of an 8-bit quantity; Selector S7 is a single bit selector for transferring one of 16 input bits to its output. Selector S6 is a 12 bit selector, and provides for transferring micro-memory address to P/MA.

The unlabeled inputs to selectors represent uncommitted logic in the basic MP design, which is specified by the system designer to provide an efficient solution to a specific system design problem. Arrows leaving a register or a data path represent data points that are available on the backpanel wiring of the MP; they are candidate bits of information for connection to the uncommitted logic of the MP. The detailed wiring of the uncommitted selector inputs is performed on the transform module, which is different for each MP application.

3.3.1.1 Micromemory and Microcontrol Section

The MP micromemory is an IC memory with a basic memory size of 256 words. Each word is 64 bits long, containing two 32-bit microinstructions. The basic memory size can be expanded to a maximum of 8192 micro-instructions. The micromemory is available in two forms:

1. A read/write micro-memory (RAM) is available for use during program development and in applications that require the MP to be re-programmed or reorganized for multiple applications. This read/write micro-memory can

be loaded from an external device, or data can be written into micro-memory under control of the microprogram. The RAM micro-memory expansion cards are available in only one version, this version contains 512 micro-instructions per card.

2. A read only micro-memory (ROM) is available for fixed applications of the MPP. This is programmed during manufacture. It is available only on the transform board of the MP17 1700 emulator, and provides for 1,024 instructions.

The micromemory is addressed by the P/MA register. The P portion of the register is a page control and consists of four bits specifying up to 16 pages of micromemory. A page consists of 256 words or 512 microinstructions. The MA register is an 8-bit register that specifies one of the 256 microinstruction pairs within the page that is to be the source of the next microinstruction. The T field of the current microinstruction is used to select the upper or the lower of the microinstruction pair as the next microinstruction to be executed. Microinstruction sequencing is designed so that no automatic overflow of addressing from the MA register to the P register occurs; any transfer of control between pages is initiated by a page jump operation, MA transform operation, or clear page operation.

3.3.1.1.1 Page/Micromemory Address Register

The P/MA register provides addressing for the micromemory. The micromemory is organized into pages of 256 locations, which provide for 512 microinstructions. The P portion of the register selects one of 16 pages, while the MA provides for selection one of the 256 locations within the page.

The P portion of the register can be set from a microinstruction, from the PS, from the RTJ register, or from an output of the transform module. The micromemory address portion of the register can be set from a microinstruction,from the MAC, from the RTJ register, or from an output of the transform module. The MA portion of the register has no sequencing capability; this is provided by the MAC. The combination N/X register can provide an address to P/MA for micromemory Read/Write operand references.

3.3.1.1.2 Memory Address Counter

The MAC is a counter which is used to determine the next location within a page following the current location specified in the MA register. In operation, the contents of the MA register is transferred to the MAC at each micromemory reference, then the contents of the MAC is incremented by one to point to this next location. Depending on the sequencing operation specified in the microinstruction, the MAC may or may not be used to obtain the next microinstruction. Sequencing of the MAC is such that location 0 within a page follows location 255 on that page.

3.3.1.1.3 Page Storage Register (PS)

The PS register is a four bit holding register which, is used to determine the page for the next instruction. In operation the contents of the page register is transferred to the PS at each micromemory reference. Depending on the sequencing operation specified in the microinstruction the PS may or may not be used to obtain the next microinstruction.

3.3.1.1.4 Microinstruction Register

The MIR is a 32-bit register which is used to hold the microinstruction during execution. Data is normally entered into the MIR from the micromemory; either the upper or lower 32 bits of the contents of the micromemory location are gated to the MIR, based on the value of the test bit determined during the preceding microinstruction. A test bit of 0 specifies the upper microinstruction, and a test bit of 1 specifies the lower microinstruction.

Data may also be transferred into MIR from the maintenance panel interface. This is done as a result of an operator request, it may or may not result in offline mode depending on the function requested.

A transform design option allows data to be transferred from the transform module to the upper 16-bits of the MIR for specialized applications.

3.3.1.1.5 Return Jump Register

The RTJ register is provided to capture the location of the next microinstruction pair at any time, when specified by a microinstruction. When this capture is specified, the contents of MAC is incremented and PS and MAC are stored in the RTJ register. The contents of the RTJ register is unchanged until the next command is given to save a new address. This saving of the next instruction pair location is independent of any actual transfer of control. The output of the RTJ register can be gated to the P/MA register to perform the return operation or the MA portion may be read into the organization of the MPP through selector S2.

3.3.1.2 Transforms and the Transform Module

The transform feature of the MPP provides the microprogram with the capability to select any pattern of bits from the data transmission paths of the MPP, to form micromemory addresses to sequence the microprogram, and to set the contents of the K and N register and upper 16 bits of the MIR register.

The transform hardware is packaged on a separate module and is specialized for each particular application. The transforms are implemented on two selectors, S5 and S8 which are located on the transform module along with the bit test selector S7.

Selector S5 is an eight bit wide two position to sixteen position selector which is used to form micromemory addresses A maximum of sixteen different micromemory address (MA) constructions can thus be specified. The MA transformed specifies one of 256 64-bit microwords within a page. The T field code specifies which 32 bit microinstruction is to be executed. If SM207 is set to 1, the transform address will be to the page denoted in the S1 field of the microinstruction. If SM207 is set to 0, the transform address will reside in the same page as the microinstruction currently being executed. The lower (least significant) eight bits of the output of S2 shall represent MA transform number 7 (j=0111 in C field) on all configurations.

Selector S8 is an eight bit wide two position to sixtten position selector which is used to select a source for loading the K or N register. Two of the sources are fixed; KN transform 0 (j=0000) shall be the output of the lower (least significant) eight bits of S2 and KN transform 6 (j=0110) shall be the output of the lower (least significant) eight bits of the MIR.

A specialized transform of the upper (most significant) 16 bits of the MIR register may be implemented to provide more efficient emulation. In order to utilize this feature, the macro instruction register (referred to as IXT) must be located on the transform module. This feature allows the upper 16 bits of MIR to be loaded directly with information encoded off the IXT register in order to control MPP arithmetic function during macro emulation RNI cycles.

Selector S7 is a one bit wide,two position to thirty two bit position selector,which is used to test a specific condition to determine which microinstruction to execute from the next microinstruction pair. This capability is used by placing a BTU command in the T field of the microinstruction; the lower five bits of the C field (j) determine which of the thirty two possible bits to test. If the tested bit is a one, the upper microinstruction is executed; otherwise, the lower microinstruction is executed.

Bits that can be tested include all the bits available for use in creating transforms and bits associated with any specialized I/O logic.

The following MPP outputs are available for monitoring by the transform module.

. S1
. S2
. SM Registers

. MIR
. Main Memory Read Data Bus
. Bits 19 and 24 through 31 of micro memory read data

3.3.1.3 Arithmetic and Logical Unit and Data Transfer Organization

3.3.1.3.1 The ALU provides the arithmetic and logical capabilities of the CPU. This unit combines two input words of the system word length, one from the A input provided by S1 and the other from the B input provided by S2.

These two inputs are combined according to the function code specified in the microinstruction; the result is delivered simultaneously at the output of the ALU for possible shifting and delivery to the destination register, to the ALU buffer circuit for delivery to the SM and mask registers, and to the memory bus and the panel interface. It is possible to ignore the output of the ALU on any instruction. The result of the ALU operation as to the sign, zero, and magnitude {by means of the carryout test} is available to the test bit logic for instruction sequencing.

Arithmetic operations may be in 1's or 2's complement arithmetic and can operate on either single precision operands or on double precision operands, using the double precision hardware available as an option. Selection of single or double precision and 1 or 2's complement mode is controlled by the microprogram which controls the states of the applicable bits in the SM register.

Also available as an option is the ability to perform split adder operations. For example, in a 16 bit CPU, a split may be made between the upper and lower 8-bit groups and would allow operations on each group to take place simultaneously. The selection of the split adder is controller by setting or clearing a bit in the SM register. The split has no effect on logical operations, since these do not involve a carry between bits.

NOTE: Design allowances have been made for optional special algorithms of which arithmetic is one. However, design of these options is not presently included in the development.

The data transfer organization of the CPU provides for storing data in one of six working registers and two files and for selecting data to be processed through the ALU. ALU results are transferred back to one of the registers or out of the organization to control external equipment.

The primary data registers are I, P, A, F, X, and Q. The names given to the registers are descriptive of the normal use of the registers in an emulation; however, the names are not intended to limit the use of the registers.

3.3.1.3.2 I Register

The I register is a word length register used primarily to hold the emulated software instruction being executed by the CPU. It is also available as input to S1, and therefore, to the A input of the ALU.

Data is entered into the I register from the output of S1. This connection is provided so that datamay be transferred from memory directly to the I register, in addition to performing some other operation on the memory data or transferring the memory data to another register through the ALU.

3.3.1.3.3 P Register

The P register is a word length register which receives data from the ALU and whose output is provided to S1 and thus to the A input of the ALU. This is a general purpose register; however, it normally is used to contain the software instruction counter for the emulation of a computer.

3.3.1.3.4 A Register

The A register is a word length, general purpose register which receives data from the ALU and provides output to S1 and thus to the A input of the ALU.

The A register is mechanized as a shifting regiser, and it can be shifted left or right without the use of the ALU. The A register may also be combined with the Q register to form a double length shifting register, which operates independently of the ALU.

3.3.1.3.5 F Register

The F register is a word length, general purpose register which can be read into the A or B input of the ALU. This register is also used as the file entry register and contains the information written into file 1 or file 2 when these files are used as the destination of an ALU operation.

3.3.1.3.6 X Register

The X register is a word length, general purpose register which can be read into the A or B input of the ALU.

3.3.1.3.7 Q Register

The Q register is a word length, general purpose register which receives data from the ALU and provides output to S2 and thus to the B side of the ALU. The Q register is mechanized as a shifting register; it may be shifted left or right in conjunction with the A register, without the use of the ALU.

3.3.1.3.8 Register File Usage

Files 1 and 2 are word length, scratchpad, storage register files which are addressed by the K and N registers, respectively.

3.3.1.3.9 File 1

File 1 is a file of 256 general purpose, word-sized registers which are addressed by the contents of the K register; the output of the addressed file is delivered to S1 and S2, and thus to the A and B side of the ALU on demand. A S/M bit is provided for selecting either the output of File 1 or output from the transform moduleas this input to S1 or S2.

3.3.1.3.10 File 2

File 2 is a 32-word file and is addressed by the lower five bits of theN register. It is intended primarily as a source of constants, but may be used as a general purpose, word-sized register which delivers its output to S1 and S2, and thus to the A and B side of the ALU.

3.3.1.3.11 Bit Generator

The bit generator is a circuit that generates one bit at any bit position in a word as input to the B side of the ALU. Bits are numbered from left to right as bits 0 to N, where N is the number of bits in a word minus one. Control todrive the bit generator is derived either from the microinstruction {bits 27 to 31} or from the lower five bits of the N register. Control is usually obtained from the microinstruction. The choice of which input drives the bit generator is based on the setting of a bit in the SM register..

3.3.1.3.12 Selector 1

S1 provides for the selection of one of eight inputs for delivery to the ALU or for delivery to the I register if the I register is selected as a destination in the microinstruction. The selector also provides data for transforms.

Input to the selector is from the following:

- Input bus
- I register
- P register
- F register
- File 1 or external input {transform}
- X register
- File 2
- A register

Output from the selector goes to the following:

- ALU
- External output or transform
- I register

Input Bus to Selector 1

The input bus is one input to S1 and provides for selection of input data to be transmitted to S1. The data may come from:

- Data from the externalmain memory
- Output of the SMregisters or the interrupt mask registers
- Algorithm

3.3.1.3.13 Selector 2

S2 provides for the selection of one of eight inputs for delivery to theB input of the ALU. In addition, the second output provides for the transfer of information from this selector to the transformmodule and to themicromemory.

Input to S2 is from the following:

- F register
- File 1 or external input {transform}
- X register
- File 2
- Q register
- N and RTJ registers
- Bit generator
- Input bus

Input Bus to Selector 2

The input bus provides for submultiplexing of data for input to S2. Inputsto the input bus are:

- Main memory
- Interrupt address
- Micromemory
- I/O system, maintenance panel interface
- K register

3.3.1.3.14 Status/Mode Register

The SM register allows the microprogram to control the mode of operation and also allows the microprogram to examine the status of certain internal and external conditions. The CPU can access one or two SM registers, referred to as SM1 and SM2. Each SM register contains the same number of bits as the basic processor word; sixteen bits for a sixteen bit processor or thirty two bits for a thirty two bit processor. Therefore, the maximum total number of SM bits in a CPU is twice the basic processor word length.

An SM register module contains 16 bits of SM1 and 16 bits of SM2. All 32 bits of an SM module can be set or reset by the microprogram by transferring information to the SM register from the output of the ALU.

The SM register bits are numbered as follows for 16 bit and 32 bit processors.

16 bit CPU: SM1 bits numbered S100 to S115;
SM2 bits numbered S200 to S215.

32 bit CPU: SM1 bits (first module) numbered
S100 to S115
SM1 bits (second module) numbered
S116 to S131
SM2 bits (first module) numbered
S100 to S115
SM2 bits (second module) numbered
S216 to S231.

The individual bits of an SM module have the following additional characteristics.

Module Bits	Functional Characteristics
SM1 (0-3) SM1 (16-19) SM2 (0-3) SM2 (16-19)	These bits are flag bits which can be set or reset by the SETF and CLRF commands in a microinstruction. These commands provide for addressing up to 16 flags. In a 16 bit processor bits 0-3 of SM1 are address as flags 0-3 and bits 0-3 of SM2 are addressed as flags 4-7.

In a 32 bit processor with two SM modules, bits 0-3 of SM1 are flags 0-3, bits 16-19 of SM1 are flags 4-7, bits 0-3 of SM2 are flags 8-11 and bits 16-19 of SM2 are flags 12-15.

The flag bits are not cleared by a processor master clear, and no external input is provided to these bits.

SM1 (0-3) and SM1 (16-19) are available as a true output from the SM module. SM2 (0-3) and SM2 (16-19) are available as a complement output from the SM module.

SM1 (4-7)
SM1 (20-23)
SM2 (4-7)
SM2 (20-23)

These bits can be set by an external signal and cleared by a processor master clear. These bits are all available as a true and complement output from the SM module.

SM1 (08-11)
SM1 (24-27)
SM2 (08-15)
SM2 (24-31)

These bits can be set by an external signal and can be cleared by a processor master clear. SM2 (14-15) and SM2 (30-31) can also be cleared by an external signals. These bits are available only as true outputs from the SM module.

SM1 (12-15)
SM1 (28-31)

These bits can be set by an external signal and can be cleared by a processor master clear. SM1 (14-15) and SM1 (30-31) can also be cleared by an external signal. These bits are available only as complement outputs from the SM module.

The assignment of status and mode conditions to specific bits of the SM register is a design function. The following assignment of status and mode fits in the minimum standard assignment. Bit assignments which are optional are listed with parenthesis. Bits are identified as S for status and M for mode.

BIT	TYPE	FUNCTION
100	M	(Double Precision)
101	M	1's complement
102	M	Bit generator input from N
103	M	(Adder Split)
104	S	Main Memory Parity Error
105		Open
106	M	(Enable Interrupt System)
107		Open
108		Open
109	M	Micro-Halt
110	S	Overflow
111	M	(Enable F1)
112		Open
113	M	Select XT/MA (R/W MM)
114		Open
115		Open
200		Open
201		Open
202		Open
203		Open
204	M	Autoload
205		Open
206		Open
207	M	Select Page XT/MA
208		Open
209		Open
210	M	Enable MM to MIR
211	M	Enable DMA to MIR
212		Open
213	S	Console Request Control
214		Open
215		Open

3.3.1.3.14.1 CPU Operating Modes

The following CPU operating modes are incorporated into a system during manufacture. The operations are controlled by a mode fit in the SM register, but if an option is not included, the associated mode bit is open in the SM register.

Double Precision Arithmetic (SM100)

If this mode bit is set to 1 and the double precision hardware option is included, the ALU and the ALU* on the double precision option module are combined to form a double-word length ALU. This double-word length ALU operates for addition and subtaction but not for logical operations.

If this mode bit is set to 0, the double precision ALU* is disconnected from the ALU and no operation takes place in ALU*.

1's Complement (SM101)

If this mode bit is set to 1, the ALU (and ALU* for double precision operation) operates in 1's complement arithmetic mode for addition and subtraction.

If the mode bit is set to 0, operations are in 2's complement arithmetic mode for addition and subtraction.

BG Input From N (SM102)

If this mode bit is set to 1, the bit generator is controlled by the lower five bits of the N register.

If this mode bit is set to 0, the bit generator is controlled by the lower five bits of the microinstruction.

Adder Split (SM103)

If this mode bit is set to 1,the ALU is split into two independent ALU's at the adder split point for arithmetic operation. The split point is between bits 07 and 08 on a 16-bit processor and between bits 15 and 16 on a 32-bit processor. On a 16-bit processor, both the upper and lower portion of the ALU operate in 2's complement arithmetic. In a 32-bit processor if 1's complement mode is selected, the upper adder operates in 2's complement mode while the lower adder operates in 1's complement mode.

If this mode bit is set to 0, the ALU operates as a single word-size ALU.

Enable Interrupt System (SM106)

If this mode bit is set to 1, the interrupt system of the processor is activated and the INTU test can examine the interrupt system.

The interrupt lines to be enabled/disabled by this mode bit are selectable in groups of four interrupt lines. Therefore, on a 16-bit processor any combination of four groups can be controlled by this mode bit and on a 32-bit processor any combination of eight groups can be controlled by this bit. Groups not controlled by the mode bit can be constantly enabled or disabled.

If this mode bit is set to 0, the interrupt system is disabled and the INTU test will always receive a reply of "no interrupt present" on groups controlled by the mode bit.

Micro Halt (SM109)

If this mode bit is set to 1, any microinstruction with a HALT code in the S field stops the operation of the processor on completion of the microinstruction.

If this mode bit is set to 0, the HALT code in the S field of any microinstruction is ignored.

Enable F1 (SM111)

If this mode bit is set to 1, the output of File 1 is enabled to Selector S1 or S2. This overrides any selection of an external source.

If this mode bit is set to 0, the output of File 1 is disabled as an input source to Selector S1 or S2. If an external source (such as the transform module) is available it may be enabled to selector S1 or S2 at the same position as defined for F1.

Note that this mode bit must be set to 1 in order to write into F1.

Select XT/MA (R/W MM) (SM113)

If this mode bit is set to 1 the MA transform determines the micromemory address for read/write micromemory operand references.

If this mode bit is set to 0, the combination N/K register determines the micromemory address for read/write micromemory operand references.

Autoload (SM204)

If this mode bit is set to 1 the autoload mode is selected on the console interface.

If this mode bit is set to 0 the autoload mode is disabled on the console interface.

Select Page XT/MA (SM207)

If this mode bit is set to 1 an MA transform can be executed across a page boundary, by setting the desired page in the S field of the microinstruction. Normal S field operations are disabled.

If the mode bit is set to 0 an MA transform is limited to the page in which the microinstruction containing the MA transform command resides. The S field of the microinstruction is decoded for normal operation.

Enable MM to MIR (SM210)

If this mode bit is set to 1, the output of the micromemory is pre-enabled to the MIR. A page jump or return jump microinstruction must be executed with this mode bit set to 1 to force the actual selection to the page and address selected.

If the mode bit is set to 0, the output of the console interface is pre-enabled to the MIR. A page jump or return jump microinstruction must be executed with this mode bit set to 0 to force the actual selection. If the processor is master cleared, the console interface is enabled to the MIR.

Enable DMA to MIR (SM211)

If this mode bit is set to 1 and SM210 is set to 0 and a page jump or return jump microinstruction is executed, main memory data via the console interface is selected to the MIR for microinstruction execution at the page and address selected.

If this mode bit is set to 0 and SM210 is set to 1 and a page or return jump microinstruction is executed, control is transferred back to micromemory at the page and address selected.

3.3.1.3.14.2 MPP Status

The following CPU status bit assignments are incorporated into a system during manufacture. All of these status bits are set by the condition detected; the clearing of the status bit must be performed by the microprogram with the exception of SM1 (bits 14, 15, 30, 31) and SM2 (bits 14, 15, 30, 31) which have an external clear input.

Main Memory Parity Error (SM104)

This status bit is set to 1 if the main memory interface detects a parity error on data read from memory.

Overflow (SM110)

This status bit is set to 1 on detection of an overflow condition. The processor provides for three selectable overflow conditions to be tested.

. Arithmetic overflow - microinstruction performing the arithmetic operation is an add or subtract with overflow test and the arithmetic result is inconsistent with the sign of the operands and the arithmetic operation.

. Binary overflow - microinstruction performing the arithmetic operation is an add or subtract with overflow test and a carry out of the most significant bit of the ALU occurred. This condition is tested if an optional binary overflow SM bit is set to 1.

. Decimal Overflow - microinstruion performing the arithmetic operation is a decimal add or subtract with overflow test and decimal overflow occurred. This condition is tested if an optional decimal arithmetic SM bit is set to 1 and the decimal arithmetic algorithm is included on the transform module.

Console Request Control(SM213)

This status bit is set to 1 if the control console interface is requesting control via the MIR. The microprogram should return control to the console via SM210. This status bit is wired to an interrupt line.

3.3.1.3.15 Interrupts and Mask Register

The interrupt system is implemented as a sampled data system at the microprogram level instead of as a true interrupt as used in conventional computers. That is, the interrupt system provides a sampling capability in which a microinstruction can sample the interrupt system to see if there is any interrupt present that has its corresponding mask register bit set to 1. This sample is taken by performing an INTU operation in the T field of any microinstruction. If there is an interrupt in the system whose mask register bit is 1 and the interrupt system is enabled the next microinstruction is executed from the upper of the next microinstruction pair. If there is no such interrupt, the next microinstruction is executed from the lower of the next microinstruction pair.

When an interrupt is recognized, the microprogram samples the interrupt address encoder to identify the most significant (highest priority) interrupt. The interrupt address encoder must be read in the microinstruction following the interrupt test, to be sure of a correct interrupt line address. If the interrupt address is read earlier or later, there is a possibility that the address encoder output is unstable due to a newly-arrived interrupt. The interrupt address is read by performing an INTA operation in the B' field of any microinstruction.

No standard interrupts are defined; the use of the interrupt system and the design of interrupts are functions of the application. Interrupts are identified by the corresponding mask bits which are assigned to control the interrupt recognition. The bits in the mask registers are identified as follows:

TABLE 3-19. MPP INTERRUPT ADDRESSES (M1)

	Interrupt Address (Base 10)	
Mask Bit	16 Bit Processor	32 Bit Processor
M100	15	47
M101	14	46
M102	13	45
M103	12	44
M104	11	43
M105	10	42
M106	9	41
M107	8	40
M108	7	39
M109	6	38
M110	5	37
M111	4	36
M112	3	35
M113	2	34
M114	1	33
M115	0	32
M116	--	15
M117	--	14
M118	--	13
M119	--	12
M120	--	11
M121	--	10
M122	--	9
M123	--	8
M124	--	7
M125	--	6
M126	--	5
M127	--	4
M128	--	3
M129	--	2
M130	--	1
		0

TABLE 3-20. MPP INTERRUPT ADDRESS (M2)

	Interrupt Address (Base 10)	
Mask Bit	16-Bit Processor	32-Bit Processor
M200	31	63
M201	30	62
M202	29	61
M203	28	60
M204	27	59
M205	26	58
M206	25	57
M207	24	56
M208	23	55
M209	22	54
M210	21	53
M211	20	52
M212	19	51
M213	18	50
M214	17	49
M215	16	48
M216	--	31
M217	--	30
M218	--	29
M219	--	28
M220	--	27
M221	--	26
M222	--	25
M223	--	24
M224	--	23
M225	--	22
M226	--	21
M227	--	20
M228	--	19
M229	--	18
M230	--	17
M231	--	16

Mask Register 1 (M1)

M100 through M115 (16-bit processor)
M100 through M131 (32-bit processor)

Mask Register 2 (M2)

M200 through M215 (16-bit processor)
M200 through M231 (32-bit processor)

Interrupt addresses are generated by the interrupt address encoder according to the assignments for a 16-bit and 32-bit processor given in Tables 3-19and 3-20.

The interrupt priorities correspond to the interrupt address generated; that is, interrupt address 0 is associated with the highest priority interrupt line and interrupt address 31 is associated with the lowest priority interrupt line in a 16-bit processor. As an example, an interrupt associated with M112 would have priority over an interrupt associated with M111 in a 16-bit processor and an interrupt address of 3_{10} would be developed by the interrupt address encoder.

The output from the interrupt address encoder is the complement of the interrupt address for input to S2; thus the transfer of the interrupt address to the X register, for example, would be coded using a -B code in the F field, INTA in the B' field, and X in the D field. This results in the correct interrupt address being transferred.

A design option in the interrupt system provides for activating interrupts in groups of four interrupt lines. Therefore, on a 16-bit processor, any combination of four groups can be controlled by the enable interrupt SM bit(s) and on a 32-bit processor any combination of eight groups can be controlled by SM bit(s). Groups not controlled by SM bit(s) can be allowed to remain active while the remaining interrupts may be enabled or disabled.

Interrupt signals must be steady state when inputted to the interrupt system and indicate the presence of an interrupt when set to a 0. If a pulse type interrupt is required, the pulse interrupt signal is used to set a bit in the SM register; this SM bit is then wired to the interrupt system. On recognizing this interrupt, the microprogram is able to clear the interrupt condition by clearing the SM bit.

3.3.1.3.16 K Register

The K register is an 8-bit counter that can be cleared, incremented, or decremented. It is used to address file 1 in addition to any program usage as a counter. The original value of K can be tested against zero by the microinstruction. The contents of K is selectable as an input to S2 via main CPU tri-state buss.

3.3.1.3.17 N Register

The N register is an 8-bit counter which may be cleared, incremented, or decremented. It is used to address file 2, control shifts, control the scale operations, and maybe used as an iteration counter which controls microinstruction execution for operations such as multiplication and division. It may also be used as a programmed counter, since the original value of N can be tested against zero by the microinstructions. N is selectable as an input to S2 with RTJ.

3.3.1.3.18 N/K Register

The K and N register may be combined to provide operand addresses outside the current operating page. They are gated through S6 to P/MA for this operation. In this operation the five least significant bits of N provide the upper portion of the twelve bit address. The seven most significant bits of K provide the lower portion of the twelve bit address. The least significant bit of K is used to determine upper or lower. A S/M bit determines whether to use N/K Data or the output of the transform module for this operation

3.3.1.4 Microinstruction Formats

Each micromemory address specifies the location of two microinstructions. Each 32-bit microinstruction is divided into five main sections and is numbered from left to right as bits 0 to 31.

Bits	Function
0,1	Mode (M) field specifies format of S and C field and sequencing mode to obtain next microinstruction pair
2-15	ALU control field specifies ALU operation, sources of operands, and destination of result of operation
16-18	Test (T) field specifies method of selecting which microinstruction of next microinstruction pair to execute
19-23	Special (S) field specifies subformat selection and special operation
24-31	Constant (C) or suboperation field specifies constants, micromemory addresses, or other codes

The total instruction appears as follows:

0 1	2 ... 15	16 ... 18	19 ... 23	24 ... 31
M	ALU Control	T	S	C

The M field specifies one of three addressing modes to be used to obtain the next microinstruction pair from micromemory and also specifies the format to be used in interpreting bits 19 to 31 of the microinstruction as follows:

M	Addressing Mode	Format for bits 19 to 31
00	Return	Format 1
01	Sequential	Format 1
10	Jump	Format 2
11	Sequential	Format 3

Figure 5 shows all microinstruction formats. Note that format 1 and format 2 microinstructions have two subformats, which are selected by the value of bit 19.

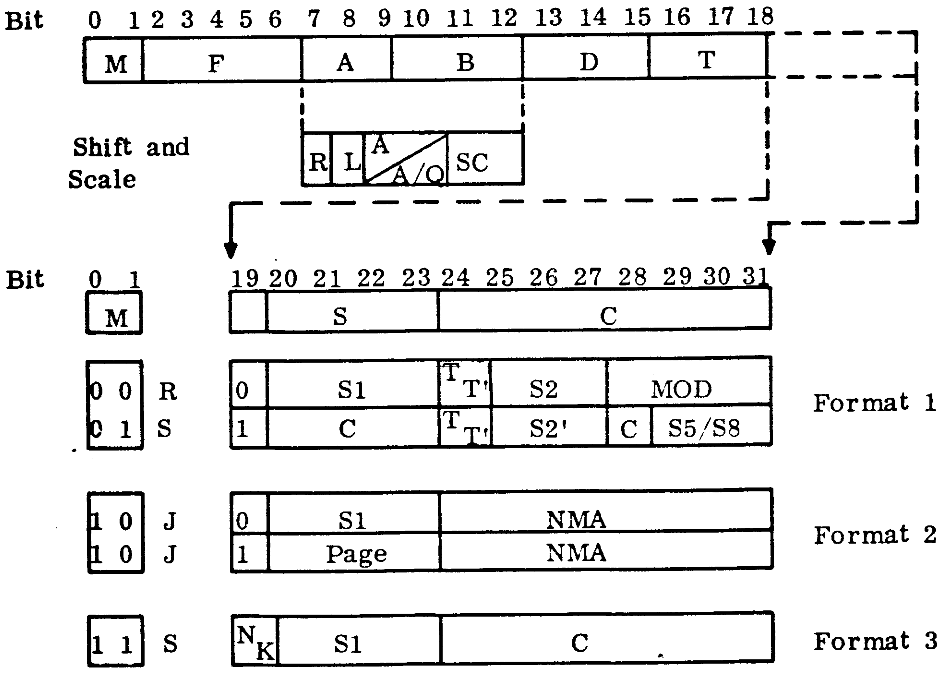


Figure 5 Microinstruction Formats

The ALU control fields specify the sources of two operands on which an arithmetic, logical, shift, or scale operation is to be performed and specify the destination of the result of the operation. For arithmetic and logical operations, the ALU control fields consist of the ALU function (F), A source (A), B source (B), and destination (D) fields, shown as follows:

Bit	2 6	7 9	10 12	13 15
	F	A	B	D

For shift and scale operations, the A and B fields are interpreted as follows:

Bit	2 6	7	8	9	10	11	12	13 15
	F	R	L	A AQ	▨	S	C	▨

The F field specifies shift or scale operation. Bits 7 and 8 specify right or left shifting. Bit 9 specifies whether the A register alone or the A and Q registers together are to be shifted or scaled. Bit 10 is not used, and bits 11 and 12 specify the shift control code. The D field contains a no-operation code for shift and scale operations.

The T field is the conditional branch of the microinstruction and specifies which microinstruction, upper or lower, of the next microinstruction pair to execute. The test can be based on the result of the ALU operation of the current microinstruction or on some other condition.

The codings in the S and C fields depend upon the contents of the M field. The S and C fields are coded in three formats. Format 1 is specified when the M field contains 00 (return mode) or 01 (sequential mode) as follows:

Bit	19	20 23	24	25 27	28 31
	0	S1	T T'	S2	MOD

Format 1

Bit	19	20 23	24	25 27	28	29 31
	1	C	T T'	S2'	C	S5/S8

The S1 field specifies operations such as main memory read or write operations; alternate codings to be used in the A, B, and D fields; etc. The T/T' bit specifies that the code in the T field is to be interpreted as the normal T code (T/T' = 0) or as the alternate T' code (T/T' = 1). The subformat select bit, bit 19, determines whether bits 25 through 31 are to be interpreted as S2 codes or as S2' codes. The S2 code can be a constant for driving the bit generator, additional information to control the main memory read or write, or a code to initiate an I/O transfer or other operation. The S2' and S5/S8 codes are associated with transforms.

