COP

CONTROL PROCESSOR

CHAPTER 4

4. CONTROL PROCESSOR

4.0 INTRODUCTION

The Control Processor is the control center of the AD-10 during run time. While the host processor exercises overall system control through the HIC, it is not capable of effectively monitoring and controlling the high speed internal operations of the AD-10. This capability is provided by the COP. Specifically, the COP is a programmed processor designed to:

- Make logical decisions based upon a variety of inputs
- Implement program loops and branches through control of its own and other processor program counters
- Initiate system halts
- Control the Active/Wait status of all other processors
- Control the access of other processors to the SM
- Provide program initiation of the memory refresh sequence
- Place specific data on the DM
- Control a set of 128 General Registers, including increment/decrement control
- Control the operation of the IOCC and all interface devices therein.

4.1 COP ORGANIZATION

The organization of the COP is illustrated in Figure 4.1. That part of the diagram in black is common to all processors and is described in Section 2.6. The part in red is specific to the COP. This consists of additional inputs to the Program Counter, direct access to both the DM and AM, the addition of the 128 General Registers with increment/decrement and compare capability, and a direct Halt input to the HIC. All of these relate to specific instructions, and are described in subsequent paragraphs.

4.1.1 PROCESSOR ADDRESS

The processor address for the COP is 7.

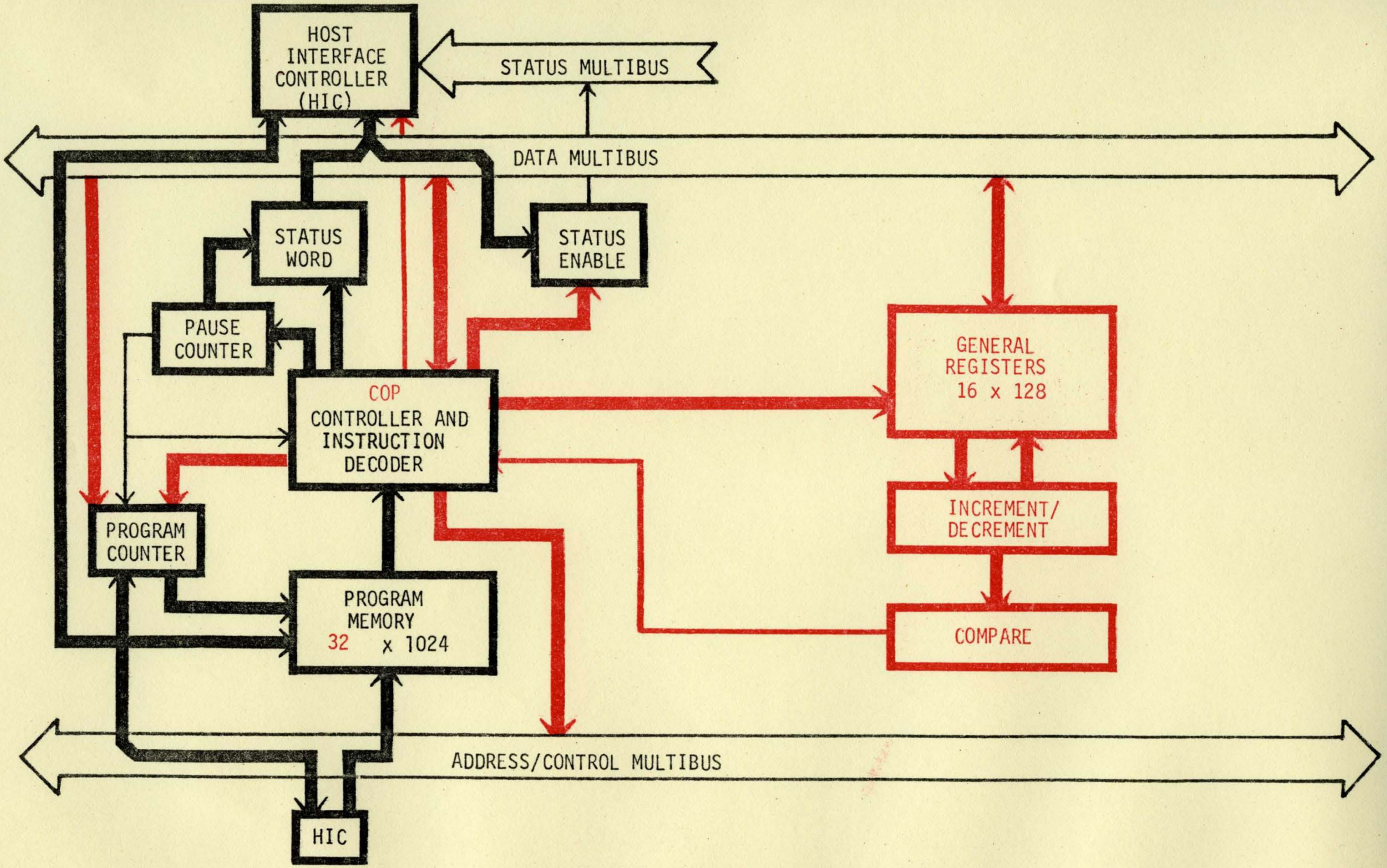
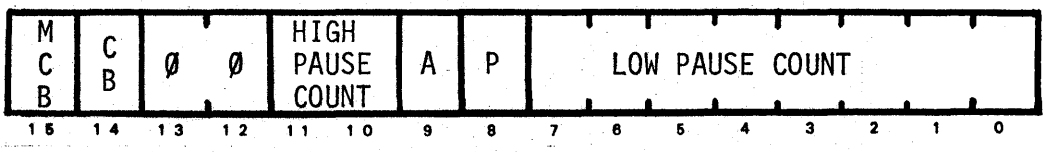


FIGURE 4.1 COP BLOCK DIAGRAM

4.1.2 PROCESSOR STATUS WORD



BITS	DESCRIPTION
Ø-7,1Ø-11	Remaining PAUSE Count (1Ø Bits)
8	COP is present when set
9	COP is active (not in PAUSE) when set
12-13	UNASSIGNED
14	Current state of the condition bit
15*	Current state of the Modified condition bit

* Once set, this bit remains set until cleared by a READ operation.

The COP Processor Status Word (PSW) contains information on the current status of the COP. The PSW is a read-only register.

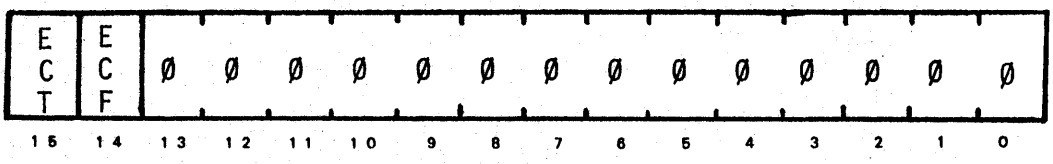
4.1.2.1 CONDITION AND MODIFIED CONDITION BITS

The state of the Condition Bit is controlled by any of the following sources:

a. A direct COP instruction (set or clear):
b. The state (true or false) of any single bit on DM; or
c. The comparison of the contents of a General Register to zero following an increment or decrement operation.

The Modified Condition Bit is set whenever an instruction is executed that could change the state of the Condition Bit. This bit then remains set until a READ operation of this register is performed by the host processor.

4.1.3 PROCESSOR STATUS ENABLE



BIT	DESCRIPTION
Ø-13	UNASSIGNED
14	Enable the Condition Bit, when false, to the AER line in the STATUS MULTIBUS
15	Enable the Condition Bit, when true, to the AER line in the STATUS MULTIBUS

The Processor Status Enable Register (PSE) is a write-only register. This register has the same address as the COP Processor Status Word.

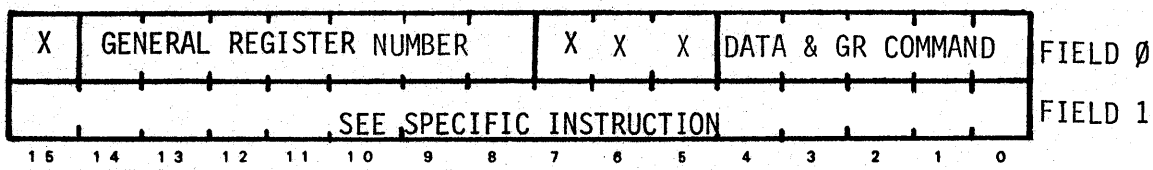
4.1.4 PROGRAM MEMORY

The COP program memory is comprised of 1024 words of 32 bits each. The word is divided into two 16-bit fields with Field Ø containing the low order bits of the word. Each field may be accessed from the host processor for a READ or WRITE operation.

4.2 THE COP INSTRUCTION SET

4.2.1 COP INSTRUCTION WORD FORMAT

The COP Instruction word format is:



Within this format, unused bits are denoted by an X. These bits are always read as a Ø. A summary of the bit patterns for all instructions is provided in Figure 4.2.

The description of the COP instruction set is divided into three categories: General Control Instructions, Data LOAD/STORE Instructions, and IOCC related instructions.

4.2.2 MICRO-PROGRAMMING COP INSTRUCTIONS

Figure 4.2 provides a summary of the COP instruction set. This summary has been organized into four groupings of instructions to indicate as clearly as possible the microprogramming possibilities which exist.

