6940B Multiprogrammer Users Guide

for the

Hewlett • Packard 9825A Calculator

with

GPIO Interface

HP-IB Interface

6940B MULTIPROGRAMMER USERS GUIDE *for the* HEWLETT-PACKARD 9825A CALCULATOR

HP Part No. 59500-90005 September 1976

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

Chapter I
INTRODUCTION

1-1 SCOPE

1-2 This guide provides programming, installation, and verification procedures for the 6940B multiprogrammer system as controlled by an HP 9825A Programmable Calculator using either the HP 98032A-Option 040 General Purpose I/O (GPIO) Interface or the HP 98034A HP-IB Interface card. The GPIO interface provides a 16-bit word directly to the 6940B while the HP-IB interface provides an 8-bit ASCII coded output which must be converted to the 16-bit format required by the 6940B. Thus, when the HP-IB interface is used, a 59500A interface unit must be employed to convert the ASCII code to the 6940B format. This guide includes condensed circuit descriptions of the 6940B/6941B multiprogrammer (Chapter VII) and the 59500A multiprogrammer interface (Chapter VIII). Although these descriptions do not provide a complete hardware analysis showing every signal and circuit, they are quite comprehensive and provide the user with a good understanding of the multiprogrammer system capabilities. The intent of this guide, therefore, is to provide all of the information to program and use the 6940B multiprogrammer with the 9825A calculator and minimize the need for constant referrals to other documents (manuals, data sheets, etc.).

1-3 The following is an abbreviated listing of all of the major information contained in this guide together with a brief description concerning the scope of the material.

(1) **Installation and Checkout.** Chapter II contains all of the information necessary for interconnecting the units, setting the address switches, and turning on the equipment in the proper sequence. Checkout procedures are included to verify that the equipment is connected correctly and that the major functions of the multiprogrammer system can be programmed by a 9825A calculator. Verification program listings are provided for both the GPIO interface and the HP-IB interface. The programs verify operation of the 6940B mainframe (and 59500A unit when using HP-IB interface) excluding plug-in cards and 6941B extender units. The verification programs are essentially "GO/NO–GO" checks with hardware diagnosis left to the Operating and Service Manuals for the 6940B and 59500A.

(2) **Programming Fundamentals.** Chapter III outlines the fundamentals of programming a 6940B/6941B multiprogrammer system with a 9825A using the GPIO interface; while Chapter IV outlines the fundamentals of programming a 59500A/6940B/6941B multiprogrammer system with a 9825A using the HP-IB interface. In addition, Chapter VII Paragraphs 7-40 through 7-65, include a theoretical explanation of the 6940B programming sequences, complete with flow diagrams. Note that the information in this guide relates only to programming requirements that are unique to the multiprogrammer system. It is assumed that the user is familiar with the fundamentals of programming a 9825A calculator.

(3) **Plug-In Card Descriptions and Programs.** Chapter V contains a brief functional description for each type of multiprogrammer plug-in card. In addition, both GPIO and HP-IB interface programming examples are provided for each card type.

(4) **Other Programming Techniques.** Chapter VI contains programming techniques such as using string variables and equating names with variables using the extended I/O ROM. This additional information may be helpful when programming the multiprogrammer system.

(5) **6940B and 59500A Circuit Descriptions.** As mentioned previously, simplified circuit descriptions of these two units are given in Chapters VII and VIII. Users who are not familiar with the multiprogrammer or have complex programming requirements are strongly urged to read these two chapters. The 59500A description (Chapter VIII) is pertinent only to operating the multiprogrammer on the HP-IB

(6) **Number System Theory and Utility Subroutines.** Appendix A contains descriptions of the number systems applicable to the multiprogrammer system. Included are explanations of the decimal, octal, binary, and BCD systems and how to convert from one system to another. Appendix A also contains utility subroutines which are additional programming aids for the user. The subroutines can be called from the users main program to perform required operations.

1-4 RELATED PUBLICATIONS

1-5 Other multiprogrammer literature that will be helpful to the user, includes the Operating and Service Manuals for the multiprogrammer mainframes (6940B and

6941B), the interface unit (59500A), and for each type of plug-in card. Each manual contains troubleshooting instructions, circuit theory, circuit diagrams and a replaceable parts list. Information on a more general level is contained in the 6940B multiprogrammer brochure (5952-3982 D).

1-6 Publications for the 9825A include the following:

	Publication	HP Part No.
9825A	Operating and Programming	09825-90000
9825A	Quick Reference Guide	09825-90010
9825A	String Variable Programming ROM	09825-90020
9825A	Advanced Programming ROM	09825-90021
9825A	General I/O Programming ROM	09825-90024
9825A	Extended I/O Programming ROM	09825-90025
98032A	16-bit Interface (GPIO) Installation and Programming	98032-90000
98034A	HP-IB Interface Installation and Programming	98034-90000

Chapter II
INSTALLATION AND CHECKOUT

2-1 This chapter provides procedures for interconnecting the multiprogrammer system with a 9825A calculator using either the General Purpose I/O (GPIO) interface card (98032A-Option 040) or an HP-IB interface card (98034A). The installation procedures include gathering the proper equipment, installing the applicable ROM's, setting address switches, and connecting the cables. Checkout procedures are included to verify that the equipment is installed correctly and that the multiprogrammer system can be programmed properly by the 9825A calculator. Paragraphs 2-2 through 2-15 provide installation and checkout procedures for the GPIO interface while paragraphs 2-16 through 2-34 provide similar coverage for the HP-IB interface.

2-2 GPIO INTERFACE

2-3 Equipment Required

2-4 The following equipment is required to assemble the system:

1. Programmable calculator HP 9825A.
2. GPIO interface (16-bit) card HP 98032A equipped with option 040.
3. ROM's: General I/O, Extended I/O, Advanced Programming, and String Variables
4. Multiprogrammer HP 6940B (plus 6941B extender units if needed).
5. Input/Output plug-in cards for 6940B/6941B multiprogrammer.
6. Interconnecting Cables:
   a. 9825A-to-6940B: Supplied with 98032A–Option 040 interface card.
   b. 6940B-to-6941B: Standard 18-inch (0.46 meter) chaining cable No. 14541A, purchased separately. Lengths up to 100 feet (30 meters) are available on special order.

2-5 Installing ROM's and GPIO Interface Card

2-6 Each ROM card can be plugged into any of the four slots in the front of the calculator (below the keyboard). ROM's are installed with the calculator switched off and with the ROM label right-side-up.

2-7 The 98032A-Option 040 GPIO interface card is configured to operate with the 6940B multiprogrammer and can be installed in any of the three slots in the rear of the calculator. The select code for the Option 040 GPIO card is preset to "9" which is the code assigned to the multiprogrammer system. Throughout this guide, all examples and sample programs for the GPIO interface use a select code of 9. Setting the select code and installing the GPIO interface card are described in Chapter 2 of the 98032A Installation and Programming Manual (98032A-90000).

2-8 Connecting Cables

2-9 Connect cables as shown in Figure 2-1. The Option 040 GPIO interface cable is equipped with the proper connector to mate with the J1 DATA INPUT connector on the rear of the 6940B multiprogrammer. One 14541A chaining cable is required for each 6941B extender unit used in the system. The standard length of the 14541A cable is 46cm (18 inches) with cables up to 30 meters (100 feet) available on special order.

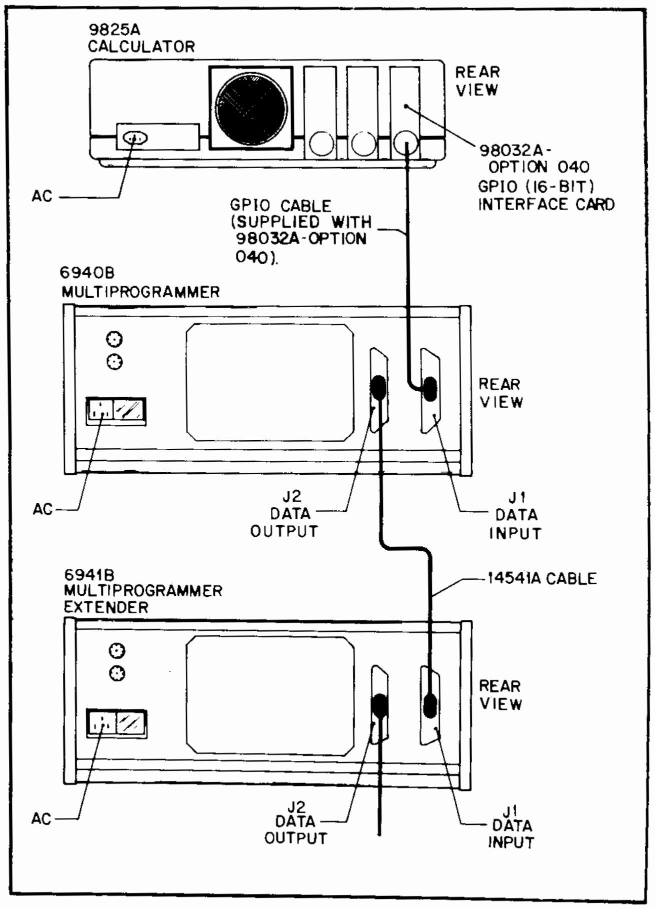


Figure 2-1. Interconnecting Cables, Multiprogrammer System/9825A (GPIO Interface)

2-10 Pre-Operational Checklist

2-11 Run through the following checklist before programming the system.

1. Correct ac line voltage is selected on rear of 6940B and 6941B's (if used). See Chapter 2 in 06940-90005 and 6941-90003 Operating and Service manuals.
2. 98032A-Option 040 GPIO interface card is installed with code "9" selected.
3. If desired, verify that the 6940B (excluding all plug-in cards and 6941B extenders) is operating properly by performing the checkout procedures given in Paragraphs 2-12 through 2-15. Checkout procedures for each type of plug-in card are provided in Chapter V.
4. Required ROM's are installed.
5. Required input/output plug-in cards are installed in 6940B and 6941B (if used) mainframe(s). Make a list of all 6940B/6941B plug-in cards and their mainframe slot addresses. This list will be useful when writing and debugging programs.

CAUTION

Ensure that power is turned off at the 6940B and 6941B's (if used) when inserting or removing cards. Failure to do so may result in equipment malfunctions.