Format 2 is specified when the M field contains 10 (jump mode), as follows:

Bit 19	20 23	24 31
0	S1	NMA
1	PAGE	NMA

Format 2

If a jump is specified to a microinstruction pair within the same micromemory page, the subformat select bit is 0; bits 20 to 23 contain a special operation code as in format 1. Bits 24 through 31 contain the micromemory address of the next microinstruction pair. The subformat select bit is 1 when a jump is specified to a different micromemory page; bits 20 through 31 contain the complete micromemory address of the next microinstruction pair.

Format 3 is specified when the M field contains 11 (sequential mode), as follows:

Bit 19	20 23	24 31
N/K	S1	C

This format allows one special operation to be performed as specified by the S1 code and also causes the eight bits of the C field to be transferred to the N register (bit 19 = 1) or to the K register (bit 19 = 0).

3.3.1.4.1 M Field (Bits 0 and 1)

The M field defines the major operation taking place in the microinstruction and also specifies the type of sequencing which will be used to obtain the next instruction pair. The operations specified in the M field are listed in Table 3.1.

TABLE 3.1. M FIELD OPERATIONS

M Code	Mnemonic	Operation
00	R	Select next microinstruction pair in current page from address contained in RTJ register. Use format 1 for special operations.
01	S	Select next microinstruction pair in current page from address contained in MAC (normally next sequential pair, unless suppressed by T field coding). Use format 1 for special operations.
10	J	A jump or page jump. Select next microinstruction pair from address specified by bits 24 to 31 of this microinstruction. Address is in current MM page if bit 19 is 0 or from page specified in bits 20 through 23 if bit 19 is 1. Use format 2 for special operations.
11	S	Transfer bits 24 through 31 of this microinstruction to N or K register as specified by bit 19 of this microinstruction. N register is specified if bit 19 is 1, and K register is specified if bit 19 is 0. Select next microinstruction pair in current page from address contained in MAC (normally next sequential microinstruction pair). Use format 3 for special operations.

3.3.1.4.2 F Field (Bits 2 through 6)

The F field specifies the logical or arithmetic operation to be performed by the ALU or the shift or scale operations performed with the A and Q registers. The split adder and the double precision hardware options are described in the following paragraphs.

The split adder option allows the main ALU to be split into two independent adders. This split is activated by setting the adder split flag in the SM register. The split blocks the carry between the two portions of the adder. The upper portion of the adder always functions as a 2's complement adder; the lower portion can function as a 1's complement or as a 2's complement adder, depending upon the state of the 1's complement SM register flag (32-bit processor only). In 1's complement mode the carryout of the lower portion is used as the end-around carry bit. In 2's complement mode, both portions of the adder act as independent 2's complement adders. The split adder has no effect on logical operations because no carry is involved in these operations.

The double precision hardware arithmetic option provides the capability to perform arithmetic on double-length operands. The double precision logic contains three additional registers (A*, Q*, X*) and a second ALU (ALU*) distinct from the main MPP elements. The A* register is unconditionally inputted to ALU*. The output of ALU* can be shifted left and right in a multiply or divide operation, and the output goes to the A* register. The X* and Q* registers are loadable only; they cannot be specified as destinations for the results from ALU*. The Q* register can be shifted during double precision multiply or divide iterations. The double precision logic, if present, is enabled when the double precision flag is set in the SM register. Figure 6 is a block diagram of the double precision logic.

LOGICAL OPERATIONS

The logical operations perform bit-by-bit combinations of the A input and the B input for delivery to the destination. Double precision logical operations cannot be performed on the A* and X* registers.

The logical operations are described in Table 3-2.

ARITHMETIC OPERATIONS

The arithmetic operators (Table 3-3) operate on either single precision operands (using the main ALU) or double precision operands (if the double precision logic is present and the double precision flag is set in the SM register). Two additional options are provided and coded in the arithmetic function code. The first option provides for a carry input to the adder (indicated by a plus sign in the microinstruction mnemonic). This is used for doing multiple precision arithmetic beyond that provided by the hardware logic. A T field code to check for a carryout of the ALU is provided to determine whether a carry into the ALU should be used on the next arithmetic operation. With the double precision bit set in the SM register, the carryin is entered in the lower bit of the double precision ALU; otherwise, the carryin is entered in the main ALU.

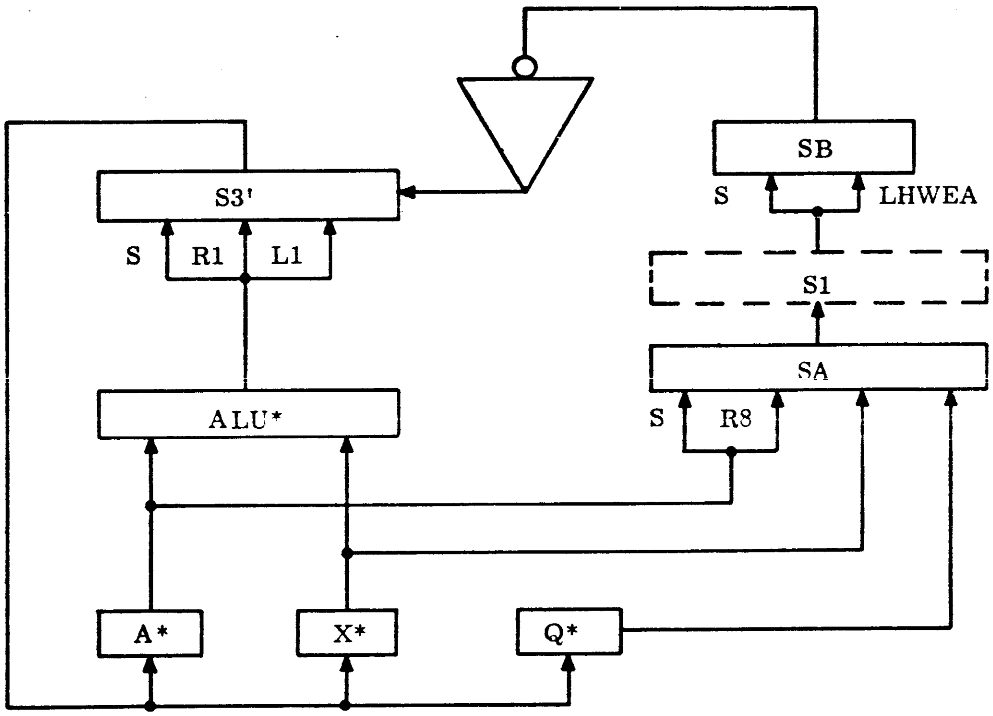


Figure 6. Double Precision Option Block Diagram

TABLE 3-2. LOGICAL OPERATIONS

F Code	Mnemonic	A input = 0 0 1 1	B input = 0 1 0 1
		Bit Result	
01100	ZERO	0 0 0 0	
01110	A•B	0 0 0 1	
01101	A•-B	0 0 1 0	
01111	A	0 0 1 1	
01000	-A•B	0 1 0 0	
01010	B	0 1 0 1	
01001	EOR	0 1 1 0	
01011	A+B	0 1 1 1	
00100	-A•-B	1 0 0 0	
00110	-EOR	1 0 0 1	
00101	-B	1 0 1 0	
00111	A+-B	1 0 1 1	
00000	-A	1 1 0 0	
00010	-A+B	1 1 0 1	
00001	-A+-B	1 1 1 0	
00011	ONE	1 1 1 1	

TABLE 3-3. ARITHMETIC OPERATIONS

F Code	Mnemonic	Operation
10100	SUB	Subtract B input from A input.
11000	ADD	Add A and B inputs.
10101	SUBT	Subtract with an overflow test.
11001	ADDT	Add with an overflow test.
10110	SUB-	Perform A-B-1 input (2's complement only).
11010	ADD+	Perform A + B + 1.
10111	SUB-T	Perform SUB- with an overflow test (2's complement only).
11011	ADD+T	Perform ADD+ with an overflow test.

SCDD

The second option allows capture of the overflow condition in the SM register (indicated by a T in the instruction mnemonic). If this is indicated, the overflow test is performed by checking the sign of the two inputs to the ALU and by setting a status/mode bit if the result is inconsistent. The status/mode overflow bit is set to 1 when the overflow occurs; it must be set to 0 by a microinstruction which sets that status/mode bit to a 0. Two other optional S/M bits allow checking for binary or decimal arithmetic overflow.

SHIFT OPERATIONS

The shift operations in the F field specify a shift of the A register or the AQ register of the main MP organization only; no shift is possible in the double precision registers from this command. The ALU is not used to perform the shift, but will perform some operation based on its decoding of the F field (which should be considered as unknown). The destination will receive this meaningless output unless an NOP is chosen for the destination of the D field.

The type of shift is determined by the coding in bits 7 to 12 of the microinstruction, and the amount of the shift is determined by the number contained in the N register. The operation examines the N register, and, if it is zero, the next microinstruction is executed. The T field codes normally used with a shift are U, L, BIT TEST, and LQL. Other T codes must be used with caution.

If the N register is not zero, a shift of one bit position is taken as specified, N is decremented by one, and the test for zero is repeated as above.

The shift conditions are:

- Shift A - Shift the A register only.
- Shift AQ - Shift the combined A and Q register. The Q register is considered to be the least significant bits of the combined AQ register.
- Shift External (Transform)
- Shift left or right.
- Enter 0 - Enter a 0 in the vacated bit position at the end of the register.
- Enter 1 - Enter a 1 in the vacated bit position at the end of the register.
- Extend sign - Extend the sign (for a right shift only).
- End-around carry - Enter the bit coming off the end of the register into the vacated on position at the other end.

All shifts are performed with an F code of 111X0. The type of shift is determined by bits 7 through 12 of the microinstruction. The shifts are defined in Table 3-4. If F = 11100 the shifting of A or AQ is inhibited and an external shift is provided for optional use on a transform module.

If F = 11110 the external shift is inhibited and A or AQ shifts are enabled.

TABLE 3-4. SHIFT OPERATIONS

Bit Code		Mnemonic	Operation
7 8 9	11 12		
1 0 0	0 0	AR0E	A is right shifted (N) bits, with 0 entered at most significant bit.
1 0 0	0 1	ARSE	A is right shifted (N) bits, with sign extension.
1 0 0	1 0	AREA	A is right shifted (N) bits, with end-around carry.
0 1 0	0 0	AL0E	A is left shifted (N) bits, with 0 entered as least significant bit.
0 1 0	0 1	AL1E	A is left shifted (N) bits, with 1 entered as least significant bit.
0 1 0	1 0	ALEA	A is left shifted (N) bits, with end-around carry.
1 0 1	0 0	AQR0E	AQ is right shifted (N) bits, with 0 entered as most significant bit in A.
1 0 1	0 1	AQRSE	AQ is right shifted (N) bits, with sign extension.
1 0 1	1 0	AQREA	AQ is right shifted (N) bits, with end-around carry.
0 1 1	0 0	AQL0E	AQ is left shifted (N) bits, with 0 entered at least significant bit in Q.
0 1 1	1 0	AQLEA	AQ is left shifted (N) bits, with end-around carry.

NOTE: External shift operations are controlled by same mode lines as the A register.

SCALE OPERATIONS

The scale operations are similar to the shift operations but the stopping of the shift is conditioned on bits 0 and 1 of A not being equal. (The scale point is normally between bits 0 and 1 of the A register. A design option allows the scale point to be specified between different bits in the A register if necessary for efficient floating point emulation.) The maximum number of bits to scale is contained in the N register, and, on completion of the scale, N is decremented by the number of shifts which were necessary to scale the number.

The scale operation is performed as follows:

1) Examine N; if it is zero, exit the microinstruction.

2) Examine bits 0 and 1 of the A register; if they differ, exit the microinstruction.

3) Shift the A or AQ register left by one bit position as specified in the instruction.

4) Decrement the N register by one count and go to step 1.

The scale operation is coded the same in bits 7 through 12 of the microinstruction and allows the same left shift options as the shift command. The same comments on exiting the shift and the usable T field codes apply to the scale operation.

All scale operations are performed with an F code of 11111. The type of shift for the scale is determined by bits 7 through 12 of the instruction. The scales are given in Table 3-5.

TABLE 3-5. SCALE OPERATIONS

Bit Code			Mnemonic	Operation
7 8 9	11	12		
0 1 0	0	0	SL0E	A is scaled left, with 0 entered as least significant bit.
0 1 0	0	1	SL1E	A is scaled left, with 1 entered as least significant bit.
0 1 0	1	0	SLEA	A is scaled left, with end-around carry.
0 1 1	0	0	SDL0E	AQ is scaled left, with 0 entered as least significant bit in Q.
0 1 1	1	0	SDLEA	AQ is scaled left, with end-around carry.

3.3.1.4.3 A Field {Bits 7 through 9}

The A field specifies the input to S1 and thus to the A side of the ALU. The eight A codes are expanded by eight A' codes by placing 1010 or 0111 in the S1 field.

The eight A inputs to S1 and to the A side of the ALU are indicated when the S1 field is not 0111 or 1010; the eight A codes specify inputs from the files, the registers, or main memory, according to Table 3-6.

TABLE 3-6. A INPUT OPERATIONS

A Code	Mnemonic	Operation
000	F2	Use contents of file 2 register as A source input. Current value of N register is used to address register file 2. If value of N is changed in current microinstruction, its initial value is used to reference file register.
001	P	Use contents of P register as A source.
010	I	Use contents of I register as A source.
011	X	Use contents of X register as A source.
100	A	Use contents of A register as A source.
101	F	Use contents of F register as A source.
110	F1	Use contents of file 1 register or external source as A source. Current value of K register is used to address register file 1. If value of K is changed in current microinstruction, initial value of K is used to reference file register.
111	MEM	Obtain data read from main core memory and use it as A source. RESTRICTION: A main memory READ command must be given in the preceding microinstruction. This command is restricted to a microinstruction with type A, B or C, execution time.

The eight A' inputs to S1 and to the A side of the ALU are indicated if the S1 field is 0111 or 1010. The A' codes specify input from the SM registers or mask registers. The A' codes are given in Table 3-7.

TABLE 3-7. A' INPUT OPERATIONS

A' Code	Mnemonic	Operation
000	SM1	Use contents of SM register 1 as A source.
001	M1	Use contents of interrupt mask register 1 as A source.
010	SM2	Use contents of SM register 2 as A source.
011	M2	Use contents of interrupt mask register 2 as A source input.
100	A*R8	Use contents of double precision A* register, shifted right eight bits with end-around carry, as A source. A* register remains unshifted.
101	A*	Use contents of double precision A* register as A source.
110	X*	Use contents of double precision X* register as A source.
111	Q*	Use contents of double precision Q* register as A source.

3.3.1.4.4 B Field (Bits 10 through 12)

The B field specifies the input to S2 and thus to the B side of the ALU. The 11 possible B codes are expanded by the seven B' codes when the S field contains 1000. The N and RTJ codes are controlled by bits 28 and 29 from the C field.

The 11 B inputs to S2 and thus to the B side of the ALU are indicated if the S field is not 1000. Code 001 of the B field is expanded by the use of bits 28 and 29 of the microinstruction for enabling the N or RTJ register to S2. This use of bits 28 and 29 is independent of the other use of the C field, and thus, by judicious use of commands in the C field, the input for the N and RTJ may be used in conjunction with commands or constants in the C field. The codes for B inputs are given in Table 3-8.

The B' inputs are specified in the B field when the S field contains 1000. The codes and actions are given in Table 3-9.

TABLE 3-8. B CODES

B Code	28 29	Mnemonic	Operation
000		F2	Use contents of file 2 register as B source. Value of N register, before instruction is executed, is used to address register file 2.
001	1 1	ZERO	B source is all zeros.
001	1 0	N	Use contents of N register as B source. Since N is 8-bit register, this source uses N as upper eight bits and zeros as lower bits.
001	0 1	RTJ	Use contents of RTJ register as B source. Since RTJ is 8-bit register, upper bits are zeros and (RTJ) serves as lower eight bits.
001	0 0	N, RTJ	Use contents of N and RTJ registers as B source. These registers are combined, with N register as upper eight bits of source and RTJ as lower eight bits. All other bits (if any) are zero.
010		BG	Use contents of BG register as B source. This register has only one bit set to 1, and position of bit in BG register is specifiable. Position is specified either on value in the N register or by number in C field, depending on state of controlling SM register bit.
011		X	Use contents of X register as B source.
100		Q	Use contents of Q register as B source.
101		F	Use contents of F register as B source.
110		F1	Similar to F2, but uses the contents of file 1 register addressed by (K) as B source.
111		MEM	Obtains data read from main core memory and uses it as B source. RESTRICTION: A main memory READ command must be given in the preceding microinstruction. This command (MEM) is restricted to a microinstruction with Type A, B or C, execution time.

TABLE 3-9. B' CODES

B' Code	Mnemonic	Operation
000	OPEN	
001	K	Transfer the K register to the eight LSB's of S2.
010	INRD	Input data/status from I/0 channel.
011	INRS	Input to 52 I/0 response signals.
100	MMU	Transfer upper 16 bits of data from micromemory to X register in 16-bit MPP; 32-bit MPP transfers total 32-bit word. F field must make a reference to B source. Address is specified by transform or NK. "D" field must be a NOP.
101	MML	Same as above, but takes lower 16 bits of data in a 16-bit MPP.
110 or 111	INTA	Use contents of interrupt address encoder as B source. The output of this encoder represents the complement of the interrupt address of the highest priority interrupt line active having it's corresponding mask bit set. RESTRICTION: A INTU test command must be given in the preceding microinstruction.

SCDD

3.3.1.4.5 D Field (Bits 13 through 15)

The D field specifies the destination of information from the main organization of the MPP. There are four sources of this information as follows:

- An optionally shifted ALU output. This shifting occurs in the S3 shift network that connects the ALU output to the P, A, F, X, or Q registers
- The output of the ALU
- The A source
- The B source

All D destinations except the I register are optionally shiftable by S3 when specified by a code in the C field or by the L8EA command in the S1 field, if the alternate codings, D' or DD", are not specified. The I destination differs from the others in that the output of the A source is the input to the I register. The codes and their operations are given in Table 3-10.

The D' destinations are specified by the D field if the S1 field is set to 1001 or 1010. The codes and action are given in Table 3-11.

The D" destinations are specified by the D field if the S1 field is set to 1011. These destinations transfer data to the double precision logic from S1. The codes and actions are given in Table 3-12.

The DD" option allows the performance of an operation on A, X, or F; this changes the register, but keeps a copy of the original register in a double precision register. This is another way of getting data to the double precision registers. The DD" option is specified when the S field contains 0001. Table 3-13 lists the DD" codes and their operations.

TABLE 3-10. D CODE TRANSFERS

D Code	Mnemonic	Operation
000	NOP	Do not transfer data to any destination.
001	P	**Transfer output of S3 to P, AB
010	I	Transfer output of S1 to I , AB
011	Q	Transfer output of S3 to Q, AB
100	F1	*Transfer output of S3 to F register, AB, and write this data in file 1 at address specified by K at completion of this instruction.
101	A	Transfer output of S3 to A, AB
110	X	Transfer output of S3 to X, AB
111	F	Transfer output of S3 to F, AB

*NOTE

The writing of data into the file 1 register takes place during the first part of the next microinstruction and takes advantage of the updated value of K from this microinstruction. Also, the next microinstruction must not specify a read of file 1.

**NOTE

If a D field command to load the AB is issued in the next microinstruction following the microinstruction with this command, the transfer to AB is inhibited.

TABLE 3-11. D' CODE TRANSFERS

D' Code	Mnemonic	Operation
000	IOD	Transfer output of S3 to I/O Data Register.
001	IOA	Transfer output of I/O Data Register to I/O Address Register. Destroys contents of I/O data register.
010	MMU	Transfer output of S2 to upper 16 bits of micromemory in 16-bit MP, or transfer output of S2 to 32-bit word in micromemory in 32-bit MP.
011	MML	Transfer output of S2 to lower 16 bits of micromemory location in 16-bit MP.
100	M1	Transfer output of ALU to mask register 1.
101	SM1	Transfer output of ALU to SM register 1.
110	M2	Transfer output of ALU to mask register 2.
111	SM2	Transfer output of ALU to SM register 2.

NOTE

Outputs to the mask and SM registers are direct from the ALU and are not shiftable.

TABLE 3-12. D" CODE TRANSFERS

D" Code	Mnemonic	Operation
000	NOP	Do not transfer data to any destination.
001	A*LHW	Transfer output of S1 to A* register, shifted left one-half word, with end-around carry.
010	X*LHW	Transfer output of S1 to X* register, shifted left one-half word, with end-around carry.
011	Q*LHW	Transfer output of S1 to Q* register, shifted left one-half word, with end-around carry.
100	NOP	Do not transfer data to any destination.
101	A*	Transfer output of S1 to A* register.
110	X*	Transfer output of S1 to X* register.
111	Q*	Transfer output of S1 to Q* register.

TABLE 3-13. DD" CODES

DD" Code	Mnemonic	Operation
101	AA*	Transfer output of S3 to A register, and transfer output of S1 to A* register.
110	XX*	Transfer output of S3 to X register, and transfer output of S1 to X* register.
111	FQ*	Transfer output of S3 to F register, and transfer output of S1 to Q* register.

3.3.1.4.6 T Field {Bits 16 through 18}

The purpose of the T field is to select the upper or lower microinstruction of the next microinstruction pair to execute. The selection of the next microinstruction may be a conditional selection or an unconditional selection, depending on the T field code. This field is available to all addressing modes in addition to I/O operations. A conditional selection may test the ALU output, the value of certain registers, certain internal conditions such as interrupts, and particular bits wired to the transform board. The only exception to these uses of the T field is when micromemory data is being read or written; in these cases, the T field is used as part of the micromemory data reference address and the upper instruction in the next sequential microinstruction pair is always selected.

The T field codes consist of two groups, T codes, and T' codes. Similarly to the A, B, and D fields that are extended by using the S field, the T field is always extended in the following sense. Bit 24 of format 1 microinstructions is either 0 or 1 if T or T', respectively, is specified. The T' codes are available only for the return and sequential addressing modes (format 1). The T/T' codes select the upper or lower portion of the next microinstruction pair as the next microinstruction to execute. The T codes are listed in Table3-14;the T' codes are listed in Table 3-15.

TABLE 3-14. T ADDRESSING MODES

T Code	Mnemonic	Operation
000	*L	Execute lower microinstruction of this microinstruction pair as next microinstruction. This operation overrides M field addressing mode.
001	U	Execute upper microinstruction of next microinstruction pair.
010	L	Execute lower microinstruction of next microinstruction pair.
011	KZU**	If initial contents of K register is zero, execute upper microinstruction or next microinstruction pair; otherwise, execute lower microinstruction of next microinstruction pair. If decrement K command is included in same microinstruction, K will contain all ones on satisfying zero test.
100	NZU**	If initial contents of N register is zero, execute upper microinstruction of next microinstruction pair; otherwise, execute lower microinstruction of the next microinstruction pair. If decrement N command is included in same microinstruction, N will contain all ones on satisfying zero test.
101	INTU	If there is an interrupt and its corresponding interrupt mask bit is set, execute upper microinstruction of next microinstruction pair; otherwise, execute lower microinstruction of next microinstruction pair.
110	NU	If sign bit of ALU output is negative on completion of this microinstruction, execute upper microinstruction of next microinstruction pair; otherwise, execute lower microinstruction of next microinstruction pair.
111	ZL	If output of ALU is zero on completion of this instruction, execute lower microinstruction of next microinstruction pair; otherwise, execute upper microinstruction of next microinstruction pair.

**NOTE: Restriction - These T field commands can not be used in microinstruction with a C field INCK or INCN command respectively.

TABLE 3-15. T' ADDRESSING MODES

T' Code	Mnemonic	Operation
000	*L	Execute lower microinstruction of this microinstruction pair. This operation overrides M field addressing mode.
001	LQL	If, at start of this microinstruction, least significant bit of Q is 1, execute lower microinstruction of next microinstruction pair. Otherwise, execute upper microinstruction of next microinstruction pair.
010	K7L	If the LSB of K register is set execute lower of next micro-instruction. If clear execute upper microinstruction of next microinstruction pair.
011	ØVFL	If overflow exists execute lower microinstruction of next microinstruction pair, if not execute upper microinstruction of next microinstruction pair.
100	BTU	Bit test. Lower order four bits in C field of this microinstruction specify a setting of bit test selector. If bit at this position is 1, execute upper microinstruction of next microinstruction pair; otherwise, execute lower microinstruction of next microinstruction pair. Bit test is general purpose testing facility that allows wiring any bit of organization available on machine's backpanel to bit test selector. This wiring is defined on transform board.
101	LQ*L	If, at start of this microinstruction, least significant bit of Q* is 1, execute lower microinstruction of next microinstruction pair; otherwise, execute upper microinstruction of next microinstruction pair.
110	COL	Carryout lower. If as result of arithmetic operation, there is carryout of ALU, execute lower microinstruction of next microinstruction pair. Otherwise, execute upper microinstruction of next microinstruction pair. This instruction allows a test for carryout of ALU during multiple-precision arithmetic.
111	Z*L	Same as ZL (Table 3-14, except ALU* is tested.

3.3.1.4.6 Subformat Select Bit (Bit 19)

Bit 19 is used to select either variations in format 1 and format 2 decoding or the choice of addressing the N or K register in format 3.

3.3.1.4.7 S Field (Bits 20 through 23)

The S field of the microinstruction is used to specify a special command (including alternate codings in the A, B and D fields), in addition to page or constant information (as required by the code in the C field). The following S codes specify actions which take place at the same time as the ALU operation specified in the F, A, B, and D fields. The codes and operations are given in Table 3-16.

TABLE 3-16. S FIELD CODES

S Code	Mnemonic	Operation
0000	NOP	No operation for S field.
0001	DD	Alternate D field coding, D' .
0010	RPT	If N register is not equal to zero, selection of next microinstruction pair is inhibited and current microinstruction pair is next microinstruction pair. N register is decremented by one. Normal T field selection applies. If N register is equal to zero, normal next microinstruction pair is used.
0011	READ	Read main memory command. Read word from main memory at address contained in Address Buffer (AB). Instruction execution is delayed until a resume is received from memory acknowledging command. Restrictions: AB must be loaded in a previous microinstruction (D field command). If D field command reloading AB is issued in same microinstruction as READ command, the new data will not be loaded into AB until completion of read portion of cycle. A READ command can only be given in a microinstruction with type A,B, C or D execution time. Memory data must be input to system in following microinstruction with type A,B, or C execution time only.

TABLE 3-16. S FIELD CODES (Continued)

S. Code	Mnemonic	Operation
0100	WRITE	Write main memory. Transmit output of S3 to main memory as data to be written at address contained in Address Buffer (AB). Instruction execution is delayed until a resume is received from memory acknowledging command. Data is stored in memory at completion of this instruction. RESTRICTIONS: AB must be loaded in a previous microinstruction (D Field Command). If D field command reloading AB is issued in same microinstruction as WRITE command, the new data will not be loaded into AB until completion of WRITE portion of cycle.

TABLE 3-16. S FIELD CODES (Cont.)

S Code	Mnemonic	Operation
0110	F2WR	Write data contained in F register into file 2 at address specified by contents of N register at beginning of current microinstruction. Actual writing takes place during first part of instruction.
0111	AP	Alternate A field coding, A'.
1000	BP	Alternate B field coding, B'.
1001	DP	Alternate D field coding, D'.
1010	APDP	Alternate A and D field coding, A'D'.
1011	DPP	Alternate D field coding, D''.
1100	GATEI	Gate output of S1 to I register.
1101	HALT	If halt bit of SM register is 1, stop operation of MPP on completion of this microinstruction. When start signal is received, continue with next microinstruction specified by addressing mode and T field. If halt bit is 0, continue with microinstruction sequencing.
1110	RTJ	Transfer address of next sequential microinstruction pair to RTJ register. This is done regardless of actual addressing mode used in this instruction.
1111	CLRNP	Clear N register and page register.

3.3.1.4.8 C Field (Bits 24 through 31)

The C field is used to specify an additional special operation, an address for a jump, or a constant for setting the K or N register. Bit 24 in format 1 specifies the T field interpretation.

The codes for this field are listed in Table 3-17.

TABLE 3-17. C CODE ACTIONS

C. Code	Mnemonic	Operation
00xxxxx		xxxxx is a constant for use in driving bit generator or in any other commands using lower five bits of instruction for control.
0100000 0100001 0100010 0100011	WRCH/0 WRCH/1 WRCH/2* WRCH/3*	Write 8-bit character specified from output of S3 at memory address specified by output of AB. See WRITE command in S field for details of operation and restrictions. Character 0 is bits 0 through 7 (MSB's); character 1 is bits 8 through 15, etc. Remainder of word in memory is unchanged. Character is not repositioned in WRITE command.
0100100	RMW	READ MODIFY WRITE - Perform a read of information from main memory in same manner as READ instruction in S field. Memory system will perform a read cycle and lock up before performing write cycle. Memory must be forced to complete write cycle by issuing WRITE instruction, using same address, before it will respond to any other read operations.
0100101	WRHW0*	Write bits 0 through 15 (MSB's) from output of S3 at memory address specified by output of AB. Bits 16 through 31 in memory are unchanged. See WRITE command in S field for details of operation and restrictions.
0100111	WRHW1*	Write bits 16 through 31 (LSB's) from output of S3 at memory address specified by output of AB. Bits 0 through 15 in memory are unchanged. See WRITE command in S field for details of operation and restrictions.

TABLE 3-17. C CODE ACTIONS - Continued

C Code	Mnemonic	Operation
0101000	WRPB	WRITE PROTECT BIT - See protect system discussion (Appendix 1).
011xxxx		Open

*NOTE: Commands applicable to 32-bit processor only.

TABLE 3-17. C CODE ACTIONS (Cont.)

C Code	Mnemonic	Operation
1000101	INCK	Increment number contained in K register by one.
1001101	INCN	Increment number contained in N register by one.
1000100	DECK	Decrement number contained in K register by one.
1001100	DECN	Decrement number contained in N register by one.
1000000	CLRK	Clear K register.
1001000	CLRN	Clear N register.
101xxxx	SETF/j	xxxx is value of j, from 0 to 15. Set SM register flag j to 1.
110xxxx	CLRF/j	xxxx is value of j, from 0 to 15. Clear SM register flag j to 0.
1110000 or 1110001	RQLXN	Destination register (P, A, F, or X) and Q register are considered as one double-length register with Q register as lower order bits. Combined register is shifted left one bit position with complement of ALU sign bit entered into lowest bit position of Q register.
1110011	RQR1E	Shift combined destination and Q register right one bit, and enter 1 in sign position of destination register. This command is used in multiply iteration.
1110010	RQR0E	Shift combined destination and Q register right one bit, and enter 0 in sign position of destination register. This command is used in multiply iteration.
1110100	RL0E	Shift destination register left one bit, entering 0 in lowest bit position of the register. This operation can not be performed when Q is destination register.
1110101	RL1E	Shift destination register left one bit, entering 1 in lowest bit position. This operation can not be performed when Q is destination register.

TABLE 3-17. C CODE ACTIONS (Continued)

C CODE	MNEUMONIC	OPERATION
1110110	RR0E	Shift destination register right one bit, entering 0 in sign position of register. This operation can not be performed when Q is destination register.
1101111	RR1E	Shift destination register right one bit, entering 1 in sign position. This operation can not be performed when Q is destination register.

The following transform codes have a 1 in bit 19 to specify the S2' format.

See discussion of transforms and transform module

C CODE	MNEUMONIC	OPERATION
000xxxx	TMA/j	xxxx = j, with values from 0 to 15. Obtain next microinstruction pair from address specified by MA transform selector setting j.
001xxxx	TMAK/j	xxxx = j, from 0 to 15. Obtain next instruction pair from address specified by MA transform selector setting j. Also, set K register to value specified by K transform selector setting j.
0100xxx	GITMAK/j	Gate output of main memory to IXT register (on transform module) and perform TMAK/j operation. Note that j = xxx with values of 0 to 7. Restrictions: This command must be executed in the microinstruction following a READ command. This command is applicable only if transform module is configured with IXT register.
0101xxx	GITMAK/xt	Gate output of main memory to IXT Register (on transform module) and perform a transform of the upper 16-bits of MIR and select one of eight transforms of MA from the decoded and encoded macro instruction loaded into IXT. Restrictions: This command must be executed in the microinstruction following a READ command. This command is applicable only if configuration includes the required specialized transform module hardware.