A COP instruction can consist of any of the following:

a. <A>
b. <B> <C>
c. <B> <D> <D>

An example of the form <B><C> is:

```
LPC $ARP; LGRF 7.
```

This instruction will cause the contents of General Register #7 to be loaded into the Program Counter of the ARP.

As indicated in c. above, the START and STOP instructions (which belong to Group D) may be microprogrammed with each other. Thus,

```
STOP $ARP; START $DEP
```

causes the ARP to be put in the WAIT state, the DEP to be put into the ACTIVE state, and leaves the ACTIVE/WAIT status of the MAP unchanged.

Note: Since the PAUSE Instruction belongs to group C, it can only be microprogrammed with instructions in group B.

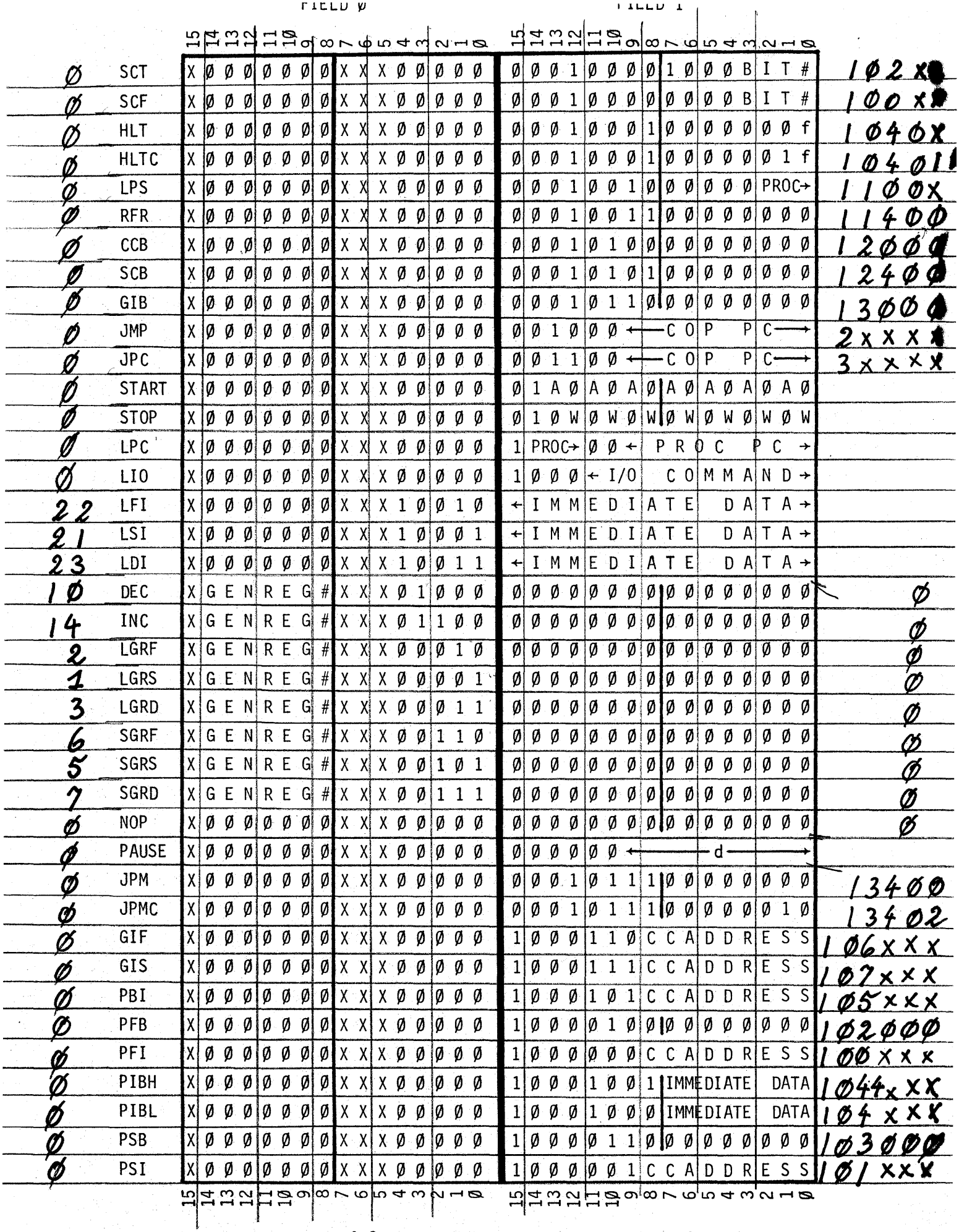


FIGURE 4.2 CONTROL PROCESSOR INSTRUCTION SET

4.2.3 GENERAL CONTROL INSTRUCTIONS

There are several subsets within this category. These include:

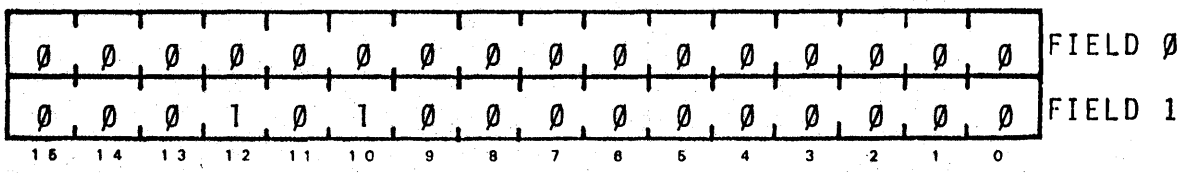
a. Instructions relating to setting (or clearing) the Condition Bit, including the General Register Increment/Decrement instructions.
b. Jump instructions, where the contents of the COP's Program Counter are changed.
c. Halt instructions, allowing the COP to initiate a system Halt via the HIC.
d. Processor control instructions including control of the ACTIVE/WAIT status of each processor, Program Counter, and Status Enable Register.
e. Other control instructions including PAUSE, NOP, and RFR (Refresh Data Memory).

The General Control Instructions are:

CCB	Clear Condition Bit
SCB	Set Condition Bit
SCB b	Set Condition Bit if Bit b is False
SCT b	Set Condition Bit if Bit b is True
DEC n	Decrement a General Register
INC n	Increment a General Register
JMP a	Jump
JPC a	Jump Conditionally
JPM	Jump to the Address on the DM
JPMC	Jump to the Address on the DM Conditionally
HLT f	Halt on Flag f True
HLTC f	Halt Conditionally on Flag f True
LPC p,a	Load the Program Counter of a Processor
LPS p	Load the Processor Status Enable Register of a Processor
START	Start Processor(s)
STOP	Stop
NOP	No-Operation
PAUSE d	Pause d Instruction Cycles
RFR	Refresh Data Memory

CCB

CLEAR CONDITION BIT



OPERATION:

$CBIT_1 \leftarrow Ø$

ERROR CONDITION(S):

CBIT if enabled to the AER line.

DESCRIPTION:

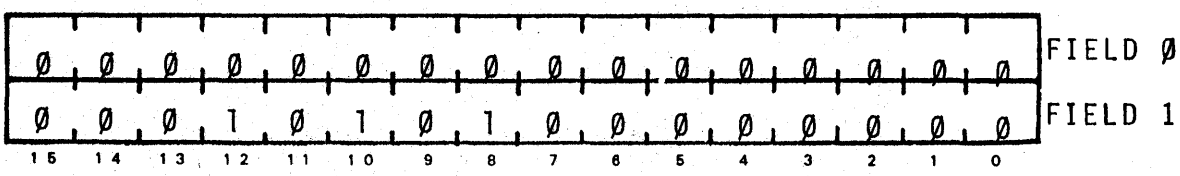
The condition bit is cleared.

EXAMPLE:

```
.
.
.
CCB
.
.
.
```

SCB

SET CONDITION BIT



OPERATION:

$CBIT_1 \leftarrow 1$

ERROR CONDITION(S):

CBIT if enabled to the AER line.

DESCRIPTION:

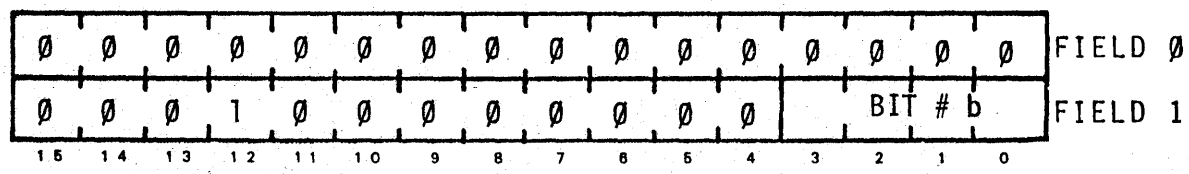
The CBIT is set.