6. Cables are connected (see Figure 2-1).
7. DATA SOURCE switch on 6940B front panel (see Figure 2-2) is in REMOTE position.
8. Power is applied to the 9825A, 6940B, and 6941B's (if used). Ensure that all 6941B's connected in the system are turned on.

2-12 Checkout Procedures

2-13 The following procedures verify that the major functions of the 6940B multiprogrammer can be programmed from a 9825A calculator using the GPIO interface. The programs are simple "GO/NO-GO" checks to establish that the system is operational. Complete hardware diagnosis of inoperable 6940B functions is provided in Operating and Service Manual HP Part No. 06940-90005. The verification programs are designed to exercise the 6940B only; do not install any plug-in cards in the 6940B mainframe. Also, do not connect a 6941B extender to the 6940B when performing the checkout procedures.

2-14 **Test Setup.** Use the following test setup for the GPIO verification tests.

a. Open door and remove all input/output plug-in cards from 6940B mainframe slots 400 through 414.

CAUTION

Ensure that power is turned off at the 6940B when removing plug-in cards.

b. Ensure that the A1 Standard Input, A2 Remote/Local, A3 Logic and Timing, and A5 Unit Select cards are installed in mainframe slots 100, 200, 300, and 500 respectively. These cards are clearly marked and must be installed to check operation of the 6940B.

c. Connect cables as shown in Figure 2-1, except do not connect a 6941B to the 6940B.

d. Install the General I/O, Extended I/O, and Advanced Programming ROM's in the slots below the 9825A keyboard.

e. Turn on the 9825A and the 6940B.

2-15 **Verification Tests.** It is recommended that the verification tests be executed in the order presented the first time the 9825A/6940B system is placed in operation. Later, after the system has been operational, the user can select any of the tests to verify function(s) that are suspected of being faulty. Each test contains operating instructions, a listing of the test program, and the test results (indications on the 6940B front panel and/or calculator display). If the test results are not as specified, the user should clear up the problem and rerun the test before continuing with subsequent verification tests. While the failure of a test often indicates that a hardware problem exists, certain operational or equipment configuration mistakes can be made and result in erroneous indications. If you get an incorrect indication, ensure that the test setup is correct and that the verification program was keyed in correctly (the check sum at the end of the keyed in program must match the check sum in the program listing).

NOTE

The program listings provided in the following GPIO verification tests are taken from an HP 9866B printer. These listings are provided only because 9866B print outs are easier to read than printouts from the 9825A's internal strip printer. Regardless of the printer used, make sure that the check sum at the end of your keyed-in program matches the check sum provided in the listing.

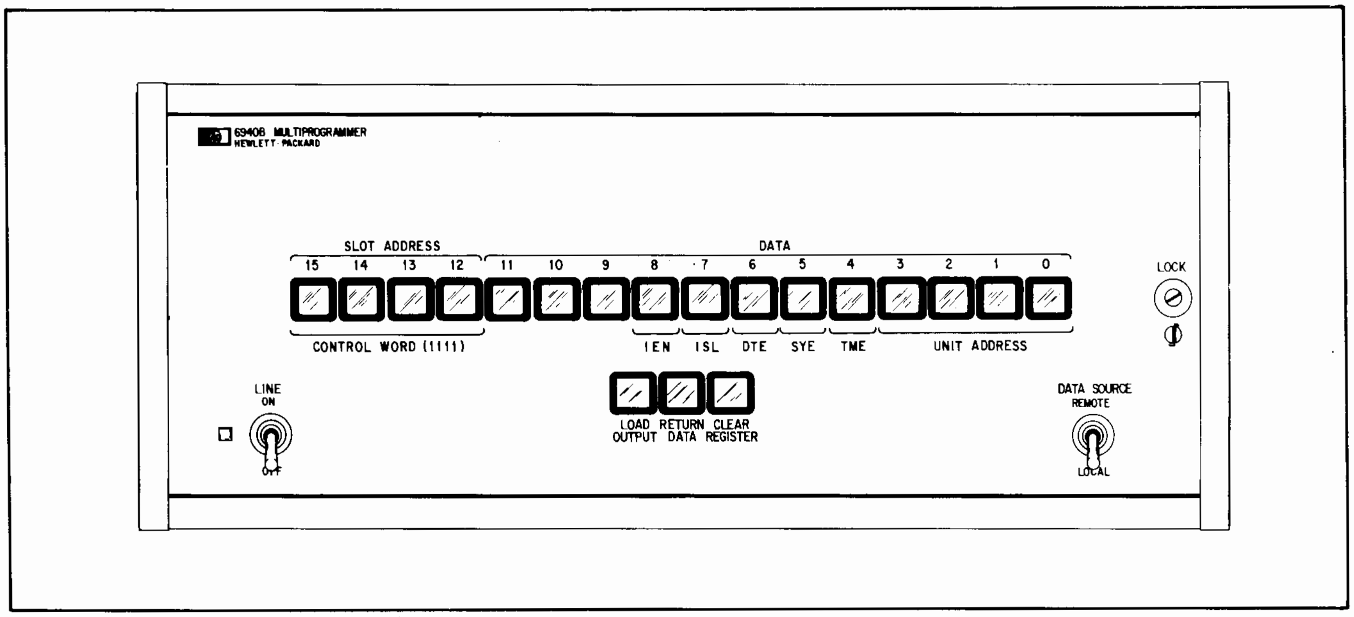


Figure 2-2. 6940B Multiprogrammer Front Panel

GP TS

1. **Interface Card Address Test.** The neral I/O and Extended I/O ROM's.
   a. Set DATA SOURCE switch on EMOTE.
   b. Ensure that the select code for card is set to "9".
   c. Key in this program and press F

```
10    WRITE IO
4;32
20    WRITE BIN 9
;-4096
30    WAIT 500
40    WRITE BIN 9          ;,40
;4095                      00
50    BEEP
60    WAIT 500
70    GOTO 20
```

   d. Bit indicators 15-12 should flash alternately with bit indicators 11-0 on 6940B front panel (see Figure 2-2).

2. **Output Data Test.** The following test requires use of the General I/O, Extended I/O, and Advanced Programming ROM's.
   a. Set DATA SOURCE switch on the 6940B front panel to REMOTE.
   b. Key in this program and press RUN.

```
0: moct;sfg 14
1: for N=0 to 15
2: wti 0,11;wti 4,dto(2↑N)
3: wait 500;next N
4: for N=0 to 15
5: wti 0,11;wti 4,dto(65535-2
6: wait 500;next N
7: gto 1
*23097
```

   c. Bit indicators on 6940B front panel should flash on one at a time and then of repeated while the program is running.

3. **Gate/Flag Test.** The following test requires use of the General I/O and Extended I/O Programming ROM's.
   a. Set DATA SOURCE switch on the 6940B front panel to REMOTE.
   b. Key in this program and press RUN.

```
0: moct
1: wti 0,11;wti 5,40;wait 200
2: wtb 9,170020,0;wait 200
3: gto 1
*15618
```

   c. LOAD OUTPUT indicator on 6940B front panel (Figure 2-2) should flash on an
   d. Press STOP, ERASE, and EXECUTE. Key in this program and press RUN.

```
0: moct;wti 0,11;wti 5,40
1: dsp "Put MP in LOCAL. Press CO
2: if iof9;dsp "RETURN DATA is no
3: if iof9=0;dsp "RETURN DATA is
4: gto 2
*19772
```

```
10    WRITE IO 9;
4;32
20    DISP "Put M
P in LOCAL.Press
 CONTINUE."
30    IF IOFLAG(9
)=1 THEN DISP "R
ETURN DATA IS NO
T PRESSED."
40    IF IOFLAG(9
)=0 THEN DISP "R
ETURN DATA IS PR
ESSED."
50    GOTo 30
```

   e. Follow instructions on calculator display.
   f. With 6940B DATA SOURCE switch in LOCAL, press and hold RETURN DATA calculator display should read "RETURN DATA is pressed".
   g. Release RETURN DATA switch. Display should read "RETURN DATA is not p

4. **Data Input Test.** The following test requires use of the General I/O and Extended I/
   a. Set DATA SOURCE switch on the 6940B front panel to REMOTE.
   b. Key in this program and press RUN.

```
0: dsp "Put MP in REMOTE. Press CONTINUE";stp
1: moct;wti 0,11;wti 5,40;wtb 9,170000
2: prt "Throw MP from","REMOTE to LOCAL.";spc 2
3: wti 0,11;fxd 0;dsp rdi 4;jmp 0
*23202
```

   c. Follow the instructions on the calculator display and printer.
   d. While the program is running, turn on all 16 data switches (0-15) on the 6940B front panel. The calculator display should read 1Ø7777.

NOTE

Data switches 12, 13, and 14 have no affect on the calculator display. They are not connected to the calculator in the input mode.

   e. Touch the CLEAR REGISTER switch on the 6940B (see Figure 2-2). The calculator display should read Ø.
   f. As you turn on any combination of data switches (15 and 11-Ø) on the 6940B, the calculator should display the octal value of the data input.

2-16 HP-IB INTERFACE

2-17 Equipment Required

2-18 The following equipment is required to assemble the system.

1. Programmable calculator HP 9825A.
2. HP-IB interface card HP 98034A
3. ROM's: General I/O, Extended I/O, Advanced Programming, and String Variables.
4. Multiprogrammer HP 6940B (plus 6941B extender units if needed).
5. Input/Output plug-in cards for 6940B/6941B multiprogrammer.
6. Multiprogrammer interface unit 59500A.
7. Interconnecting cables:
   a. 9825A-to-59500A: One cable is supplied with 98034A HP-IB interface card. Extra cables can be ordered separately.
      10631A HP-IB cable 1 meter (3.3 ft)
      10631B HP-IB cable 2 meters (6.6 ft)
      10631C HP-IB cable 4 meters (13.2 ft)
   b. 59500A-to-6940B: Standard 18-inch (0.46 meter) chaining cable No. 14541A, supplied with 59500A. Lengths up to 100 feet (30 meters) are available on special order.
   c. 6940B-to-6941B: Standard 18-inch (0.46 meter) chaining cable No. 14541A, purchased separately. Lengths up to 100 feet (30 meters) are available on special order.

2-19 Installing ROM's and HP-IB Interface Card

2-20 Each ROM card can be plugged into any of the four slots in the front of the calculator (below the keyboard). ROM's are installed with the calculator switched off and with the ROM label right-side-up.