TABLE 3-17. C CODE ACTIONS (Continued)

C CODE	MNEUMONIC	OPERATION
011xxxx	TK/j	xxxx = j, with values from 0 to 15. Set K register to value specified by K transform selector setting j.
100xxxx	TN/j	xxxx = j, from 0 to 15. Set N register to value specified by N transform selector setting j.

The following two codes are used if the optional program protect logic is included.

101xxx	SUB	Upper bounds - transfer output of S3 to upper bounds register in main memory interface.
110xxxx	SLB	Set lower bounds - transfer output of S3 to lower bounds register in main memory interface.

The following format 3 codings are for setting the K and N register; this format has the M field bits 0 and 1 set to 11, while bit 19 selects the register.

Value	K = value	When microinstruction bit 19 = 0, transfer C field value (bits 24 to 31) to K register and execute next sequential microinstruction pair.
Value	N = value	When microinstruction bit 19 = 1, transfer C field value (bits 24 to 31) to N register and execute next sequential microinstruction pair.

The following format 2 codings in the C field are used to perform a jump, specified if the M field (bits 0 and 1) is 10.

Number	Number	Number is address of next instruction pair. If page jump is required (bit 19 = 1), S field contains page setting instead of S field code.

3.3.1.5 Microinstruction Timing

3.3.1.5.1 Microinstruction Classes

The basic CPU microinstruction execution time is 168 nanoseconds. Some microinstructions have longer execution times to allow certain operations to be completed. The microinstructions have been grouped according to execution times as typ A, B, C, E, E and G as shown in Table 3-18.

TABLE 3-18. MICROINSTRUCTION EXECUTION TIME

MICROINSTRUCTION TYPE	EXECUTION TIME IN NANOSECONDS
A	168
B	224
C	280
D	336
E (Shift or Scale)	280 + 55n (where n is number of shifts)
F (Micro-memory read/write operand)	448
G (Read/Write main memory)	380 (typical)

The classification of microinstructions is shown in Figure 3-1. This figure provides execution time by types for all legal combinations of micro commands which extend the basic cycle time.

Exception to the execution times as listed in Figure 3-1 are:

FIGURE 3-1   MICROINSTRUCTION CLASSIFICATION

READ/WRITE MAIN MEMORY	READ/WRITE MICROMEMORY OPERAND	SHIFT OR SCALE	ONE'S COMPLEMENT ARITHMETIC	ADD OR SUBTRACT	TMA, TMAK, GITMAK	A', B', NU, ZL, COL, Z *L	INSTRUCTION TIME
0	0	0	0	0	0	0	A
0	0	0	0	0	0	1	B
0	0	0	0	0	1	0	C
0	0	0	0	0	1	1	C
0	0	0	0	1	0	0	B
0	0	0	0	1	0	1	C
0	0	0	0	1	1	0	C
0	0	0	0	1	1	1	C
0	0	0	1	0	0	0	A
0	0	0	1	0	0	1	B
0	0	0	1	0	1	0	C
0	0	0	1	0	1	1	C
0	0	0	1	1	0	0	C
0	0	0	1	1	0	1	D
0	0	0	1	1	1	0	C
0	0	0	1	1	1	1	D
0	0	1	0	0	0	0	E
0	0	1	0	0	1	0	E
0	1	0	0	0	0	0	F
1	0	0	X	X	X	X	G

X = 0 or 1 (Don't care condition)

. The combination of one's complement arithmetic with an ADD+ or ADD+T arithmetic operation is classified as a type B rather than a type C.

. The MMU and MML B' commands are included under Read/Write micromemory operand commands; therefore, they are a type F rathe than a type B.

. A type G microinstruction followed by a microinstruction with a GITMAK command (C field) has an additional execution time of 190 nanoseconds (typical); thus the total execution time becomes 570 nanoseconds.

3.3.1.5.2 Microinstruction and Memory Timing

Analysis of a microprogram for execution time starts by classifying each of the microinstructions as Type A,B,C,D, E, F or G. This is done by using the microinstruction classification table or by examining the assembler output listing.

Each microinstruction with a main memory read or write command has a 380 nanosecond (typical) execution time regardless of the type of cycle (assuming no DMA activity on main memory interface). However, type E and F microinstructions cannot contain a main memory reference command. An additional execution time of 190 nanoseconds (typical) is required on all read commands followed by a microinstruction containing a GITMAK command.

The total execution time required is then calculated from main memory command to main memory command. If the total time, starting with a microinstruction with a read or write command and including all microinstructions up to the following read or write command is less than the memory cycle time (600 nanoseconds), take 600 nanoseconds as the execution time for that sequence. If the total time is greater than 600 nanoseconds, the execution time for that sequence is the calculated time.

As an example the execution time for a microprogram sequence would be calculated as follows:

INSTRUCTION TYPE	TIME	SEQUENCED TIME (nanoseconds)
G (MEM REF)	380	828
A	168	
C	280	
G (MEM REF)	380	600 (548 < 600)
A	168	
G (MEM REF)	570	1074 (380 + 190=570)
C (GIT MAK)	280	
B	224	
G (MEM REF)	380	996
A	168	
F	448	
G (MEM REF)	280	
⋮	⋮	

3.3.1.6 Micromemory Operand References

The MP17 has the capability of transferring information between the micromemory and the registers of the processor. The transfer from the register to the micromemory is possible only if the micromemory is a read/write micromemory.

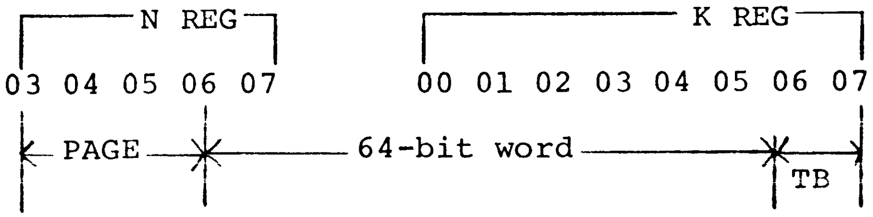
Micromemory is addressed as one to 16 pages of 256 words each, where each word is 64 bits and is divided into an upper 32-bit and lower 32-bit word. A 32-bit processor can reference 32-bit micromemory words by specifying page, address and upper or lower microinstruction. A 16-bit processor can reference only 16 bits at a time and an additional specification of the upper or lower 16 bits of each 32 bit micromemory half word is required to address all bits of micromemory.

Microinstructions for micromemory operand references may be an upper or lower microinstruction; the next microinstruction executed following the referencing microinstruction is always the upper microinstruction of the next sequential location. Microinstruction referencing a micromemory operand do not have to reside in the same page as the micromemory operand being referenced, when the combined N/K register is used to reference the micro-operands.

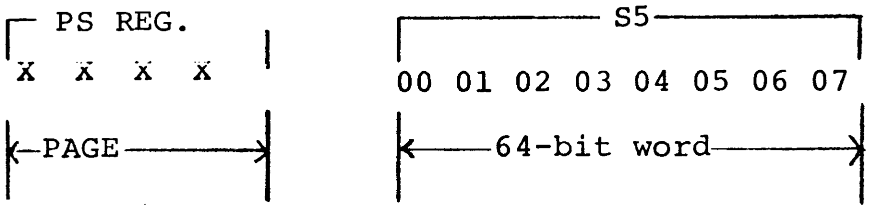
The micro commands for eading and operand from micromemory are coded in the B' field as MMU and MML; the micro commands for writing an operand into micromemory are coded in the D' field as MMU and MML.

Two addressing modes are available for operand references via status mode bit 113 (SM113) as follows:

SM113 = 0: The contents of the combined N/K registers is used to reference micro operands as indicated. The least significant bit of K (Bit 07) determines which 32-bit half word is referenced via T' code 010.



SMM113=1: An MA transform via S5 determines the 64-bit word. The 32-bit half word to be referenced must be selected by a T field test.



XXXX = Page in which referencing microinstruction resides.

3.3.2 Main Memory

Main memory is made up of 8K stacks of core memory. This 8K memory stack is configured on an 11'' x 14'' printed circuit board which requires two 0.625'' card spaces. The basic cycle for this memory is 600 nanoseconds for a read restore cycle and 700 nanoseconds for a read-modify-write cycle.

3.3.2.1 Main Memory Mini Configuration

* Mini Configuration {8 to 32K}. Maximum memory size 32K only, 1 CPU allowed, 1 bank, only one reference at a time.

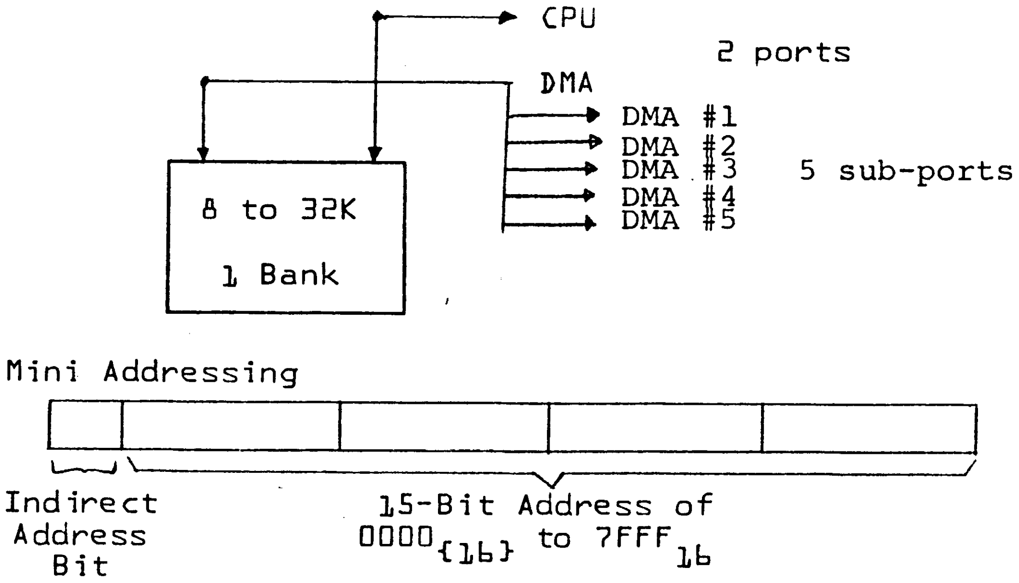


Figure 3

The mini memory configuration is a 1-bank 2-port memory with memory size options of 8, 16, 24, or 32K. 1 bank signifies only one reference may take place at one time. 2 ports signifies two independent data and control paths exist to the memory and either of the ports may request memory independent of any operation currently underway on the other port. The ports are CPU and DMA.

3.3.2.1.1 CPU Port

1. CPU data lines to Memory: These 16 multipurpose lines from the CPU to Memory Interface {CPU-D01 through D16} are used to transmit information as follows: {4TTL Loads Max.} {D01=LSB; D16=MSB}

   a. 15 Address Bits {CPU MAB 1 through MAB 15} The most significant bit is ignored, thus MAX addressing of 32K is determined by the 15 bit address. The 15 address bits are loaded into a buffer register by the LOAD ADDRESS control line. The desired CPU address must be loaded prior to initiating a CPU memory request.

   b. 15 Bit Bounds Address - The bounds registers are used to override the protect operation when a CPU address is less than the upper bound and greater than the lower bound. When this condition occurs, the memory protect bit line is forced false and the CPU protect line is forced true within the memory interface. Therefore, all instructions read are unprotected and all writes are protected. The bounds are as follows:

      1} 15 Bit Upper Bound Address - The 15 bit upper bound address is loaded into a buffer register by the load upper bound control line. It can be loaded any time except during a CPU memory cycle. The most significant bit, CPU D16, is not used.

      2} 15 Bit Lower Bound Address - The 15 bit lower bound address is loaded into a buffer register by the load lower bound control line. It can be loaded any time except during a CPU memory cycle. The most significant bit, CPU D16, is not used.

   c. 16 Normal Write Data Lines - {Nonsplit Cycle} These 16 write data lines used fornormal non-split cycle write operations must be stable within 100ns after the leading edge of CPU memory request. They must remain stable until CPU early data resume. This requirement applies to word, character, and protect bit writes. For character write operation, the CPU must place the desired character in the required character 0 or character 1 position.

d. 16 Data Lines - Split Cycle Write
These 16 write data lines used for the split cycle write operations must be stable within 10ns after the leading edge of write {split cycle initiate}. These lines must remain stable until 25ns after the leading edge of CPU early data resume following the leading edge of the write line. This requirement applies to both word and character split cycle writes. For character write operation, the CPU must place the desired character in the required character 0 or character 1 position.

2. 16 Data Lines from Memory {DFM01 through 16} {DFM 01 = LSB, DFM 16 = MSB}. These lines are, for all memory operations, common lines for both CPU and DMA. These 16 data lines from memory are stable a maximum of 122ns after the leading edge of CPU early data resume and a maximum of 17ns after the leading edge of CPU memory data resume.

a. For a read cycle, data shall remain stable a minimum of 340ns after the leading edge of CPU memory data resume.

b. For a normal write cycle, data shall remain stable a minimum of 50ns for word write and a minimum of 150ns for character write after the leading edge of CPU memory data resume.

c. For split cycle write cycles, data shall remain stable a minimum of 150ns after the leading edge of CPU write {slit cycle initiate} for both word and character split cycle write.

3. CPU Memory Request Line {$\overline{\text{CPU-REQ}}$}
This line requests a memory cycle. CPU-REQ goes true asynchronously to memory operation. DMA-CPU request resolution is 68ns minimum - 95ns normal - 129ns maximum to initiation of the memory cycle. CPU-REQ must go false within 50ns after the leading edge of CPU memory data resume.

4. CPU Early Data Resume line {$\overline{\text{CPU EARLY DS}}$}

a. For read, normal write and read portionof split cycle operations, this status line from the memory to CPU is used for an early data resume to optimize CPU performance. CPU early DS goes true minimum 95ns - normal 105ns - maximum 122ns before memory data is stable. It is a 50ns pulse.

b. For write portion of split cycle operation, CPU EARLY DS, goes true after the leading edge of write{split cycle initiate} {210ns normal}. This line is a split cycle write resume to the processor. CPU data to memory must remain stable for 25ns after leading edge of CPU EARLY DS before changing. It is a 50ns pulse.

5. CPU Memory Data Resume {$\overline{\text{CPU MDS}}$}
This status signal from memory for all operations signifies a data resume. Data from memory will be stable {at the memory interface} 17ns maximum after the leading edge of CPU-MDS. It is a 50ns pulse. The leading edge of CPU-MDS occurs 255ns minimum - 260ns normal - 265ns maximum after a memory stack cycle is initiated.

6. Character Mode Operation

a. Read Character - When character 0 and character 1 lines are both false, a read word operation is indicated. The CPU must select character 0 or character 1 as required for its character mode read. Character 0 = DFM 09 through 16; character 1 = DFM 01 through 08

b. Write Character - When character 0 is true, the memory will store CPU-D09 through D16 when no fault exists. When character 1 is true, the memory will store CPU-D01 through CPU-D08 when no fault exists. If both are true, word write is specified. If only one is true, character write is specified and the corresponding memroy data for the characterline false is restored in memory. For a split cycle write, when both character 0 and 1 are false, a word restore is performed.

1} Write Character 0 line {CPU Character 0}
When this line is true, CPU D09 through D16 is stored in memory during a write cycle. When this line is false, DFM09 through 16 is stored in memory during a write cycle.

For a normal write, this line must be stable within 100ns after a CPU memory request. It must remain stable until CPU early data resume.

For a split cycle write, this line must be stable with the leading edge of write and remain stable until 25ns after leading edge of CPU early data resume after split cycle initiate.

2} Write Character 1 Line {CPU Character 1}
When this line is true, CPU D01 through D08 is stored in memory during a write cycle. When this line is false, DFM 01 through 08 is stored during a write cycle. For a normal write, this line must be stable within 100ns after a CPU memory request. It must remain stable until CPU early data resume. For a split cycle write, this line must be stable with the leading edge of write and remain stable until 25ns after leading edge of CPU early data resume after split cycle initiate.

7. Write Line {CPU Write}
This control line from the CPU to memory provides the following control:

a. Read and Normal Write Cycles
When CPU write is true, the memory is directed to perform a write cycle. When false, the memory is directed to perform a read cycle. For read and normal write cycles, this line must be stable within 100ns after the leading edge of CPU memory request. It must remain stable until the leading edge of CPU early data resume.

b. Split Cycle Write
CPU write line must be false for the read portion of the split cycle. When this line goes true after a CPU memory data resume, the write portion of the split cycle is initiated {split cycle initiate}. It must then remain true until 25ns after the leading edge of CPU early data resume following the write initiate.

8. Split Cycle Line {$\overline{\text{CPU-SC}}$}
This control line from the CPU to memory requests a split cycle operation. CPU-SC must be stable 100ns after the leading edge of CPU memory request and remain stable until the leading edge of CPU early data resume. The write portion of the split

cycle is initiated with the write line. Characters 0 and 1 lines determine whether a word write, character write, or word restore is performed.

9. Load Upper Bound Line {$\overline{\text{LOAD UPPER BOUND}}$}
This control line from the CPU to memory directly loads CPU-D01 through D15 into a buffer register. This data {in the true state} is used as the upper bounds address used for bounds operation control. This line latches data with the positive going edge of load upper bound. A data setup time of 20ns {for CPU-D01 through D15} is required. The minimum required pulse width is 50ns. The upper bound can be loaded any time except during a CPU memory cycle. The most significant bit, CPU D16, is not used.

10. Load Lower Bound Line {$\overline{\text{LOAD LOWER BOUND}}$}
This control line from the CPU to memory directly loads CPU-D01 through D15 into a buffer register. This data {in the true state} is used as the upper bounds address used for bounds operation control. This line latches data with the positive going edge of load lower bound. A data setup time of 20ns {for CPU-D01 through D15} is required. The minimum required pulse width is 50ns. The lower bound can be loaded any time except during a CPU memory cycle. The most significant bit, CPU-D16, is not used.

11. Load CPU Address Line {LOAD CPU ADDRESS}
This control line from CPU to memory directly loads CPU-D01 through D15 into a buffer register. This data {in the true state} is used as CPU memory address. This line latches data with the positive going edge of load CPU address. A data setup time of 20ns is required. The minimum required pulse width is 50ns. The CPU memory address register can be loaded any time CPU memory request is false. The most significant bit, CPU-D16, is not used.

12. CPU Protect Bit Write Line {CPU Write Protect Bit}
This control line from CPU to memory directs the memory to write protectbit {memory bit 18} in memory. This is performed as a normal write cycle. When CPU-D10 is true, the memory clears the protect bit. When CPU-D10 is false, the memory sets the protect bit. The requirements for CPU-D10 are the same as for normal write data operation. CPU protect write must be stable within 100ns after the

leading edge of CPU memory request. CPU protect write must remain stable until the leading edge of CPU early data resume. A protect parity bit {memory bit 19} is stored with the protect bit {compliment of protect bit}. Memory word data and memory word parity are restored in memory.

NOTE: The CPU allows only protected instructions to perform a protect bit write.

13. Program Protect Line {$\overline{\text{CPU PROTECT}}$}
This control line from CPU to memory indicates CPU operation protect status. This line must be stable within 100ns after the leading edge of CPU memory request. It must remain stable until the leading edge of CPU early data resume.

a. For read {only} memory cycles the CPU protect line has no effect on operation.

b. For all write operations within bounds where the CPU memory address is less than the upper bound and greater than lower bound, the memory unconditionally stores the specified CPU word or character.

c. For normal and split cycle write memory cycles not in bounds area. When CPU protect is true, the memory will unconditionally store the specified CPU word or character. When CPU protect is false, the memory will:

1} When protect bit {bit 18} is true, set CPU protect fault and restores original memory data.

2} When protect bit {bit 18} is false, store the specified CPU word or character.

14. Memory Program Protect Bit Line {$\overline{\text{MEMORY PROTECT BIT}}$}
This status line from the memory to CPU {common status line with DMA} {timing is same as data from memory lines};

a. When CPU memory address is within bounds address, memory protect bit is forced false. This forces all instructions to be unprotected instructions. However, the memory protect bit is unchanged.

b. When CPU memory address is not within bounds address, memory protect bit line is equal to the memory protect bit {bit 18}. This provides program protect status to the CPU.

15. Program Protect Fault Line {CPU-PROTECT FAULT}
This status line from the memory to CPU indicates:

a. When CPU-protect fault is true, the CPU protect line was false, memory protect bit was true and CPU memory address was not within bounds address when a write operation was attempted. The line is pulsed {175ns minimum - 250ns maximum pulse width}. The leading edge of CPU-protect fault will occur:

1} Within 100ns after CPU memory data resume for protect bit write operations and normal write operations.

2} Within 300ns after CPU memory data resume or within 100ns after write initiate {whichever is longer after CPU memory data resume} for split cycle write operations.

NOTE: If CPU performs a protect write operation with program protect line false, a program protect fault will occur when memory protect bit is true {with or without a protect parity error}. However, the CPU does not allow a protect bit write by an unprotected instruction.

b. When CPU-protect fault is false, one or more of the following applies:

1} CPU protect false and memory protect bit false.

2} CPU protect true.

3} CPU memory address is within bounds addres.

4} Protect parity error and cpu protect false {memory protect bit is a don't care} {P.P.E. CPU protect = "0"}.

16. Word/Character Parity Line {$\overline{\text{MEMORY PARITY LINE}}$} This status line from memory to CPU provides parity {of the specified word or character data only} read from or stored in memory. Protect bit is not included in parity since it has its own parity bit. Therefore, any requirements for parity including protect bit must be generated by the CPU by use of memory parity line and memory protect bit lines. When memory parity line is true, the selected word or character has an even number of 1 bits; when false, an odd number of 1 bits. The line is defined as follows:

   a. All normal read cycles - memory parity line shall be stable within 45ns after the leading edge of CPU-memory data resume. It shall remain stable for a minimum of 340ns after the leading edge of CPU-MDS.

   b. Normal writecycles - word and character memory parity line is stable with the leading edge of CPU memory data resume. It remains stable for 225ns minimum after the leading edge of CPU memory resume.

   c. Split Cycle

      1} Read cycle - Memory parity line shall be stable within 45ns after the leading edge of CPU-MDS. It will remain stable until 10ns after the leading edge of CPU write {split cycle initiate} or CPU CHARO+1.

      2} Write cycle - Memory parity line shall be stable within 215ns after the leading edge of CPU write. It will remain stable while CPU write, CPU character 0, character 1 and CPU-DXX are stable or until 350ns after leading edge of CPU write, whichever occurs first.

17. CPU Memory Parity Error Line {$\overline{\text{CPU PARITY ERROR}}$} This status line from memory to CPU indicates a parity error occurred due to: {{WPE + PPE} . {READ-L+SC} . $\overline{\text{PW}}$. $\overline{\text{WRITE-L}}$} DATA WORD PARITY ERROR OR PROTECT PARITY ERROR during a normal read cycle or a read cycle portion of a split cycle. Parity error line cannot go true during a normal write cycle or

a write protect bit cycle. CPU parity error goes true within 150ns after leading edge of CPU-MDS. It is a pulse width minimum pulse width of 65ns; maximum pulse width 175ns.

a. A word parity errorindicates that one of the memory word bits is wrong or the parity bit is wrong. Word parity error has no effect on memory operation other than setting the parity error line.

b. A protect parity error indicates that either the protect bit or the protect parity bit is wrong. When a P.P.E. occurs and CPU protect is true during a write operation, actual protect bit {bit 18} is stored back in memory protect bit {DI18} and the complement of actual protect bit {bit} is stored in memory protect parity bit {DI19}. When a P.P.E. occurs and CPU protect is false, actual DO18 and DO19 are restored in DI18 and DI19 respectively.

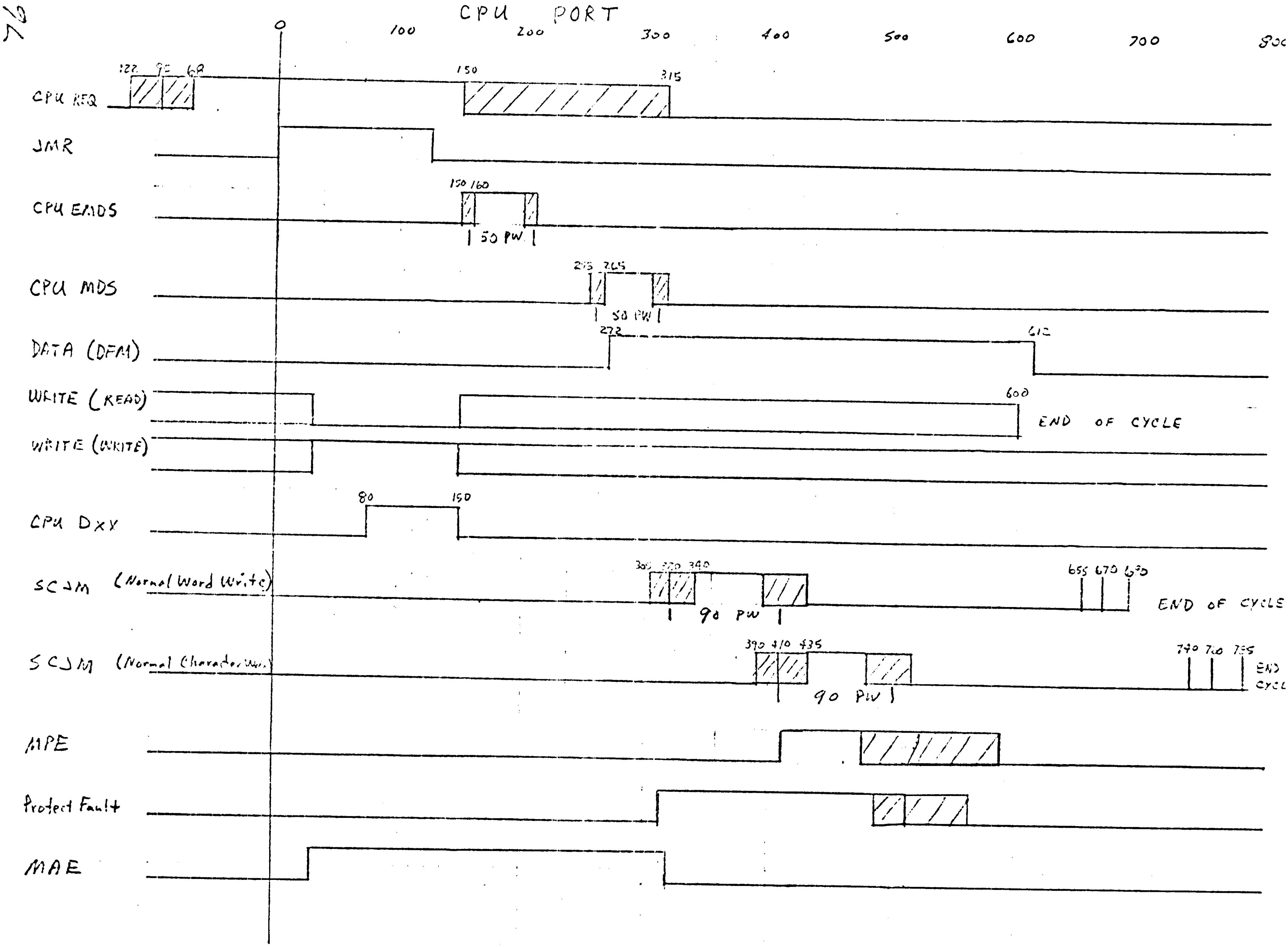
18. CPU Memory Addressing Error {$\overline{\text{CPU MAE}}$}
This status from memory to CPU indicates the CPU attempted to access a nonexistent memory stack, CPU MAE will be stable within 30ns after stack memory cycle is initiated. It will remain stable until leading edge of CPU-MDS which will force CPU MAE false.

When a CPU MAE occurs, the memory interface will force a read cycle from stack 1 {lowest address stack}. Address for stack will be CPU memory address bits 1 through 13. {Bit 1 is least significant address bit.} A normal read cycle is then completed.

19. Master Clear {$\overline{\text{CPU-MC}}$}

20. Existing Stack Control Lines {$\overline{\text{STACK 2,3,4}}$}
When a memory stack exists in a system, input pins must be grounded as follows:

PIN	SIGNAL	A14	A15	COMMENT
48	$\overline{\text{STACK 2}}$	1	0	Gnd when present
248	$\overline{\text{STACK 3}}$	0	1	Gnd when present
49	$\overline{\text{STACK 4}}$	1	1	Gnd when present
NA	$\overline{\text{STACK 1}}$	0	0	Always Present


CPU PORT
0
100
200
300
400
500
600
700
800
CPU REQ
122
95
68
150
315
JMR
CPU EMDS
150 160
50 PW
CPU MDS
215 265
50 PW
DATA (DFM)
272
612
WRITE (READ)
600
END OF CYCLE
WRITE (WRITE)
CPU DXX
80
150
SCJM (Normal Word Write)
305 320 340
90 PW
655 670 690
END OF CYCLE
SCJM (Normal Character Wri
390 410 435
90 PW
740 760 785
END
CYCL
MPE
Protect Fault
MAE

21. Stack Drive Inhibit {CPU SAVE}
When CPU SAVE is true, the memory interface inhibits all stack drive voltages.

3.3.2.1.2 DMA Port

1. DMA data lines to memory {DMA D01-D16} TTL Open Collector gated with DMA Req. Accept.

   a. 16 Normal Write data lines - {nonsplit cycle}
   These 16 write data lines used for normal nonsplit cycle write operations must be stable within 50ns after the leading edge of DMA request accept. They must remain stable until DMA request accept goes false. This requirement applies to word and character mode. For character write operation, the DMA must place the desired character in the required character 0 or character 1 position.

   b. 16 Data Lines - split cycle write
   These 16 write data lines used for the split cycle write operations must be stable within 10ns after the leading edge of write {split cycle initiate}. These lines must remain stable until the leading edge DMA request accept goes false. This requirement applies to both word and character split cycle writes. For character write operation, the DMA must place the desired character in the required character 0 or character 1 position.

2. DMA Memory Address Bits {DMA MAB1 through MAB15}
Open Collector TTL gated with DMA Req Accept. These 15 address lines must be stable within 50ns after leading edge of DMA request accept. They must remain stableuntil DMA MA goes true, or DMA MCTM goes true or DMA REQ ACCEPT goes false. Most significant address bit, DMA MAB16, is not used.

3. 16 Data Lines from Memory {DFM01 through 16}
{DFM01 = LSB, DFM16 = MSB}
These lines are, for all memory operations, common lines for both CPU and DMA. These 16 data lines from memory are stable a maximum of 17ns after the leading edge of DMA memory data resume for fast

DMA cycles and a minimum of 30ns before DMA MDS for slow DMA cycles.

a. For a read cycle, data shall remain stable a minimum of 340ns for fast DMA and 800ns for slow DMA after the leading edge of DMA memory data resume.

b. For a normal write cycle for fast DMA only, data shall remain stable a minimum of 50ns for word write and a minimum of 150ns for character write after the leading edge of DMA memory data resume.

c. For split cycle write cycles for fast DMA only, data shall remain stable a minimum of 150ns after the leading edge of CPU write {split cycle initiate} for both word and character split cycle write.

4. DMA Memory Request Line
{DMA 1- MR $\overline{\text{DMA2- MR}}$ $\overline{\text{DMA3- MR}}$ $\overline{\text{DMA4- MR}}$ $\overline{\text{DMA5- MR}}$
This line requests a memory cycle. DMA-MR goes true asynchronously to memory operation. DMA-CPU request resolution is 78ns minimum - 120ns normal - 165ns maximum to initiation of the memory cycle. Fast DMA-MR must go false within 50ns and slow DMA MR must go false within 300ns after the leading edge of DMA memory data resume unless continuous requests desired, in which case data address and write lines must be stable 700ns for slow DMA and 350ns for fast DMA after leading edge of DMA-MDS. DMA1 has highest priority with DMA5 having lowest priority. Any DMA has priority over CPU.