EXAMPLE:

```
      .
      .
      .
    SCB
      .
      .
      .
```

SCF b

SET CONDITION IF BIT b IS FALSE



OPERATION:

IF $DM_{Ø}$(b) IS FALSE

THEN $CBIT_{1} \leftarrow 1$

ELSE $CBIT_{1} \leftarrow CBIT_{Ø}$

ERROR CONDITION(S):

None

DESCRIPTION:

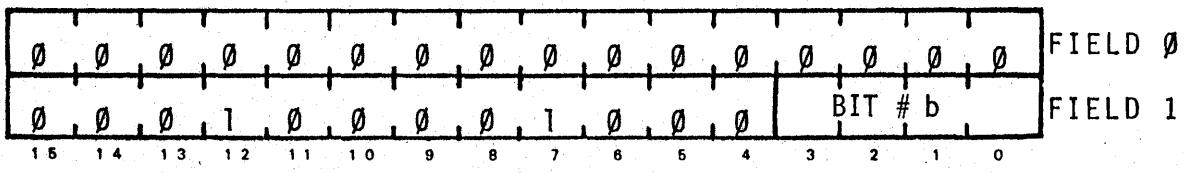
The specified bit of the DATA MULTIBUS is tested. If it is false, the condition bit is set. If it is true, the condition bit is not modified.

EXAMPLE:

```
INSTRUCTION
  CYCLE
                .COP              .DEP
                .                 .
                .                 .
n               START $DEP        .             !GET THE CONDITION BIT
n+1             NOP               LCF;PAUSE 3   !SEND THE CONDITION BIT
n+2             SCF 15            .             !IF FALSE ...
n+3             JPC ERROR         .             !WE HAVE AN ERROR
                .                 .
                .                 .
                .                 .
```

SCT b

SET CONDITION IF BIT b IS TRUE



OPERATION:

IF $DM_{Ø}$ (b) IS TRUE

THEN $CBIT_{1} \leftarrow 1$

ELSE $CBIT_{1} \leftarrow CBIT_{Ø}$

ERROR CONDITION(S):

None

DESCRIPTION:

If the specified bit of the DATA MULTIBUS is true, the condition bit is set. If it is false, the condition bit is not modified.

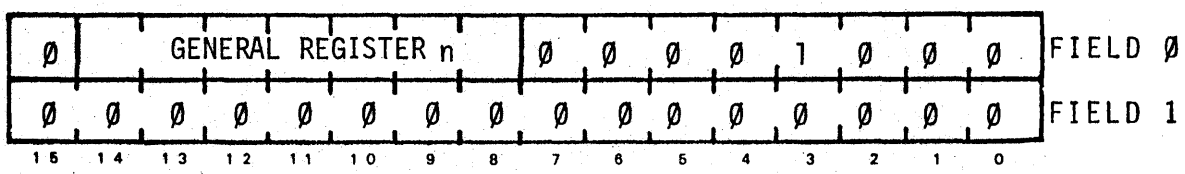
EXAMPLE:

```
    .
    .
    .
LGRF  1Ø        ! SEND THE CONDITION BIT
SCT   8         !IF TRUE...
JPC    ERROR    !...WE HAVE AN ERROR
    .
    .
    .

                !SEND THE CONDITION BIT
```

DEC n

DECREMENT A GENERAL REGISTER



OPERATION:

$GR_1(n) \leftarrow GR_Ø(n) - 1$

IF $GR_1(n) \rightarrow \neq Ø$

THEN $CBIT_1 \leftarrow 1$

ELSE $CBIT_1 \leftarrow Ø$

ERROR CONDITION(S):

CBIT if enabled to the AER line.

DESCRIPTION:

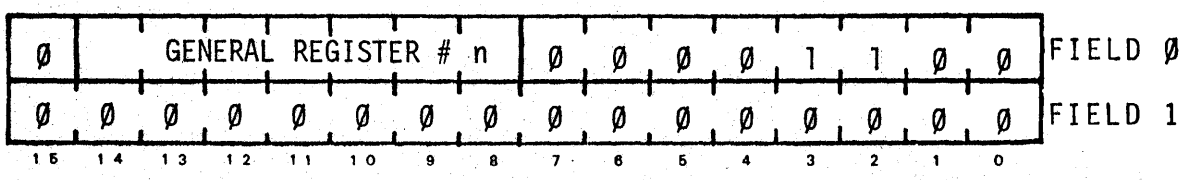
The contents of the specified register are decremented by one; if the result is zero, the CBIT is cleared; otherwise it is set.

EXAMPLE:

```
LOOP    .
        .
        .
        DEC  67          !DONE?
        JPC  LOOP        !IF NOT, DO LOOP AGAIN
         .
         .
         .
```

INC n

INCREMENT A GENERAL REGISTER



OPERATION:

$GR_1(n) \leftarrow GR_\emptyset(n)+1$

IF $GR_1(n) \neq \emptyset$

THEN $CBIT_1 \leftarrow 1$

ELSE $CBIT_1 \leftarrow \emptyset$

ERROR CONDITION(S):

CBIT if enabled to the AER line.

DESCRIPTION:

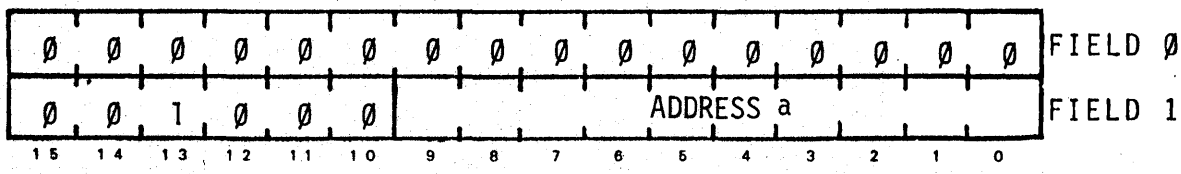
The contents of the specified register are incremented by one. If the result is zero, the CBIT is cleared; otherwise it is set.

EXAMPLE:

```
LOOP    .
        .
        .
        INC 77      !DONE?
        JPC  LOOP   !IF NOT, DO LOOP AGAIN
         .
         .
         .
```

JMP a

JUMP



OPERATION:

COP PC_1 ← ADDRESS

ERROR CONDITION(S):

NONE

DESCRIPTION:

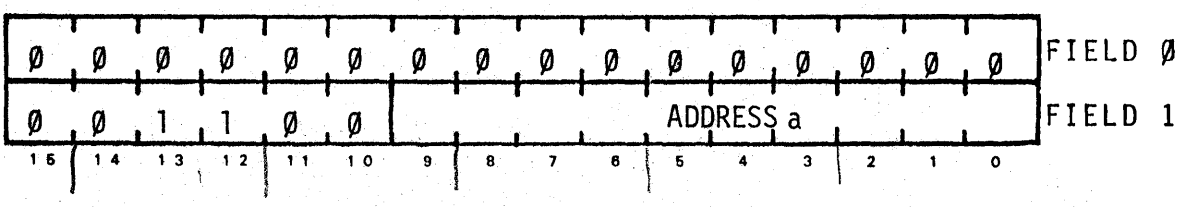
The address portion of the instruction is placed in the COP PROGRAM COUNTER so that the next instruction executed will be fetched from that address in program memory.

EXAMPLE:

```
BEGIN  .
       .
       .
       JMP  BEGIN     !ONE MORE TIME
```

JPC a

JUMP CONDITIONALLY



OPERATION:

IF CBIT IS TRUE

THEN COP PC_1 ← ADDRESS

ELSE NOP

ERROR CONDITION(S):

NONE

DESCRIPTION:

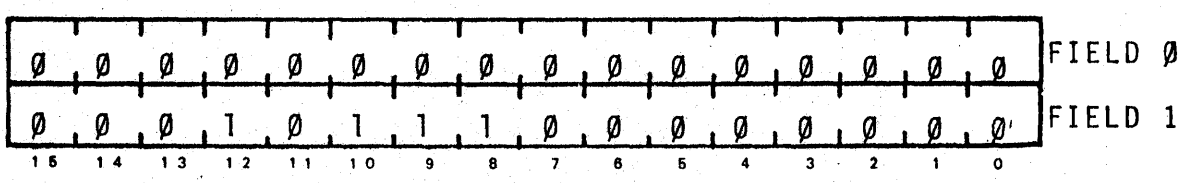
If the condition bit is true, the address portion of the instruction is placed in the COP PROGRAM COUNTER so that the next instruction executed will be fetched from that address in program memory. Otherwise, no operation is performed and the next sequential instruction is executed.

EXAMPLE:

```
BEGIN  .
       .
       .
       JPC  ERROR
       JMP  BEGIN
ERROR  STOP $ARP,$MAP,$DEP
       HLT  Ø
       .
       .
       .
```

JPM

JUMP TO THE ADDRESS ON THE DATA MULTIBUS



OPERATION:

COP PC_1 ← $DM_Ø$(0:9)

ERROR CONDITION(S):

NONE

DESCRIPTION:

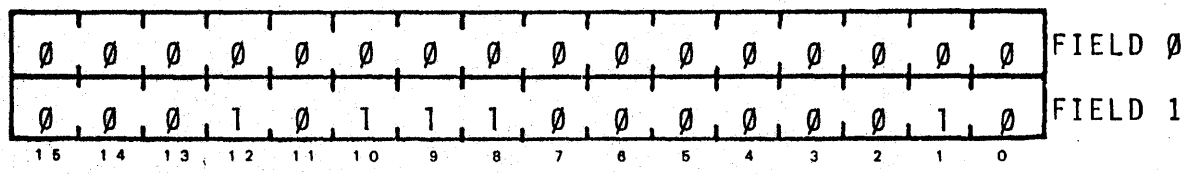
The low order 1Ø bits of the DATA MULTIBUS FIRST are placed in the COP PROGRAM COUNTER. This causes the next instruction to be fetched from that address.