2-21 The 98034A interface card provides HP-IB capability for the 9825A calculator and can be installed in any of the three slots in the rear of the calculator. The select code for the HP-IB card is preset to "7". Throughout this guide, all examples and sample programs for the HP-IB interface use a select code of "7". Setting the select code and installing the HP-IB interface card are described in Chapter 2 of the 98034A Installation and Programming Manual (98034A-90000).

2-22 Setting Multiprogrammer System Addresses

2-23 The talk and listen addresses for the multiprogrammer system are selected by address switches on the rear of the 59500A interface unit. The switches are set to the suggested talk address of "W" and listen address of "7" as the unit is shipped from the factory. The "W" and "7" correspond to an address of "23" as defined in the 9825A General I/O Programming Manual (09825-90024) Chapter 4. Throughout this guide, all examples and sample programs for the HP-IB interface use a multiprogrammer address of "23".

2-24 As shown in Figure 2-3, there are seven address switches on the rear of the 59500A. The last two switches, 6 and 7, are ignored. Check that switches 1 through 3 and 5 are set to "1" and switch 4 is set to "0" to select a talk address of "W" and a listen address of "7".

2-25 Connecting Cables

2-26 Connect cables as shown in Figure 2-4. The HP-IB interface card 98034A is equipped with the proper cable to mate with the J1 HP-IB connector on the rear of the 59500A multiprogrammer interface unit. The 14541A chaining cable connecting the 59500A to the 6940B is supplied with the 59500A unit. Additional 14541A chaining cables must be purchased separately. One cable is required for each 6941B extender unit used in the system. The standard length of the 14541A cable is 46cm (18 inches) with cables up to 30 meters (100 feet) available on special order.

2-27 Pre-Operational Checklist

2-28 Run through the following checklist before programming the system.

1. Correct ac line voltage is selected on rear of 6940B, 59500A, and 6941B's (if used). See Chapter 2 in 06940-90005, 59500-90001 and 06941-90003 Operating and Service manuals.
2. 98034A HP-IB interface card is installed with code "7" selected.
3. Address switches on rear of 59500A are set for a multiprogrammer talk address of "W" and listen address of "7" (see Paragraph 2-24).
4. If desired, verify that the 59500A/6940B system (excluding all plug-in cards and 6941B extenders) is operating properly by performing the checkout procedures given in Paragraphs 2-31 through 2-34. Checkout procedures for each type of plug-in card are provided in Chapter V.
5. Required ROM's are installed.
6. Required input/output plug-in cards are installed in 6940B and 6941B (if used) mainframe(s). Make a list of all 6940B/6941B plug-in cards and their mainframe slot addresses. This list will be useful when writing and debugging programs.

——— CAUTION ———

Ensure that power is turned off at the 6940B and 6941B's (if used) when inserting or removing cards. Failure to do so may result in equipment malfunctions.

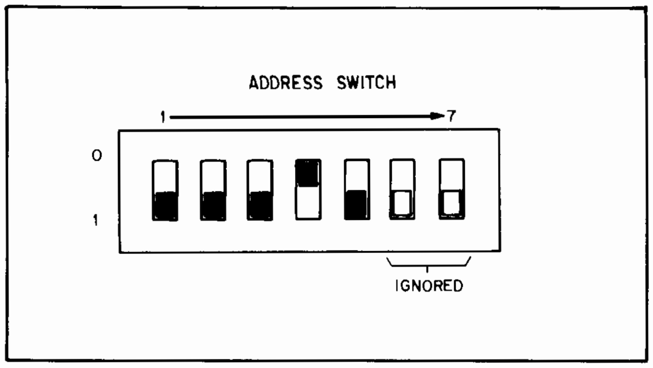


Figure 2-3. Address Switches on Rear of 59500A

7. Cables are connected (see Figure 2-4).
8. DATA SOURCE switch on 6940B front panel (see Figure 2-2) is in REMOTE position.

2-29 System Turn-On Sequence

2-30 To prevent the random setting of the 59500A's service request (SRQ) and the multiprogrammer's system enable (SYE) signal at turn-on, the instruments must be turned-on in the following sequence:

1. Turn on calculator (and printer if applicable).
2. Set 6940B to LOCAL and then turn power on.
3. Turn on 59500A and 6941B's (if used). Ensure that all 6941B's connected in the system are turned on.
4. Now switch 6940B to REMOTE.

2-31 Checkout Procedures

2-32 The following procedures verify that the major functions of the 59500A/6940B system can be programmed from a 9825A calculator using the HP-IB interface. The programs are simple "GO/NO-GO" checks to establish that the system is operational. Complete hardware diagnosis of inoperable functions is provided in Operating and Service manuals for the 59500A and 6940B units. The verification programs are designed to exercise the 59500A/6940B only; do not install any plug-in cards in the 6940B mainframe. Also, do not connect a 6941B extender to the 6940B when performing the checkout procedures.

2-33 Test Setup. Use the following test setup for the HP-IB verification tests.

a. Open door and remove all input/output plug-in cards from 6940B mainframe slots 400 through 414.

——— CAUTION ———

Ensure that power is turned off at the 6940B when removing plug-in cards.

b. Ensure that the A1 Standard Input, A2 Remote/Local, A3 Logic and Timing, and A5 Unit Select cards are

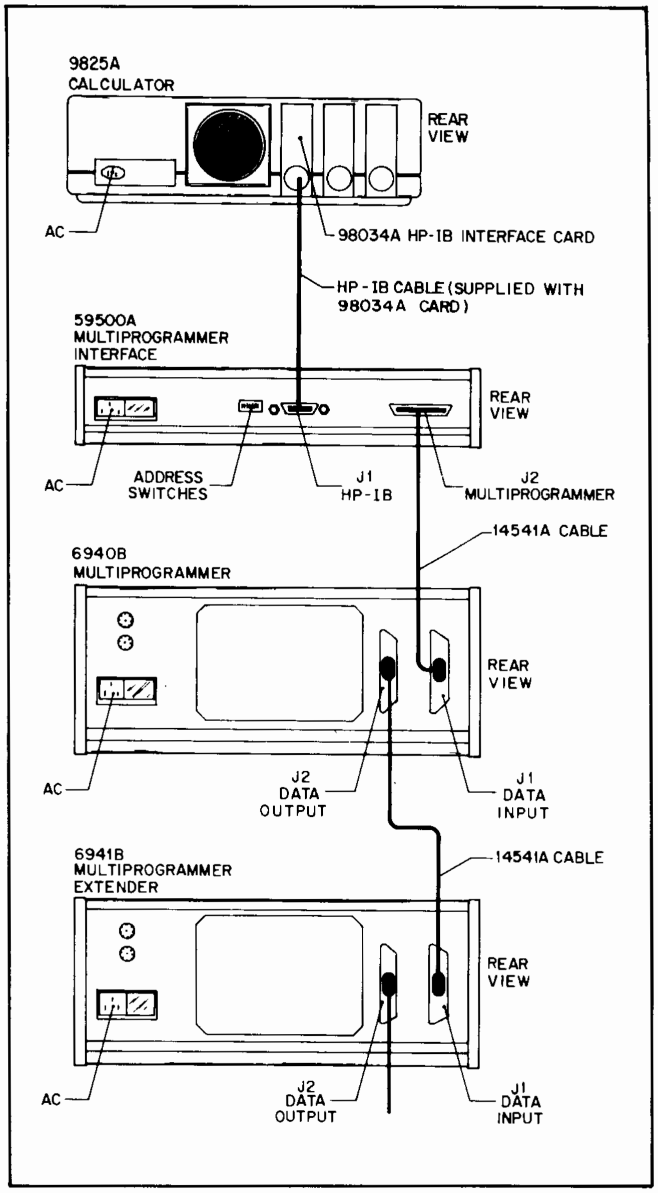


Figure 2-4. Interconnecting Cables, Multiprogrammer System/9825A (HP-IB Interface)

installed in mainframe slots 100, 200, 300, and 500 respectively. These cards are clearly marked and must be installed to check operation of the 6940B.

c. Connect cables as shown in Figure 2-4, except do not connect a 6941B to the 6940B.

d. Install the General I/O, Extended I/O, and Advanced Programming ROM's in the slots below the 9825A keyboard.

e. Turn on the system (see Paragraph 2-30).

2-34 Verification Tests. It is recommended that the verification tests be executed in the order presented the first time the 9825A/59500A/6940B system is placed in operation. Later, after the system has been operational, the user can select any of the tests to verify function(s) that are suspected of being faulty. Each test contains operating instructions, a listing of the test program, and the test results (indications on the 59500A/6940B front panels and/or calculator display). If the test results are not as specified, the user should clear up the problem and rerun the test before continuing with subsequent verification tests. While the failure of a test often indicates that a hardware problem exists, certain operational or equipment configuration mistakes can be made and result in erroneous indications. If you get an incorrect indication, ensure that the test setup is correct and that the verification program was keyed in correctly (the check sum at the end of the keyed-in program must match the check sum of the program listing).

NOTE

The program listings provided in the following HP-IB verification tests are taken from an HP 9866B printer. These listings are provided only because 9866B print outs are easier to read than print outs from the 9825A's internal strip printer. Regardless of the printer used, make sure that the check sum at the end of your keyed-in program matches the check sum provided in the listing.

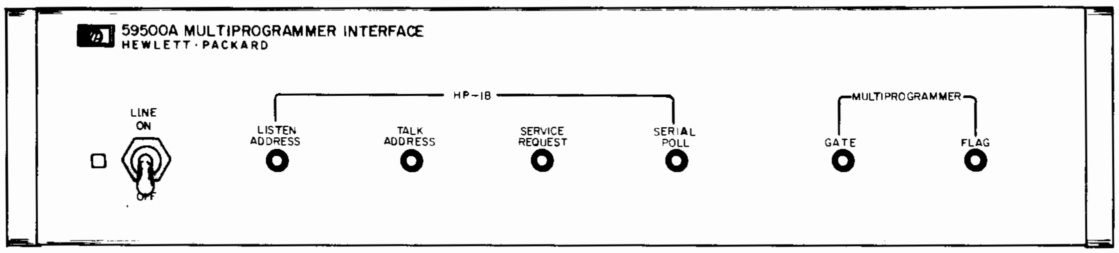


Figure 2-5. 59500A Multiprogrammer Interface, Front Panel

HP-IB VERIFICATION TESTS

1. **Talk/Listen Test.** This test verifies that the 59500A responds to its listen and talk addresses. The General I/O ROM is required to perform the following test.
   a. Set DATA SOURCE switch on 6940B front panel (see Figure 2-2) to REMOTE.
   b. Key in this program and press RUN.

```
0: wrt 723;wait 200;red 723;wait 200;jmp 0
*20565
```

   c. LISTEN ADDRESS and TALK ADDRESS indicators on 59500A front panel (see Figure 2-5) should alternately light.