5. DMA Request Accept
{DMA1-RA, $\overline{\text{DMA2-RA}}$, $\overline{\text{DMA3-RA}}$, $\overline{\text{DMA4-RA}}$, $\overline{\text{DMA5-RA}}$}
When the DMA-RA lines goe true, normal write data, address and writeline must be stable within 50ns. Where used, character 0 and 1, DMA protect, DMA split cycle, slow DMA, must be stable within 100ns. These lines must remain stableuntil DMA req accept goes false.

6. Slow DMA Line {SLOW-DMA} TTL Open Collector
When Slow DMA line is true, all timing requirements for data lines and DMA data strobe defined by standard NCR 605 DMA are met. Where DMAs are designed to use the performance of the 600ns stack, gating false the slow DMA line will optimize memory performance. A time-out circuit is used to match slow DMA requirements and 600ns stack operation.

This line has pull up resistor and therefore requires no wiring when slow DMA operation is desired.

7. Priority Line {$\overline{\text{PRIORITY}}$}
Pull up resistor on-line, therefore, no connection required. When priority is true, the DMA device determined by ENAB1-5 {handwired to the desired DMA1 through 5 -- only one available priority DMA} has absolute priority. All other memory requests -- both DMA and CPU -- are blocked out until priority goes false. The current memory cycle is completed normally at which time the memory accepts only priority DMA requests. It must be noted that split cycles in process when priority goes true potentially could cause a problem if the write initiate is delayed or excessively long; in such cases, system software restrictions may be required.

Use of this line can give a guaranteed response time to request when required and when split cycle operation is allowed for. This is compatible with 1700DSA.

8. Priority Enable {$\overline{\text{ENAB1 through ENAB5}}$}
When these lines are handwired to a specified DMA {as defined below}, the specified DMA has absolute priority over all requests which are subsequently locked out.

ENAB1	ENAB2	ENAB3	ENAB4	ENAB5	PRIORITY DMA
0	1	1	1	1	DMA 1
1	0	1	1	1	DMA 2
1	1	0	1	1	DMA 3
1	1	1	0	1	DMA 4
1	1	1	1	0	DMA 5

9. DMA-MCTM {Memory Cycle Time}
MCTM is an interface timing signal generated in the memory. It occurs 120ns after the start of the cycle and is 50ns wide. It is normally high going to ground when on. It is driven by a 74H08 TTL Gate. One load maximum per DMA.

10. DMA-LTP2 {Timing Pulse}
LTP2 is another timing pulse generated by the memory. It is referenced to the end of the cycle occurring about 100ns before the end. It is 68ns wide. It is a negative going pulse driven by a 74H08 TTL gate. One load maximum per DMA.

11. DMA MA {Memory Available}
MA is a signal generated by the memory indicating the memory is in an active state {active low}. MA goes low 115ns after the start of the cycle and remains low until the end of the cycle. It is driven by a 74H08 TTL gate. One load maximum per DMA.

12. DMA Memory Data Resume {DMA MDS}
This status signal from memory for all operations signifies a data resume. Data from memory will be stable {at thememory interface} 17ns maximum after the leading edge of DMA-MDS for fast DMA, and a minimum of 30ns before leading edge of DMA MDS for slow DMA. It is a 50ns pulse. The leading edge of DMA MDS occurs 255ns minimum - 260ns normal - 265ns maximum after a memory stack cycle is initiated for fast DMA and 300ns minimum - 315 normal - 331 maximum after initiation of memory stack cycle for slow DMA. 2TTL loads maximum each DMA.

13. Character Mode Operation

Pull up resistors on character 0, 1 therefore no connection required and word mode always selected.

a. Read character - When character 0 and character 1 lines are undefined and DMA write is false, a read operation is indicated. The DMA must select character 0 or character 1 as required for its character mode read. Character 0 = DFM 09 through 16; character 1 = DFM 01 through 08.

b. Write Character - When character 0 is true, the memory will store CPU-D09 through D16 when no fault exists. When character 1 is true, the memory will store CPU-D01 through DMA-D08 when no fault exists. If both are true, word write is specified. If only one is true, character write is specified and the corresponding memory data for the character line false is restored in memory. For a split cycle write, when both character 0 and 1 are false, a word restore is performed.

1) Write character 0 line {DMA character 0} TTL Open Collector
When this line is true, DMA D09 through D16 is stored in memory during a write cycle. When this line is false, DFM 09 through 16 is stored in memory during a write cycle.

For a normal write, this line must be stable within 100ns after a DMA memory request accept. It must remain stable until DMA request accept goes false. For a split cycle write, this line must be stable with the leading edge of write and remain stable until after DMA request accept goes false.

2) Write character 1 line {DMA character 1} {TTL open collector}
When this line is true, DMA D01 through D08 is stored in memory during a write cycle. When this line is false, DFM 01 through 08 is stored in memory during a write cycle. For a normal write, this line must be stable within 100ns after a DMA memory request accept. It must remain stable until DMA request accept goes false. For a split cycle write, this line must be stable with the leading edge of write and remain stable until after DMA request accept goes false.

14. Write Line {DMA Write} {TTL Open Collector}
This control line from the DMA to memory provides the following control:

a. Read and Normal Write Cycles - When DMA write is true, the memory is directed to perform a write cycle. When false, the memory is directed to perform a read cycle. For read and normal write cycles, this line must be stable within 100ns after the leading edge of DMA request accept. It must remain stable until and be cleared with DMA request accept going false.

b. Split Cycle Write - DMA write line must be false for the read portion of the split cycle. When this line goes true after DMA request accept, the write portion of the split cycle is initiated {split cycle initiate}. It must then remain true until DMA request accept goes false.

15. Split Cycle Line {$\overline{\text{DMA-SC}}$}
TTL Open Collector's pull-up resistor allows no connection. This control line from the DMA to memory requests a split cycle operation. DMA-SC must be stable 100ns after the leading edge of DMA request accept and remain stable until DMA request accept goes false. The write portion of the split cycle is initiated with the write line. Character 0 and 1 lines determine whether a word write, character write or word restore is performed.

16. Program Protect Line {$\overline{\text{DMA PROTECT}}$} TTL Open Collector
This control line from DMA to memory indicates DMA operation protect status. This line must be stable within 100ns after the leading edge of DMA request accept. It must remain stable until DMA request accept goes false.

a. For read {only} memory cycles the DMA protect line has no effect on operation.

b. For normal and split cycle write memory cycles when protect is true, the memory will unconditionally store the specified DMA word or character. When DMA protect is false, the memory will:

1} When protect bit {bit 18} is true, set DMA protect fault and restores original memory data.

2} When protect bit {bit 18} is false, store the specified DMA word or character.

When no connection is made to this line, all operations are unprotected {pull up resistor}.

17. Memory Program Protect Bit Line {MEMORY PROTECT BIT} This status line from the memory to DMA {common status line with CPU}: {Timing is same as data from memory lines} memory protect bit line is equal to the memory protect bit {bit 18}. This provides program protect status to the DMA.

18. Program Protect Faule Line {DMA-PROTECT FAULT};
2 TTL Loads Maximum
This status line from the memory to CPU indicates:

   a. When DMA-protect fault is true, the DMA protect line was false and memory protect bit was true when a write operation was attempted. The line is pulsed 175ns minimum - 250ns maximum pulse width. The leading edge of DMA-protect fault will occur:

      1} Within 100ns after DMA memory data resume for fast DMA and within 35ns after DMA-MDS for slow DMA for normal write operations.

      2} Within 300ns after DMA memory data resume or within 100ns after write initiate {whichever is longer after DMA memory data resume} for split cycle write operations.

   b. When DMA-protect fault is false, one or more of the following applies:

      1} DMA protect false and memory protect bit false.

      2} DMA protect true.

      3} {Protect parity error and DMA protect} false {Memory protect bit is a don't care}{P.P.E. . CPU PROTECT = "0"}

19. Word/Character Parity Line {MEMORY PARITY LINE} This status line from memory to DMA {common with CPU} provides parity of the specified word or character data only read from or stored in memory. Protect bit is not included in parity since it has its own parity bit. Therefore any requirements for parity including protect bit, i.e., 1700 DSA, must be generated by the DMA by use of memory parity line and memory protect bit lines. When memory parity

line is true, the selected word or character has an even number of 1 bits; when false, an odd number of 1 bits. The line is defined as follows:

a. All normal read cycles - memory parity line shall be stable

1) within 45ns after the leading edge for fast DMA, and

2) with for slow DMA, the leading edge of DMA memory data resume. It shall remain stable for a minimum of 340ns for fast DMA and 300ns for slow DMA after the leading edge of DMA MDS.

b. Normal write cycles - word and character memory parity line is stable with the leading edge of DMA memory data resume. It remains stable for 1) 225ns for fast DMA minimum, and 2) 180ns for slow DMA, after the leading edge of DMA memory data resume. It remains stable for 1) 225ns for fast DMA minimum, and 2) 180ns for slow DMA after the leading edge of DMA memory resume.

c. Split Cycle

1) Read cycle - memory parity line shall be stable

a) within 45ns after for fast DMA, and

b) with for slow DMA, the leading edge of DMA-MDS. It will remain stable until 10ns after the leading edge of DMA write {split cycle initiate}.

2) Write cycle - memory parity line shall be stable within 215ns after the leading edge of DMA write. It will remain stable while DMA write, CPU character 0, character 1 and DMA-DXX are stable or until 300ns after leading edge of DMA write whichever occurs first.

20. DMA Memory Parity Error Line {DMA PARITY ERROR} This status line from memory to DMA indicates a parity error occurred due to:

{{WPE + PPE}.{READ-L + SC}.$\overline{\text{PW}}$.$\overline{\text{WRITE-L}}$} DATA WORD PARITY ERROR OR PROTECT PARITY ERROR during a normal read cycle or a read cycle portion of a split cycle. Parity errors line cannot go true during a normal write cycle. DMA parity error goes true 1} within 100ns after for slow DMA, 2} within 150ns after for fast DMA leading edge of DMA-MDS. It is a pulse width minimum pulse width of 65ns; maximum pulse width 175ns.

a. A word parity error indicates that one of the memory word bits is wrong or the parity bit is wrong. Word parity error has no effect on memory operation other than setting the parity error line.

b. A protect parity error indicates that either the protect bit or the protect parity bit is wrong. When a P.P.E. occurs and DMA protect is true during a write operation, actual protect bit {bit 18} is stored back in memory protect bit {DI18} and the complement of actual protect bit is stored in memory protect parity bit {DI19}. When a P.P.E. occurs and DMA protect is false, actual DO18 and DO19 are restored in DI18 and DI19 respectively.

21. DMA Memory Addressing Error {$\overline{\text{DMA MAE}}$} This status from memory to DMA indicates the DMA attempted to access a nonexistent memory stack. DMA MAE will be stable within 30ns after stack memory cycle is initiated. It will remain stable until leading edge of DMA-MDS which will force DMA MAE false.

When a DMA MAE occurs, the memory interface will force a read cycle from stack 1 {lowest address stack}. Address for stack will be DMA memory address bits 1 through 13. {Bit 1 is least significant address bit.} A normal read cycle is then completed.

22. DMA Master Clear
Received from CPU.

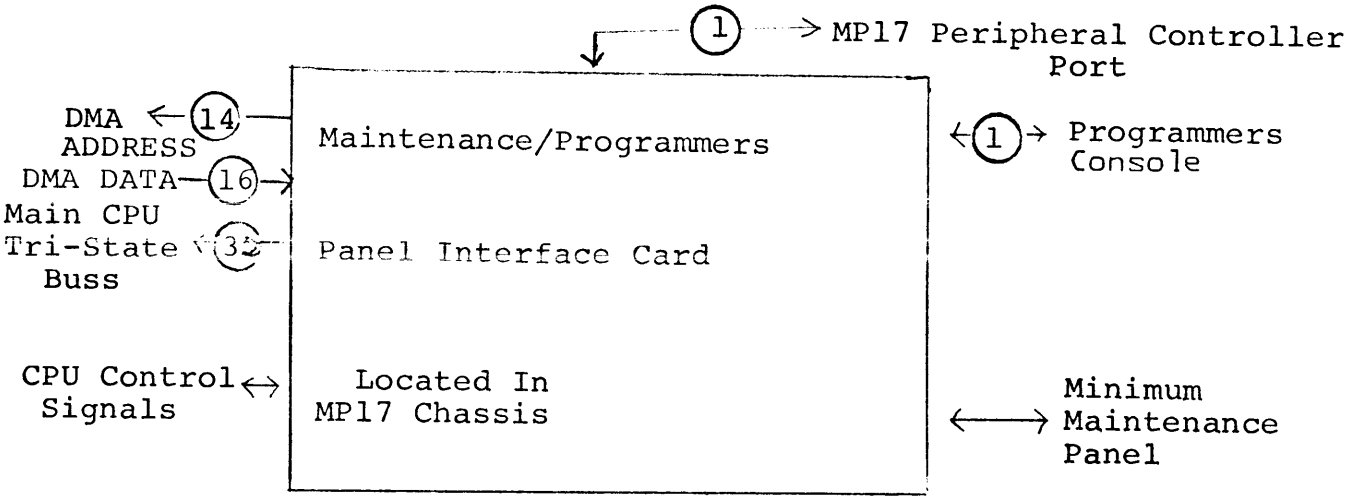
3.3.2.3 Main Memory 32-Bit Controller Configuration

The 32-bit controller has:

A} 8K of 16 bit {data memory
B} 16K of 16 bit {data} memory
C} 8K of 32 bit {data memory
D} 16K of 32 bit {data} memory

3.3.3 Maintenance/Programmers Panel Interface:

All configurations of the MP17 contain the maintenance/programmers panel interface card. Also standard will be a minimum maintenance panel which interfaces through the panel interface to the CPU. The programmers console is optional on all MP17 configurations, and interfaces through the same card as the minimum maintenance panel. The basic signal flow of the panel interface is as follows.



The interface card interfaces to the MP17 through DMA, main CPU buss, and internal CPU control signals. The interface to the programmers console is RS232 compatable. No "Special" programmers console is included. The programmers console will consist of a teletype, a display or any device which has two-way simultaneous serial (full duplex) ASCII characteristics. The programmers console is capable of being remoted, but no modem or modem control signals are provided. The interface to the minimum maintenance panel is not RS232 compatable.

The interface provides for all common computer control panel functions. These functions can be performed equally well through the maintenance panel or the programmers console. The interface also provides a data path for MP17 peripheral controllers to transmit ASCII characters to the panel interface. This feature actually allows these peripheral controllers to perform most operations that can be performed from the maintenance panel or programmers console. In practice this feature will generally be used to load main memory or micro-memory from devices like the card reader.

Basic to the operation of the panel interface is the Function Control Register (FCR). This is a 32-bit register which is accessible to the CPU in a manner similar to the way switches on a conventional panel are accessible to the CPU. The function control register bit assignments are listed in Table 9.

BIT		DIGIT	BIT DEFINITION
0 0	0 0		Overflow
0 1	0 1	0	Protected Instruction
0 2	0 2		Protect Fault
0 3	0 3		Parity Error
0 4	0 4		Interrupt System Active
0 5	0 5	1	Auto-Restart Enabled
0 6	0 6		Micro Running
0 7	0 7		Macro Running
0 8	0 8		
0 9	0 9	2	
1 0	0 A		
1 1	0 B		
1 2	0 C		
1 3	0 D	3	Multi-Level Ind. Add. Mode
1 4	0 E		Execute Micro via DMA
1 5	0 F		Suppress Console Transmit
1 6	1 0		
1 7	1 1	4	
1 8	1 2		BP Int. (BP Stop if Clr)
1 9	1 3		Micro BP, Step, Go, Stop (Macro if Clr)
2 0	1 4		Step
2 1	1 5	5	Selective Stop
2 2	1 6		Selective Skip
2 3	1 7		$\overline{\text{Protect Switch}}$
2 4	1 8		
2 5	1 9	6	DISPLAY 1
2 6	1 A		
2 7	1 B		
2 8	1 C		
2 9	1 D	7	DISPLAY 2
3 0	1 E		
3 1	1 F		

Status Only

0	0	BP Off
0	1	Int. Ref
1	0	Store Op.
1	1	All Ref.

FUNCTION CONTROL REGISTER (FCR)

TABLE 9

CODE		DISPLAY 1	DISPLAY 2
0	0 0 0 0	FCR	F2
1	0 0 0 1	P	N,RTJ
2	0 0 1 0	I	
3	0 0 1 1		X
4	0 1 0 0	A	Q
5	0 1 0 1	MIR	F
6	0 1 1 0	BP	F1
7	0 1 1 1	P-MA	MEM
8	1 0 0 0	SM1	
9	1 0 0 1	M1	K
A	1 0 1 0	SM2	
B	1 0 1 1	M2	
C	1 1 0 0		MM
D	1 1 0 1	A*	
E	1 1 1 0	X*	
F	1 1 1 1	Q*	

DISPLAY CODE DEFINITIONS

3.3.3.1 Functional Operation

The interface is capable of accepting eight different control characters. The eight control characters are H, I, J, K, L, Cr, Lf and ?. Cr denotes Carriage Return, Lf denotes Line Feed and ? denotes Question Mark.

H through L identify the type of data or operation entered or returned. The combination of Cr Lf terminates all entries except Master Clear.

A Question Mark accompanied by correct parity will generate a Master Clear to the computer, memory and peripherals. There is no response to this entry.

A normal entry consists of one control character H through L; 0, 2, 4 or 8 hexadecimal digits 0 through F; and Cr Lf in that order. A normal response consists of one control character that identifies the data that follows; 4 or 8 hexadecimal digits and Cr Lf in that order. If a transmission or operator error occurred on the entry, the control character is replaced by an asterisk (*) and the Function Control Register is unconditionally displayed. All entries except ? cause a response unless bit 15 of FCR is set.

3.3.3.1.2 H Control Function.

This function is used to clear a specific bit in the FCR.

Example: H14 Cr Lf

This would clear bit 14_{16} in FCR (Step mode), and the response would be a display of the updated FCR.

3.3.3.1.3 I Control Function

This function is identical to H except it sets a bit in FCR. H and I are also used for Stop/Run Control (See below).

3.3.3.1.4 J Control Function

The J Control Function is used to replace the contents of the function control register in a digit mode. While it may be used to change the value of any FCR digit, it generally is used to change digits 6 and 7, display #1 and display #2. The value of display #1 and display #2 specifies which MP17 parameter is displayed on display requests, or entered on enter requests. J functions always consist of J followed by two hexidecimal digits and Cr Lf. The first hexidecimal digit specifies the FCR digit 2 thru 7 and the second hexidecimal digi specifies the value the digit is to assume, 0 through F.

Example: J64 Cr Lf

This would set FCR digit 6 to 4 (select the "A" register), and the response would be a display of the updated FCR.

The J code is also used to alternately display the upper and lower 16 bits of a 32-bit register on the maintenance panel.

Example: J Cr Lf

This would cause the other 16 bits to be displayed and compliment the U/L indicator on the maintenance panel.

3.3. 3.1.5 K Control Function

The K control function is used to display or enter data into the parameter specified by Display #1. The K functions use two formats. The first format is a request to display the parameter specified by Display #1. It is as follows:

```
K Cr Lf
```

The second format is an enter data request. The data is entered into the parameter specified by Display #1. It consists of K followed by 4 or 8 hexidecimal digits, terminated by Cr Lf. The hexadecimal digits are the data to be entered. Several examples follow:

1) To display the "P" register the following is necessary:

```
J61 Cr Lf Set Display #1 to "P" register (1).
K Cr Lf   Display parameter selected in Display #1.
```

2) To enter $14FE_{16}$ into the breakpoint register the following is necessary:

```
J66 Cr Lf      Set display #1 to BP register (6)
K14FE Cr Lf    Enter into parameter selected in
                Display #1.
```

3.3.3.1.6 L Function

The L function is operationally the same as the K function except it is associated with Display #2.

Note: When main memory is displayed or entered, the register selected in Display #1 is the main memory address. The Display #1 selection must be the P or A register. This register is incremented by 1 after the display.

When micro-memory is displayed or entered, the K register is the least significant 8 bits of the address, and the N register provides the remaining bits. The K register is incremented by 1 after the display.

3.3.3.1.7 Stop/Go Control

The following entry will cause a go:

H Cr Lf

This is a micro go if bit 19 of FCR is set. It is both a micro and macro go if bit 19 of FCR is clear.

The following entry will cause a stop:

I Cr Lf

This is a micro stop if bit 19 of FCR is set. It is a macro stop if bit 19 of FCR is Clear.

The response to a start or stop entry is a display of FCR.

If a macro stop occurs for any reason, a display of FCR will result. The same is true if a main memory parity error occurs.

3.3.3.1.8 Master Clear

A master clear can be generated in several ways:

1) A question mark from remote console.
2) The MC button on the maintenance panel.
3) The autoload button.
4) A signal from a peripheral controller.
5) A power on Master Clear.

3.3.3.1.9 Auto-Restart

If the auto-restart switch is enabled, or if the maintenance panel is not present, a macro Go will be generated when power is applied to the computer. The Go signal is generated after all voltages have reached normal level.

3.3.3.1.10 Breakpoint

There are two types of breakpoint - micro and macro. If bit 19 of FCR is set, micro BP is selected. If bit 19 is clear, macro BP is selected.

1) Macro BP. Bits 16 and 17 of FCR are used to select three types of macro BP as follows:

Bit 17	Bit 16	
0	0	Breakpoint not selected
0	1	Instruction reference BP
1	0	Store operand BP
1	1	All references BP

A macro BP occurs if the BP register is equal to the main memory address and the select conditions are met.

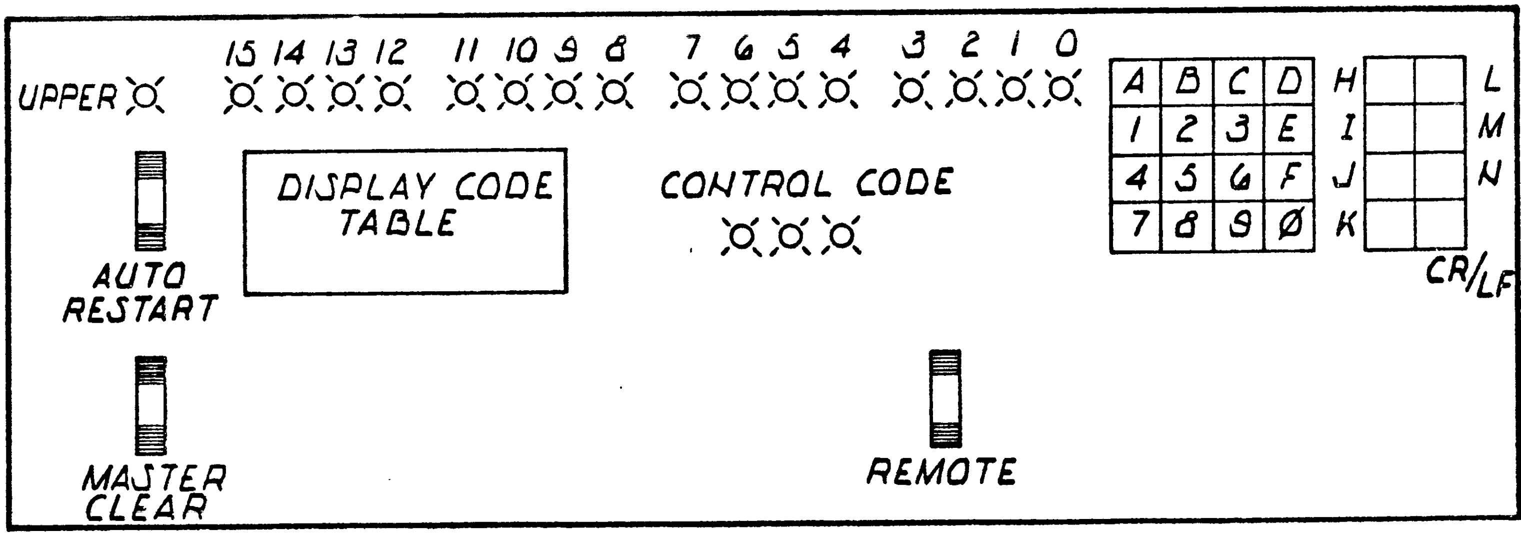
If bit 18 of FCR is set, an interrupt occurs when the breakpoint conditions are met rather than a stop.

2) Micro BP. For micro BP, P-MA is compared to the lower 12 bits of the breakpoint register. In addition, the upper/lower selection (32-bit select) is compared to the thirteenth bit of the BP register. If all bits are equal, and the combination of FCR bits 16 and 17 is not zero, then a micro stop occurs.

3.3.3.2 Minimum Maintenance Panel

The minimum maintenance panel consists of a 16" x 4½" PC board that is mounted directly above the main chassis. It connects the Panel Interface Module through a flexible cable. Power and ground connections are also provided in the cable.

The following functional drawing illustrates the features available.



The following describes the above drawing.

DATA DISPLAY - This sixteen bit panel provides display of data. The data displayed is determined by functions previously described. Display indicators are LED's.

UPPER - This LED indicates whether the display is displaying the upper or lower sixteen bits. When lit it indicates upper.

CONTROL CODE - This 3-bit LED display indicates the last control character entered. It is interpreted as a binary number. The equivalence table is as follows:

0	H	4	L
1	I	5	UNDEFINED
2	J	6	UNDEFINED
3	K	7	ERROR

AUTO RESTART - This two position switch selects the auto restart capability.

MASTER CLEAR - This momentary contact switch causes a Master Clear to the CPU, Memory and peripheral controllers.

REMOTE - This two position switch is used to enable the panel or the remote programmers console. In the local position the panel is enabled, in the remote position the programmers console is enabled. If the panel is physically removed, remote is selected.

FUNCTION DEFINITION TABLE - This table will consist of operator assistance information such as "FCR" definition control character definition, etc.

DATA ENTRY - These sixteen momentary pushbuttons are used to enter hexidecimal data.

CONTROL CHARACTERS - These six momentary pushbuttons are used for entering control characters.

NOTE: When using the minimum maintenance panel it is only necessary to depress one switch to generate Cr Lf.

3.4 Physical

3.4.1 Dimensions

Logic Chassis - 18''H x 17.5''W x 14''D

Power Supply8-3/4''H x 17.5''W x 16''D

Weight - to be determined

Cooling - Forced Air

3.5 Reliability

3.5.1 Mean Time Between Failure - MTBF

The Mean Time Between Failure for each module is unknown at this time. The design goal is 5000 hours.

3.5.2 Fail Safe Features

There shall be a circuit breaker which breaks the incoming power lines. No operating fail safe features exist in the MP.

3.5.3 Failure Detection

The MP has both memory parity and protect failure detection features.

3.6 Maintainability

3.6.1 Operating Adjustments

No operating adjustments shall be required for any modules described in this engineering specification.

3.6.2 Maintenance Philosophy

Maintenance philosophy for the MP shall be to use diagnostics requiring minimal operator intervention through the operators panel to isolate to the logic module failing for 80% of all failures. The failing module shall be replaced in the field and shall be taken to a depot for repair.

3.6.3 Mean Time to Repair MTTR

Based on the maintenance philosophy the MTTR shall be 30 minutes after the required module is available on site.

3.6.4 Preventive Maintenance

Preventive maintenance shall be scheduled in conjunction with emergency maintenance to ensure satisfactory operating performance. This maintenance shall include but not be limited to checking air filters, running voltage margins, setting normal voltages, and diagnostics.

3.6.5 Interchangeability

Any logic module shall be capable of operating in any slot where it satisfies the defined card placement requirements.

3.6.6 Technical Manuals

A separate manual shall be required for each module. The manuals shall define both hardware and software characteristics of the applicable module. Systems manuals are beyond the scope of this specification and require the end user to generate such technical manuals.

3.6.7 Diagnostic Requirements

Diagnostics are required to isolate 80% of all logic failures to the module level. Capability shall be provided to inform the operator which module is failing.

3.6.8 Voltage Margins

All modules individually and in a system configuration shall operate with a plus or minus 5% voltage variation from nominal voltages.

3.7 Environmental

3.7.1 Nonoperating Environmental

All modules individuall and in a system configuration shall withstand a temperature range from -30°F to +150°F {-35°C to +65°C} and a maximum thermal shock not to exceed 20°F/hour with no permanent effect on operating characteristics. All requirements covering altitude, temperature, humidity, shock, and vibration stated in CDC Standard 1:30:011, section 3.12.1 for Nonoperating Environmental Requirements, shall be met.

3.7.2 Operating Environmental

All modules individually and in a system configuration shall operate at a temperature range from 40°F to 120°F, a maximum thermal shock of 0.2°F per minute and a relative humidity range from 10 to 90 percent. All requirements stated in CDC Standard 1:30:011, section 3.1.2.2 for Operating Environmental Requirements, shall be met.

3.8 Refurbishment

To be determined.

3.9 Design Construction

Appropriate CDC, National, and International standards shall be followed where possible and conformance shall be stated where applicable.

3.9.1 Communications Standards

All communication channels shall conform to and meet the requirements of EIA RS-232-C standard: Interface Between Data Terminal Equipment and Data Communications Equipment.

3.9.2 Electromagnetic Compatibility Requirements

All modules in a system configuration only shall meet all Electromagnetic Compatibility {EMC} requirements defined in CDC Standard 1:30:022.

3.9.3 Computer Equipment Standards

All modules individually and in a system configuration shall conform to all requirements of CDC Standard 1:30:011 Computer and Peripheral Equipment Design Requirements.

3.9.4 Safety Standards

The MP17 shall conform to all requirements of CDC Standard 1:30:003 Equipment Safety Requirements.

3.9.5 Human Engineering

The design of the cabinet and operators panel shall comply with appropriate human factors engineering. The design shall be user-orientated and intended for use by a typical operator.

4.0 Quality Assurance Provisions (This section is for production units. It is assumed that the design verification and production verification tests are complete.)

4.1 Destructive Test

No destructive type testing shall be performed on shippable items.

4.2 Tests and Checks

4.2.1 Performance Test

a. Speeds
b. Media handled
c. Capabilities
d. Error indicators
e. Addressability
f. Micro-instruction repertoire
g. Macro-instruction repertoire

4.2.2 Reliability Test

a. Fail safe features
b. Marginal checking

4.2.3 Characteristic Measurement Test

a. Power on auto restart verification
b. Parity verification
c. Program protect verification
d. Interrupt verification
e. Real time clock verification . (Short term and long term accuracy)

4.2.4 Maintainability

a. Interchangeability test
b. Diagnostic test (software and hardware)
c. Technical manuals

4.2.5 Environmental Test (Periodic QA Audit Test performed per NCR System Environmental Standard)

a. Speeds
b. Fail safe features
c. Marginal checking
d. Diagnostics (race conditions and settling times)
e. Real time check (short term and long term accuracy)

4.3 Workmanship Verification

NCR Workmanship Standard

4.4 Special Test

.4 Environmental

Periodic QA audit conducted at CDC SCDD to verify conformance to Engineering Electrical Design Standard CDC-STD 1.30.000 and Mechanical Design Standard CDC-STD 1.20.000.

Workmanship

Periodic QA audit conducted at CDC SCDD to verify conformance to Quality Specification CDC-Spec 10120300 Volume II.

5. Preparation for Delivery

NCR Standard

6.0 NOTES

6.1 Equipment Data Sheet

To be defined.

7.0 APPENDIX I -- MP17 CONFIGURATION

7.1 PROGRAMMING

This section contains a description of the concepts which must be understood to properly program the 1700 computer.

7.1.1 Instruction Formats -- There are three types of instructions in the 1700 computer. The types are Storage Reference, Register Reference, and Skip. Five instruction formats are required; one for each type of instruction plus one for the Shift instructions and one for the Interregister instruction, which are subgroups of the Register Reference instructions.

Hexadecimal notation is used in this computer for ease in expressing the 4-bit groups which occur in the instruction format. {Hexadecimal is to the base 16.} The additional characters required are A, B, C, D, C, and F. The relationships between binary, decimal, and hexadecimal are shown in Table 7-1.