EXAMPLE:

```
        LFI     PROGL
        SGRF    RETURN
        JMP     SUB       !CALL SUBROUTINE
PROGL   .
        .
        .
SUB     .
        .
        .
        LGRF    RETURN    !RETURN FROM SUBROUTINE
        JPM
        .
        .
        .
```

JPMC

JUMP TO THE ADDRESS ON THE DATA MULTIBUS CONDITIONALLY



OPERATION:

IF CBIT IS TRUE

THEN COP PC_1 ← $DM_Ø$(Ø:9)

ELSE NOP

ERROR CONDITION(S):

NONE

DESCRIPTION:

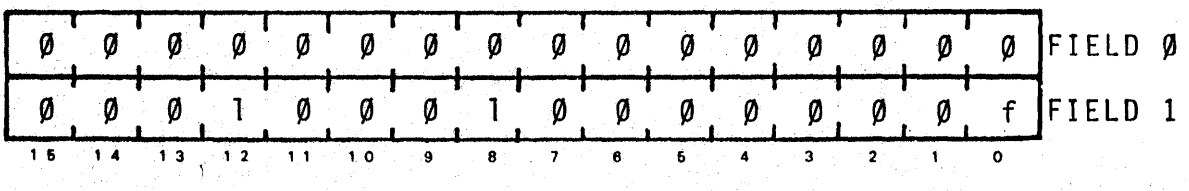
If the condition bit is true, then the low order 1Ø bits of the DATA MULTIBUS FIRST are placed in the COP PROGRAM COUNTER. Otherwise, no operation is performed.

EXAMPLE:

```
    .
    .
    .
LGRF  RETURN  !PUT "RETURN" ON DM
JPMC          !JUMP TO "RETURN" IF CBIT IS TRUE
    .
    .
    .
```

HLT f

HALT ON FLAG (f) TRUE



OPERATION:

IF FLAG f IS TRUE

THEN AM_1 ← HALT

ELSE NOP

ERROR CONDITION(S):

NONE

DESCRIPTION:

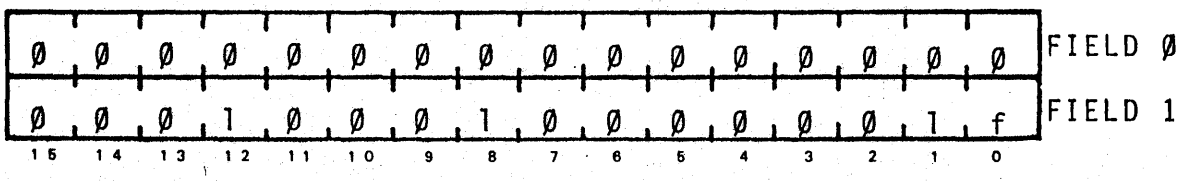
If the HALT f flag is enabled in the HALT MASK REGISTER, all processors will HALT at the end of the current instruction cycle.

EXAMPLE:

```
BEGIN  .
       .
       .
       HLT  Ø     !WAIT FOR USER
       .
       .
       .
       HLT  1     !DONE?
       JMP  BEGIN !NO, ONE MORE TIME
```

HLTC f

HALT CONDITIONALLY ON FLAG f TRUE



OPERATION:

```
IF CBIT IS TRUE

   THEN IF FLAG f IS TRUE

    THEN AM1  ←  HALT

    ELSE NOP

   ELSE NOP
```

ERROR CONDITION(S):

NONE

DESCRIPTION:

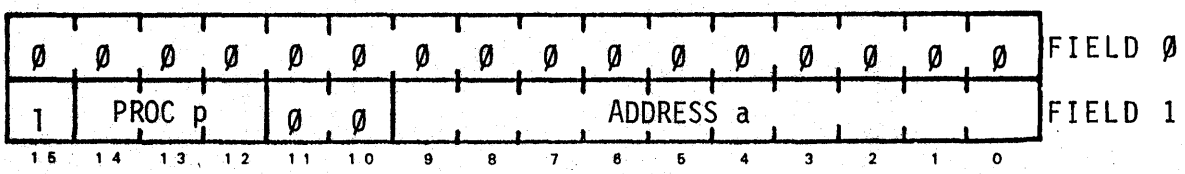
If the condition bit is set and the HALT f flag is enabled in the HALT MASK REGISTER, all processors will HALT at the end of the current instruction cycle.

EXAMPLE:

```
BEGIN  .
       .
       .
       HLTC  1     !ERROR ABOVE?
       JMP   BEGIN !NO, ONE MORE TIME
```

LPC p,a

LOAD THE PROGRAM COUNTER OF A PROCESSOR



OPERATION:

$AM_1(Ø:9) \leftarrow$ ADDRESS

$AM_1(16:18) \leftarrow$ PROC #

PROC# $PC_1 \leftarrow AM_1(Ø:9)$

ERROR CONDITION(S):

BUS CONFLICT (ADDRESS)

DESCRIPTION:

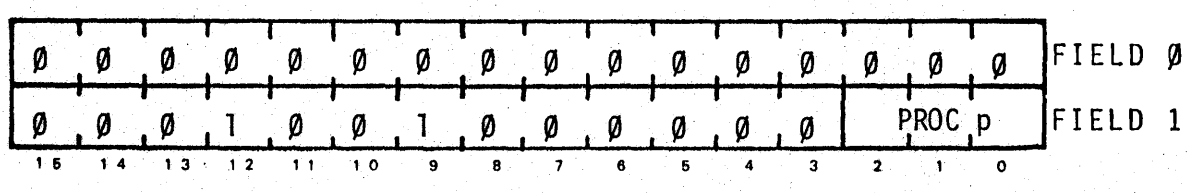
The address is placed in the PROGRAM COUNTER of the specified processor. The processor must not be active (otherwise, the processor PROGRAM COUNTER is not changed). The PAUSE count of the processor is cleared.

EXAMPLE:

```
.
.
.
STOP   $ARP
LPC    $ARP,FIRST
START  $ARP
.
.
.
```

LPS p

LOAD THE PROCESSOR STATUS ENABLE REGISTER OF A PROCESSOR



OPERATION:

$AM_1(16:18) \leftarrow$ PROC #

PROC # $PSE_1 \leftarrow DM_1$

ERROR CONDITION(S):

BUS CONFLICT (ADDRESS)

DESCRIPTION:

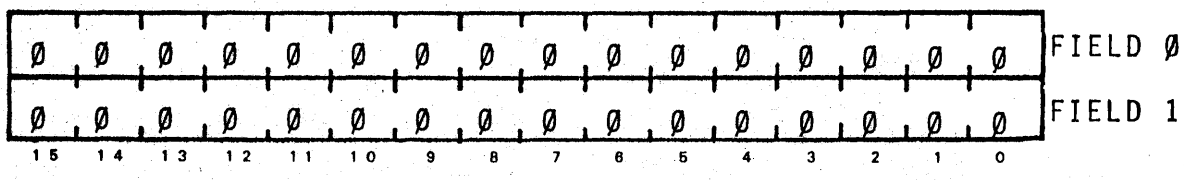
The PROCESSOR STATUS ENABLE REGISTER of the specified processor is loaded with the data on the DATA MULTIBUS.

EXAMPLE:

```
.
.
.
LPS  $ARP; LGRF 6
.
.
.
```

NOP

NO OPERATION



OPERATION:

None

ERROR CONDITION(S):

None

DESCRIPTION:

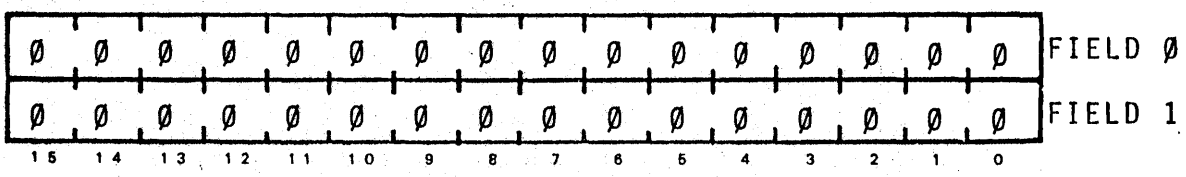
Suspends execution for one instruction cycle.

EXAMPLE:

```
NOP
```

NOPC

NO OPERATION CONDITIONALLY



OPERATION:

IF CBIT IS TRUE

THEN NOP
ELSE NOP

ERROR CONDITION(S):

None

DESCRIPTION:

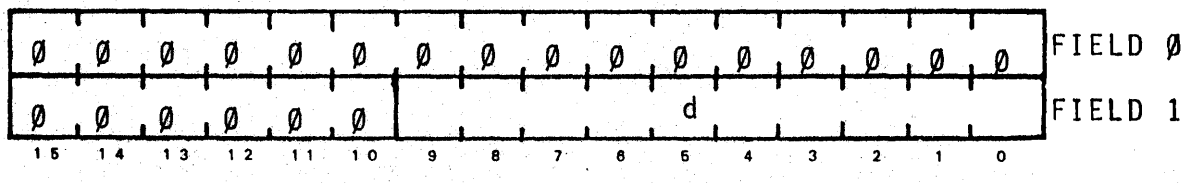
If the condition bit is set, then there will be no operation performed. Otherwise, no operation is performed. For further information refer to the NOP instruction.