2. **Data Output Test.** This test verifies that the 59500A correctly outputs valid data patterns to the 6940B. The General I/O, Extended I/O, and Advanced Programming ROM's are required to perform the following test.
   a. Set DATA SOURCE switch on the 6940B front panel to REMOTE.
   b. Key in this program.

```
0: for N=0 to 11;fmt 5,c,f4.0,"X",z
1: wrt 723.5,"OT@",dto(2↑N);wait 400
2: next N
3: wrt 723.5,"A";wait 400
4: wrt 723.5,"B";wait 400
5: wrt 723.5,"D";wait 400
6: wrt 723.5,"H";wait 400
7: gto 0
*57
```

c. Press RUN and check that each of the 16 data indicators on the 6940B front panel (Figure 2-2) flash on one at a time.
d. Press STOP, ERASE, and EXECUTE. Key in this program and press RUN.

```
0: for N=0 to 11;fmt 5,c,f4.0,"X",z
1: wrt 723.5,"OTO",dto(4095-2↑N);wait 400
2: next N
3: wrt 723.5,"N7777";wait 400
4: wrt 723.5,"M7777";wait 400
5: wrt 723.5,"K7777";wait 400
6: wrt 723.5,"G7777";wait 400
7: gto 0
*29705
```

e. Data indicators on 6940B will flash off one at a time.

3. **Gate/Flag Test.** This test verifies the gate/flag features of the 59500A/6940B. The General I/O ROM is required to perform the following test.
a. Set DATA SOURCE switch on the 6940B front panel to REMOTE.
b. Key in this program and press RUN.

```
0: fmt 1,c,z;wrt 723.1,"O20T@T";wait 400;wrt 723.1,"X";wait 400;jmp 0
*6589
```

c. GATE indicator on 59500A (Figure 2-5) and LOAD OUTPUT indicator on 6940B (Figure 2-2) should flash on and off together.
d. Set DATA SOURCE switch on 6940B to LOCAL. GATE indicator on 59500A should keep on flashing but the LOAD OUTPUT on 6940B should not flash.
e. Press and hold RETURN DATA on 6940B (Figure 2-2). GATE on 59500A stays off, FLAG on 59500A (Figure 2-5) is on. GATE resumes flashing when RETURN DATA is released.

4. **Data Input Test.** This test instructs the user on how to input data to the 9825A from the 6940B/59500A, and then verifies if the data is correct. The General I/O ROM is required to perform the following test.
a. Set DATA SOURCE switch on the 6940B front panel to REMOTE.
b. Key in this program and press RUN.

```
0: dsp "Put MP in REMOTE. Press CONTINUE";stp
1: fmt 1,c,z;wrt 723.1,"OT@Z"
2: prt "Throw MP from","REMOTE to LOCAL.";spc 2
3: red 723,X;fxd 0;dsp X;jmp 0
*27926
```

c. Follow the instructions on the calculator display and printer.
d. While the program is running, turn on all 16 data switches (0-15) on the 6940B front panel. The calculator display should read 17777.

NOTE

Data switches 12, 13, and 14 have no affect on the calculator display. These bits are not connected to the calculator in the input mode.

e. Touch the CLEAR REGISTER switch on the 6940B (Figure 2-2). The calculator display should read Ø.
f. As you turn on any combination of data switches (15 and 11-0) on the 6940B, the calculator should display the octal value of the data input.

5. Serial Poll Test. This test verifies the serial poll mode capabilities of the 59500A/6940B. The General I/O and Extended I/O ROM's are required to perform the following test.

a. Set DATA SOURCE switch on the 6940B front panel to REMOTE.

b. Key in this program and press RUN.

```
0: mdec;fmt 1,c,z;wrt 723.1,"020T";wait 500;rds(723)→C;wait 500
1: if C#64;dsp "Status error";beep;stp
2: rds(723)→C;if C#0;dsp "SRQ not reset";beep;stp
3: gto 0
*4241
```

c. On the 59500A (Figure 2-5), the SERVICE REQUEST and LISTEN ADDRESS indicators should blink on together, alternating with the TALK ADDRESS indicator. (In a dimly lit room, notice that the SERIAL POLL indicator flickering each time the other indicators change).

Chapter III
GPIO PROGRAMMING FUNDAMENTALS

3-1 This chapter outlines the fundamentals of programming the multiprogrammer system with the 9825A calculator using the General Purpose I/O (GPIO) interface (98032A-Option 040). Included in this chapter are explanations and examples showing how to program the basic word formats, as well as general programming techniques for multiprogrammer input and output cards. All examples assume the GPIO interface is assigned a select code of 9.

3-2 CALCULATOR OPERATING MODES

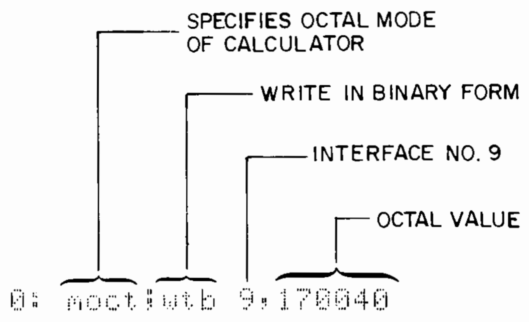
3-3 The 9825A calculator operates in the decimal mode (mdec) when switched on. However, all data transfers between the calculator and multiprogrammer will use the octal mode (moct). Upon completion of the data transfer with the multiprogrammer, the user may re-specify the decimal mode for I/O operations with other devices, if desired. This is an important point to remember since not knowing what mode of operation the calculator is operating in can cause a substantial amount of confusion. It is recommended that the user read chapter 2 of the Extended I/O ROM Manual, HP Part No. 09825-90025, for an understanding of Decimal and Octal operating modes.

3-4 MULTIPROGRAMMER WORD FORMATS

3-5 The next portion of this chapter is devoted to the fundamentals of communicating between the calculator and the multiprogrammer system (6940B/6941B). For calculator-to-multiprogrammer exchanges, three different types of codes (words) can be sent and for multiprogrammer-to-calculator exchanges one type of word is received. The following paragraphs describe the basic construction and purpose of each word type, showing how to send the output words or read in the input word. Each word type is treated separately without too much regard for the programming sequences which can be used to accomplish a specific task. Programming examples, using these basic word formats, are included later in this chapter, under general programming techniques.

3-6 Example 1 illustrates the technique that would be used to write an octal value in binary form to interface number 9.

Example 1. Writing an Octal Value to Interface Number 9



```
0: moct;wtb 9,170040
```

3-7 Calculator Output Words

3-8 Three types of output words are sent from the calculator to the multiprogrammer system: control words, data words, and address words.

3-9 **Control Word.** Usually, the first word sent to a multiprogrammer system will be a control word. It is used to specify a unit address and the operating mode of the multiprogrammer system. The unit address portion selects which mainframe in the chain (of one 6940B Multiprogrammer and up to fifteen 6941B Extenders) will respond to subsequent data or address words from the calculator. The mode portion of the word selects one of several output or input modes of multiprogrammer operation (System Enable, Timing Mode Enable, Input Select, etc.). How the multiprogrammer system responds to these various control word modes is described in Chapter VII of this guide, as well as in the 6940B Service Manual. A good understanding of these modes is essential for effective programming involving specific applications.

3-10 Figure 3-1 illustrates the structure of a control word as transmitted by the GPIO. Octal 17 in the address field identifies the word as a control word and tells the 6940B unit that the data is to be interpreted as control mode and unit address information.

3-11 The data field consists of four octal numbers. The most significant digit, D_4, is always zero in a control word. Digits D_3 and D_2 specify the operating mode as indicated by the codes shown in Table 3-1. Only the most commonly used codes are shown in Table 3-1; but combinations for other codes are possible and become obvious through inspection.

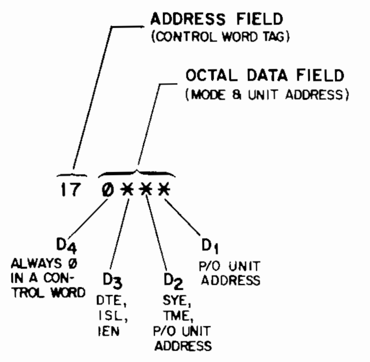


Figure 3-1. Control Word

Table 3-1. Operating Mode Codes

Multiprogrammer Operating Mode		Data Field*
Output Mode	Input Mode	
SYE Off		ØØØØ
SYE On		ØØ4Ø
DTE, SYE On		Ø14Ø
DTE, SYE, TME On		Ø16Ø
	ISL, SYE On	Ø24Ø
	ISL, SYE, TME On	Ø26Ø
	IEN, SYE, TME On	Ø46Ø

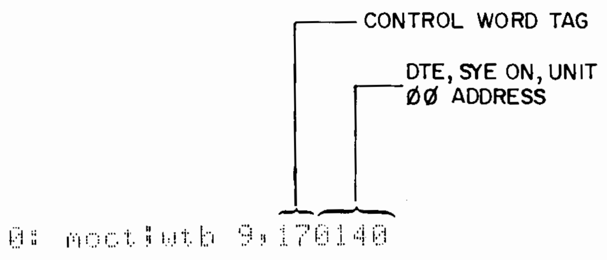
*Data field codes shown are valid for a unit address of ØØ only.

3-12 Digits D_2 and D_1 specify the unit address. The first unit in the chain (6940B, Master) has an address of ØØ, the second unit (6941B Extender) has an address of Ø1, and so forth, up to a unit address of 17_8 (15_{10}). The two unit address digits are simply added (in octal) to the last two control mode digits shown in Table 3-1 to obtain the final data field code. For example, if the user wants to select unit number one (second unit in chain) and program the DTE, SYE, and TME modes, he would send a code of Ø161 in the data field. To select unit number 15 (last unit) and the same modes, he must send an Ø177 ($Ø16Ø_8 + 17_8$).