TABLE 7-1.

DECIMAL	HEXADECIMAL	BINARY
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

7.1.1.1 Storage Reference -- This class of instructions is that which reference storage for operands. Bits 15-12 specify the specific instructions, bits 11-8 specify the addressing mode, and bits 7-0 is a value used for relative addressing. Refer to section 7.2.1 for a more explicit explanation.

7.1.1.2 Register Reference -- This class of instructions operates on the computer registers or control FFs. A Register Reference instruction is identified when bits 15-12 are all zero. Bits 11-8 identify the specific Register Reference instruction. Bits 7-0 may be a constant, an instruction modifier, or program address modifier, depending on the specific instruction. Refer to section 7.2.2 for a more explicit explanation.

7.1.1.3 Shift -- This instruction is a special case of the Register Reference instruction. It is identified when bits 15-12 are all zero and bits 11-8 equal 1111. Bit 7, if set, means the Shift is left; if clear, means the Shift is right. Bit 6, if set, means the A register will be shifted, Bit 5, if set, means the Q register will be shifted. If bits 6 and 5 are set, both the Q and A register will be shifted. Bits 4-0 contain the count which determines the number of shifts to be made. Refer to section 7.2.2.8.

7.1.1.4 Skip -- The Skip instructions are those instructions which sense the existence of specific conditions within the computer. If the condition sensed for is not present, the next instruction comes from address P+1, where P is the address of the skip instruction.. If the condition sensed for is present, the next instruction comes from the address P+1+SK where SK is the skip count contained in bits 3-0 of the Skip instruction. The Skip instructions are identified by bits 15-12 = 0 and bits 11-8 = 0001. Bits 7-4 specify the specific Skip instruction and condition. Bits 3-0 are the Skip count. Refer to section 7.2.3. There are eight basic conditions sensed for with the Skip instruction. The inverse of these conditions is specified when bit 4 of the instruction is set, thus giving a total of 16 Skip instructions.

7.1.1.5 Inter-Register -- These instructions transfer data from certain combinations of origin registers through the Adder to certain combinations of destination registers. The Interregister instructions are identified by bits 15-12 = 0, and bits 11-8 = 1000. Refer to Section 7.2.2.5.

7.1.2 Addressing Modes -- There are seven types of addressing modes: Absolute, Relative, Indirect, 16-Bit Relative, Relative Indirect, Storage, and Constant. They apply to the storage reference instructions only. Section 7.2.1 gives a detailed explanation of each mode.

7.1.3 Interrupt System -- This interrupt system gives the program the ability to easily establish priority of interrupts such that an interrupt of high priority can interrupt the machine while processing an interrupt of a lower priority. The return path to the lower priority interrupt routines and then to the main program is clearly established and saved.

7. .3.1 Interrupt Trap locations are established for each interrupt line. They are in the range of addresses 0100 through 013C. The assignment for each interrupt state or line is shown in Table 7-2. The first column is the interrupt state. The second column is the value of △ to be used in the exit interrupt instruction to exit from that state.

The third column is the address where the contents of the program address register are stored when an interrupt occurs. The fourth column is the address of the first instruction to be executed following an interrupt. These addresses are reserved exclusively for interrupts unless that particular interrupt is not being used.

7.1.3.2 Mask Register -- The Mask Registeris the enable for each interrupt state or line. Bit 0 of themask register corresponds to interrupt line 0, bit 1 to line 1, etc. To enable an interrupt line, its corresponding bit in the mask register must be set. The Mask Register is set by the Interregister instruction.

7.1.3.3 Priority -- The priority of interrupts is under control of the computer program. The program assigns priority by establishing an interrupt mask for each interrupt state which enables all higher priority interrupts and disables all lower priority interrupts. When an interrupt state is entered, the mask for that state is placed in the mask register. Therefore, there may be up to 16 levels of priority. Note that it is possible to change priority during execution of a program.

TABLE 7-2. INTERRUPT STATE DEFINITIONS

Interrupt State	Value of △ To Exit State	Location of Return Address	Location of First Instruction After Interrupt Occurs
00	00	0100	0101
01	04	0104	0105
02	08	0108	0109
03	0C	010C	010D
04	10	0110	0111
05	14	0114	0115
06	18	0118	0119
07	1C	011C	011D
08	20	0120	0121
09	24	0124	0125
10	28	0128	0129
11	2C	012C	012D
12	30	0130	0131
13	34	0134	0135
14	38	0138	0139
15	3C	013C	013D

7.1.3.4 Internal Interrupts -- Interrupts are also generated by certain conditions arising within the computer. They are called internal interrupts. If such a condition occurs, it generates interrupt 00_{10} {interrupt mask bit 00_{10}}. Normally, internal interrupts are assigned the highest priority. The internal interrupts are:

1. Storage Parity Error
2. Program Protect Fault
3. Power Failure

7.1.3.5 Operation -- The computer can distinguish between up to 16 {1 internal, 15 external} different interrupts. Each of these interrupts has its respective address to which control is transferred upon recognizing the interrupt.

When the computer is processing a particular interrupt, it will be defined as being in that interrupt state {state 00 through 15}. Thus, the interrupts and their respective bits in the interrupt mask register are numbered 00 through 15. An interrupt in bit 7 will put the computer in interrupt state 7, etc.

Before the computer can recognize any interrupt, the mask bit for that interrupt must be set and the interrupt system must be activated. The mask register may be set by an Interregister command and the interrupt system can be activated by an enable interrupt command.

Upon recognizing an interrupt, the computer automatically stores the return address in the lower 15 bits of the storage location reserved for that interrupt state. The 16th bit is set or cleared to record the overflow indicator if the computeris in 32K mode. If the computer is in 65K mode, all 16 bits are required to save the return address. The program must check for an overflow condition with an SOV or an SNO instruction. The computer then deactivates the interrupt system and transfers control to another address also specified by the interrupt state. The program would then store all registers including the mask register in addresses reserved for this interrupt state and load the mask register with themask to be used while in this state. Ones in the mask denote interrupts that have higher priority than the interrupt being processed. Themask should not have a 1 in the position of the interrupt being processed. If an interrupt was allowed into the same state which is being processed, the return link would be lost. The program then activates the interrupt system and processes the interrupt.

The computer exits from an interrupt state in the following way. The program inhibits interrupt and restores the registers, including the mask register. If the computeris in 65K mode, the program must restore the overflow condition that existed when the interrupt occurred by forcing an overflow condition or by executing an SOV or SNO instruction. After loading the register, the program executes the exit interrupt command with Δ equal to the lower 8 bits of the base address of the interrupt state. This command reads the storage location where the return address is stored. The overflow indicator is set or cleared in accordance with bit 16 if the computer is in 32K mode. The interrupt system is activated, and control transferred to the return address.

7.1.3.6 Example -- The following table and sample program steps apply if we had five different possible interrupts and we wanted three levels of priority such that interrupt 01 had high priority, interrupt 02,05 had next priority, and interrupt 03,04 had low priority. This example assumes the computer is in 65K mode. If the computer is in 32K mode, skip the steps marked with {*}.

Bit	5	4	3	2	1	0	
Mask 1	1	1	1	1	1	1	Mask used for Main Program
Mask 2	1	0	0	1	1	1	Mask used for State 03, 04
Mask 3	0	0	0	0	1	1	Mask used for State 02, 05
Mask 4	0	0	0	0	0	0	Mask used for State 01

Main Program

```
 Set mask reg. to Mask 1
 enable int.
        ---
        ---
        ---
        ---
        ---
        ---
```

State 01 Program

```
 Store Registers
*Check overflow with SOV
    or SNO
 Set Mask to Mask 4
 enable int.
        ---
        ---
 Inhibit int.
*Reset overflow condition
 exit int. 01
```

State 03 Program

```
 Store Registers
*Check overflow with SOV or SNO
 Set Mask to Mask 2
 enable int.
        ---
        ---
 Inhibit int.
*Reset overflow condition
 replace registers
 exit int. 03
```

State 04 Program

```
 Store Registers
*Check overflow with SOV or SNO
 Set Mask to Mask 2
 enable int.
        ---
        ---
 Inhibit int.
*Reset overflow condition
 replace registers
 exit int. 04
 State 02 Program

 Store Registers
*Check overflow with SOV
    or SNO
 Set Mask to Mask 3
 enable int.
      ---
      ---
 Inhibit int.
*Reset overflow condition
 Replace registers
 exit int. 02
```

```
 State 05 Program

 Store Registers
*Check overflow with SOV
    or SNO
 Set Mask to Mask 3
 enable int.
      ---
      ---
 Inhibit int.
*Reset overflow condition
 Replace registers
 exit int. 05
```

7.1.4 Program Protect -- The MP17 computer has a Program Protect system which makes it possible to protect a program in the computer from any other nonprotected program also in the computer. The system is built around a program protect bit contained in each word of storage. If the bit is set, it means that word is an operand or an instruction of the protected program. All operand and instruction locations of the protected program must have theprogram protect bit set. None of the instructions or operands of the nonprotected program can have the program protect bit set.

Whenever a violation of the Program Protect system other than a Direct Storage Access violation is detected, the program protect fault FF is set and an internal interrupt is generated. A violation indicates that the nonprotected program has attempted an operation which could harm the protected program.

7.1.4.1 Program Protect Violations -- This section is an inclusive list of all possible Program Protect Violations.

7.1.4.1.1 Nonprotected instruction attempts to write in a protected storage location. The contents of the storage location are not changed.

7.1.4.1.2 Attempt to write into a protected storage location via External Storage access when a nonprotected instruction was the ultimate source of the attempt. The contents of the storage location are not changed.

7.1.4.1.3 An attempt is made to execute a protected instruction following the execution of a nonprotected instruction. The protected instruction is executed as a nonprotected Selective Stop instruction. It is not violation, however, if an interrupt caused this sequence of instructions.

7.1.4.1.4 An attempt is made to execute the following instructions when they are not protected; any Interregister instructions with bit 0=1, EIN, IIN, EXI, SPB, or CPB. Those instructions become a nonprotected Selective Stop instruction under these circumstances.

7.1.4.2 Program Protect is manually enabled by a two-position switch on the computer console. If the switch is not enabling program protect, none of the above violations are recognized with the exception of 7.1.4.1.2.

7.1.4.3 Set/Clear program protect bit -- The Program Protect instruction (SPB or CPB) is the only way in which the program protect bit may be set or cleared in each word of storage.

7.1.4.4 Programming requirements. In order forthis program protect system to work, the following program requirements must be met:

a. There must be a completely checked-out program package which handles all interrupts for the nonprotected program. This program must also be part of the protected program.

b. The protected program must be a completely checked-out program.

7.1.4.5 Peripheral Equipment Protection -- All peripheral equipment which is essential to operation of the protected program must have a switch which designates if it is a protected device. If the switch is on, the peripheral device responds with a reject to all nonprotected commands (except status request) addressed to it. All protected commands are responded to in the "normal" manner. If the switch is off, the peripheral device responds in the "normal" manner to protected and nonprotected commands.

7.2 REPERTOIRE OF INSTRUCTIONS

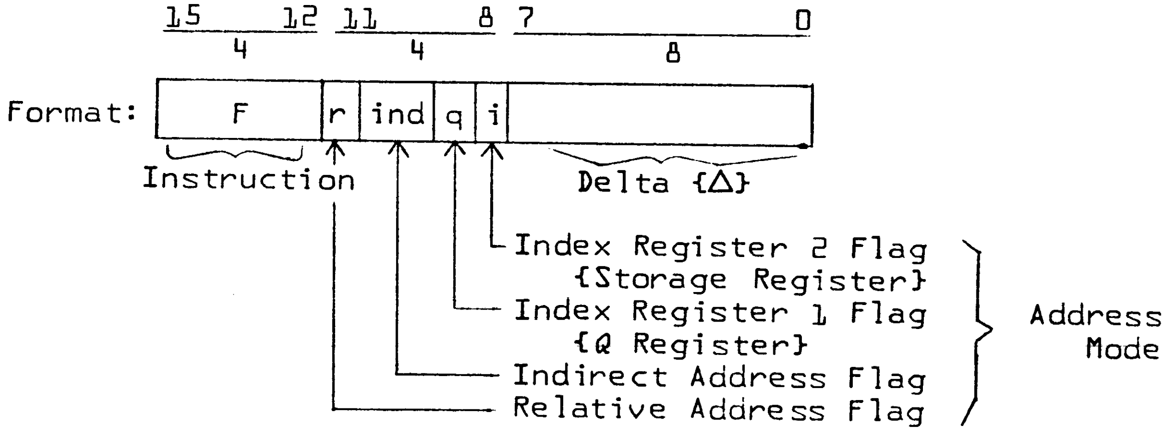
The MP17 computer shall be designed to execute the following groups of instructions:

Storage Reference
Register Reference
Skips

7.2.1 Storage Reference

The Storage Reference instructions contain three fields: Instruction, Address Mode, and Delta. The instruction field shall contain the operation code.

The address mode contains flags for indexing, indirect addressing, and relative addressing and the delta field is a signed 8-bit address modifier where the most significant bit is the sign bit. Delta is treated as zero when its contents = 00000000.



The Storage Reference instructions have seven different types of addressing modes:

a. Absolute
b. Constant
c. Indirect
d. Storage
e. Relative
f. 16-bit Relative
g. Relative Indirect

The following definitions apply to the descriptions of these addressing modes.

Instruction Address: The address of the instruction being executed = P.

Indirect Address: A storage address which contains an address rather than an operand.

Base Address: The operand address after all indirect addressing but before modification by Index registers. The base address is the effective address if no indexing is specified.

Effective Address: The final address of the operand. In certain cases, the effective address equals the operand for read operand type instructions. These cases are noted in Table 3-2.

Indexing: The computer has two Index registers. Index register number 1 is the Q register, and Index register number 2 is storage location $00FF_{16}$. The base address may be modified by either one or both of the Index registers. If the Index register number 1 flag is set, the contents of the Q register are added to the base address to form the effective address. If the Index register number 2 flag is set, the contents of storage location $00FF_{16}$ are added to the base address to form the effective address. If both Index register flags are set, the contents of Q are added to the base address; then the contents of $00FF_{16}$ are added to the result to form the effective address. Indexing occurs after indirect addressing has been completed.

The computer uses the 16-bit one's complement adder during Indexing operations. Consequently, Index register contents are treated as signed quantities {bit 15 = sign bit}.

Storage reference instructions have seven different types of addressing modes:

a. Absolute {address mode bits = 0, 1, 2, or 3}. Relative and indirect flags both equal ▽0▽ and delta ≠ ▽0▽. The base address equals delta. The sign bit of delta is not extended. The contents of the Index registers, when specified, are added to the base address to form the effective address.

   In the case where the address mode bits = 0, P+1 is the effective address.

b. Constant {address mode bits = 0, 1, 2, or 3}. Relative and indirect flags both equal ▽0▽ and delta = ▽0▽.

   In the case where the address mode bits = 1, 2, or 3, the contents of P+1 plus the contents of one or both Index registers form the effective address. The effective address is taken as the operand for read-operand type instructions.

c. Indirect {address mode bits = 4, 5, 6, or 7}. Relative address flag equals ▽0▽, indirect flag equals ▽1▽, and delta ≠ ▽0▽. The 8-bit value of delta is an indirect address. Delta is a magnitude quantity for this operation {no sign bit}.

d. Storage {address mode bits = 4, 5, 6, or 7}. Relative address flag equals ▽0▽, indirect flag equals ▽1▽, and delta = ▽0▽. The contents of location P+1 is an indirect address. When the base address is formed {indirect addressing complete}, the contents of one or both Index registers, if specified, are added to form the effective address.

e. Relative {address mode bits = 8, 9, A, or B}. The relative flag equals ▽1▽, indirect address flag equals ▽0▽, and delta does not equal ▽0▽. The base address is equal to the instruction address, P, plus the value of delta with sign extended. The contents of the index registers, when specified, are added to the base address to form the effective address.

Mode	Binary	Hex	△ Delta	Effective Address	Address of Next Instruction
Absolute	0000	0	≠0	Delta	P+1
Constnat			=0	P+1	P+2
Absolute	0001	1	≠0	△+{00FF}	P+1
Constant			=0	{P+1}+{00FF}*	P+2
Absolute	0010	2	≠0	△+{Q}	P+1
Constant			=0	{P+1}+{Q}*	P+2
Absolute	0011	3	≠0	△+{Q}+{00FF}	P+1
Constant			=0	{P+1}+{Q}+{00FF}*	P+2
Indirect	0100	4	≠0	{△}	P+1
Storage			=0	{P+1}	P+2
Indirect	0101	5	≠0	{△}+{00FF}	P+1
Storage			=0	{P+1}+{00FF}	P+2
Indirect	0110	6	≠0	{△}+{Q}	P+1
Storage			=0	{P+1}+{Q}	P+2
Indirect	0111	7	≠0	{△}+{Q}+{00FF}	P+1
Storage			=0	{P+1}+{Q}+{00FF}	P+2
Relative	1000	8	≠0	P+△	P+1
16-Bit Relative			=0	P+1+{P+1}	P+2
Relative	1001	9	≠0	P+△+{00FF}	P+1
16-Bit Relative			=0	P+1+{P+1}+{00FF}	P+2
Relative	1010	A	≠0	P+△+{Q}	P+1
16-Bit Relative			=0	P+1+{P+1}+{Q}	P+2
Relative	1011	B	≠0	P+△+{Q}+{00FF}	P+1
16-Bit Relative			=0	P+1+{P+1}+{Q}+{00FF}	P+2
Relative Ind.	1100	C	≠0	{P+△}	P+1
Relative Ind.			=0	{P+1+{P+1}}	P+2
Relative Ind.	1101	D	≠0	{P+△}+{00FF}	P+1
Relative Ind.			=0	{P+1+{P+1}}+{00FF}	P+2
Relative Ind.	1110	E	≠0	{P+△}+{Q}	P+1
Relative Ind.			=0	{P+1{P+1}}+{Q}	P+2
Relative Ind.	1111	F	≠0	{P+△}+{Q}+{00FF}	P+1
Relative Ind.			=0	{P+1+{P+1}}+{Q}+{00FF}	P+2

*Effective Address = operand for read operand type instructions.

f. 16-Bit Relative {address mode bits = 8, 9, A, or B}. The relative address flag equals ▿1▿, the indirect address flag equals ▿0▿, and delta equals ▿0▿. If no indexing is specified, the instruction address P+1, plus the contents of location P+1, form the base address = effective address. If indexing is specified, the contents of the specified Index register{s} are added to the base address to form the effective address.

g. Relative Indirect {address mode bits = C, D, E, or F}. Both relative and indirect flags equal ▿1▿.

1} Delta ≠ ▿0▿. The value of the instruction address, P, plus the value of delta with sign extended is an indirect address. If bit 15 of the contents of this indirect address is ▿0▿, the contents of this indirect address is the base address. If bit 15 of the contents of the indirect address is a ▿1▿, the contents of the indirect address is another indirect address if the computer is in 32K mode.

In 65K mode, all 16 bits of the first indirect address is the base address, and indirect addressing does not continue. In 32K mode, indirect addressing continues until bit 15 of the contents of the indirect address is ▿0▿. The contents of the Index registers, when specified, are then added to the base address to form the effective address.

2} Delta = ▿0▿. If bit 15 of P+1+{P+1} equals ▿1▿ and the computer is in 32K mode, P+1+{P+1} is indirect address. Indirect addressing continues until bit 15 of the contents of an indirect address is ▿0▿. In 65K mode, the contents of P+1+{P+1} is the base address. Then the contents of the Index register{s}, when specified, are added to the base address to form the effective address.

The preceding table shows all the addressing possibilities for storage reference instructions which may be obtained through combinations of flag bits.

Note: { } Denotes ▿contents of▿.

7.2.1.1 LDA Load A {F=A}

Load the A register with the contents of the storage location specified by the effective address. The contents of the storage location are not altered.

7.2.1.2 STA Store A {F=6}

Store the contents of the A registerin the storage location specified by the effective address. The contents of A are not altered.

7.2.1.3 LDQ Load Q {F=E}

Load the Q register with the contents of the storage location specified by the effective address. The contents of the storage location are not altered.

7.2.1.4 STQ Store {F=4}

Store the contents of the Q register in the storage location specified by the effective address. The contents of Q are not changed.

.1.5 ADD Add to A {F=8}

Add the contents of the storage location specified by the effective address to the contents of the A register. One's complement arithmetic is used. The overflow indicator will be set if the magnitude of the sum is greater than the capacity of the A register. Once set, the overflow indicator will remain set until a Skip on Overflow instruction is executed.

7.2.1.6 SUB Subtract from A {F=9}

Subtract the contents of the storage location specified by the effective address from the contents of the A register. One's complement arithmetic is used. Operation of overflow is the same as in ADD.

7.2 1.7 ADQ Add to Q {F=F}

Add the contents of the storagelocation specified by the effective address to the contents of the Q register. One's complement arithmetic is used. Operation of overflow is the same as in ADD.

7.2.1.8 AND And with A {F=A}

Form the logical product, bit by bit, of the contents of the storage location specified by the effective address and the contents of the A register. The result replaces the contents of A.

7.2.1.9 EOR Exclusive OR with A {F=B}

Form the logical difference {Exclusive OR}, bit by bit, of the contents of the storagelocation specified by the effective address and the contents of the A register. The results replace the contents of A.

7.2.1.10 RAO Replace Add One in Storage {F=D}

Add one to the contents of the storage location specified by the effective address. The contents of A and Q are not changed. One's complement arithmetic is used. Operation of overflow is the same as in ADD.

7.2.1.11 MUI Multiply Integer {F=2}

Multiply the contents of the storage location specified by the effective address by the contents of the A register. The 32-bit product replaces the contents of Q and A with the most significant bits in the Q register. One's complement arithmetic is used.

7.2.1.12 JMP Jump {F=1}

Replace the contents of the program counter {P} with the effective address.

7.2.1.13 RTJ Return Jump {F=5}

Replace the contents of the storage location specified by the effective address with the address of the next consecutive instruction. The address stored in the effective address will be P+1 or P+2, depending on the addressing mode of RTJ. The contents of P are then replaced with the effective address +1.

7.2.1.14 DVI Divide Integer {F=3}

Divide the combined contents of the Q and A registers with the contents of the effective address. The Q register contains the most significant bits before execution. The quotient is in the A register and the remainder in the Q register at the end of execution. The overflow indicator is set if the magnitude of the quotient is greater than the capacity of the A register. Once set, the overflow indicator remains set until a skip on overflow instruction is executed.

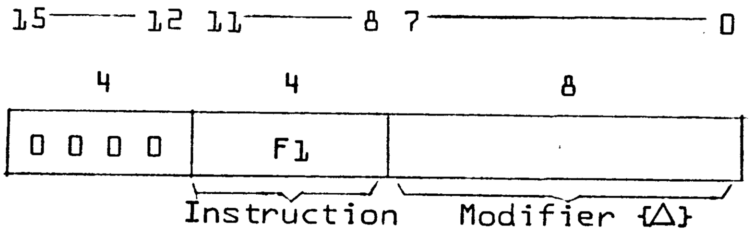
7.2.1.15 SPA Store A, Parity to A {F=7}

Store the contents of the A register in the storage location specified by the effective address. Set the A register to 1 if the parity bit of the word stored at the effective address is set. If theparity bit is not set, set the A register to 0.

7.2.2 Register Reference

Register reference instructions shall use the address mode field for the operation code. Register reference instructions are identified by the upper four bits of an instruction being zero.

Format:



7.2.2.1 SLS Selective Stop {F1=0} Δ=0

Stops the machine if this instruction is executed when the Stop switch is on. The instruction becomes a Pass when the switch is off.

7.2.2.2 INP Input to A {F1=2}

Read one word from an external device into the A register. The word in the Q register selects the sending device. If the device sends a reply, the next instruction comes from P+1. If the device sends a reject, the next instruction comes from P+1+Δ, where Δ is an 8-bit signed number including sign. An internal Reject causes the next instruction to come from P+Δ.

7.2.2.3 OUT Output from A {F1=3}

Output one word from the A registerto an external device. The word in the Q register selects the receiving device. If the device sends a reply, the next instruction comes from P+1. If the device sends a reject, the next instruction comes from P+1+Δ, where Δ is an 8-bit signed number including sign. An internal Reject causes the next instruction to come from P+Δ.

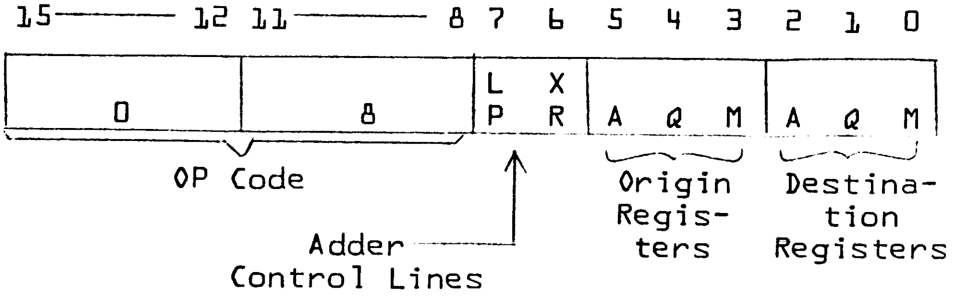
7.2.2.4 ENA Enter A {F1=A}

Replace the contents of the A register with the 8-bit delta, sign extended.

7.2.2.5 Inter-Register {F1=8}

These instructions cause data from certain combinations of origin registers to be sent through the Adder to any combination of destination registers. Various operations, selected by the Adder control lines, are performed on the data as it passes through the Adder.

Format:



Instructions:	Bit 7	6	5	4	3
SET Set to Ones	1	0	0	0	0
CLP Clear to Zero	0	1	0	0	0
TRA Transfer A	1	0	1	0	0
TRQ Transfer Q	1	0	0	1	0
TRB Transfer Q or M	1	0	0	1	1
TCA Transfer Complement A	0	1	1	0	0
TCM Transfer Complement M	0	1	0	0	1
TCQ Transfer Complement Q	0	1	0	1	0
TCB Transfer Complement Q + M	0	1	0	1	1
AAM Transfer Arithmetic Sum A, M	0	0	1	0	1
AAB Transfer Arithmetic Sum A, Q, + M	0	0	1	1	1
EAM Transfer Exclusive OR A, M	0	1	1	0	1
EAQ Transfer Exclusive OR A, Q	0	1	1	1	0
EAB Transfer Exclusive OR A, Q, + M	0	1	1	1	1
LAM Transfer Logical Product A, M	1	0	1	0	1
LAQ Transfer Logical Product A, Q	1	0	1	1	0
LAB Transfer Logical Product A, Q, + M	1	0	1	1	1
CAM Transfer Comp. Logical Product A, M	1	1	1	0	1
CAQ Transfer Comp. Logical Product A, Q	1	1	1	1	0
CAB Transfer Comp. Logical Product A, Q, + M	1	1	1	1	1

The origin registers are considered as operands, of which there are two kinds, defined as follows:

Operand 1 may be:

a. $FFFF_{16}$ (bit 5=0)

or b. The contents of A (bit 5=1)

Operand 2 may be:

a. $FFFF_{16}$ (bit 4=0 and bit 3=0)

or b. The contents of M (bit 4=0 and bit 3=1)

or c. The contents of Q (bit 4=1 and bit 3=0)

or d. The OR, bit by bit, of the contents of Q and M (bit 4=1 and bit 3=1)

7.2.[illegible].5.1 Operations Possible

LP=0 and XR=0. The data placed in the destination register(s) is the arithmetic sum of operand 1 and operand 2. The overflow indicator operates the same as in ADD.

LP=1 and XR=0. The data placed in the destination register(s) is the logical product, bit by bit, of operand 1 and operand 2.

LP=0 and XR=1. The data placed in the destination register(s) is the exclusive OR, bit by bit, of operand 1 and operand 2.

LP=1 and XR=1. The data in the destination register(s) is the complement of the logical product, bit by bit, of operand 1 and operand 2.

Operand 1	Operand 2	LP=0 XR=1	LP=1 XR=0	LP=1 XR=1	LP=0 XR=0
0	0	0	0	1	Arithmetic Sum
0	1	1	0	1	
1	0	1	0	1	
1	1	0	1	0	

Table 7-3. Interregister Instruction Truth Table

7.2.2.5.2 Notes

a. Register transfers can be accomplished with LP=0, XR=0, and making operand 1 or operand 2 equal to $FFFF_{16}$.

b. Magnitude comparisons without destroying either operand can be done with LP=0, XR=0, no destination register selected and testing the overflow indicator.

c. Complementing registers can be done with LP=0, XR=1, and making operand 1 or operand 2 equal to $FFFF_{16}$.

7.2.2.6 INA Increase A {F1=9}

Replaces the contents of A with the sum of the initial contents of A and delta. Delta is treated as a signed number with the sign extended into the upper 8 bits. Operation of overflow is the same as in ADD.

7.2.2.7 INQ Increase Q {F1=D}

Replaces the contents of Q with the sum of the initial contents of Q and delta. Delta is treated as a signed number, with the sign extended into the upper 8 bits. Operation of overflow is the same as in ADD.

7.2.2.8 ENQ Enter Q {F1=C}

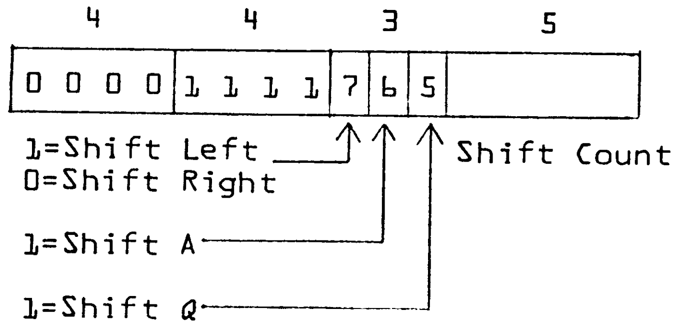
Replaces the contents of Q with the 8-bit delta, sign extended.

7.2.2.9 NOP No Operation {F1=B} Δ=0

7.2.2.10 Shifts As Defined Below

Mnemonic	Instruction Name
ARS	A Right Shift
QRS	Q Right Shift
LRS	Long Right Shift {QA}
ALS	A Left Shift
LLS	Long Left Shift

Format:



Shifts A or Q, or QA, left or right the number of places specified by the 5-bit shift count. Sign is extended on right shifts. Left shifts are end-around.

7.2.2.11 The following instructions are legal only if the PROGRAM PROTECT switch is off or if the instructions themselves are protected. If an instruction is illegal, it becomes a Selective Stop and interrupt on Program Protect Fault is possible {if selected}

Switch ON: Selective Stop unless instruction is protected.

Switch OFF: Normal execution {no program protection}.

7.2.2.11.1 EIN Enable Interrupt {F1=4} Δ=0

Activates the interrupt system. The interrupt system must be active and the mask bit set for an interrupt to be recognized.

7.2.2.11.2 IIN Inhibit Interrupt {F1=5} Δ=0

Deactivates interrupt system.

7.2.2.11.3 EXI Exit Interrupt State {F1=E}

This instruction must be used to exit from any interrupt state. Delta defines the interrupt state from which the exit is taken. {Refer to Table X}. This instruction automatically reads the address containing the return address, resets the Overflow indicator according to bit 16 if the computer is in 32K mode, activates the interrupt system and jumps to the return address.

7.2.2.11.4 SPB Set Program Protect {F1=6} Δ=0

Sets the program protect bit in the address specified by Q.