EXAMPLE:

```
      .
      .
      .
    NOPC
      .
      .
      .
```

PAUSE d

PAUSE FOR d INSTRUCTION CYCLES



OPERATION:

None

ERROR CONDITION(S):

None

DESCRIPTION:

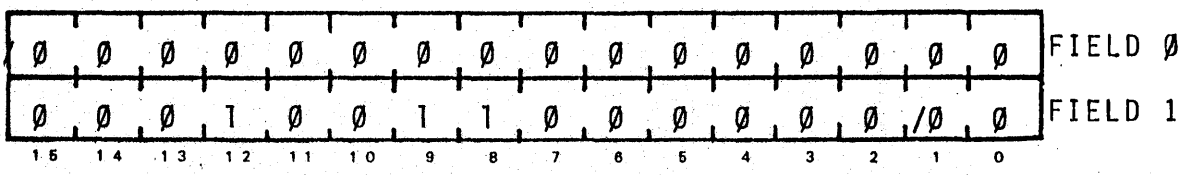
Suspend execution for d instruction cycles following the PAUSE instruction. This has the same effect as (d+1) NOP's.

EXAMPLE:

```
      .
      .
      .
    START $MAP,$DEP,$ARP  !START IT UP
    PAUSE 20              !WAIT 21 INSTRUCTIONS
    STOP  $MAP,$DEP,$ARP  !ALL DONE
      .
      .
      .
```

RFR

REFRESH DATA MEMORY



ERROR CONDITION(S):

None

DESCRIPTION:

This instruction causes a memory REFRESH. Automatic REFRESH of 4 rows of DATA MEMORY is performed every 102.4µs. A REFRESH will cause a delay of 2.5µs in the execution of all processors. The operations performed during a REFRESH are:

1. Halt all processors (save all necessary status information);
2. Perform a shutdown sequence capturing multibus data;
3. REFRESH memory; and
4. Restart processors (reset REFRESH counter) and replay shutdown/restart buffer onto the multibus.

EXAMPLE:

```
.
.
.
LIO   STARTADC
RFR
PAUSE 73
LIO   READADC
.
.
.
```

4.2.4 DATA LOAD/STORE INSTRUCTIONS

The three Load Immediate Data instructions allow a 16 bit data word contained in Field X of the instruction word to be loaded directly onto the DM.

The General Register LOAD/STORE instructions provide the capability to load the contents of any of the 128 General Registers onto the DM or to store the data on the DM in any one of the General Registers.

The Load Immediate Data instructions are:

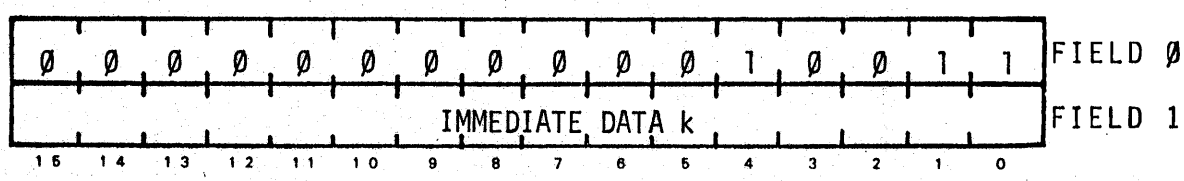
LDI	Load Double Immediate
LFI	Load First Immediate
LSI	Load Second Immediate

The General Register LOAD/STORE instructions are:

LGRD	Load General Register Double
LGRF	Load General Register First
LGRS	Load General Register Second
SGRD	Store General Register Double
SGRF	Store General Register First
SGRS	Store General Register Second

LDI k

LOAD DOUBLE IMMEDIATE



OPERATION:

DM_1 ← IMMEDIATE DATA

$DM_{1.5}$ ← IMMEDIATE DATA

ERROR CONDITION(S):

BUS CONFLICT (DATA)

DESCRIPTION:

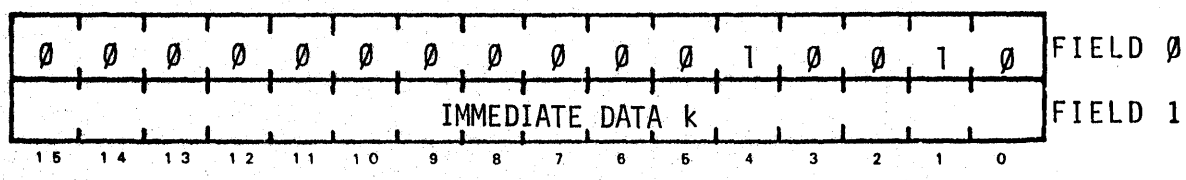
The immediate data is placed on the DATA MULTIBUS FIRST and SECOND.

EXAMPLE:

```
      .
      .
      .
LDI   DATA
      .
      .
      .
```

LFI k

LOAD FIRST IMMEDIATE



OPERATION:

DM_1 ← IMMEDIATE DATA

ERROR CONDITION(S):

BUS CONFLICT (DATA)

DESCRIPTION:

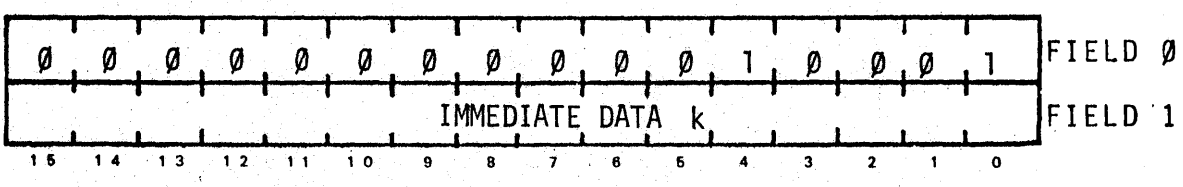
The immediate data is placed on the DATA MULTIBUS FIRST.

EXAMPLE:

```
    .
    .
    .
LFI  DATA
    .
    .
    .
```

LSI k

LOAD SECOND IMMEDIATE



OPERATION:

$DM_{1.5}$ ← IMMEDIATE DATA

ERROR CONDITION(S):

BUS CONFLICT (DATA)

DESCRIPTION:

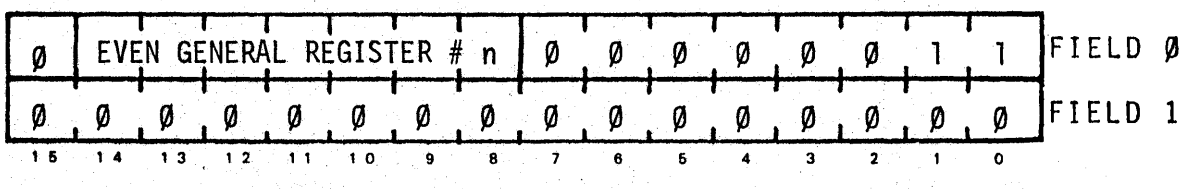
The immediate data is placed on the DATA MULTIBUS SECOND.

EXAMPLE:

```
.
.
.
LSI  DATA
.
.
.
```

LGRD n

LOAD FROM GENERAL REGISTERS DOUBLE



OPERATION:

$DM_1 \leftarrow GR_{\emptyset}(n)$

$DM_{1.5} \leftarrow GR_{\emptyset}(n+1)$

ERROR CONDITION(S):

BUS CONFLICT (DATA)

DESCRIPTION:

The contents of the specified even* register and next register are placed on the DATA MULTIBUS FIRST and SECOND. The low order bit of the register number is ignored.

* An even register is of the set:

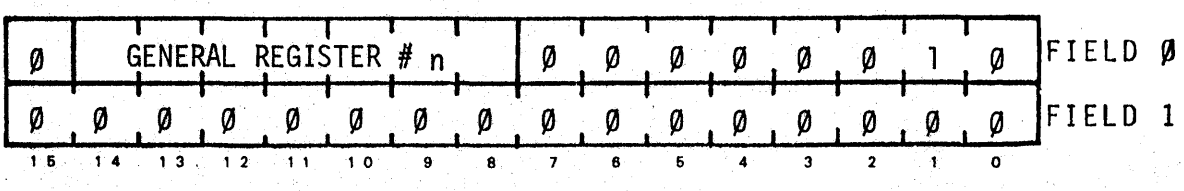
$$\{n \mid n=0,2,4,6,\ldots,126_{10}\}$$

EXAMPLE:

```
      .
      .
      .
    LGRD  PAIR   !PASS THEM ON
      .
      .
      .
```

LGRF n

LOAD FROM GENERAL REGISTER FIRST



OPERATION:

$DM_1 \leftarrow GR_Ø(n)$

ERROR CONDITION(S):

BUS CONFLICT (DATA)

DESCRIPTION:

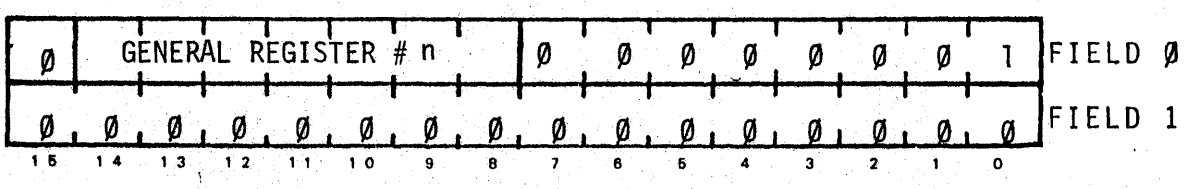
The contents of the specified register are placed on the DATA MULTIBUS FIRST.