3-13 A typical control word, with DTE and SYE on, would be sent as shown in Example 2.

3-14 Data Word. Data words select and control output cards and a few "special purpose" input cards. The data word selects an individual card within the mainframe previously specified by a control word.

Example 2: Typical Control Word



3-15 Figure 3-2 shows the basic construction of a data word sent from the calculator. The address field selects the particular card (one of 15 slots in a mainframe) that will receive the octal data in the data field portion of the word. Table 3-2 shows the card slot address that corresponds to octal data in the address field. The data field consists of four octal digits that can be of any value between ØØØØ and 7777 (Appendix A contains an explanation on converting numbers from octal to decimal and vice-versa).

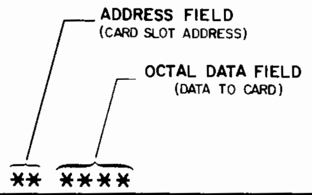
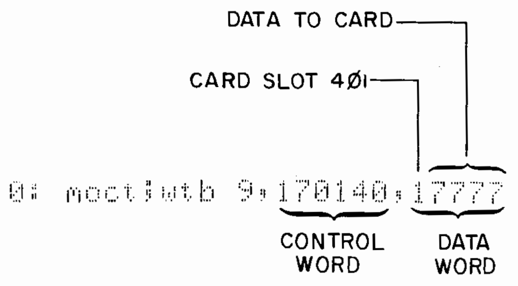


Figure 3-2. Data Word

3-16 Example 3 shows sending a typical data word preceded by the same sample control word shown in Example 2. Notice that it is not necessary to include leading zeroes in the card address field. In Example 3, an output card (in slot 401, Unit ØØ) is directed to output data represented by the octal number "7777".

Example 3: Typical Data Word



3-17 Address Word. An address word selects the input card that is to send data (a return data word) back to the

Table 3-2. Card Slot Addresses

OCTAL DATA IN ADDRESS FIELD	MULTIPROGRAMMER CARD SLOT ADDRESS
ØØ	Slot 400
Ø1	Slot 401
Ø2	Slot 402
Ø3	Slot 403
Ø4	Slot 404
Ø5	Slot 405
Ø6	Slot 406
Ø7	Slot 407
1Ø	Slot 408
11	Slot 409
12	Slot 410
13	Slot 411
14	Slot 412
15	Slot 413
16	Slot 414

calculator. Before an address word can be sent, a control word must have first selected an input operating mode (and the unit address).

3-18 Figure 3-3 shows the basic construction of an address word sent from the calculator. The address field contains the card slot address as in a data word, but the data field contains zero data (four octal zeroes). Zero data is only used for convenience in visualizing address words and, in fact, if the input mode has been selected (ISL on) a data word sent to the multiprogrammer would be interpreted as an address word and the data field would have no effect.

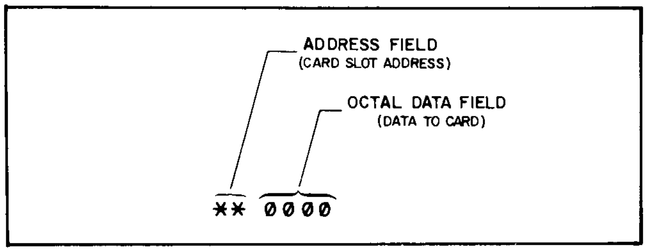
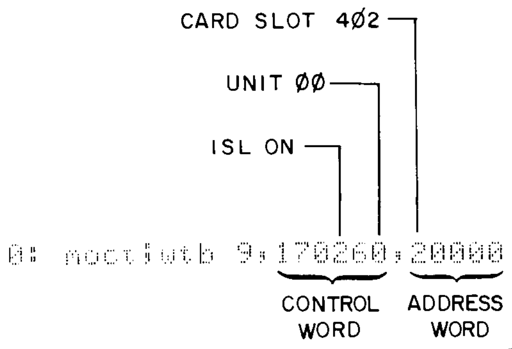


Figure 3-3. Address Word

3-19 As will be explained in detail later in this chapter, there are two ways of sending address words to input cards; with or without a gate. A write binary statement (wtb) is used to address a card with a gate, and the write interface (wti) function is used to address a card without a gate. Both methods require that a control word be sent first specifying the input mode of operation, followed by the address word specifying the card slot.

3-20 Example 4 shows a typical address word sent to an input card installed in multiprogrammer slot 402. Notice that the preceding control word calls for the input mode of operation (ISL on). This example uses a write binary statement (wtb) to address the card with a gate.

Example 4: Typical Address Word



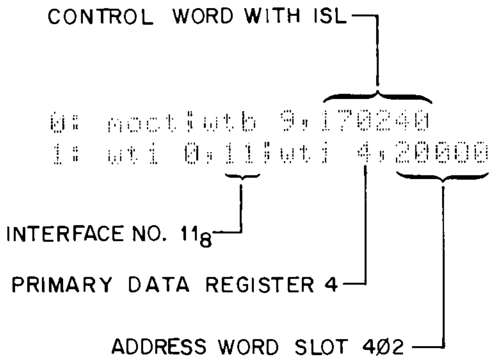
```
0: moct;wtb 9,170260,20000
```

NOTE

Since TME is included in the control word of Example 4, do not attempt to execute this example without first reading the description of the timing mode operation as it applies to input cards (Paragraph 3-80).

3-21 Example 5 shows the same address word as Example 4, being sent without a gate to slot 402 by means of the write interface function (wti).

Example 5: Sending an Address Word without a Gate



```
0: moct;wtb 9,170240
1: wti 0,11;wti 4,20000
```

3-22 In line 1 of Example 5, the first wti function selects the GPIO interface. Note that a select code of 11_8 is used because the wti function is affected by the octal mode of the calculator (octal 11 equals decimal 9). The second wti function in line 1 selects primary data register 4 in the GPIO. A general description of the interface control registers is given in Chapter 4 of the Extended I/O ROM Manual, HP Part No. 09825-90025.

3-23 Summary. Figure 3-4 summarizes the requirements for constructing control, data, and address words.

Control Word = 170000 + control data (Table 3-1)
Data Word = Octal card address (Table 3-2) X 10000 + output data (in octal)
Address Word = octal card address (Table 3-2) X 10000

Figure 3-4. Constructing Control, Data, and Address Words

3-24 Calculator Input Word

3-25 Addressing a card will cause the card to send its data back to the GPIO interface of the calculator. Then, the read interface function (rdi) is used to transfer the data from the interface into a variable in the calculator.

3-26 Example 6 illustrates reading data from an input card and storing it in variable X. It is assumed that the calculator was previously placed in the octal mode, the multiprogrammer input mode (ISL on) was selected, and the input card in slot 402 was addressed and gated (see Example 4).

Example 6. Reading Input Card Data

```
1: wti 0,11;rdi 4→X
```

3-27 If the card is addressed without a gate, (Example 5), it is merely necessary to follow the write interface function specifying the slot address with the read interface function. For example, line 1 of Example 5 would then become:

```
1: wti 0, 11; wti 4, 20000; rdi 4→X
```

3-28 Words sent from the multiprogrammer to the calculator are known as return data words. Return data words contain information from an input card previously selected by an address word. Each word (Figure 3-5) contains an input request character, a zero character, and four octal data characters.

3-29 The input request (IRQ) character can be either a "1" or a "0" A "1" indicates that the input card has returned a flag and its input data is valid. Moreover, during an interrupt sequence (TME and IEN on), a "1" in the IRQ field indicates that this card has generated an interrupt. A "0" indicates that: (1) the input card has not returned a flag (card is still busy or has not interrupted) and its' data is not valid; or (2) the card does not contain an IRQ function. Some input cards; such as the voltage monitor card (69421A), do not contain an IRQ function and a "0" in the IRQ field can be ignored.

3-30 The zero character located between the IRQ and the octal data field represents multiprogrammer bits 12-14 which are not returned to the calculator and therefore are always zero.

3-31 The four data characters, representing the input information from the multiprogrammer card, can be any octal number, between 0000 and 7777. Hence a return data word with all "7" 's in the data field and a "1" in the IRQ field would appear as a 107777 on the calculator display.

3-32 Return data words, stored in variables, can be utilized directly unless the word is obtained from a card that contains the IRQ function. In these cases, the return data must first be checked to determine if there is a "1" in the IRQ field. If the IRQ character is a "1", it must be subtracted from the return data word to yield the card's input data. How to check for IRQ and subtract it, if necessary, is described under general programming techniques for input cards in this chapter and in the applicable input card programs of Chapter V.

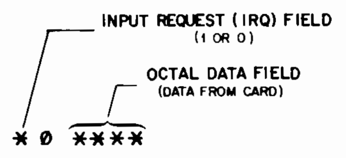


Figure 3-5. Return Data Word

3-33 GENERAL PROGRAMMING INFORMATION

3-34 The next portion of this chapter describes the programming sequences associated with the various plug-in cards of the multiprogrammer system. If a detailed description of multiprogrammer operation is desired, see Chapter VII.

3-35 Programming Aids

3-36 Number System Theory. Appendix A contains a description of the number systems applicable to the multiprogrammer system. Included are explanations of the decimal, octal, binary, and BCD systems, how to convert numbers from one system to another, and how the number systems relate to some of the plug-in cards.

3-37 Control Mode Reference Table. Table 3-3 lists all of the plug-in cards available at the time of this writing and which operating mode bit (in a control word) will affect the operation of each card. Also included is an IRQ column which indicates the cards that contain IRQ cicruitry.

3-38 The following two examples show how to use Table 3-3:

1. If the user has a 69321B D/A output card, he can see at a glance that it will be affected by the DTE and SYE control mode bits. He can also determine that if TME is on (extended handshake), it will require 30μsec (nominal) for the card to return a CTF flag.

2. If the user has a 69431A digital input card, he can deduce that ISL must be on to read-in data from the card and that the card has interrupt capability (IEN and IRQ). The user can also determine from the table that this card is capable of providing a gate/flag data transfer cycle with its external device when TME is programmed on. In this case, the length of the flag period is, of course, determined by the external device (Note 1 of table).