7.2.2.11.5 CPB Clear Program Protect {F1=7} Δ=0

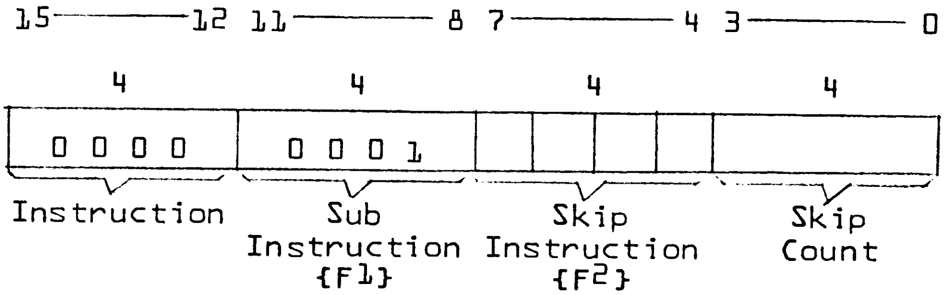
Clears the program protect bit in the address specified by Q.

7.2.2.11.6 Any Interregister instruction in which bit 0 is a ▽1▽. This bit selects M as a Destination Register.

7.2.3 Skips

Skip instructions shall be identified when the address mode field is 1 and the instruction mode field is 0.

Format:



When the skip condition is met, the contents of the skip count +1 is added to P to obtain the address of the next instruction {e.g., when the skip count is zero, go to P+1}. When the skip condition is not met, the address of the next instruction is P+1 {skip count ignored}. The skip count does not have a sign bit.

7.2.3.1 SAZ Skip if A=+0 {F2=0}

7.2.3.2 SAN Skip if A≠+0 {F2=1}

7.2.3.3 SAP Skip if A=+ {F2=2}

7.2.3.4 SAM Skip if A=- {F2=3}

7.2.3.5 SQZ Skip if Q=+0 {F2=4}

7.2.3.6 SQN Skip if Q≠+0 {F2=5}

7.2.3.7 SQP Skip if Q=+ {F2=6}

7.2.3.8 SQM Skip if Q=- {F2=7}

7.2.3.9	SWS	Skip if switch set	{F2=8}
7.2.3.10	SWN	Skip if switch not set	{F2=9}
7.2.3.11	SOV	Skip on overflow	{F2=A}
7.2.3.12	SNO	Skip on no overflow	{F2=B}
7.2.3.13	SPE	Skip on storage parity error	{F2=C}
7.2.3.14	SNP	Skip on no storage parity error	{F2=D}
7.2.3.15	SPF	Skip on Program Protect Fault	{F2=E}

Program protect fault is set by:

1. A nonprotected instruction attempting to write into a protected address.

2. A protected instruction executed immediately following a nonprotected instruction except if an interrupt caused the instruction sequence.

3. Execution of any nonprotected instruction affect-interrupt mask or enables.

The program protect fault is cleared when it is sensed. The program protect fault cannot be set if the program protect system is disabled.

7.2.3.16 SNF Skip on no Program Protect Fault {F2=F}

Refer to SPF for conditions for a Program Protect Fault.

7.3 Enhanced Instructions

Instruction formats for enhancement to the 1700 Instruction Repertorie are upward compatible with the existing 1704/14/84 computer.

7.3.1 Type 2 Storage Reference Instructions

FORMAT:

15 12	11 8	7	6	5 3	2 0
0	4	r	i	Ra	Rb
F4	F5	Δ			

Type 2 storage reference instructions are identified when the F field is zero¬ the F1 field is equal to four¬ and registers Ra and Rb are not both specified to be the I register {i.e.¬ both not zero; see below}.

The Type 2 storage reference instruction contains four parts: Instruction Field {F4}¬ Instruction Mode Field {F5}¬ Addressing Mode Fields {delta¬ r¬ i¬ and Ra}¬ and Register Rb. From these¬ two operands {A and B} are specified for the executing of the instruction.

The F4 field determines the instruction {e.g.¬ add¬ subtract¬ etc.}; the F5 field determines the instruction mode {e.g.¬ word or character}. The Addressing Mode Fields contains four fields to determine the addressing:

1} Delta determines 8 or 16 bit addressing {i.e.¬ whether the instruction is two or three words}.

2} Flag r is the relative address flag.

3} Flag i is the indirect address flag.

4} Register Ra is the index register.

The Addressing Mode Fields determine the address for operand A; register Rb and Instruction Mode Field F4 determines the addressing for operand B. Table 1 specifies the addressing modes¬ the effective address¬ and the address of the next instruction.

7.3.1 Type 2 Storage Reference Instructions (Continued)

The following definitions apply to the description of addressing modes.

- Instruction Address: the address of the instruction being executed, also called P.

- Indirect Address: a storage address which contains an address rather than an operand. (Currently, there is no implementation of indirect addressing).

- Base Address: the operand address after all indirect addressing but before modification by an index register. The base address is the effective address if no indexing is specified.

- Effective Address: the final address of the operand.

- Indexing: If specified, the contents of register Ra is added to the base address to form the effective address. Indexing occurs after indirect addressing has been completed.

  The computer uses the 16-bit one's complement adder during Indexing operations. Consequently, index register contents are treated as signed quantities (bit 15 is the sign bit).

- Registers: the registers Ra and Rb are defined as follows:

Register	Value
I	0
1	1
2	2
3	3
4	4
Q	5
A	6
None	7

7.3.1 Type 2 Storage Reference Instructions {Continued}

Type 2 storage reference instructions have four different types of addressing modes. They are:

1. 8-Bit Absolute {r is zero¬ i is zero¬ and delta is not zero}. The base address equals delta. The sign bit of delta is not extended. The contents of index register Ra¬ when specified¬ is added to the base address to form the effective address.

   If no indexing takes place¬ the addresses which can be referenced in the 8-bit Absolute mode are restricted to the low core area. Delta is eight bits in length and thus the computer references a location between 0001 and 00FF inclusive.

2. 8-Bit Relative {r is one¬ i is zero¬ and delta is not zero}. The base address is equal to the instruction address plus one¬ P+1¬ plus the value of delta with sign extended. The contents of index register Ra¬ when specified¬ is added to the base address to form the effective address.

   If no indexing takes place¬ the addresses which can be referenced in the 8-Bit Relative mode are restricted to the program area. Delta is eight bits in length and thus the computer references a location between P-7E and P+80 inclusive.

3. 16-Bit Absolute {r is zero¬ i is one¬ and delta is zero}. The base address equals the contents of P+2. The contents of index register Ra¬ when specified¬ is added to the base address to form the effective address.

4. 16-Bit Relative {r is one¬ i is zero¬ and delta is zero}. The base address equals the contents of P+2 plus P+2. The contents of index register Ra¬ when specified¬ is added to the base address to form the effective address.

7.3.1 Type 2 Storage Reference Instructions (Continued)

TYPE 2 STORAGE ADDRESSING RELATIONSHIPS

Mode	Delta	r	i	Ra	Effective Address	Address Of Ne Instruction
8-Bit Absolute	Δ≠ 0	0	0	0	Δ + {I}	P+2
				1	Δ + {1}	
				2	Δ + {2}	
				3	Δ + {3}	
				4	Δ + {4}	
				5	Δ + {Q}	
				6	Δ + {A}	
				7	Δ	
8-Bit Relative		1		0	P+1+Δ{SE}+ {I}	
				1	P+1+Δ{SE}+ {1}	
				2	P+1+Δ{SE}+ {2}	
				3	P+1+Δ{SE}+ {3}	
				4	P+1+Δ{SE}+ {4}	
				5	P+1+Δ{SE}+ {Q}	
				6	P+1+Δ{SE}+ {A}	
				7	P+1+Δ{SE}	
16-Bit Absolute	Δ = 0	0	1	0	{P+2} + {I}	P+3
				1	{P+2} + {1}	
				2	{P+2} + {2}	
				3	{P+2} + {3}	
				4	{P+2} + {4}	
				5	{P+2} + {Q}	
				6	{P+2} + {A}	
				7	{P+2}	
16-Bit Relative		1	0	0	P+2+{P+2} + {I}	
				1	P+2+{P+2} + {1}	
				2	P+2+{P+2} + {2}	
				3	P+2+{P+2} + {3}	
				4	P+2+{P+2} + {4}	
				5	P+2+{P+2} + {Q}	
				6	P+2+{P+2} + {A}	
				7	P+2+{P+2}	

Note: { } denotes "contents of"

7.3.1. Type 2 Storage Reference Instructions {Continued}

Notes:

A. The format is similar to the Type 1 interregister instructions.

B. F1=7 was chosen because Type 1 interregister instructions have F1=8. Thus the F1 codes will be adjacent.

C. Note that if F2a is zero and registers Ra and Rb are specified to be the I register, the instruction is a clear program protect bit {CPB}.

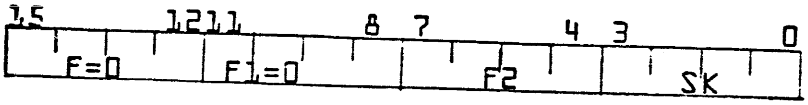
D. Field F2a is reserved for future expansion for the Type 2 interregister instruction format.

E. The value of 7 {none} cannot be used for either Ra or Rb.

F. The former mnemonic {TRR} was changed to avoid conflict with TRA and TRQ.

7.3.2 Type 2 Skip Instructions

FORMAT:



Type 2 skip instructions are identified when the F and F1 fields are both zero, and the skip count {SK} field is not zero. When the skip condition is met, the skip count plus one is added to the P register to obtain the address of the next instruction {e.g., when the skip count is one, go to P+2}. When the skip condition is not met, the address of the next instruction is P+1 {skip count ignored}. The skip count does not have a sign bit and cannot be zero.

S4Z SK	{F2=0}	Skip if register 4 is a positive zero {all bits are zero}.
S4N SK	{F2=1}	Skip if register 4 is not a positive zero {not all bits are zero}.
S4P SK	{F2=2}	Skip if register 4 is positive {bit 15 is a zero}.
S4M SK	{F2=3}	Skip if register 4 is negative {bit 15 is a one}.
S1Z SK	{F2=4}	Skip if register 1 is a positive zero {all bits are zero}.
S1N SK	{F2=5}	Skip if register 1 is not a positive zero {not all bits are zero}.
S1P SK	{F2=6}	Skip if register 1 is positive {bit 15 is a zero}.
S1M SK	{F2=7}	Skip if register 1 is negative {bit 15 is a one}.
S2Z SK	{F2=8}	Skip if register 2 is a positive zero {all bits are zero}.
S2N SK	{F2=9}	Skip if register 2 is not a positive zero {not all bits are zero}.
S2P SK	{F2=A}	Skip if register 2 is positive {bit 15 is a zero}.

7.3.2 Type 2 Skip Instructions {Continued}

S2M SK	{F2=B}	Skip if register 2 is negative {bit 15 is a one}.
S3Z SK	{F2=C}	Skip if register 3 is a positive zero {all bits are zero}.
S3N SK	{F2=D}	Skip if register 3 is not a positive zero {not all bits are zero}.
S3P SK	{F2=E}	Skip if register 3 is positive {bit 15 is a zero}.
S3M SK	{F2=F}	Skip if register 3 is negative {bit 15 is a one}.

Notes:

1. The format is exactly the same as the Type 1 skip instructions.

2. F1=0 was chosen because Type 1 skip instructions have F1=1. Thus the F1 codes will be adjacent.

3. Note that if F2 and SK are zero, the instruction is a selective stop {SLS}.

7.3.3 Field Reference Instructions

FORMAT:

15 12	11 8	7	6	5 3	2 0
D	S	r	i	Ra	F3a
FLDSTR	FLDLTH	Δ			

Field reference instructions are identified when the F field is zero, the F1 field is equal to five, and the F3a field is not zero.

Field reference instructions contain four parts: instruction field (F3a), addressing mode fields (delta, r, i, and Ra), FLDSTR and FLDLTH fields. The F3a field determines the instruction (e.g., load, store, etc). The addressing mode fields are defined exactly as the Type 2 storage reference instructions (see I).

FLDSTR defines the starting bit of the field: FLDSTR=0 means the field starts at bit 0; FLDSTR=15 means the field starts at bit 15. FLDLTH defines the length of the field minus one: FLDLTH=0 means the field is one bit long; FLDLTH=15 means the field is 16 bits long.

A field starts at the bit specified by FLDSTR and includes FLDLTH contiguous bits to the right of that bit. No field may cross a word boundary; e.g., if FLDSTR=0, the field length must be one bit long (FLDLTH=0).

SFZ

(F3a=2)

If the contents of the specified field of the storage location specified by the effective address are zero (all bits are zero), skip one location; otherwise execute the next instruction.

SFN

(F3a=3)

If the contents of the specified field of the storage location specified by the effective address are non-zero (not all bits are zero), skip one location; otherwise execute the next instruction.

Note: Each skip field instruction assumes that a one word instruction follows it.

7.3.3 Field Reference Instructions {Continued}

LFA

{F3a=4}

Load register A, right justified, with the contents of the specified field of the storage location specified by the effective address. All other bits of register A are cleared to zero. The contents of storage are not altered.

SFA

{F3a=5}

Store the contents of the field from register A, right justified, into the specified field of the storage location specified by the effective address. All other bits of storage are unchanged. Memory is locked until completion of the instruction. The contents of A are not altered.

CLF

{F3a=6}

Clear to all zeros the specified field of the storage location specified by the effective address. All other bits of storage are unchanged. Memory is locked until completion of the instruction.

SEF

{F3a=7}

Set to all ones the specified field of the storage location specified by the effective address. All other bits of storage are unchanged. Memory is locked until completion of the instructions.

<u>Notes:</u>

1. Format is similar to the Type 1 interregister instructions.
2. The used F3a code is for future expansion.
3. Note that if r and i are zero, register Ra is specified to be the I register, and F3a equals zero, the instruction is an inhibit interrupt {IIN}.
4. The value of seven {none} is used for Ra, if no indexing is specified.
5. The mnemonics LDF and STF have been changed to LFA and SFA to be compatible with type 2 storage referenc instructions {see I}.

7.3.4 Decrement and Skip Backwards Instructions

FORMAT:

15 — 12	11 — 8	7 — 5	4	3 — 0
0	6	Ra	0	SK

The decrement and skip backwards instruction is specified when the F field is zero, the F1 field is equal to six, and the skip count {SK} is not zero. When the skip condition is met, the skip count is subtracted from the P register to obtain the address of the next instruction {e.g., when the skip count is one, go to P-1}. When the skip condition is not met, the address of the next instruction is P+1. The skip count does not have a sign bit and cannot be zero. Register Ra specifies the register to be decremented and tested.

DIP

{Ra=0}

Decrement by one the contents of register I and skip backwards SK locations if the contents of register I are positive; otherwise execute the next instruction.

D1P

{Ra=1}

Decrement by one the contents of register 1 and skip backwards SK locations if the contents of register 1 are positive; otherwise execute the next instruction.

D2P

{Ra=2}

Decrement by one the contents of register 2 and skip backwards SK locations if the contents of register 2 are positive; otherwise execute the instruction.

D3P

{Ra=3}

Decrement by one the contents of register 3 and skip backwards SK locations if the contents of register 3 are positive; otherwise execute the next instruction.

D4P

{Ra=4}

Decrement by one the contents of register 4 and skip backwards SK locations if the contents of register 4 are positive; otherwise execute the next instruction.

7.3.4 Decrement and Skip Backwards Instructions {Continued}

DQP

{Ra=5}

Decrement by one the contents of register Q and skip backwards SK locations if the contents of register Q are positive; otherwise execute the next instruction.

DAP

{Ra=6}

Decrement by one the contents of register A and skip backwards SK locations if the contents of register A are positive; otherwise execute the next instruction.

Notes:

1. The format is similar to Type 1 and Type 2 skip instructions.

2. Bit 4 must be equal zero; bit 4=1 is reserved for future expansion.

3. Note that if the skip count is zero and register Ra is specified as the register I, the instruction is a set program protect bit {SPB}.

7.3.5 Miscellaneous Instructions

FORMAT:

15 – 12	11 – 8	7 – 5	4	3 – 0
0	B	Ra	0	F3

Miscellaneous instructions are identified when the F field is zero, the F1 field is equal to a decimal eleven {hex B}, and the F3 field is not zero. Various operations are selected via the F3 field; while register Ra may be used to specify an operand. Register Ra is defined as follows:

Register	Value
I	0
1	1
2	2
3	3
4	4
Q	5
A	6
None	7

{Note that the symbol R below is equivalent to Ra}.

GPE

Generate Character Parity Even {F3=2}

Set or clear bit 7 of register A to cause the parity of bits 0-7 to be even. The rest of the contents of register A are not altered. Register Ra is not used.

GPO

Generate Character Parity Odd {F3=3}

Set or clear bit 7 of register A to cause the parity of bits 0-7 to be odd. The rest of the contents of register A are not altered. Register Ra is not used.

LUB R

Load Upper Unprotected Bounds Register {F3=4}

Load the upper unprotected bounds register from the contents of register R.

7.3.5 Miscellaneous Instructions {Continued}

LLB R	Load Lower Unprotected Bounds Register {F3=5} Load the lower unprotected bounds register from the contents of register R.
EMA R	Execute Firmware Sequence from Macro-memory {F3=6 Transfer machine control to the storage location {macromemory} specified by the contents of register R with further machine control being from storage rather than micromemory. After completion of the firmware sequence, control may be returned to the next instruction address {P+1}.
EMI R	Execute Firmware Sequence from Micro-memory {F3=7 Transfer machine control to the micromemory address specified by the contents of register R. After completion of the firmware sequence, control may be returned to the next instruction address {P+1}.
SIO	Set/Sample Output or Input {F3=8} For output, one word from the register A is set {output} to an external device. The word in register Q selects the receiving device. For input, one word from an external device is sampled {input} to register A. The word in register Q selects the sending device. This instruction permits transmission to/from MOS peripheral devices. Register Ra is not used.
LMP R	Load Memory Page Register {F3=9} Load the memory page register specified by the contents of bits14 and15 of register R with the contents of bits 0-3 of register R.

7.3.5 Miscellaneous Instructions {Continued}

Notes:

1. Note that if register Ra is specified as register I and the F3 field is zero, the instruction will be a no operation {NOP}.

2. Bit 4 must be equal to zero; bit 4=1 {along with unused F3 codes} is reserved for future expansion.

3. Further definition for the contents of register Q in the SIO instruction will be needed.

4. All miscellaneous instructions are protected.

7.4 1700 Emulation

This section contains a description of the 16 bit micro processor configuration applicable to 1700 emulation, as well as the concepts which must be understood regarding the 1700 transform module to properly emulate the 1700 macro instruction repertorie.

7.4.1 1700 Transform - Three types of micro program selectable transform execution are possible; they are an MA transform, K or N transform, or a combined MA and K transform. These transform commands are coded in the C field of a micro instruction as TMA/j, TK/j,TN/j, and TMAK/j or GITMAK/j. The letter j is decoded as the lower 4 bits of the micro instruction for MA transforms {16 selections}, and as the lower 3 bits of the micro instruction for K or N transforms {8 selections}. These bits specify the selector position to use for MA selector S5 {16 positions} and K or N selector S8 {8 positions}.

The 1700 transform module provides for more efficient emulation of the basic 1700 instructions by including specialized hardware. A block diagram of the 1700 transform module is shown in Figure 7-1. The IXT register holds the current micro instruction being emulated. The IXT^1 register holds the least significant 8 bits of the macro instruction and is capable of being shifted left or right.

The IXT register is loaded by executing a main memory READ micro instruction followed by a micro instruction with a GITMAK/j or GITMAK/XT C field micro command. The execution of the micro instruction with the GITMAK command results in the following operations:

■ GITMAK/j - 1} Output data from main memory is gated into IXT register.

2} An MA and K transform are executed based on the value of j {limited to positions 0-7}.

■ GITMAK/XT - 1} Output data from main memory is gated into IXT and IXT^1 registers.
2} One of eight MA transforms is executed based on the macro instruction loaded into the IXT register {selected from S5 positions 8-15}. The K register will always be transformed from S8 position 7.

7.4.1 1700 Transforms {Continued}

- GITMAK/XT 3} The most significant 16 bits of the micro instruction register {MIR} are loaded with a micro command encoded off the macro instruction residing in the IXT register. This operation is referred to as a transform of MIR {XT/MIR}. The least significant 16 bits of MIR are loaded from micro memory.

The special transforms selected by the transform hardware whenever a GITMAK/XT command is executed are shown in Figures 7-2, 7-3, and 7-4.

The MP17 MA transform assignments are shown in Figure 7-5. MA transforms 8 through 15 are used for emulation of the basic 1700 instructions. MA transforms 0 through 7 are used for emulating the 1700 enhanced instructions. The letter C in these transforms indicates a bias to be determined when the emulation micro program is written. Transforms 1, 3 and 5 may not be required and may be eliminated from the final transform design.

MODE	F1 (BINARY)	HEX	DELTA	MA TRANSFORM	MIR TRANSFORM
Absolute	0000	0	≠0	XT/F	$\Delta \longrightarrow$ X, AB
Constant			=0	XT/F1	P+1 $\longrightarrow$ P, AB
Absolute	0001	1	≠0	XT/F	Δ +[00FF] $\longrightarrow$ X, AB
Constant			=0	EX/F1^1	P+1 $\longrightarrow$ P, AB
Absolute	00010	2	≠0	XT/F	Δ +[Q] $\longrightarrow$ X, AB
Constant			=0	XT/F1^1	P+1 $\longrightarrow$ P, AB
Absolute	0011	3	≠0	XT/CON	Δ +[00FF] $\longrightarrow$ X, AB
Constant			=0	XT/F1^1	P+1 $\longrightarrow$ P, AB
Indirect	0100	4	≠0	XT/F1^1	$\Delta \longrightarrow$ X, AB
Storage			=0	XT/F1^1	P+1 $\longrightarrow$ P, AB
Indirect	0101	5	≠0	XT/F1^1	$\Delta \longrightarrow$ X, AB
Storage			=0	XT/F1^1	P+1 $\longrightarrow$ P, AB
Indirect	0110	6	≠0	XT/F1^1	$\Delta \longrightarrow$ X, AB
Storage			=0	XT/F1^1	P+1 $\longrightarrow$ P, AB
Indirect	0111	7	≠0	XT/F1^1	$\Delta \longrightarrow$ X, AB
Storage			=0	XT/F1^1	P+1 $\longrightarrow$ P, AB
Relative	1000	8	≠0	XT/F	P+Δ(SE) $\longrightarrow$ X, AB
16-Bit Relative			=0	XT/F1	P+1 $\longrightarrow$ P, AB
Relative	1001	9	≠0	XT/F1	P+Δ(SE) $\longrightarrow$ X, AB
16-Bit Relative			=0	XT/F1	P+1 $\longrightarrow$ P, AB
Relative	1010	A	≠0	XT/F1	P+Δ(SE) $\longrightarrow$ X, AB
16-Bit Relative			=0	XT/F1	P+1 $\longrightarrow$ P, AB
Relative	1011	B	≠0	XT/F1	P+Δ(SE) $\longrightarrow$ X, AB
16-Bit Relative			=0	XT/F1	P+1 $\longrightarrow$ P, AB
Relative Ind.	1100	C	≠0	XT/F1^1	P+Δ(SE) $\longrightarrow$ X, AB
Relative Ind.			=0	XT/CON	P+1 $\longrightarrow$ P, AB
Relative Ind	1101	D	≠0	XT/F1^1	P+Δ(SE) $\longrightarrow$ X, AB
Relative Ind.			=0	XT/CON	P+1 $\longrightarrow$ P, AB
Relative Ind.	1110	E	≠0	XT/F1^1	P+Δ(SE) $\longrightarrow$ X, AB
Relative Ind.			=0	XT/CON	P+1 $\longrightarrow$ P, AB
Relative Ind.	1111	F	≠0	XT/F1^1	P+Δ(SE) $\longrightarrow$ X, AB
Relative Ind.			=0	XT/CON	P+1 $\longrightarrow$ P, AB

FIGURE 7-2: 1700 STORAGE REFERENCE TRANSFORMS
{GITMAK/XT}

F1 (BINARY)	MA TRANSFORM	MIR TRANSFORM	COMMENT
0000	XT/F1	No-Op	Sel Stop (Δ=0) Inst. Enh. (Δ≠0)
0001	XT/SKIP	P+Δ(SKIP)→X, AB	SKIP
0010	XT/F1	P+Δ(SE)→F, AB	Input to A
0011	XT/F1	P+Δ(SE)→F, AB	Output from A
0100	XT/F1	No-Op	Enable Int. (Δ=0) Inst. Enh. (Δ≠0)
0101	XT/F1	No-Op	Inhibit Int. (Δ=0) Inst. Enh. (Δ≠0)
0110	XT/F1	Q→AB	Set Prog. Protect (Δ=0) Inst. Enh. (Δ≠0)
0111	XT/F1	Q→AB	Clr. Prog. Protect (Δ=0) Inst. Enh. (Δ≠0)
1000	*	*	Inter Register
1001	Xt/F1	P+1→P, AB	Increase A
1010	XT/F1	P+1→P, AB	Enter A
1011	XT/F1	No-Op	Pass (Δ=0) Inst. Enh. (Δ≠0)
1100	XT/F1	P+1→P, AB	Enter Q
1101	XT/F1	P+1→P, AB	Increase Q
1110	XT/F1	Δ(INT)→X, AB	Exit Int.
1111	XT/SHIFT	A→F	Shift

*See 1700 Inter Register Transforms

FIGURE 7-3: 1700REGISTER REFERENCE TRANSFORMS (GITMAK/XT)

MA TRANSFORMS	CONDITION
XT/IR	M not origin reg.
XT/IRM	M is origin reg.

MIR TRANSFORMS	CONDITION

	MIR FIELDS			Origin					Dest.		
F	A	B	D	LP I07	XR I06	A I05	Q I04	M I03	A I02	Q I01	M I00
ADDT	-	-	-	0	0	X	X	0	X	X	X
A•B	-	-	-	1	0	X	X	0	X	X	X
A⊕B	-	-	-	0	1	X	X	0	X	X	X
$\overline{A}+\overline{B}$	-	-	-	1	1	X	X	0	X	X	X
ADD+	-	-	-	X	X	X	X	1	X	X	X
	One's	-	-	X	X	0	X	0	X	X	X
	A Reg	-	-	X	X	1	X	0	X	X	X
	P Reg	-	-	X	X	X	X	1	X	X	X
		One's	-	X	X	X	0	0	X	X	X
		Q Reg	-	X	X	X	1	0	X	X	X
		Zero	-	X	X	X	X	1	X	X	X
			No-Op	X	X	X	X	0	0	0	0
			A Reg*	X	X	X	X	0	1	X	X
			Q Reg*	X	X	X	X	0	0	1	X
			F Reg	X	X	X	X	0	0	0	1
			P Reg*	X	X	X	X	1	X	X	X

*No-Op if protect violation detected.

FIGURE 7-4: 1700 INTERREGISTER TRANSFORMS (GITMAK/XT)

XT	MIR 28-31 (2)	S5 MA TRANSFORM							
							IXT^1		
XT/REG	0	C	C	C	C	C	10	11	12
						IXT			
XT/F4	1	C	C	C	C	00	01	02	03
								IXT^1	
XT/F2a	2	C	C	C	C	C	C	08	09
						IXT^1			
XT/F2	3	C	C	C	C	08	09	10	11
							IXT^1		
XT/F3a	4	C	C	C	C	C	13	14	15
						IXT^1			
XT/F3	5	C	C	C	C	14	15	08	09
		S1							
XT/S1	6	08	09	10	11	12	13	14	15
		S2							
XT/S2	7	08	09	10	11	12	13	14	15

XT	MIR 28-31 (2)	S5 MA TRANSFORM							
					IXT				
XT/SKIP	8	1	1	0	08	09	10	11	0
							IXT		
XT/SHIFT	9	1	1	1	1	0	08	09	10
						IXT			
XT/IR	10	1	0	1	1	12	13	14	15
					IXT				
XT/F	11	1	0	0	00	01	02	03	0
						IXT			
XT/IRM	12	1	1	1	0	08	09	10	11
					IXT				
XT/F1	13	0	1	F=0	04	05	06	07	Δ=0
							IXT		
XT/F1^1	14	1	0	1	0	X0	06	07	0
							Δ≠ 0 +X1		↓
XT/CON	15	1	0	0	0	0	0	X1	

X0: F1=00XX + Multi-Level Indirect Mode
X1: Protect Violation

FIGURE 7-5: MP17 MA TRANSFORMS

XT	MIR 29-31 (2)	S8 K/N TRANSFORM								
XT/S2	0	S2	08	09	10	11	12	13	14	15
XT/SHIFT	1	IXT (11–15)	0	0	0	11	12	13	14	15
XT/EXTINT	2	S2 (12–15)	1	1	12	13	14	15	0	0
XT/REG	3	IXT[1] (10–12)	0	0	0	0	0	10	11	12
XT/S1	4	S1	08	09	10	11	12	13	14	15
XT/$\overline{\text{FLDSTR}}$	5	IXT	0	0	0	0	$\overline{00}$	$\overline{01}$	$\overline{02}$	$\overline{03}$
XT/MIR	6		24	25	26	27	28	29	30	31
XT/$\overline{\text{FLDLTH}}$	7	IXT	0	0	0	0	$\overline{04}$	$\overline{05}$	$\overline{06}$	$\overline{07}$

FIGURE 7-6: MP17 K/N TRANSFORMS

7.4.1 1700 Transforms (Continued)

The MP17 K/N transform assignments are shown in Figure 7-6. Transforms 0, 4 and 6 are general purpose transforms. Transforms 1, 3, 5 and 7, are used for 1700 emulation and transform 2 is used for processing external interrupts during auto data transfers.

The IXT^1 register is utilized during emulations of 1700 enhanced instructions, and serves two functions.

A. Provides a holding register for least significant eight bits of first macro instruction word for double word formats I and IV.

B. Provides the capability of shifting the least significant eight bits of first macro instruction right or left for more flexibility in emulating the enhanced instruction formats.

The IXT^1 register is shifted by executing a micro instruction with F = 11100. The type of shift is determined by the coding of bits 7 to 12 of the micro instruction as shown below, and the number of shifts is determined by the contents of the N register.

MICRO INSTRUCTION BITS					Operation
7	8	9	11	12	
1	0	0	1	0	IXT^1 right shifted (N) bits end around
0	1	0	1	0	IXT^1 left shifted (N) bits end around

7.4.2 MP17 Bit Test Assignments - The following bit test assignments are applicable to 1700 emulation.

BIT	Operation
0	Open

7.4.2 MP17 Bit Test Assignments (Continued)

BIT	OPERATION
1	Open
2	Execute upper micro instruction if "i" field of 1700 enhanced format I or IV is a one.
3	Execute upper micro instruction if "r" field of 1700 enhanced format I or IV is a one.
4	Execute upper micro instruction if bit "0" of F5 field of 1700 enhanced format I is a one.
5	Execute upper micro instruction if bit "1" of F5 field of 1700 enhanced format I is a one.
6	Execute lower micro instruction if selective stop switch set.
7	Execute lower micro instruction if selective skip switch set.
8	Execute upper micro instruction if the memory parity line is true (indicates even parity).
9	Execute upper micro instruction if STORE OOFF (index "i") status is true.
10	Execute upper micro instruction if DELTA = 0.
11	Execute lower micro instruction if 1700 Storage reference address mode indicates Effective Address = Operand.
12	Execute lower micro instruction if Storage Parity Error detected.
13	Execute lower micro instruction if Storage Protect Fault detected.
14	Execute upper micro instruction if Multilevel Indirect Mode selected.
15	Execute upper micro instruction if previous main memory write cycle was aborted.

7.4.3 1700 S/M Register Bit Assignments - The following assignment of status and mode bits in the S/M register is applicable to 1700 emulation.