EXAMPLE:

```
.
.
.
LGRF  7   !PASS IT ON
.
.
.
```

LGRS n

LOAD FROM GENERAL REGISTER SECOND



OPERATION:

$DM_{1.5} \leftarrow GR_{Ø}(n)$

ERROR CONDITION(S):

BUS CONFLICT (DATA)

DESCRIPTION:

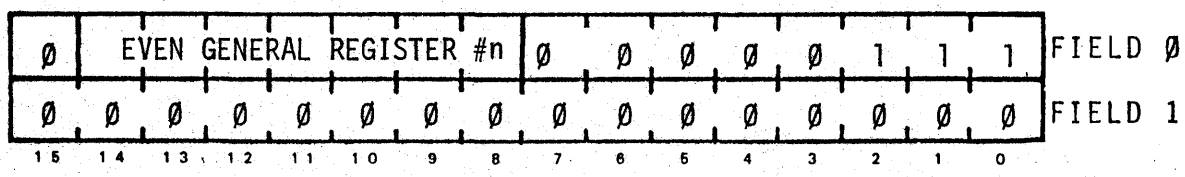
The contents of the specified register are placed on the DATA MULTIBUS SECOND.

EXAMPLE:

```
.
.
.
LGRS  43  !PASS IT ON
.
.
.
```

SGRD n

STORE TO GENERAL REGISTERS DOUBLE



OPERATION:

$GR_{.5}(n) \leftarrow DM_{Ø}$

$GR_{1}(n+1) \leftarrow DM_{.5}$

ERROR CONDITION(S):

None

DESCRIPTION:

The contents of the DATA MULTIBUS FIRST and SECOND is placed in the specified even* and next registers. The low order bit of the register number is ignored.

* An even register is of the set:

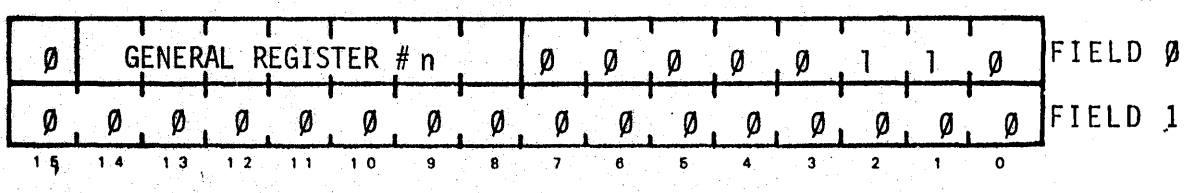
$$\{n \mid n=Ø,2,4,6\ldots,126_{10}\}$$

EXAMPLE:

```
      .
      .
      .
SGRD PAIR  !SAVE THEM
      .
      .
      .
```

SGRF n

STORE TO GENERAL REGISTER FIRST



OPERATION:

$GR_{.5}(n) \leftarrow DM_{Ø}$

ERROR CONDITION(S):

None

DESCRIPTION:

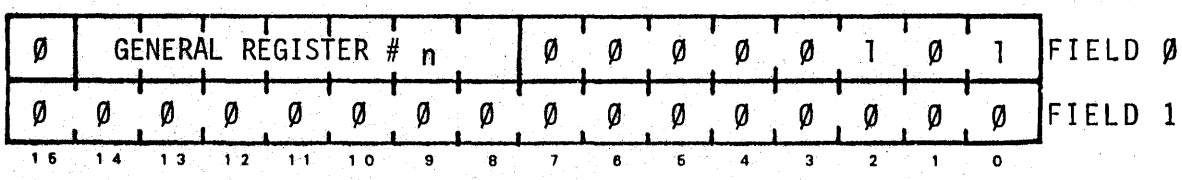
The contents of the DATA MULTIBUS FIRST are placed in the specified register.

EXAMPLE:

```
.
.
.
SGRF 32     !SAVE IT
.
.
.
```

SGRS n

STORE TO GENERAL REGISTER SECOND



OPERATION:

$GR_1(n) \leftarrow DM_{.5}$

ERROR CONDITION(S):

None

DESCRIPTION:

The contents of the DATA MULTIBUS SECOND are placed in the specified register.

EXAMPLE:

```
.
.
.
SGRS  16     !SAVE IT
.
.
.
```

4.2.5 I/O CHANNEL CONTROLLER INSTRUCTIONS

The I/O Channel Controller (IOCC) is described in detail in Chapter 8. However, a summary is presented here as an aid to understanding the COP instructions that relate to the IOCC.

4.2.5.1 IOCC DESCRIPTION

Figure 4.3 illustrates the overall organization of the IOCC. The IOCC consists of an address and instruction decoder that receives its inputs from the COP via the AM. An Output Data Buffer is also provided which allows data to be transmitted to several output devices without requiring multiple DM transactions. This buffer may be loaded by one transaction from the DM or by two transactions from the AM. Inputs via the AM are limited to the data contained in the COP instruction word. The contents of the Output Data Buffer may be read back via the DM, primarily as a diagnostic aid. Finally, the Channel Controller provides the transmitter/receiver pairs matched to those in the remotely located Device Controller. The Device Controller may contain up to 128 devices in any mixture of A/D or D/A converters or logic (sense and control) buffers. Each device has a physical address decoded from the IOCC Address Bus. In addition, each device may be assigned to any one of up to 32 control groups for command purposes. This allows groups of devices to be simultaneously commanded to perform their specific function. Examples include simultaneous up-dating of the double-buffered D/A converters and/or simultaneous initiation of A/D conversion.

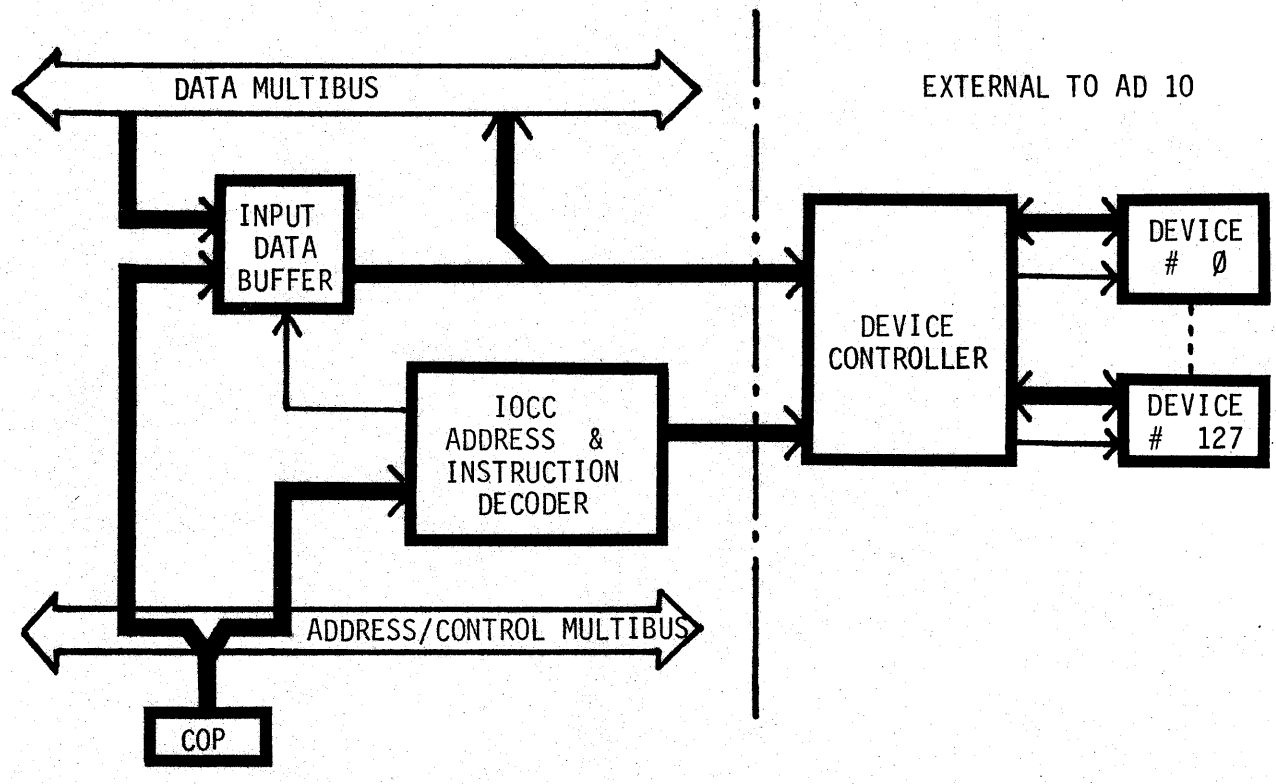


FIGURE 4.3 IOCC BLOCK DIAGRAM

The output of an A/D converter can be read at any time, and care should be exercised to ensure that adequate time has been allowed for the conversion cycle to be completed. A microprogrammed instruction may be used to read an A/D converter output and then initiate another conversion cycle.