3-39 Output Cards, General Programming Techniques

3-40 Three multiprogrammer control mode bits (SYE, DTE, and TME) are used when programming output cards. Table 3-3 lists the cards affected by these bits and Table 3-1

Table 3-3. Control Mode Summary for Plug-in Cards

MODEL	DESCRIPTION	IEN	ISL	DTE	SYE	Flag (CTF) From Card (4)	IRQ
69321B	D/A Voltage	–	–	Yes	Yes	30μsec	–
69325A	BPSA Voltage Cont.	–	–	–	Yes	9.5msec	–
69326A	BPSA Current Cont.	–	Yes	–	Yes	4.3msec	–
69327A	BPSA Current Cont.	–	Yes	–	Yes	4.3msec	–
69328A	BPSA Gain Cont.	–	–	–	Yes	9.5msec	–
69330A	Relay Output	–	–	Yes	Yes	12msec (1)	–
69331A	Digital Output	–	–	Yes	Yes	(1)	–
69332A	Open Collector Output	–	–	–	–	(5)	–
69335A	Stepping Motor	–	–	–	Yes	(2)	–
69370A	D/A Current	–	–	Yes	Yes	30μsec	–
69380A	Breadboard Output	–	–	–	–	(5)	–
69421A	Voltage Monitor	–	Yes	–	–	6msec	–
69430A	Iso. Digital Input	–	Yes	–	–	(5)	–
69431A	Digital Input	Yes	Yes	–	–	(1)	Yes
69433A	Relay Output/Readback	–	Yes	–	Yes	4msec	–
69434A	Event Sense	Yes	Yes	–	–	(3)	Yes
69435A	Pulse Counter	–	Yes	–	–	(5)	–
69436A	Process Interrupt	Yes	Yes	–	–	(3)	Yes
69480A	Breadboard Input	–	Yes	–	–	(5)	–
69500-06A	12 Bit Resistance	–	–	–	Yes	6msec	–
69510-13A	Dual 6 Bit Resistance	–	–	–	–	6msec	–
69600B	Programmable Timer	Yes	Yes	Yes (7)	–	(2)	Yes
69601B	Frequency Reference (6)	–	–	–	–	(5)	–

(1) Max time depends on flag from external device.
(2) Time depends on number of pulses and pulse frequency.
(3) Max time depends on external inputs to detectors.
(4) CTF flag is ignored unless TME has been programmed on.
(5) These cards have no CTF circuitry and must not be sent a data or address word with a gate ("T") while TME is on. If this is done, no flag will be returned from the card causing the system to hang-up.
(6) Free-running frequency generator (not programmable).
(7) With jumper W4 installed.

shows the four control mode codes most often used for output card programming. Explanations of sending constants and variables to output cards as well as descriptions of each control mode (SYE, DTE, TME) are given in the following paragraphs. Programming examples are included where applicable.

NOTE

A unit address of ØØ is used in all control words shown in the programming examples.

3-41 Sending Constants and Variables. As a rule, output cards are programmed with control words and data words. (There are some output cards that can also be operated in the input mode as will be pointed out in the individual card programming chapter.) The data field portion of a data word must contain an octal constant in the range from "ØØØØ" to "7777". To send a variable value, assume the data field consists of 4 zeroes (multiply the address field by 1ØØØØ) and add the variable data to the data word. Write Binary (wtb) statements will always be used to send constants or variables.

3-42 Example 7 illustrates sending a data word containing a constant data value to an output card. In this example, a data value of "7777" is sent to the output card in slot 402.

Example 7: Sending a Constant Data Value

```
                                   DATA WORD ─┐
                   CONTROL WORD ─┐            │
                                 │            │
                             ────┴────   ─────┴──
Ø: moct;wtb 9,17Ø14Ø,27777
```

3-43 Example 8 shows how to send a variable data value to an output card. Data is represented by variable A and can be any positive octal value between "ØØØØ" and "7777". Note that if the variable were equal to "7777", Example 8 would perform the same operation as Example 7.

3-44 Line 1Ø in Example 8 allows the user to work with decimal values when calculating variables. In this case, the decimal to octal function (dto) of the calculator is used to convert a decimal value in variable A to octal. In addition, line 1Ø checks to see if the octal value exceeds 12 bits (7777 octal) and, if it does, goes to a user provided error routine. Of course, if variable A is already an octal value, the dto function must be omitted from line 1Ø.

3-45 If the dto function was not used and variable A was not an octal value, error E6 would occur when the calculator attempted to execute line 11. If variable A was an octal value exceeding 7777, it would change the card slot address if added to the data word. This is why the value of A is tested for a maximum of 7777 in line 1Ø before being added.

3-46 System Enable (SYE). The SYE control mode is used to control the outputs of most output cards (see Table 3-3). To enable the outputs, SYE must be programmed on. When SYE is programmed off, the outputs of all cards affected by SYE are disabled; i. e., are placed in a predetermined condition. For example, resistance outputs are shorted, voltage outputs are held at 0 volts and digital outputs are held in the high or zero state, depending on the card option.

3-47 As shown in previous examples, SYE can be programmed on in a control word prior to sending the data word to the output card. In some instances, the user may wish to take advantage of the SYE off condition which removes the outputs from all affected output cards simultaneously. In this case, transmitting a control word with SYE off will cause all affected cards to go to a predetermined condition. Example 9 shows a control word with SYE off. When Example 9 is executed, all output cards responding to SYE will go to a predetermined condition.

Example 8: Sending a Variable Data Value

```
1Ø: dtoA→A;if A>7777;gto "error"
11: moct;wtb 9,17Ø14Ø,2ØØØØ+A
```

Explanation:

1Ø: Conversion of variable A from decimal to octal (optional).
11: Control word is sent to multiprogrammer (DTE, SYE on). Variable A is then added to data word and sent to multiprogrammer output card in slot 402.

Example 9: Transmitting a Control Word With SYE Off

```
0: noct;wtb 9,170000
```

3-48 Thus, all output cards affected by SYE are in a predetermined condition when SYE is off and will output data when SYE is on, whether or not the card is addressed.

3-49 Data Transfer Enable (DTE). The DTE control mode affects output cards having either external gate/flag capability or dual rank storage. DTE need not be programmed on if a particular card does not have either of these capabilities, but no harm is done if it is programmed on. When using the timing mode, it is generally good practice to program both TME and DTE on together. The reason for this is that the four cards affected by DTE (69321B, 69330A, 69331A, and 69370A) cannot drive the CTF flag line unless DTE is on. Thus, if any of these cards are used in the timing mode, with DTE off, a hang-up will occur. When DTE is on and one of the affected cards is addressed and gated it will immediately return the leading edge of the CTF flag. Depending upon the card model, the trailing edge of the flag is either controlled by circuits on the card itself or by an external circuit connected to the card. For models 69321B and 69370A (D/A cards), the trailing edge of the flag is controlled by circuits on the card itself. For models 69330A (Relay Output) and 69331A (Digital Output), the trailing edge of the flag is normally controlled by an external device.

3-50 Cards With External Gate/Flag Capability. The digital output card (69331A) and the relay output card (69330A) have external gate/flag capability. This feature allows these cards to transfer data to an external device which requires a gate input to receive data.

3-51 Normally DTE will be programmed on in a control word, before sending a data word to the card, as shown in Examples 7 and 8. As each data word is received, it is loaded into the addressed card and immediately strobed (gated) in*o an external device.

3-52 Some multiple card applications may require data to be loaded into several output cards, then simultaneously gated into external devices. In this case, a control word may be sent with DTE off, followed by appropriate data words, then another control word with DTE on. As each data word is received, the data will be immediately transferred to the output of the addressed card. However, the gate from the output card will not be enabled until a control word containing DTE is received.

3-53 Example 10 illustrates loading data into several output cards with DTE off, then turning DTE on to simultaneously generate gates to external devices.

3-54 In Example 10, output cards located in multiprogrammer slots 401, and 402 are sequentially loaded with data. Notice that SYE is on, allowing the data to be immediately transferred to the output of the cards, and maintaining the output of previously programmed output cards. After all cards have been loaded with data, DTE is turned on by the control word "170140", resulting in simultaneous generation of gate signals to external devices.

3-55 If it is not required to simultaneously gate external devices from multiple cards, DTE can be included in the first control word of Example 10, (i. e., 170140) and the control word following the data words can be eliminated. Each card would then generate a gate as it receives a data word, as previously described.

3-56 Cards With Dual Rank Storage. The D/A voltage (69321B) and current (69370A) converter output cards have dual rank storage capability. This feature allows the outputs of any number of D/A cards to be changed to new values simultaneously. In this case, a control word is sent with DTE off, followed by appropriate data words, then another control word with DTE on. As each data word is received, the data will be loaded into the first rank storage of the addressed card. When a subsequent control word with DTE is received, all previously addressed D/A cards will simultaneously transfer the contents of the first rank storage into second rank storage and to the output D/A conversion circuits.

3-57 Example 10, which was previously shown for programming DTE for digital output and relay output cards, can also be used for D/A cards. In this case, the first rank storage of D/A cards located in multiprogrammer slots 401 and 402 would be loaded with new data. After the cards have received new data, DTE is turned on by the control word "170140", resulting in both cards transferring data from first rank to second rank storage and converting the data to analog outputs.

3-58 As was the case for the 69330A and 69331A cards, DTE can also be programmed on first, before sending data words to the D/A cards. As each data word is received, it will be loaded into the addressed card and converted to an analog voltage/current that will be present at the output of the card.

3-59 Timing Mode (Extended Handshake). The TME control mode allows output cards with CTF circuits to control the multiprogrammer flag. Table 3-3 lists the cards which may be used in the TME mode. When programming output cards with TME on, the receipt of the trailing edge

Example 10: Programming Multiple Cards With DTE:

```
Ø: moct;wtb 9,17ØØ4Ø,17777,27777,17Ø14Ø
```

of a CTF card flag will notify the calculator that the output card(s) has timed out (completed data transfer).

3-60 Serial or parallel methods may be used when programming output cards in the timing mode. The serial method programs one output card at a time, while the parallel method involves programming several output cards at once.

3-61 Serial Method. Example 11 illustrates programming two output cards serially (one at a time) in the timing mode. The interrupt capability of the 9825A extended I/O ROM will be utilized to detect the trailing edge of the multiprogrammer flag. First, a linkage is established between GPIO interface select code 9 and a service routine called "next". Then a control word with both TME and DTE on is transmitted, placing the multiprogrammer in timing mode. Since certain output cards cannot drive CTF unless DTE is on, DTE and TME should be programmed on together (see Paragraph 3-49). After the data word is sent to the first card, interface 9 is enabled to receive interrupts and the main program continues running.