BIT	Function	DESCRIPTION
100	Double Precision	Not included
101	One's Complement	See general specification
102	BG Input From N	See general specification
103	Adder Split	See general specification
104	Memory Parity Error	See general specification
105	Protect Fault	Set - Indicates 1700 protect fault occured
106	Active Interrupt System	Set - Activates sixteen 1700 interrupt levels.
107	Enable Decimal Arithmetic	Set - Enables decimal correction codes to selectors S1 and S2 via F1 input and enables decimal overflow test. (SM111 must be cleared).
108	Macro Halt	Set and then cleared (Strobed) via firmware when selective stop condition detected. Causes console to initiate a constant interrupt until a "GO" function is received from external source.
109	Micro Halt	See general specification.
110	Overflow	See general specification.
111	Enable F1	Set - Enables F1 to selectors S1 and S2. Disables decimal corrections codes and data inputs to S1 and S2.
112	Binary Overflow Test	Set - Monitors ALU output for binary overflow for setting SM110. (SM107 must be cleared).
113	Select XT/MA (R/WMM)	See general specification.
114	Pre-enable Interrupt System	Set - Activates interrupt system (sets SM106) on second GITMAK/XT micro instruction executed after setting pre-enable. Cleared by hardware. Utilized on 1700 EIN Macro.
115	Macro BKP Interrupt	Set via console control which initates interrupt. Must be cleared by 1700 Enhanced Macro after processing interrupt.
200	Auto-Data Transfer	See Auto Data Transfer section
201	Strobe/Read	See Auto Data Transfer section
202	Enable SPT, SSEL	See Auto Data Transfer section
203	Terminate	See Auto Data Transfer section
204	Autoload	See general specification
205	Open	
206	Open	

7.4.3 (Continued)

BIT	FUNCTION	DESCRIPTION
207	Select Page XT/MA	See general specification
208	Open	
209	Enable Fault Detection	Set on GITMAK/XT micro instruction. Must be cleared by firmware during interrupt processing routine prior to executing first 1700 interrupt subroutine macro.
210	Enable MM to MIR	See general specification
211	Enable DMA to MIR	See general specification
212	Enable 1700 Enhancements	Set - Provides special controls on transform module for enhanced format I.
213	Console Request Control	See general specification
214	Open	
215	Open	

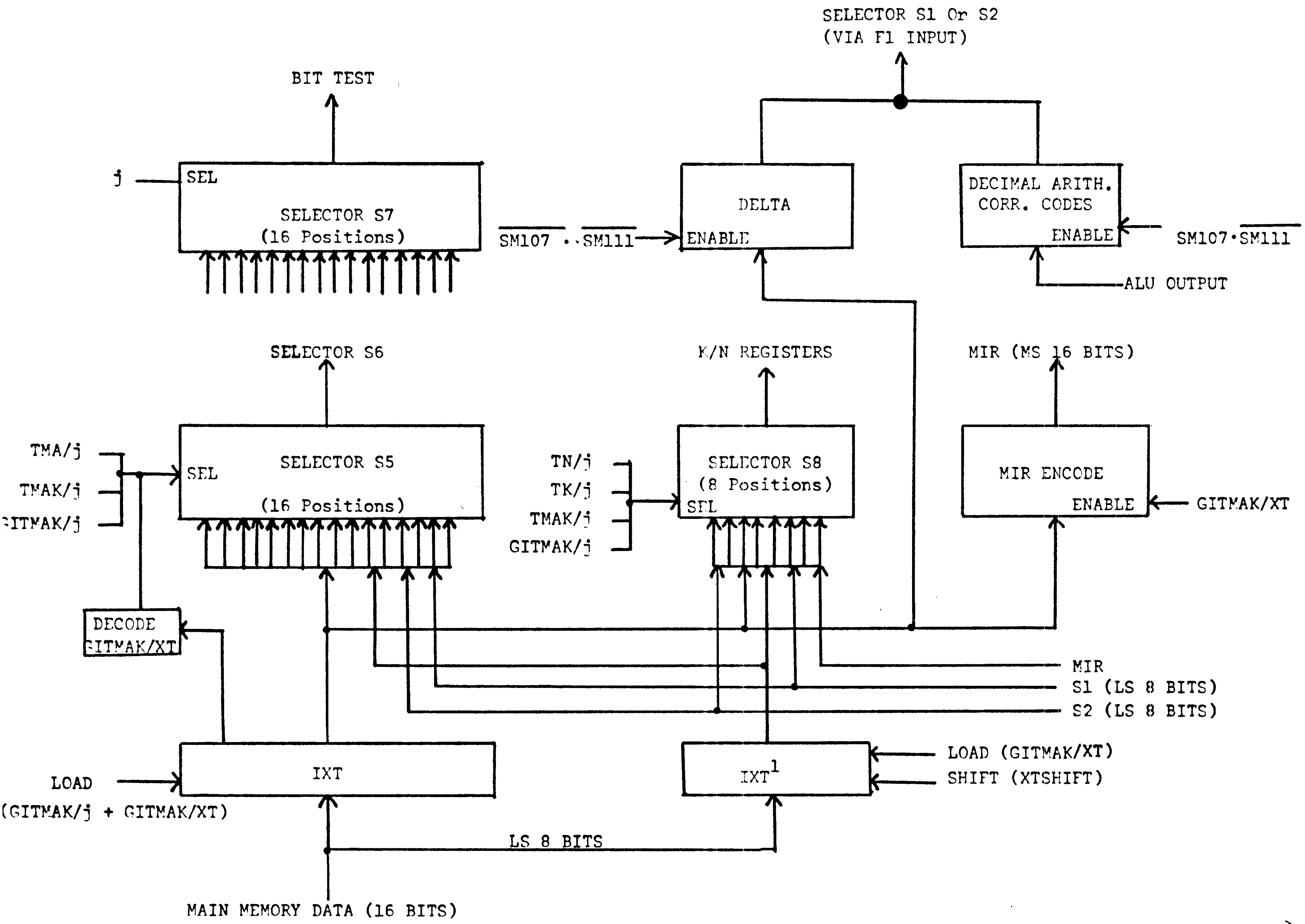
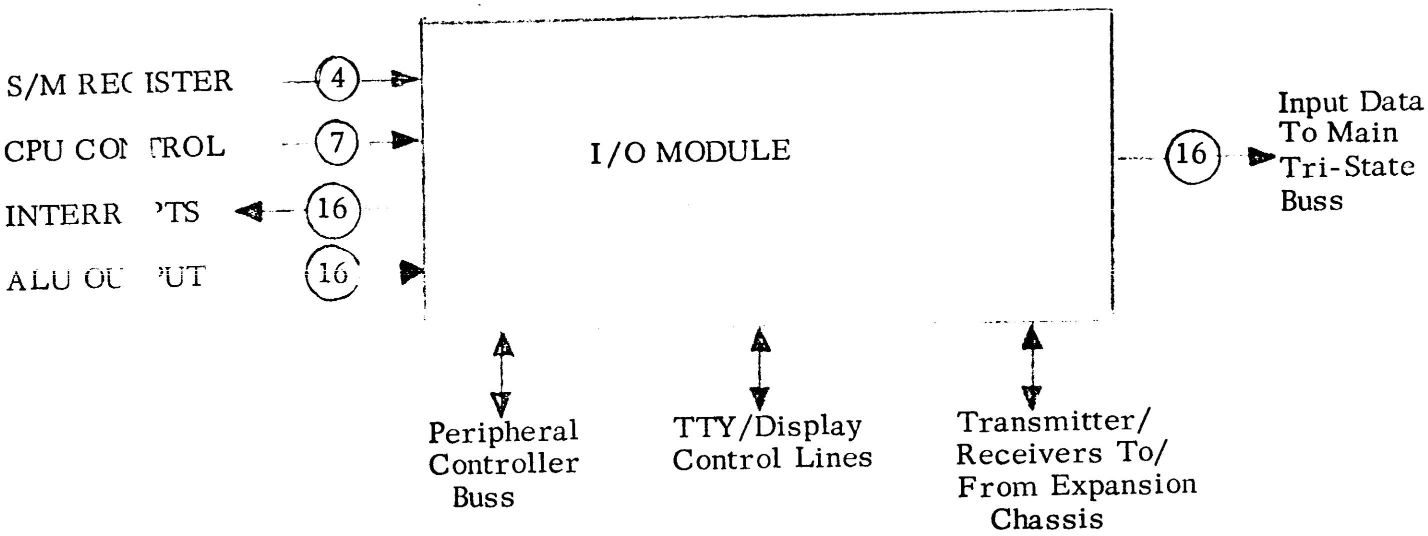


FIGURE 7-1: 1700 TRANSFORM MODULE BOLCK DIAGRAM

7.5 Internal I/O and Teletype/Display Controller:

This module, which is standard in the Mini and Maxi configurations and optional in peripheral controller configurations, occupies one card slot in the MP17 chassis. The following figure illustrates major signal flow paths to and from the I/O module.


S/M REGISTER
4
CPU CONTROL
7
INTERRUPTS
16
ALU OUTPUT
16
I/O MODULE
16
Input Data To Main Tri-State Buss
Peripheral Controller Buss
TTY/Display Control Lines
Transmitter/ Receivers To/ From Expansion Chassis

The following basic functions are provided by this module:

- Real Time Clock - This module contains a real time clock, which in conjunction with micro-code appears as a 1700 peripheral to the macro-level programmer. (Providing the MP17 is being utilized as a 1700 emulator.)

- I/O Teletype/Display Controller - Contained as an integral part of this module is an I/O teletype controller. The teletype to be used is the Teletype Corp. Models ASR/KDR 33/35. The display to be used is the Control Data Model 92423 conversational display terminal. When the MP17 is used as a 1700 emulator this controller will be compatible with CDC 1711 TTY Controller.

- Internal Peripheral Controller Buss - This module provides all I/O data lines, interrupts, and control signals necessary to generate, in conjunction with micro-code, both an internal CDC 1700 A/Q (Input/Output) buss and an NCR 605 (set/sample)I/O buss. This buss is TTL level and is intended to interface to controllers located in the basic MP17 chassis. Physically the 1700 AQ buss and the NCR 605 buss utilize the same data and control paths. Only the controlling micro-code, and the controller differ.

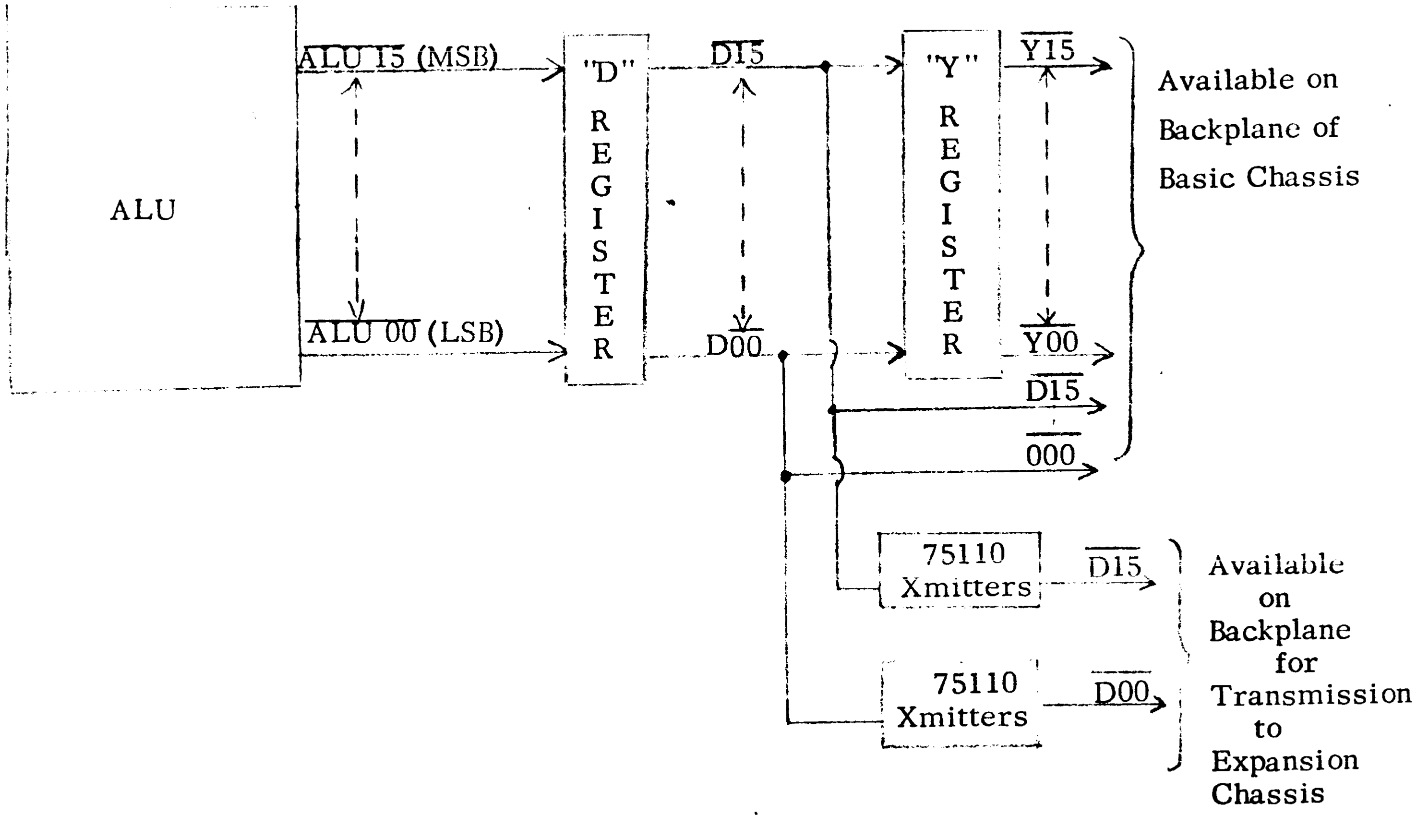
- Transmitter/Receiver to/from Expansion Chassis: Differential transmitter/receivers are provided on this module for connection to an I/O extender module in the expansion chassis. This I/O extender module will recreate the Internal Peripheral controller buss in the expansion chassis.

The CPU is interfaced to the I/O module in the following basic manner.

- ALU Output - All output data and address information is provided from the output of the ALU.
- S/M Register - All commands to peripheral controllers are generated by micro-code manipulation of the CPU status mode register.
- CPU Control - Timing and control information for controlling internal I/O module data gating is provided from CPU control signals.
- Interrupts - Although shown functionally as originating in the I/O module for transmission to the CPU interrupt circuitry, interrupts from peripheral controllers (within the basic chassis) actually are wired directly from the peripheral controller module to the CPU. Interrupts from the expansion chassis are received by the I/O module and then transmitted to the CPU.
- Input Data and Peripheral Response Signals - All of these are provided to the CPU on the main CPU tri-state buss.

7.5.1 Functional Operation: The following sub-sections describe the functional operation of this module.

7.5.1.1 Output Data and Peripheral Addresses: All output data and peripheral address information is obtained from the output of the ALU, and held in registers contained on the I/O module. The following diagram illustrates this signal flow.

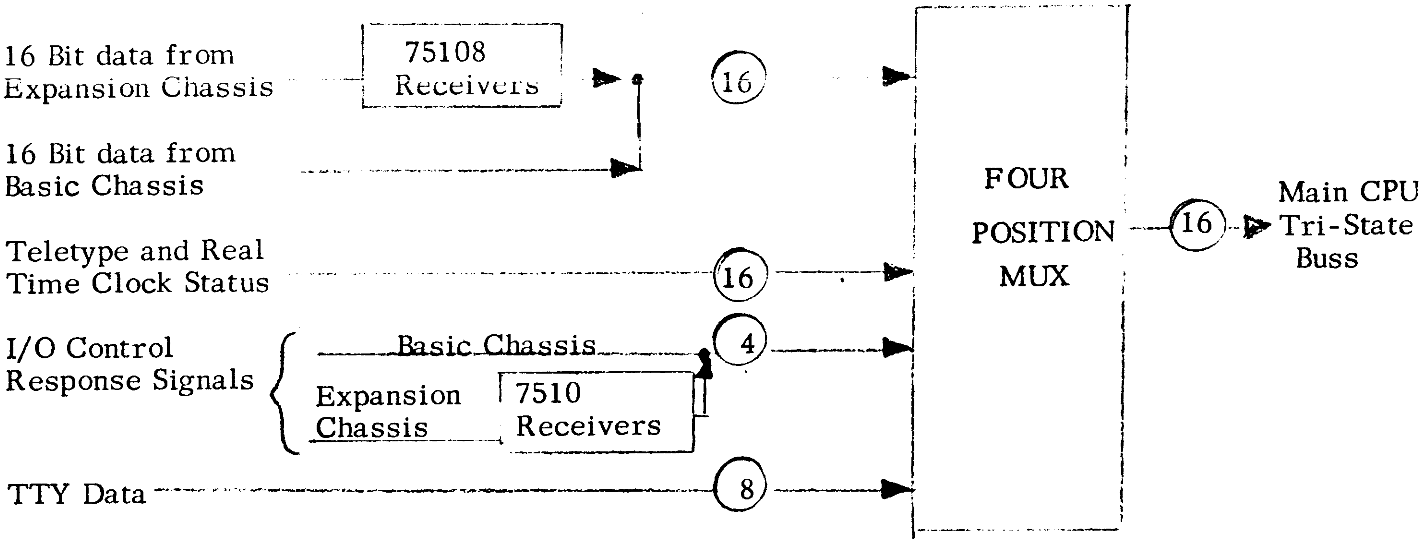


The loading of the "D" and "Y" registers is controlled by micro-code instructions. The "D" register is used as the output data register. The outputs of the "D" register are available at periphe al controller ports on the MP17 backplane. The outputs of the "D" register are also transmitted to the I/O buss extender in the expansion chassis through 75110 transmitters. Functionally, when operating as a 1700 emulator, the "D" register output provides the same data as the 1700 "A" register during I/O.

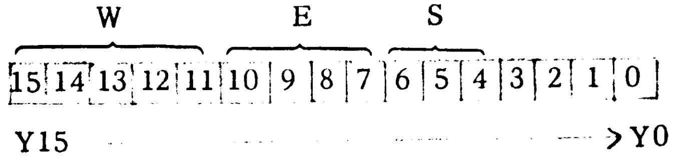
The "Y" register is used as the peripheral address register. The "Y" register outputs are not transmitted to the expansion chassis directly. However, the "Y" register data is loaded through the outputs of the "D" register and the buss extender loads its own "Y" register from the "D" register outputs at the same time "D" is transferred to "Y" in the basic chassis. When operating as a 1700 emulator the "Y" register functionally provides 1700 "Q" register information to peripherals.

The "D" register is loaded by execution of a micro-instruction with D' = 000 (IOD) or D' = 001 (IOA). In the case where D' = 001 (IOA) the data is immediately transferred from "D" to "Y". The loading of these registers is accomplished by a control signal from the CPU when D' = 000 or D' = 001. This signal is a 56 nano-second negative going pulse. The leading edge of this pulse strobes ALU data into the "D" register, and in the case where D' = 001 the trailing edge strobes the "D" register into "Y". This control signal is transmitted to the expansion chassis through a 75110 transmitter.

7.5.1.2 Input Data and I/O control response signals: All input data is provided to the CPU via the main Tri-state buss. The following Figure illustrates the basic signal flow


16 Bit data from Expansion Chassis
75108 Receivers
16
16 Bit data from Basic Chassis
Teletype and Real Time Clock Status
16
I/O Control Response Signals
Basic Chassis
Expansion Chassis
7510 Receivers
4
TTY Data
8
FOUR POSITION MUX
16
Main CPU Tri-State Buss

Data from the expansion chassis is received via 75108 open collector receivers. Data from the basic chassis is received via the open collector buss. Data to be gated onto the Tri-state buss is selected by a combination of the B' field of a micro-instruction and the contents of the "Y" register at the time the micro-instruction is executed. The "Y" register for this selection is decoded accodring to the 1700 W-E-S convention. This convention follows:



Selection of the four I/O response control signals is accomplished by execution of a micro-instruction with B'=011 {INRS} without regard to the contents of "Y". The data gated to the tri-state buss will have the following format:

Description		Tri-state Buss
605 Peripheral	1700 Peripheral	Bit Position
Undefined	Undefined	00 & 01
Position 00	Undefined	02
Position 01	Reply	03
Position 02	Reject	04
Direction	Character Mode	05
Undefined	Undefined	06 - 15

To select the other three groups of signals it is necessary to execute a micro-instruction with B'=010 {INRD}. The data actually gated to the tri-state buss when this is done is determined according to the following table.

W	E	S	Yo	DATA GATED TO TRI-STATE
0	1	1	0	TTY DATA
0	1	1	1	TTY STATUS
0	1	7	-	REAL TIME CLOCK STATUS
-	-	-	-	DATA FROM PERIPHERAL CONTROLLERS

7.5.1.3 Interrupts: When used as a 1700 emulator, the firmware treats each of the two groups of interrupts in a different manner. The sixteen lower priority interrupts {Program interrupts} are utilized to generate the standard sixteen 1700 macro interrupts. The 16 higher priority interrupts {Data Interrupts} are used as micro-level interrupts and are transparent at the macro-level.

7.5.1.4 Micro-Programmable I/O Control Signals:

Four status mode signals are used to generate control signals to the I/O ports. 75110 transmitters are provided for transmitting these four signals to the expansion chassis.The specific status/mode bits utilized and their functions are defined in the following table.

	FUNCTION	
S/M Bit	NCR 605 Peripherals	CDC 1700 Peripherals
To be defined	Auto-Data xfer	Auto-Data xfer
"	Strobe	Read
"	Enable SPT, SSEL	Write
"	Terminate	Terminate

These signals are controllable directly by micro-code as with any other S/M register bits. The 1700 emulator uses these to generate 1700 Input from A, and Output from A instructions. It also uses them to generate NCR 605 Set/Sample instructions.

1 Other CPU control signals: The following control signals are used internally by the I/O module. These signals originate in the CPU and are indirectly controllable by micro-code. ie. They occur because of CPU conditions.

MASTER CLEAR - This signal originates in the CPU. It is transmitted to the I/O expansion chassis through a 75110 transmitter.

PROGRAM PROTECT - This signal orginates in the CPU control section, and indicates the processor protect status. It is transmitted to the expansion chassis through a 75110 transmitter.

MIR 12 - This signal originates as bit 12 of the memory instruction register, and is used to select control signals on the Tri-state buss.

TRI-STATE ENABLE - A negative going enable to turn on the tri-state buss. This is equivalent to the S4 enable in the MPP.

Fifty-six nano-sec positive going pulse. This pulse is used to re-sync the response signals with the CPU timing by strobing the signals into a 74175 D-type latch. The timing of this signal must be such that the output of the latch is stable when it is selected as the input to the Tri-state multiplexer.

In addition to the above signals, a K/N transform is required as follows:

Bit 0 (LSB)	-	0
Bit 1	-	0
Bits 2,3,4,5	-	True Interupt Address
Bits 6 & 7	-	Bias, to be determined

7.5.2 TTY/Display Controller

The TTY/display controller is compatible with CDC 1711 TTY controller. The logic features of the I/O TTY/ display controler are as follows. The I/O TTY to be used with this controller is the Teletype Corp. Models ASR/KDR 33/35 {refer to Teletype Cor. Bulletin 273B for details}. The display to be used is the Control Data Model 92423 Conversational Display Terminal {refer to CDC Specification Number 62018200 for details}.

The TTY is controlled via three wires. The CDT is controlled via six wires. Two additional wires are provided for an optional connection to the manual interrupt line.

The following subsection makes reference to 1700 micro-level conventions.

7.5.2.1 Device Interface

1. Communication with the TTY is accomplished via three wires - a receiver wire, a transmitter wire, and two common wires {shorted at the controller}. Interface is 20mA current loop.

2. Communication with the CDT is accomplished via six wires - receiver, transmitter, signal ground, request to send, clear to send and protective ground. Signals conform with EIA standard RS232-C and CCITT recommendation V24.

Data transmission is serial, asynchronous. Start/stop format USASCII Code - 7 are data bits and 1 is parity.

1. The TTY receives and transmits data at a maximum speed of 110 bits per second {baud}.

2. The CDT operates at one of four speeds installation selected: 110, 300, 1200, or 9600 baud.

7.5.2.2 Programming

When referencing the TTY/CDT devices, Q-register should contain either 0090_{16} or 0091_{16} according to the following table:

	Computer Instruction	
Q-Register	Output From A	Input to A
0090	Write	Read
0091	Director Function	Director Status

7.5.2.2.1 Director Function

Bit {A Register}	Function	Operation
00	Clear Controller	Clear all interrupt requests. Clear Busy, Interrupt, Data, Alarm, Manual Interrupt conditions. Select Read Mode. Connect Printer. Any interrupt request bit shall take precedence over this function.
01	Clear Interrupt	Clears all interrupt requests and the manual interrupt. Any interrupt request bit shall take precedence over this function.
02	Data Interrupt Request	Conditions the controller to send an interrupt signal whenever a Data Status is active.
03	End of Operation Interrupt Request	Conditions the controller to send an interrupt signal when the controller is not busy. When in the EOP state the controller will accept a change in its mode.

Bit {A Register}	Function	Operation
04	Alarm Interrupt Request	Conditions the controller to send an interrupt signal when the Lost Data Status is active.
05	Not Used	
06	Not Used	
07	Not Used	
08	Select Write Mode	Conditions the controller for an output operation. Clears the Alarm Status.
09	Select Read Mode	Conditions the controller for an input operation.
10	Connect Printer	Select a mode of operation in which the printer {with the paper tape punch, when exists} and the Tape Reader {when exists} are both connected to the controller. Data read from paper tape in this mode will also be printed {and punched}.
11	Not Used	
12	Not Used	
13	Disconnect Printer	Select a mode of operation in which the printer {and paper tape punch when exists} is disconnected from the controller. Data read from paper tape are not printed {nor punched}. This mode allows transmission of other than ASCII codes to the computer, including binary information.
14	Not Used	
15	Not Used	To be specified.

All nonconflicting functions may be performed simultaneously.

Select Write mode and Select Read mode are rejected when controller is busy. Other functions are always performed. When several functions are issued simultaneously and some of them can be performed, the output from A instruction exits normally {Reply} but those function which should be rejected are not performed. When none of the functions can be performed, the Output from A instruction is rejected.

7.5.2.2.2 Director Status

Bit {A Register}	Status	Description
00	Ready	Unit is ready.
01	Busy	Read Mode - The controller is in the process of receiving a character from the TTY/CDT or the holding register contains data for transfer to the computer. The Busy status will drop upon completion of the data transfer. Write Mode - The data register contains data and is in the process of transferring it to the TTY/CDT. Busy will drop upon completion of the transfer.
02	Interrupt	An interrupt condition exists in the controller.
03	Data	Read Mode - The holding register contains data for transfer to the computer. The data status will drop upon completion of the transfer. Write Mode - The controller is ready to accept another character from the computer.
04		Always the inverse of Busy.

Bit {A register}	Status	Description
05	Alarm	Parity error or lost data or field error {no stop bit when expected} occurred.
06	Lost Data	The holding register contained data for transfer to the computer and the TTY/CDT began to send a new sequence.
07	Parity Error	A parity error occurred.
08	Not Used	Always ▿0▿.
09	Read Mode	The controller is conditioned for input operation.
10		Always same as ready.
11	Manual Interrupt	A manual interrupt has occurred.
12	Not Used	Always ▿0▿.
13	Not Used	Always ▿0▿.
14	Not Used	Always ▿0▿.
15	Not Used	Always ▿0▿.

7.5.3 Real Time Clock: The real time clock which is an integral part of the I/O module is designed to appear as a 1700 peripheral to the macro-level software. Available to the macro-level program are two functions: Enable Limit Interrupt and Disable Limit Interrupt. Also available to the macro-level program are two status bits, Limit Interrupt and Lost Count.

Limit Interrupt is selected by performing a write to the real time clock ("W"=0, "E"=1, and "S"=7) and the least significant bit of the "Q" register equal to one. The limit interrupt is cleared in the same manner with the least significant bit of the "Q" register equal to zero. Either case of write will clear an existing limit interrupt and clear the status of the real time clock.

Status of the real time clock is obtained by an input from the real time clock ("W"=0, "E"=1, and "S"=7). Status is returned in the "A" register with bit 15 when true indicating a lost count, and bit 14 when true indicating a limit interrupt. The rest of the bits in the "A" register are undefined.

The limit interrupt being received by a macro-level program is dependant on the appropriate "M" register bit being set and the macro interrupt enabled as with all other 1700 peripherals.

Micro-level code is capable of receiving a data interrupt from the real-time clock every 3-1/3 milli. This interrupt is enabled anytime that the corresponding mask bit is set.

The determination of when the micro-code generates the macro-level interrupt is dependant on the emulator. To generate the limit interruptit is only necessary for the micro-instruction to set S/M bit "Terminate" before clearing the data interrupt. If this is done the data interrupt will be set providing the limit interrupt has been enabled by the micro-code.

The data interrupt is cleared by setting S/M bits "Auto-Data" and "Strobe" bits with the I/O "Y" register selecting "W"=0, "E"=1, and "S"=7. In the case where it is desired to clear the data interrupt without generating the "Limit" interrupt it is necessary to clear S/M bit "Terminate" before clearing the data interrupt.

If the data interrupt has not been cleared before another 3-1/3 milli. count occurs (and the limit interrupt has been selected) the lost count status bit will be set. This will cause a limit interrupt to occur with the lost count status bit set.

The real time clock is always "ready" and reads or writes are never rejected.

'.5.4 I/O Emulation: The I/O control module circuitry, in conjunction with 1700 emulator, will be used to generate the following macro-level I/O capabilities.

CDC 1700 A/Q (Inp, Out)

NCR 605 (Set/Sample)

NCR 605 Auto-Data Transfer

CDC 1700 Auto-Data Transfer

With respect to the first three items the I/O signals generated, and received will be in accordance with the I/O specifications for their respective I/O schemes. The fourth item will be quite similar to NCR 605 Auto Data Transfer, but will be used with 1700 type peripherals.

The following points should be noted with respect to these I/O capabilities.

1) CDC peripherals will always be assigned equipment numbers 0-7. NCR peripherals will always be assigned equipment numbers 8-15.

2) The two types of I/O structures provided (NCR 605 and CDC 1700) in most cases use the same data, and control paths between the I/O module and the peripheral controller ports. The difference between the two I/O busses is basically in the micro-program manipulating the control signals.

3) The buss is TTL level, and specifically does not employ "3000" type party line receivers/transmitters.

4) The instruction for generating NCR 605 I/O commands is not defined at the present time. The mnemonic will be referred to as set/sample for the I/O operation.

The following sub-sections define the I/O specifications for the above mentioned I/O capabilities.

7.5.4.1 Internal TTL CDC 1700 A/Q I/O

7.5.4.1.1 Read

The read signal signifies the request for an input operation. If the data is available at the time the read signal rises, a reply is returned within four microseconds; if data is not available at the time the read signal rises, a reject signal is returned within four microseconds.

7.5.4.1.2 Write

The write signal signifies the request for an output operation. If data can be used at the time the write signal rises, a reply is returned within four microseconds; if data cannot be used at the time the write signal rises, a reject signal will be returned within four microseconds.

7.5.4.1.3 Reply

Reply to Write

If the peripheral equipment can accept data when the write signal rises, the following sequence of events occurs:

1) The computer channel transfers data to the appropriate register in the peripheral equipment.

2) The peripheral equipment sends a reply to the channel a minimum of 200 nanoseconds and a maximum of four microseconds later.

3) The channel drops the write signal when it receives the reply.

4) Absence of a write signal for 100 nanoseconds drops the reply.

5) The data lines drop when the reply drops.

Reply to Read

If data is available when the read signal rises, the following sequence of events occurs:

1) The data available is gated to the data cable.

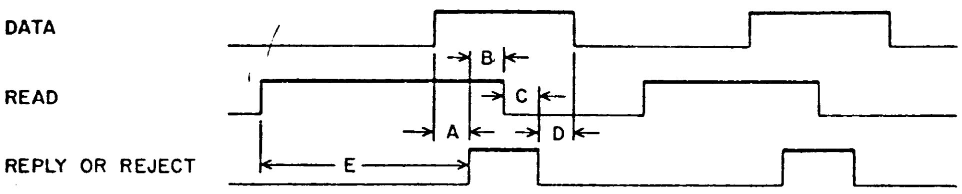
2) The reply is returned a minimum of 200 nanoseconds and a maximum of four microseconds later.

3) The reply causes the read line to drop.

4) Absence of a read signal for 100 nanoseconds causes the reply to drop.