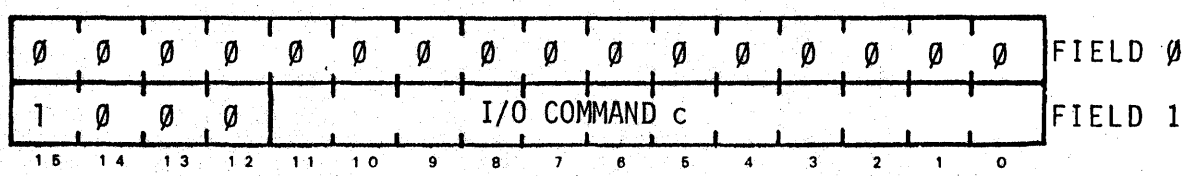
4.2.5.2 IOCC INSTRUCTIONS

The IOCC Instructions are:

LIO c	Load the IOCC with a command word.
GIB	Get the I/O Buffer.
PFB	Put First to the I/O Buffer.
PSB	Put Second to the I/O Buffer.
PIBH k	Put Immediate to the I/O Buffer High
PIBL k	Put Immediate to the I/O Buffer Low
PBI	Put I/O Buffer to the I/O.
GIF c,a	Get the I/O First
PFI c,a	Put First to the I/O.
PSI c,a	Put Second to the I/O.

LIO c

LOAD THE I/O CHANNEL CONTROLLER WITH A COMMAND WORD



OPERATION:

$AM_1(Ø:11) \leftarrow$ I/O COMMAND

ERROR CONDITION(S):

BUS CONFLICT (ADDRESS)

DESCRIPTION:

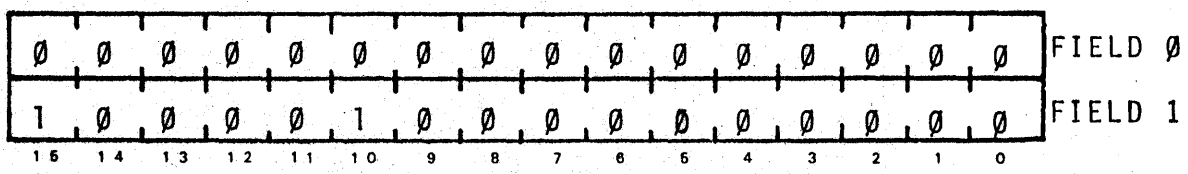
The I/O command is sent to the CHANNEL CONTROLLER over the ADDRESS MULTIBUS. The command may be self contained or used in conjunction with the DATA MULTIBUS.

EXAMPLE:

```
.
.
.
LIO WRITE; LGRF 14
.
.
.
```

PFB

PUT FIRST TO I/O BUFFER



OPERATION:

$AM_1(Ø:11) \leftarrow$ COMMAND

$IOB_2 \leftarrow DM_1$

ERROR CONDITION(S):

BUS CONFLICT (ADDRESS)

DESCRIPTION:

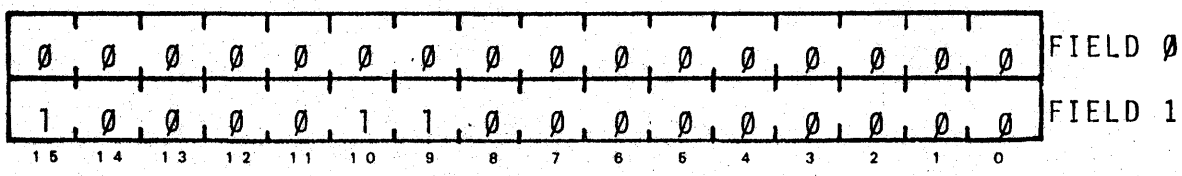
The CHANNEL CONTROLLER takes the data on the DATA MULTIBUS and places it in the I/O BUFFER. This allows the same data to be used several times without using the DATA MULTIBUS each time.

EXAMPLE:

```
.
.
.
PFB; LGRF 5     !SEND THE CONTENTS OF GR#5 TO THE I/O BUFFER
PBI  WRITE, DAC1
PBI  WRITE, DAC2
.
.
.
```

PSB

PUT SECOND TO I/O BUFFER



OPERATION:

$AM_1(Ø:11) \leftarrow$ COMMAND

$IOB_2 \leftarrow DM_{1.5}$

ERROR CONDITION(S):

BUS CONFLICT (ADDRESS)

DESCRIPTION:

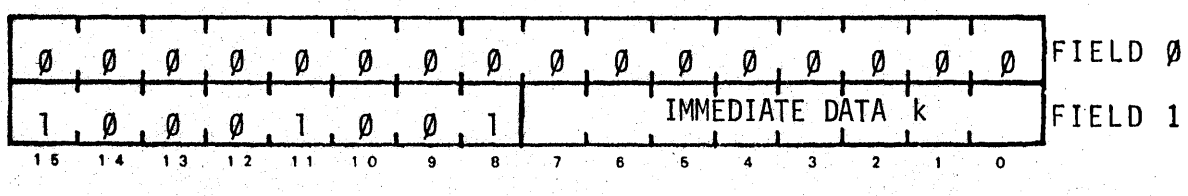
The CHANNEL CONTROLLER takes the data on the DATA MULTIBUS and places it in the I/O BUFFER. This allows the same data to be used several times without using the DATA MULTIBUS each time.

EXAMPLE:

```
.
.
.
PSB; LGRS 4
PBI  WRITE, DAC1
PBI  WRITE, DAC2
.
.
.
```

PIBH k

PUT IMMEDIATE TO I/O BUFFER HIGH



OPERATION:

$AM_1(Ø:7) \leftarrow$ IMMEDIATE DATA

$IOB_2(8:15) \leftarrow AM_1(Ø:7)$

ERROR CONDITION(S):

BUS CONFLICT (ADDRESS)

DESCRIPTION:

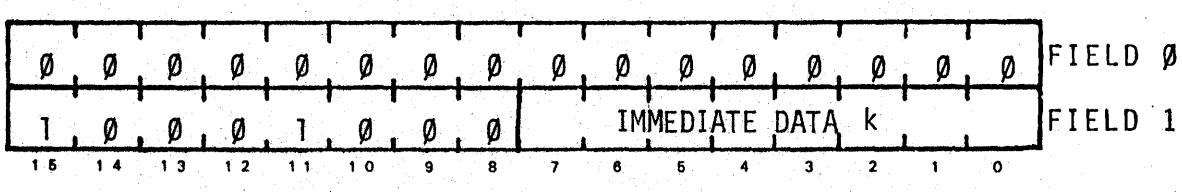
The CHANNEL CONTROLLER takes the 8 bits of immediate data and places it in the high order 8 bits of the I/O BUFFER. This allows the buffer to be loaded without using the DATA MULTIBUS.

EXAMPLE:

```
.
.
.
PIBH  HDATA
PIBL  LDATA
PBI   WRITE,DAC1
.
.
.
```

PIBL k

PUT IMMEDIATE TO I/O BUFFER LOW



OPERATION:

$AM_1(Ø:7) \leftarrow$ IMMEDIATE DATA

$IOB_2(Ø:7) \leftarrow AM_1(Ø:7)$

ERROR CONDITION(S):

BUS CONFLICT (ADDRESS)

DESCRIPTION:

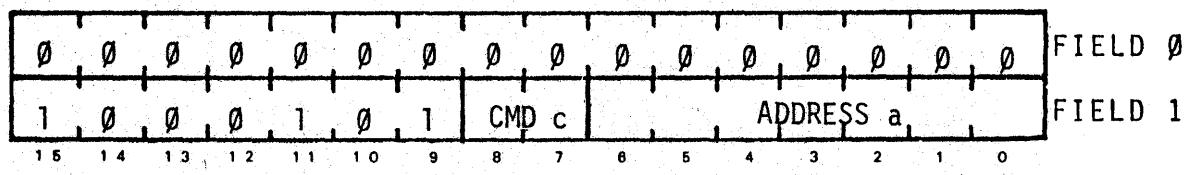
The CHANNEL CONTROLLER takes the 8 bits of immediate data and places it in the low order 8 bits of the I/O BUFFER. This allows the buffer to be loaded without using the DATA MULTIBUS.

EXAMPLE:

```
.
.
.
PIBH  HDATA
PIBL  LDATA
PBI   WRITE,DAC1
.
.
.
```

PBI c,a

PUT I/O BUFFER TO I/O



OPERATION:

$AM_1(Ø:8) \leftarrow$ COMMAND,ADDRESS

$IO_2 \leftarrow AM_1(Ø:8), IOB_1$

ERROR CONDITION(S):

BUS CONFLICT (ADDRESS)

DESCRIPTION:

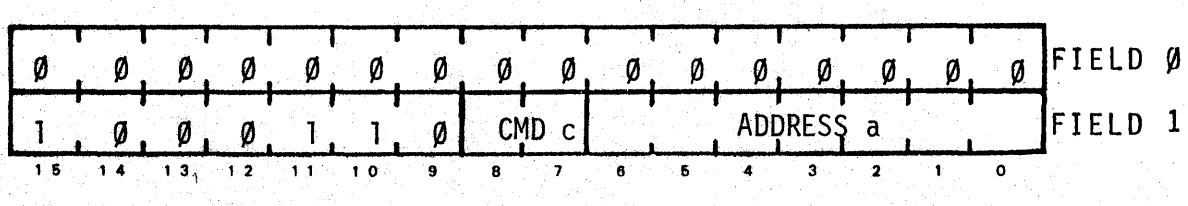
The CHANNEL CONTROLLER takes the COMMAND and ADDRESS portion of the ADDRESS MULTIBUS and the contents of the I/O BUFFER and sends them to the DEVICE CONTROLLER. The DEVICE CONTROLLER will process the information.