3-62 When the first card times out, the trailing edge of flag causes an interrupt on interface 9, which in turn causes a jump to service routine "next". Service routine "next" sends the data to the second card, establishes a new interrupt linkage between interface 9 and service routine "done", and re-enables interface 9 for further interrupts. Service routine "next" then returns program control to the main program which continues running.

3-63 When the second card times out, the interrupt on interface 9 will cause a jump to service routine "done". Service routine "done" sends a new control word to the multiprogrammer turning TME off, then returns program control to the main program.

3-64 Parallel Method. Example 12 illustrates parallel (simultaneous) programming of two output cards in the timing mode. A control word with DTE and SYE is sent to the multiprogrammer, followed by data words addressing the two cards, and then another control word with DTE, SYE, and TME. Each output card will begin its timing sequence as it is addressed. Since TME is turned on after

Example 11: Timing Mode of Operation, Serial Method

```
Ø: oni 9,"next"
1: moct;wtb 9,17Ø16Ø,27777;mdec;eir 9

                    •

                    •

                    •

2Ø: "next":moct;wtb 9,37777;mdec;oni 9,"done";eir 9;iret
21: "done":moct;wtb 9,17ØØ4Ø;mdec;iret
```

Explanation:

Ø: Interrupt linkage is established between GPIO interface 9 and routine labeled "next".

1: Control word with TME on is sent to multiprogrammer, followed by data word to slot 402. Interface 9 is enabled for interrupts after data word has been sent. Main program continues from this point until an interrupt occurs.

2Ø: First interrupt has occurred. Data word is sent to output card in slot 403. New interrupt linkage is established between interface 9 and service routine "done". Interface 9 is re-enabled for interrupts and program control returns to main program.

21: Second interrupt has occurred. Control word with TME off is sent to multiprogrammer and program control returns to main program.

the timing sequences have started, a trailing edge of the CTF flag will not be returned to generate an interrupt on interface 9 until both cards have timed out.

3-65 When both cards time out, an interrupt will be generated on interface 9, causing a jump to service routine "done". Service routine "done" sends another control word to the multiprogrammer turning TME off, then tells the calculator to print "done" as an indication that both cards have timed out. After printing "done", program execution returns to the main program. Of course, the users service routine does not have to turn TME off and print "done" but could take any action desired.

3-66 Examples 11 and 12 both specify octal mode (moct) and decimal mode (mdec) repeatedly. This is because it was assumed that these program segments were part of a program controlling other devices besides a multiprogrammer. It was further assumed that these devices required the calculator to operate in the decimal mode. Based on these assumptions, it was necessary to set the calculator to the octal mode to send data to the multiprogrammer and return the calculator to decimal mode after sending data. If the users program is such that the calculator will always be operating in octal mode, moct may be removed from lines 2Ø and 21 and mdec should be completely removed from both examples.

3-67 Notice that an interrupt from an interface disables the interface for further interrupts. If further interrupts are desired, the interface must be re-enabled for interrupts by another eir 9 statement. In addition, if it is desired to have a different service routine executed on receiving a subsequent interrupt, another oni 9 statement must be executed establishing interrupt linkage to the new service routine.

3-68 One final note on using the timing mode with output cards. It is not necessary to use the 9825A interrupt system when using the timing mode. If the user is not concerned with tying up the calculator for the duration of data transfers he may send a control word containing TME followed by a data word to the appropriate card then another control word with TME off (e. g., moct; wtb 9, 17Ø16Ø,27777,17ØØ4Ø). In this case, the calculator will pause until the data transfer with the card has been completed before sending the second control word.

3-69 Avoiding Hang-Up Conditions in the Timing Mode. During the TME mode, there is a wait-for-flag interval that starts from the time that the output card is addressed with a gate and ends when the card returns a CTF flag. During the wait-for-flag period, the calculator is said to "hang-up" if another output or input operation is attempted. That is, any additional output words sent to the multiprogrammer system will not be obeyed until the flag from the programmed card is returned. If the calculator is in hang-up because of a user error, it can remain there indefinitely; waiting for a CTF flag that will never occur. For output card programming in the TME mode, hang-up can be caused by one of

Example 12: Timing Mode of Operation, Parallel Method

```
Ø: oni 9,"done"
1: moct;wtb 9,17Ø14Ø,27777,37777,17Ø16Ø;mdec;eir 9

                    •

                    •

                    •

2Ø: "done":moct;wtb 9,17ØØ4Ø;prt "done";mdec;iret
```

Explanation:

Ø: Establishes interrupt linkage between GPIO interface 9 and routine labeled "done".
1: Control word with DTE, SYE on is sent to multiprogrammer. Data words are sent to output cards in slots 402 and 403, followed by another control word with DTE, SYE, and TME. (With TME on, the trailing edge of CTF flag will not be returned until both cards time out).
2Ø: Interrupt has occurred. New control word with TME off is sent to multiprogrammer, calculator prints "done", and program control returns to main program.

three possible problems:

(1) A card without CTF circuitry (Note 5, Table 3-3) was addressed and gated (with a data word) while TME was on.

(2) A card with external gate/flag capability (69330A or 69331A) was addressed and gated, as is normal, *but* the gate/flag terminals on the card are open (CTF flag cannot be terminated).

(3) DTE was not turned on with TME and a card affected by DTE (Table 3-3) was addressed and gated.

3-70 The hang-up of problem (1) results in the initiation of a gate within the multiprogrammer which cannot be terminated. This hang-up can be cleared by means of a read interface function, described in Paragraph 3-86, prior to sending the desired control word. To avoid this problem, the user must bear in mind that cards without CTF circuits are not intended for timing mode operation and should never be addressed and gated while TME is on. Problem (2) of the previous paragraph results in an unterminated CTF flag and cannot be cleared by sending the statement described in Paragraph 3-86. For this case, the CTF flag can be terminated by toggling the 6940B's LINE switch off and then on. To avoid this problem, the user must simply remember to jumper the gate and flag terminals together on any 69330A or 69331A cards that do not use an external timing circuit. Problem number (3) results in an unterminated gate within the multiprogrammer, the same as problem (1), and can be cleared by sending the statement described in Paragraph 3-86. As described in Paragraph 3-49, problem (3) can be avoided by programming TME and DTE on together.

3-71 Input Cards, General Programming Techniques

3-72 **Control Mode Bits for Input Card Operations.** As indicated in Table 3-3, three control mode bits are applicable to input card programming; ISL, TME, and IEN. The input select (ISL) bit must be programmed on prior to any basic read sequence in which return data is sent back to the calculator from an input card. The TME and IEN bits are used to establish the timing and interrupt modes of operation, as will be explained subsequently. Table 3-1 lists the three control mode codes that are most commonly used for input card operations.

NOTE

All control words used in this section assume a unit address of ØØ.

3-73 System Enable Bit. Notice that the SYE bit is on in all three of the input mode codes shown in Table 3-1. Although SYE does not affect input cards, it is usually good practice to have SYE on in control words specifying an input mode of operation. If SYE is turned off, and output cards are included in your system, the outputs of these cards will be disabled creating what may be an undesirable situation. Hence, all of the control word examples shown in the input portion of this chapter show SYE on.

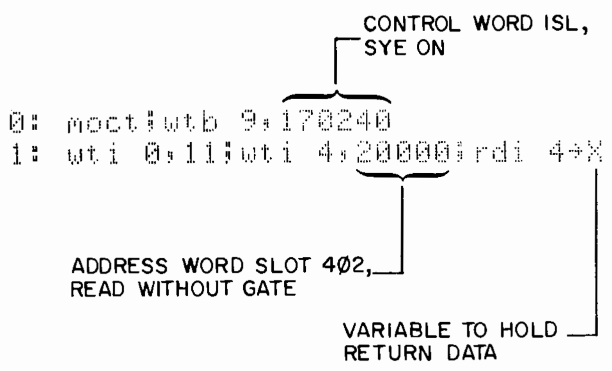
3-74 **Basic Input Operating Modes.** There are two basic programming modes for input cards; serial and parallel. A serial operation can involve the programming of a single input card or a group of cards, programmed one at a time. A parallel operation, involves the simultaneous programming of a group of cards.

3-75 Within the boundaries of serial operation there are two programming sequences which are often used with input cards. These are the simple direct read without gate sequence and the timing mode (extended handshake). In the direct read sequence the input card is addressed, without a gate, and the existing data on the card is returned immediately to the calculator. In the timing mode sequence (TME on), the card is addressed with a gate which initiates a gate/flag transfer cycle between the card and its external device. After the transfer is completed, the fresh data on the input card is read into the calculator.

3-76 Parallel operation for input cards is synonymous with the interrupt mode of operation (TME and IEN on). The interrupt mode allows the user to activate (arm) a group of input cards and then provides an indication (in the form of an interrupt) when the first card is ready with data or the first external process has been completed. Sample programs for the read without gate sequence, timing mode and interrupt mode are provided in the following paragraphs.

3-77 **Reading in a Return Data Word.** At some point(s) in any input card program, the user will want to read the return data available on the card into the calculator. An example of a simple read without gate sequence was shown in Paragraph 3-27. Because this operation is used many times when programming input cards, this example is shown again for convenience. To summarize the read without gate sequence shown in Example 13, a control word is sent with ISL on, followed by an address word directing that the input card in slot 402 return its data. The read interface statement allows the return data to be stored in variable "X". Notice that the read without gate sequence of Example 13 is the only recommended way to obtain data from input cards that do not have CTF flag circuits (see Note 5 of Table 3-3). If the user desires to employ a direct read without gate sequence for a second input card, he can simply follow line 1 of Example 13 with a similar line, omitting wti Ø,11 and changing the card slot address and variable as desired. It is not necessary to respecify the select code in the write interface statement (wti Ø,11) or transmit the

Example 13: Addressing an Input Card and Entering Its Return Data:


CONTROL WORD ISL, SYE ON
ADDRESS WORD SLOT 4Ø2, READ WITHOUT GATE
VARIABLE TO HOLD RETURN DATA

```
0: moct;wtb 9,170240
1: wti 0,11;wti 4,20000;rdi 4→X
```

control word again, since they remain set until changed by the program.