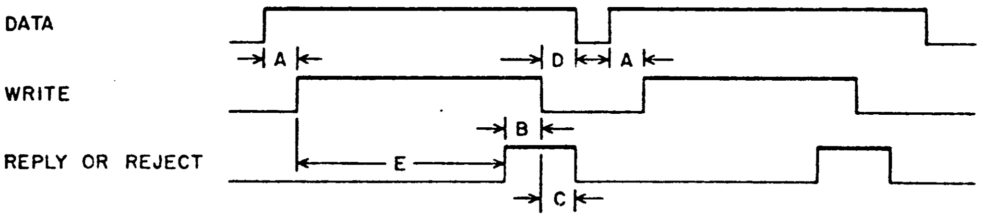
5) The data lines drop when the reply drops. See Figure 6 for timing information.

INPUT OPERATION



A = 0 μSEC MIN @ PERIPHERAL DEVICE	D = 0 μSEC MIN @ PERIPHERAL DEVICE
B = 0.0 μSEC MIN @ COMPUTER	E = 4.0 μSEC MAX @ PERIPHERAL DEVICE
C = 0.0 μSEC MIN @ PERIPHERAL DEVICE	0.2 μSEC MIN @ PERIPHERAL DEVICE

OUTPUT OPERATION



A = 0.1 μSEC MIN @ COMPUTER	D = 0.1 μSEC MIN @ COMPUTER
B = 0.0 μSEC MIN @ COMPUTER	E = 4.0 μSEC MAX @ PERIPHERAL DEVICE
C = 0.0 μSEC MIN @ PERIPERAL DEVICE	0.2 μ SEC MIN @ PERIPHERAL DEVICE

NOTE:
THE ADDRESS BITS WILL BE ON THE CHANNEL A MINIMUM OF 0.1 μSEC BEFORE AND AFTER THE READ OR WRITE SIGNAL.

Figure 6. AQ I/O Timing

7.5.4.1.4 Reject

If the specified operation cannot or should not be performed at the time a read or write signal appears, a reject will be returned within four microseconds.

7.5.4.1.5 Program Protect

The program protect signal is present if the I/O instruction requires access to a protected device. If the signal is not present, the protected device returns a reject signal.

7.5.4.1.6 Character Input

This signal is generated by the peripheral device if the data transfer is an 8-bit character or less in the low-order bit positions. Devices which never exceed an 8-bit transfer may have this line up continuously while reading.

7.5.4.1.7 Continue Bit

Bit 15 of Q is a continue bit and is used to speed the operation devices which require continuous random addressing. Such a device operates as follows:

1) Address the device with Q15=0 and the remainder of Q set to select this device. The device is now connected.

2) All succeeding addresses with Q15=1 will be recognized by this device. Thus 15 bits of address are available to this device.

3) The next address with Q15=0 will disconnect this device unless it is the address of this device.

NOTE: This type of addressing is not allowed if 1700 Auto-Data Transfer is utilized.

7.5.4.1.8 The Q register in the 1700 computer is used to send addressing codes to peripheral equipments. The format of the Q register is shown below. Each level of peripheral equipment except a unit is addressed by a unique section of the Q register.

15 11	10 7	6 0
W	E	Command

Q Register

7.5.4.1.9 The following table 7 defines the signals available at the backpanel at each port in the MP17. This table includes common signals used both by the NCR 605 and CDC 1700 peripherals, and special terms dedicated to only one type. The table headings have the following meanings.

MNEMONIC - Signal name at the port. In cases where the signal is used by both 1700 peripherals and 605 peripherals the mnemonic corresponds to NCR nomenclature.
NAME - Defines the usage of the signal

PIN NO. - Backplane pin at the port where the signal is available.

TYPE - Specifies use by NCR 605, 1700, or both.

PATH - Foil indicates backplane printed circuit connection. Wire indicates wire wrap connection.

The following should be noted with respect to 1700 usage. All of the signals listed exclusively for 1700 usage must be custom wired if they are to be used.

TABLE 7

STANDARD I/O INTERFACE LINES

MNEMONIC	NAME	PIN NO	TYPE	PATH
	INPUT			
RD 01	Data Input 01 (LSB)	204	Both	Foil
RD 02	02	206		
RD 03	03	208		
RD 04	04	210		
RD 05	05	214		
RD 06	06	218		
RD 07	07	221		
RD 08	08	224		
RD 09	09	226		
RD 10	10	229		
RD 11	11	231		
RD 12	12	234		
RD 13	13	236		
RD 14	14	239		
RD 15	15	241		
RD 16	Data Input 16 (MSB)	244		Foil
RD INT 01	Data Interrupt 01	250		Wire
02	02	250		
03	03	250		
04	04	250		
05	05	250		
06	06	250		
07	07	250		
RD INT 08	Data Interrupt 08	250		
RD IR OUT	Data Direction Out/Char Mode	47		
RP INT 01	Program Interrupt 01	249		
02	02	249		
03	03	249		
04	04	249		
05	05	249		
06	06	249		
07	07	249		
RP INT 08	Program Interrupt 08	249	Both	Wire
EXSTOP	External Macro Stop	293	Both	Wire
EXMC	External Master Clear	294	Both	Wire
EXGO	External Macro Go	295	Both	Wire

TABLE 7

STANDARD I/O INTERFACE LINES (Cont'd)

MNEMONIC	NAME	PIN NO	TYPE	PAT
	INPUT			
$\overline{\text{RPOS 01}}$	Position Address 01	213	M05	Foil
$\overline{\text{02}}$	Position Address 02/Reply	216	Both	Foil
$\overline{\text{03}}$	Position Address 03/Reject	220	Both	Foil
$\overline{\text{RINT 1}}$	Optional Int #1	49	1700	Wire
$\overline{\text{RINT 2}}$	Optional Int #2	50	1700	Wire
	OUTPUT			
$\overline{\text{SD 01}}$	Data Output 01 (LSB)	203	Both	Foil
$\overline{\text{02}}$	↓ 02	205	↓	↓
$\overline{\text{03}}$	03	207		
$\overline{\text{04}}$	04	209		
$\overline{\text{05}}$	05	211		
$\overline{\text{06}}$	06	215		
$\overline{\text{07}}$	07	219		
$\overline{\text{08}}$	08	223		
$\overline{\text{09}}$	09	225		
$\overline{\text{10}}$	10	228		
$\overline{\text{11}}$	11	230		
$\overline{\text{12}}$	12	233		
$\overline{\text{13}}$	13	235		
$\overline{\text{14}}$	14	238		
$\overline{\text{15}}$	15	240		
$\overline{\text{SD 16}}$	Data Output 16 (MSB)	243		
$\overline{\text{SMB7}}$	Mode Bit Line 1/Y01	247		
$\overline{\text{8}}$	Mode Bit Line 2/Y02	246		
$\overline{\text{9}}$	Set/Sample Mode Bit/Y03	245		
$\overline{\text{SPOS 01}}$	Position Address Line 1/Y04	212		
$\overline{\text{02}}$	Position Address Line 2/Y05	217		
$\overline{\text{03}}$	Position Address Line 3/Y06	222	Both	Foil
$\overline{\text{SPT 01}}$	Port Address Line 01	227	M05	Wire
$\overline{\text{02}}$	↓ 02	227	↓	↓
$\overline{\text{03}}$	03	227		
$\overline{\text{04}}$	04	227		
$\overline{\text{05}}$	05	227		
$\overline{\text{06}}$	06	227		
$\overline{\text{07}}$	07	227		
$\overline{\text{SPT 08}}$	Port Address Line 08	227	M05	Wire

TABLE 7

STANDARD I/O INTERFACE LINES (Cont'd)

MNEMONIC	NAME	PIN NO	TYPE	PATH
	OUTPUT			
$\overline{\text{SSEL 01}}$	Select 01	251	M05	Wire
02	02	251	↓	↓
03	03	251	↓	↓
04	04	251	↓	↓
05	05	251	↓	↓
06	06	251	↓	↓
07	07	251	↓	↓
$\overline{\text{SSEL 08}}$	Select 08	251	M05	Wire
$\overline{\text{SSTB}}$	Strobe/Read	248	Both	Foil
$\overline{\text{STERM}}$	Terminate	48	M05	Foil
$\overline{\text{MR}}$	Master Reset	46	Both	Wire
$\overline{\text{Y00}}$	Address Bit 00 (LSB)	280	1700	↓
$\overline{\text{Y07}}$	↓ 07	281	↓	↓
$\overline{\text{Y08}}$	↓ 08	282	↓	↓
$\overline{\text{Y09}}$	↓ 09	283	↓	↓
$\overline{\text{Y10}}$	↓ 10	284	↓	↓
$\overline{\text{Y11}}$	↓ 11	285	↓	↓
$\overline{\text{Y12}}$	↓ 12	286	↓	↓
$\overline{\text{Y13}}$	↓ 13	287	↓	↓
$\overline{\text{Y14}}$	↓ 14	288	↓	↓
$\overline{\text{Y15}}$	Address Bit 15 (MSB)	289	↓	↓
$\overline{\text{WRITE}}$	Write	290	↓	↓
$\overline{\text{PROG PROT}}$	Program Protect	291	↓	↓
$\overline{\text{WEO}}$	W=0	292	1700	Wire

7.5.4.2 Internal TTL NCR MOS Set/Sample

This section makes reference to signals listed in Table 7.

The MOS-I/O structure, provides 8 port addressing, and an 8-way priority scheme. Each port is numbered from 0 to 7, but designated 1 to 8. Port 7{8} has the lowest priority. Each port can communicate with a peripheral directly, or each port may optionally be further multiplexed to communicate with up to 8 peripherals. Thus, up to 64 peripherals can be controlled. The second level of 8 operates on a priority or scanning scheme which must be provided by some external means.

Information is transferred to or from the device via the I/O Set or Sample Command. The I/O Command causes 16 bits to be input to the processor or 16 bits to be transferred to the device called out in the command. The direction of transfer is implied to the interface by the address bit Y03 {See Table 8}.

When address bit Y03 is a "1", that is, a SET condition, the 16 bits of information contained in the output data register are placed on the output lines. Hence, during a SAMPLE condition, Y03 is a "0" and the 16 bits of information on the Input Lines are transferred to the Tri-state buss. When a position number is required by a port device, Y04 should be used if two positions or less are required, Y04 and Y05 if 4 or less positions are required and bits Y04, Y05 and Y06 if 8 positions are required.

Executing a set or sample command outputs a discrete signal to the port selected, {SPT 0-7}, the three position bit lines {SPOS 1-3}, the three mode bit lines {SMB 7-9} and a strobe {SSTB}. It either outputs {simultaneously} on the 16 Data Output Lines {SD1-16} or accepts the data placed {by the port device} on the 16 Data Input Lines {RD1-16}.

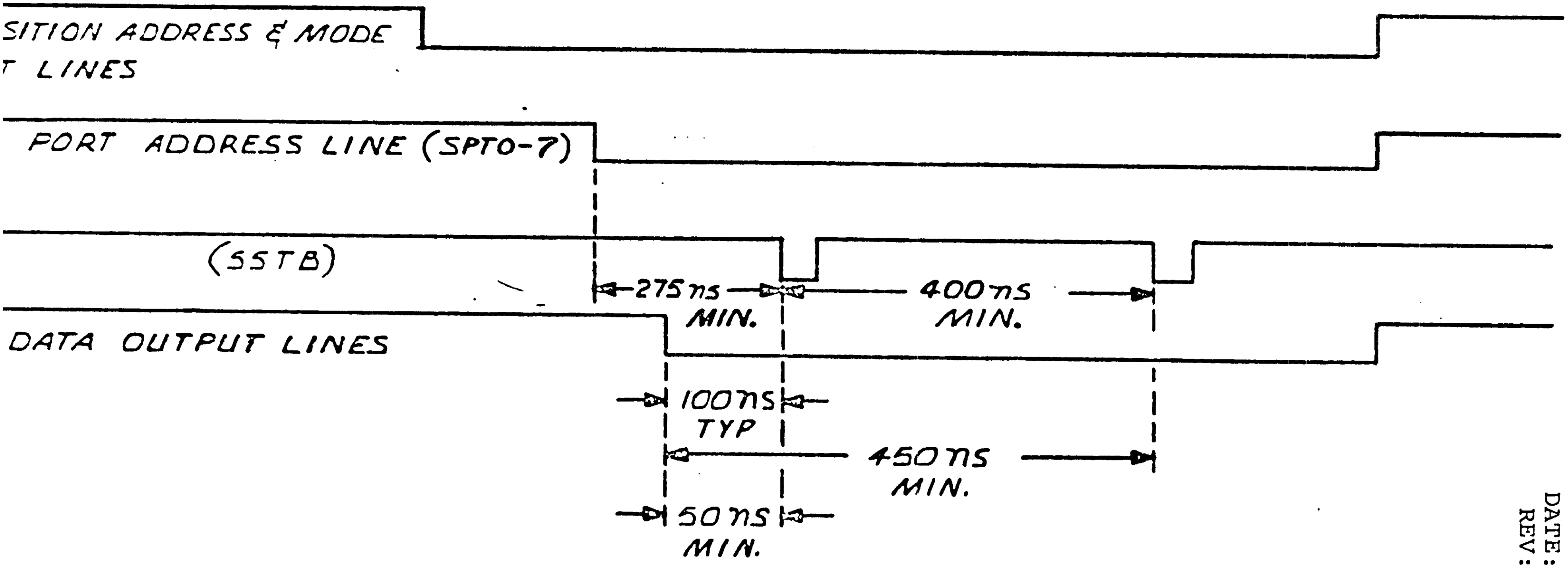
In order to insure sufficient time for the data to stabilize at a port device, the processor outputs the data for approximately 600ns. The strobe occurs near the end of this time. There are actually two strobes sent, one near the beginning of data and another near the end. Both will be sent during data transmission. Two strobes are sent to simplify control for some port devices.

Devices outputting data to the processor should utilize the port signal {SPT 0-7} to gate their data on the buss thru open collector gates.

Devices accepting data should only accept data if their port signal is true and at strobe time.

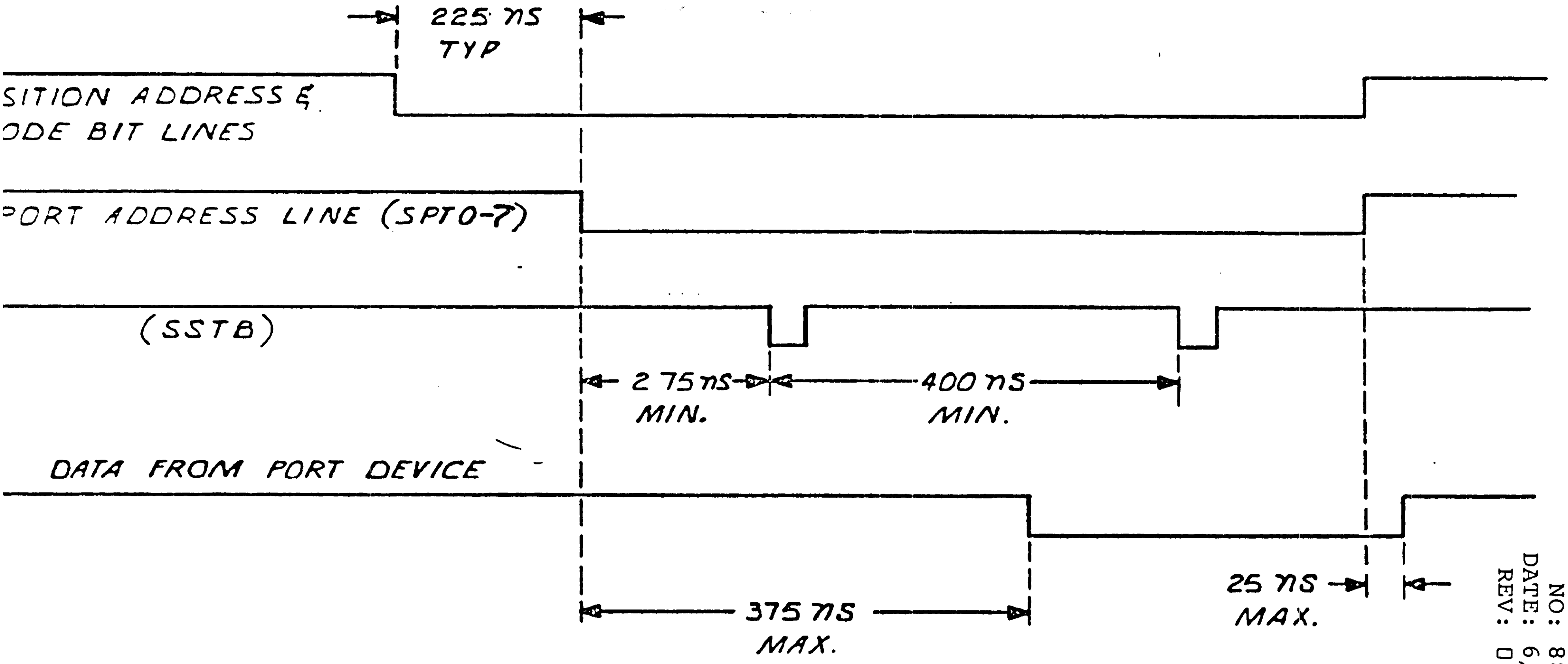
The "PROGRAM" Interrupt will cause the processor to take a program branch to a location in memory as defined later. This mode is for control of peripherals. It allows peripheral occurrences, such as initializing operations terminations, malfunctions or errors. The NCR 605 Program interrupt will be treated at the macro-level as a 1700 interrupt. The two following timing diagrams illustrate the operation of set/sample.

605-1 SET COMMAND TIMING



NO:88786800
DATE:6/13/73
REV:01

605-1 SAMPLE COMMAND TIMING



NO: 88786800
DATE: 6/13/73
REV: 01

7.5.4.3 NCR 605 and CDC 1700 Auto Data Transfer Functional Operation:

An auto-data transfer is initiated by a normal I/O operation. The exact method depends on the specific peripheral. After it is initiated, the peripheral flags the CPU with a data interrupt each time it wants to transfer a word to or from main memory. The data interrupt is transparent to the macro-code, but it does "steal" some time from it.

When the data interrupt is recognized by the CPU, the micro-code executes a K transform which directs it to a four-word block in file 1. The four-word block is broken down as follows:

Word 1	Address (a la the Q register)
Word 2	First word address
Word 3	Last word address + 1
Word 4	Relocation word

If the peripheral is M05 with multi-positions (as defined by Word 1), then the format is as follows:

Word 1	Address
Word 2	Indirect address to main memory
Word 3	Not used
Word 4	Relocation word

The address word is defined in Table 8. The first word address and the last word address + 1 define the locations in main memory from/to which the data transfer takes place. The first word address is incremented by the micro-code as the transfer progresses. The re-[illegible]on word is used on systems with more than 65K of main memory [illegible]ure that a change of the relocation word {used to address lo-[illegible]s above 65K} will not affect auto-data transfers already in progress.

	INP, OUT, SET/SAMPLE	AUTO-DATA XFER {Contents of Word 1}
Y15{MSB}	W	

Y14		▿0▿=INPUT{1700}
		▿1▿=OUTPUT{1700}

Y13		▿0▿=SINGLE POSITION PORT ▿1▿=MULTI-POSITION PORT
[illegible]		
[illegible]		
Y10	Always ▿0▿ for 1700 Always ▿1▿ for M05	Always ▿0▿ for 1700 Always ▿1▿ for M05
Y09		

Y08	Equipment{1700}
	Port{M05}
		Equipment{1700}
		SSEL{Interrupt Select}{M05}

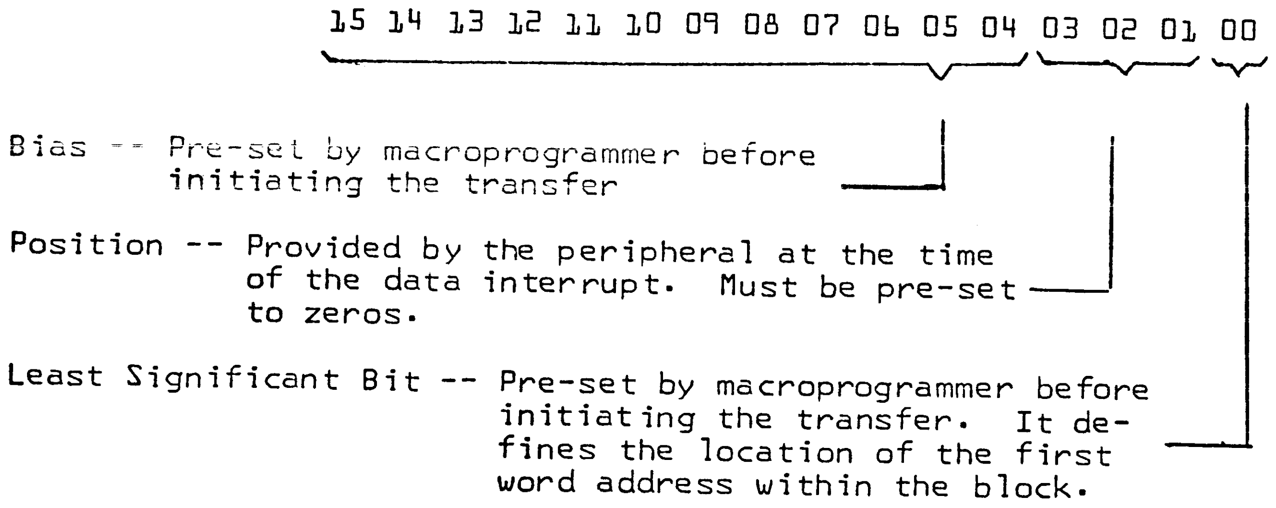
Y07		
Y06		

Y05	Station{1700}
	Position{M05}
		Station{1700}

Y04		
Y03	▿1▿=SET ▿0▿=SAMPLE	
Y02		
Y01	Director{1700} Mode{M05}	
Y00{LSB}		

TABLE 8

The first word address and the last word address + 1 are located in main memory in the case of multi-position M05 peripherals. The address in main memory of these words is determined by word 2 of the four-word block in file 1, as follows:



Bias -- Pre-set by macroprogrammer before initiating the transfer

Position -- Provided by the peripheral at the time of the data interrupt. Must be pre-set to zeros.

Least Significant Bit -- Pre-set by macroprogrammer before initiating the transfer. It defines the location of the first word address within the block.

Note that there is only one relocation word for all positions on a port, and this word is used for both indirect addressing and data transfers.

7.6 MP17 Configurations - This section illustrates the basic system configurations of the MP17. It is divided into three functional sub-sections. These subsections cover the MINI, MAXI, and programmable peripheral controller versions of the MP17.

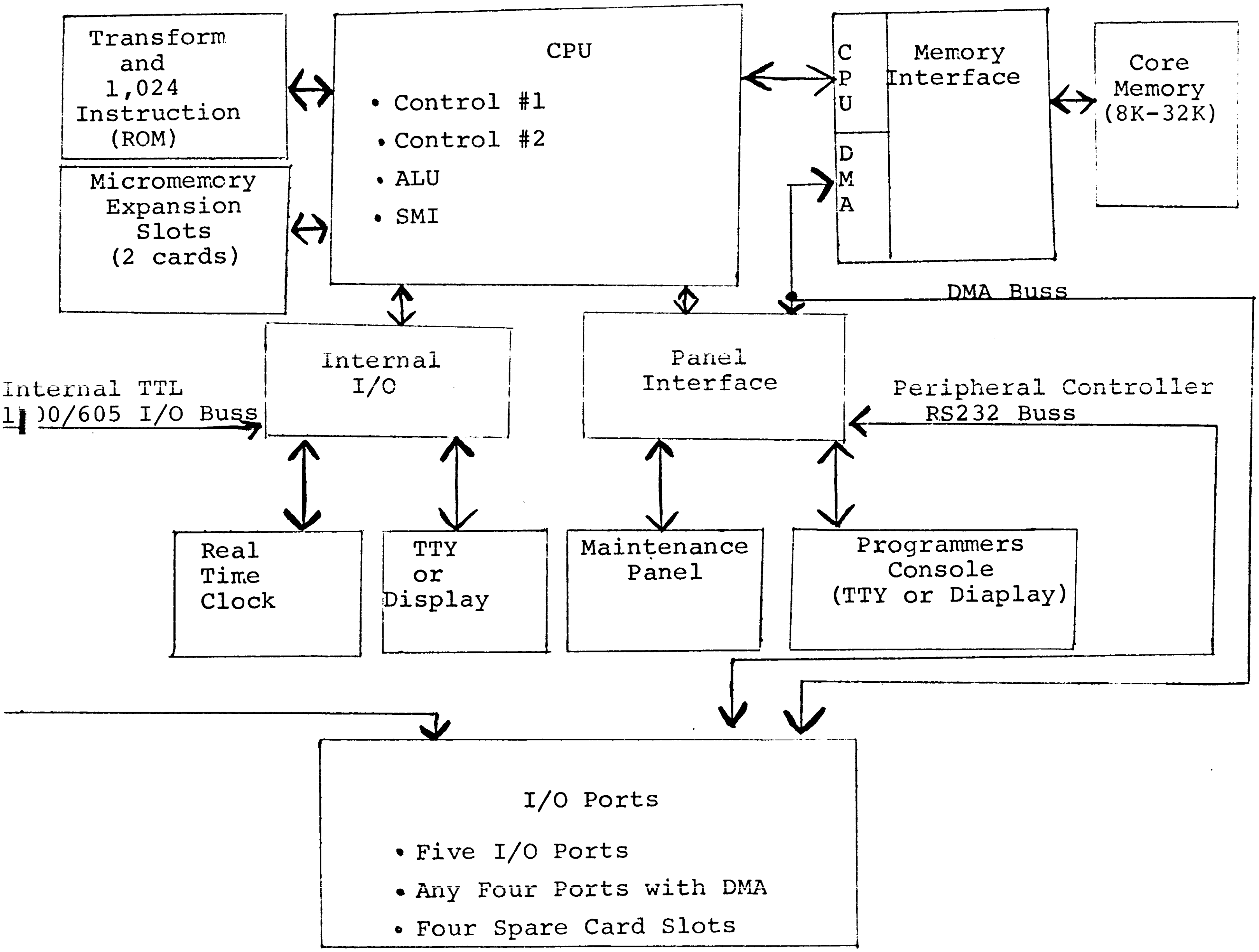
7.6.1 MINI Configuration. The MINI configuration of the MP17 is contained in one MP17 chassis module. This basic chassis provides 27 card slots. The utilization of these card slots is illustrated in the following figure.

8K CORE MEMORY	8K CORE MEMORY	8K CORE MEMORY	8K CORE MEMORY	MEMORY INTF CPU & DMA	PANEL INTERFACE	MICRO MEMORY EXPAN	MICRO MEMORY EXPAN	TRANSFORM & MICROMEMORY	CONTROL #1	CONTROL #2	ALU 16 BIT	SMI 32 BIT	INTERNAL I/O & TTY	I/O PORT	SPARE	I/O PORT	I/O PORT	SPARE	I/O PORT	SPARE	I/O PORT	SPARE
*	*	*												*	*	*	*	*	*	*	*	*
Z	Y	X	W	V	U	T	S	R	P	N	M	L	K	J	H	G	F	E	D	C	B	A

(Front View MP17 MINI Configuration)

*Denotes optional card.

The MINI configuration is generally utilized as a 1700 emulator. However it may be used as a general purpose emulator of other small computers. The following block diagram illustrates the system capabilities of the MINI.



MP17 Functional Block Diagram MINI Configuration

7.6.1.1 I/O Ports: Five I/O ports are provided in the MINI MP17 configuration. These ports are wired in the standard version to allow insertion of NCR 605 controllers without backplane modification. If it is desired to utilize 1700 A/Q type controllers in these slots fifteen signal lines must be custom wired. These are the signals used exclusively by the 1700 A/Q I/O Scheme. In the event either type controller (1700 or NCR 605) requires the use of a DMA port the signals from the DMA port must be custom wired. Also, if a peripheral controller is equipped to interface to the panel interface these signals must be custom wired.

7.6.2 MAXI Configuration to be supplied later.

7.6 3 Programmable Peripheral Controller Configuration: To be supplied later.

8.0 APPENDIX II

32-bit controller configuration - to be added.

9.0 APPENDIX III - GLOSSARY

A field
In microinstruction, A field specifies source of operand to be sent to ALU from selector 1.

A register
General purpose register.

A* register
Register included in systems containing hardware double precision option.

ALU
Arithmetic and logical unit. Performs arithmetic and logical operations on two operands sent to it from the two selectors.

B field
In microinstruction, B field specifies source of operand to be sent to ALU from selector 2.

BG
Bit generator. Allows a word to be sent to ALU with all zeros except one bit set at any bit position; used for masking or arithmetic operations.

C field
In microinstruction, constant {C} field. Can contain constants, micromemory addresses, or other codes, depending on format of microinstruction.

CPU
Central processing unit. Consists of micromemory, control section, arithmetic section, and I/O.

D field
In microinstruction, destination {D} field. Specifies destination for results of operation performed by ALU.

Dead start
Optional logic that allows read/write micromemory to be loaded from external input device.

Emulation

Process combining hardware and firmware design, by which one processor {emulator} executes programs designed for different processor, even though one-to-one hardware correspondence does not exist.

F field

In microinstruction, function {F} field. Specifies operation to be performed by ALU or shift or scale of A or AQ registers.

F register

General purpose register.

File 1

Register file addressed by contents of K register.

File 2

Register file typically addressed by contents of N register.

Firmware

General term for combination of microinstructions used in microprogram to perform a certain operation.

I register

General purpose register that can be used to hold software instruction during execution if MP is configured as emulator.

IC

Integrated circuit.

I/O

Input/output.

K register

8-bit counter that can be cleared, incremented, or decremented under microinstruction control. Also used to address file 1.

LPM

Large plane memory.

M field

In microinstruction, mode {M} field specifies addressing mode to be used to obtain next microinstruction pair from micromemory.

MA register

Micromemory address register. Holds micromemory address of current microinstruction pair.

MA transform	Micromemory address transform.
MAC	Memory address counter. Holds address of next sequential microinstruction pair.
Main memory	Core memory used by MP for storage of operands, etc.
Mask register	Used to control processing of internal and external interrupts.
Microinstruction	32-bit instruction from micromemory that controls all operations throughout 5600 system.
Micromemory	High speed semiconductor memory, which contains microprograms.
Microprogram	Set of microinstructions stored in micromemory.
MIR	Microinstruction register. Holds microinstruction being executed.
MM	Micromemory.
MPP	Microprogrammable processor. CPU portion of any specific 5600 system.
N register	8-bit counter that can be cleared, incremented, or decremented under microinstruction control. Also used to address file 2.
P register	General purpose register that can be used to hold main memory address of software instruction being executed if MPP is configured as emulator.
Program protect	Optional logic which, when enabled, prevents unprotected programs and I/O users from changing contents of protected areas of main memory.
Q register	General purpose register used in multiply and divide operations.

Q* register
Register included in systems containing hardware double precision option.

RTJ register
Return jump register. Holds micromemory address, to which control will return at completion of a subroutine.

S field
In microinstruction, special {S} field specifies operation to be performed in parallel with ALU operation.

S1, S2, etc.
Selector 1, selector 2, etc.

Selector
Multiplexer thatallows one of several sources of data to be selected for transfer from one location in MPP organization to another under microinstruction control.

SM
Status/mode register.

Split adder
Optional process by which ALU can be functionally split into two independent ALUs under microinstruction control.

SPM
Small plane memory.

Status/mode register
Contains flag bits and status/mode bits. Flag bits are set under microinstruction control to enable certain internal MPP operations {e.g., double precision flag set enables double precision logic}. Status/mode bits indicate internal or external conditions {e.g., memory parity error}.

T field
In microinstruction, test {T} field specifies whether upper or lower microinstruction of next microinstruction pair is to be executed.

Transform matrix — Selects bits from various sources in MPP organization and translates them into micromemory address in MA register, or transfers them to K or N register.

X register — General purpose processor register.

X* register — Register included in systems containing hardware double precision option.