EXAMPLE:

```
.
.
.
PFB                 !SAVE DATA
PBI  WRITE,DAC1     !WRITE IT ONE TIME
PBI  WRITE,DAC2     !WRITE IT TWICE
.
.
.
```

GIF c,a

GET I/O FIRST



OPERATION:

$AM_1(Ø:8) \leftarrow$ COMMAND,ADDRESS

$IO_2 \leftarrow AM_1(Ø:8)$

$DM_5 \leftarrow IO^*$

ERROR CONDITION(S):

BUS CONFLICT (ADDRESS,DATA)

DESCRIPTION:

The CHANNEL CONTROLLER takes the COMMAND and ADDRESS information from the ADDRESS MULTIBUS and sends it to the DEVICE CONTROLLER. The DEVICE CONTROLLER (assumming an input device and an input command) will send data via the I/O BUS and the CHANNEL CONTROLLER will place the data on the DATA MULTIBUS. The elapsed time from execution of this instruction until data is present on the MULTIBUS is 5 instruction cycles; this is the same as the delay from the time a DATA MEMORY address is sent out from the MAP until the time the memory data is present on the MULTIBUS.

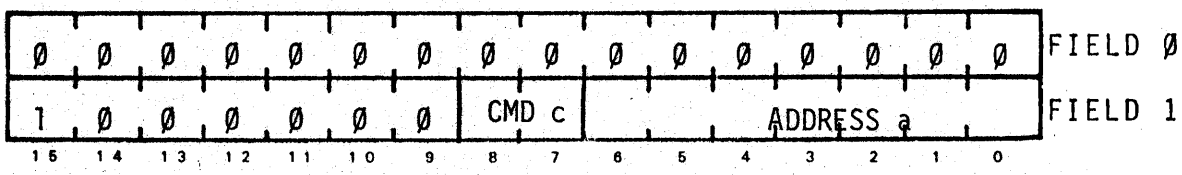
* The data portion of the I/O BUS should be considered busy from the start of the instruction until the data is on the MULTIBUS. Legal commands during this period of time are: GIF, GIS, GIB, PIBL, PIBH and PIB. Illegal instructions are PFI, PSI and PBI.

EXAMPLE:

```
.
.
.
GIF NEXT,ADC1
.
.
.
```

PFI c,a

PUT FIRST TO I/O



OPERATION:

$AM_1(Ø:8) \leftarrow$ COMMAND, ADDRESS

$IO_2 \leftarrow AM_1(Ø:8), DM_1$

ERROR CONDITION(S):

BUS CONFLICT (ADDRESS)

DESCRIPTION:

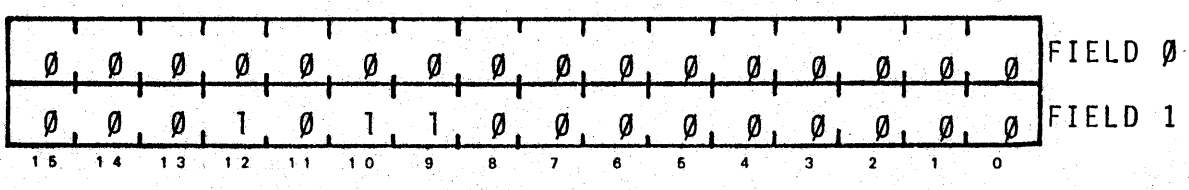
This instruction sends COMMAND and ADDRESS information to the CHANNEL CONTROLLER. The CHANNEL CONTROLLER will respond by taking the data off of the DATA MULTIBUS. The COMMAND and ADDRESS portion of the ADDRESS MULTIBUS and the data from the DATA MULTIBUS are sent "down the cable" to the DEVICE CONTROLLER which will process the information.

EXAMPLE:

```
.
.
.
LGRF 3; PFI WRITE, DAC1
.
.
.
```

GIB

GET I/O BUFFER



OPERATION:

AM_1 ← COMMAND

$DM_{1.5}$ ← IOB_1

ERROR CONDITION(S):

BUS CONFLICT(ADDRESS,DATA)

DESCRIPTION:

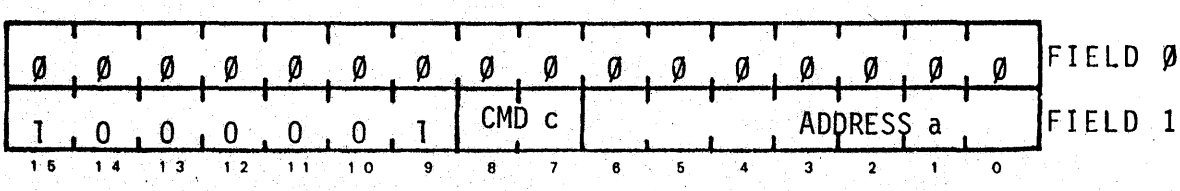
The CHANNEL CONTROLLER will place the contents of the I/O BUFFER on the DATA MULTIBUS on the SECOND BUS TRANSACTION.

EXAMPLE:

```
.
.
.
GIB
.
.
.
```

PSI c,a

PUT SECOND TO I/O



OPERATION:

$AM_1(Ø:8) \leftarrow$ COMMAND, ADDRESS

$IO_2 \leftarrow AM_1(Ø:8), DM_{1.5}$

ERROR CONDITION(S):

BUC CONFLICT (ADDRESS)

DESCRIPTION:

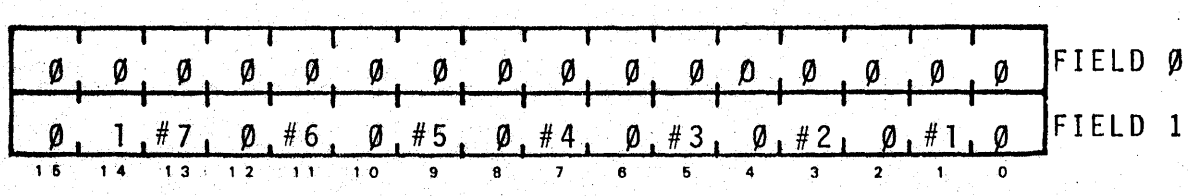
This instruction sends command and address information to the CHANNEL CONTROLLER. The CHANNEL CONTROLLER will respond by taking the data off of the DATA MULTIBUS. The COMMAND and ADDRESS portion of the ADDRESS MULTIBUS and the data from the DATA MULTIBUS are sent "down the cable" to the DEVICE CONTROLLER which will process the information.

EXAMPLE:

```
.
.
.
PSI WRITE, DAC1; LGRS 8
.
.
.
```

START p[,...[,p]]

START PROCESSOR(S)



OPERATION:

IF FIELD 1 (PROC# *2-1) IS TRUE

THEN PROC# $PS_1(9)$ ← 1

ELSE NOP

ERROR CONDITION(S):

BUS CONFLICT (ADDRESS)

DESCRIPTION:

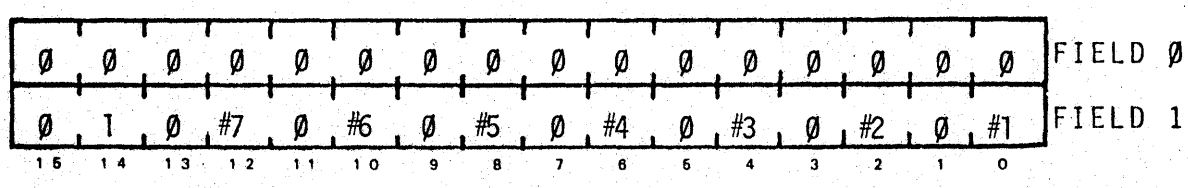
The processors whose numbers (1-7) are specified with a 1 in the appropriate bit position of Field 1 are put into the active state. They will begin execution on the next instruction cycle. Note: If a START instruction is microprogrammed with a STOP instruction, and a given processor is specified in both instructions, a NOP will result for that processor (i.e., its START/STOP status will not be changed).

EXAMPLE:

```
.
.
.
START $ARP
.
.
.
```

STOP p[,...[,p]]

STOP PROCESSOR(S):



OPERATION:

IF FIELD 1 (PROC#*2-2) IS TRUE

THEN PROC# $PS_1(9) \leftarrow Ø$

ELSE NOP

ERROR CONDITION(S):

BUS CONFLICT (ADDRESS)

DESCRIPTION:

The processors whose numbers (1-7) are specified with a 1 in the appropriate bit position of Field 1 are put into the wait state. They will halt at the end of the current instruction cycle. Note: If a START instruction is microprogrammed with a STOP instruction, and a given processor is specified in both instructions, a NOP will result for that processor (i.e., its START/STOP status will not be changed).

EXAMPLE:

```
.
.
.
STOP $ARP; START $DEP
.
.
.
```