3-78 Checking for IRQ in the Return Data Word. As indicated previously, the return data stored in the variable of Example 13 can be used directly if it is obtained from an input card that does not contain IRQ (see Table 3-3). If the card does contain the IRQ function, the return data must first be examined to determine if there is a "1" in the IRQ field. If the IRQ character is a "1", it must be subtracted from the return data before the data can be utilized. Example 14 shows how to check for IRQ and subtract it, if necessary. Example 14 would immediately follow Example 13 for any cards that have IRQ. In line 2, the return data in variable "X" is checked for the presence of IRQ. If IRQ is present, the number 1ØØØØØ is subtracted from the return data in "X". If IRQ is not present, –1 is stored in "X" as an indication that the data was not current data, and the program continues execution. Of course, –1 (used here for explanation purposes) is optional.

Example 14: Checking for IRQ and Subtracting it

```
2: if X<100000;-1→X;gto "continue"
3: X-100000→X
4: "continue":
```

3-79 Octal to Decimal Conversion. For some input cards, such as the 69431A Digital Input, the user will find it convenient to work with the octal data returned to the variable "X" in the previous examples. For other cards, such as the 69421A Voltage Monitor, it will be more convenient to work with decimal values. In these cases, the octal to decimal function of the calculator can be utilized. To use this function, simply add the following line to your program: otd X→X. The placement of this program line depends on whether or not the card used IRQ. If IRQ is not used, it can be inserted immediately after line 1 of Example 13. If IRQ is used, the line must be inserted after line 3 (subtraction of IRQ) in Example 14.

3-80 Timing Mode (Extended Handshake). The TME control mode permits a data input transfer cycle between an input card and its external device. This is accomplished via the gate/flag method in which input cards with CTF circuits control the flag line. (Table 3-3 shows the input cards that have CTF circuits.) During the TME mode, the trailing edge of the CTF flag will notify the calculator that the input card has completed its data transfer.

3-81 Example 15 shows how to program a single input card in the timing mode. As shown previously with output cards, the interrupt capability of the 9825A calculator will be used to detect interrupts from GPIO interface 9. First a linkage is established between GPIO select code 9 and a service routine called "read". Then a control word with ISL and TME is sent, placing the multiprogrammer in the input timing mode.

3-82 When the input card times out, the trailing edge of flag causes an interrupt on interface 9, which in turn causes a jump to service routine "read". Service routine "read" reads the input card data (without gate) into variable X. IRQ is then subtracted from the value in X. The calculator then prints the value read from card slot 402 and returns program control to the main program. The user has the option of taking any action desired within the service routine rather than merely printing out the data value.

3-83 If more than one input card is to be read in the timing mode, they must be read in a serial fashion. That is, the first card must complete its data transfer before the second card begins. In this case, line 1 in Example 15 could be re-transmitted using the next card address. It is not necessary to transmit the control word (with ISL, TME) again, unless a different control word was subsequently sent to the multiprogrmmer.

3-84 One final note on using the timing mode. It is not necessary to use the 9825A interrupt system when using the timing mode. If the user is not concerned with tying up the calculator for the duration of data transfers, he may follow the card slot address with another control word containing ISL and SYE (no TME). The calculator will pause until the addressed card returns its flag. When the trailing edge of flag is received, the second control word will be transmitted. The card may then be read without a gate and the data stored in a variable. Example 16 illustrates this technique.

Example 15: Timing Mode for One Input Card

```
0: oni 9,"read"
1: moct;wtb 9,170260,20000;mdec;eir 9
                    •
                    •
                    •
20: "read":moct;wti 0,11;rdi 4→X
21: X-100000→X
22: fxd 0;prt "data=",X;mdec;iret
```

Explanation:

Ø: Establishes interrupt linkage between GPIO interface 9 and routine labeled "read".
1: Send control word with ISL and TME, followed by address word to slot 402. Enables interface 9 for interrupts.
2Ø: Interrupt has occurred. Data from slot 402 is read without gate. Data is stored in variable X.
21: Subtracts IRQ from data.
22: Data read from slot 402 is printed on calculator. Program control is returned to main program.

Example 16: Using Timing Mode Without Using the 9825A Interrupt System

```
0: moct;wtb 9,170260,20000,170240
1: wti 0,11;wti 4,20000;rdi 4→X;mdec
```

3-85 Avoiding "Hang-up" Conditions in the Timing Mode. During the TME mode, there is a wait-for-flag interval that starts from the time that the input card is addressed with a gate and ends when the card returns a CTF flag. During the wait-for-flag period, the calculator is said to "hang-up" if another input or output operation is attempted. That is, any additional output words sent to the multiprogrammer system will not be obeyed until the flag from the programmed card is returned. If the calculator is in hang-up because of a user error, it can remain there indefinitely; waiting for a CTF flag that will never occur. For input card programming in the TME mode, hang-up can be caused by one of two possible problems.

(1) A card without CTF circuitry (see Note 5, Table 3-3) was addressed and gated while TME was on.

(2) A card with external gate/flag capability (69431A only) was addressed and gated, as is normal, but the gate/flag terminals on the card are open (CTF flag cannot be terminated).

3-86 The hang-up of problem number (1) results in the initiation of a gate within the multiprogrammer which cannot be terminated. Cards without CTF circuits are not intended for timing mode operation and should never be addressed and gated while TME is on. Data from these cards should be obtained only by reading without a gate, as previously recommended in Paragraph 3-77. If you think that you have improperly gated one of these cards, Example 17 shows a method of clearing the hang-up condition.

3-87 Executing the program line of Example 17 clears the multiprogrammer system by resetting the gate signal. Problem number (2) of Paragraph 3-85 is caused by an unterminated CTF flag and cannot be cleared by sending the program line of Example 17. The only remedy for an unterminated flag is to remove input power from the 6940B unit (toggle the LINE switch off and then on). To avoid this problem the user must simply remember to jumper the gate and flag terminals together on any 69431A cards that do not use an external timing circuit.

3-88 Interrupt Mode of Operation. As indicated in Table 3-3, the interrupt mode can be used only with 69431A, 69434A, 69436A, or 69600B cards. These cards must be activated (armed) before they are able to cause an interrupt. As is described in the interrupt mode description of Chapter VII, (Paragraphs 7-52 through 7-59), there are two ways of arming the cards: individually or as a group. The most commonly used method is individual arming which is permitted only when jumper W6 is removed from the card. For this method, a control word is sent first, with ISL on. The cards are then armed, one at a time, by sending each card an address word with a gate. The interrupt mode is then established by sending a control word with TME and IEN on. The first armed card that interrupts (receives a data ready indication from its external device), returns a CTF flag thereby generating an interrupt on interface 9 (which is previously enabled). TME and IEN are then turned off to prevent further interrupts and an IRQ poll (interrupt search) is performed. During the IRQ poll, a read without gate sequence is done for each card and the return data is examined for the presence of IRQ. If the IRQ character is a "1", it indicates that the addressed card had interrupted and its return data can be stored for processing. If the IRQ character is a "Ø", it indicates that the card has not yet received new data or detected a change of state in the external device. Once the cards have been polled, the interrupting card (or cards) must be reset (either disarmed or recycled) or it will cause another interrupt if TME and IEN are programmed on again. As indicated in Table 3-4, the requirements for recycling, disarming, and interrupting are different for each card. Notice also, that most of these cards are recycled or disarmed in the output mode using data words. Interrupt mode programs for these cards are given in the individual card programming chapter. Additional information can be found in the instruction manual for each of these cards.

3-89 An interrupt sequence using the group arming (jumper W6 installed) technique is similar to the procedures just described except for the arming procedures at the beginning of the program. However, in this case, all cards with jumper W6 installed are armed simultaneously when a control word is sent with TME and IEN on. Once an interrupt occurs, an IRQ poll is done in a manner similar to that described previously in the individual arming sequence. Although the group arming technique saves programming steps, it can be difficult to use unless all input cards in the system are being used in the same manner and the user fully understands all of the implications (read Paragraph 7-59 in Chapter VII. Therefore, it is recommended that the user remove jumper W6 from all cards having an interrupt capability and use the individual arming method.

3-90 Example 18 illustrates an interrupt sequence using three input cards from which jumper W6 has been removed (cards are individually armed). The example assumes that the input cards are located in slots 401, 402, and 403.

3-91 As with the timing mode examples, the interrupt capability of the 9825A calculator is used to enable a fast response to a multiprogrammer interrupt. In this example, variables X, Y, and Z are used to store either valid data if the card interrupted or a value of −1 if it did not. Service routine "poll" also prints the data and slot number of the interrupting card before returning to the main program. The user has the option of taking any action desired in the service routine rather than printing the interrupting slot number and data. In addition, any card slots or variables desired can be used.

3-92 At the completion of the interrupt sequence of Example 18, the user may wish to begin a non-interrupt operation. In this case, it is a good practice to disarm any input cards that are still active (had not interrupted) in order to avoid any possibilities of hang-ups or erroneous results in the other modes. Similarly, the user should include only the four card types listed in Table 3-4 in his interrupt sequence to avoid the same possibilities.

3-93 Once the interrupt mode is established, the user may want to proceed to an unrelated operation, such as programming a digital output card. To leave the interrupt mode before a service request occurs, the user must execute the program line illustrated in Example 17. Once this is done, interface 9 must then be disabled for further interrupts by an (eir 9,Ø) statement before programming any other cards. To return to interrupt mode of operation, a control word with TME and IEN on is sent and interface 9 is re-enabled for interrupts. (e. g., wtb 9,17Ø46Ø;eir 9). Caution is necessary when using this procedure with 69434A event sense cards, as these cards do not store interrupts that occur when the multiprogrammer is not operating in the interrupt mode.

Example 17: Clearing a Hang-Up Caused by an Unterminated Gate

```
0: moct;wti 0,11;wti 5,40
```

Example 18: Interrupt Mode of Operation, Individual Arming

```
0: oni 9,"poll"
1: moct;wtb 9,170240,10000,20000,30000,170460;mdec;eir 9

                    .
                    .
                    .

20: "poll":moct;fxd 0
21: wtb 9,170240;wti 0,11;wti 4,10000;rdi 4→X
22: if X<100000;-1→X;jmp 2
23: X-100000→X;wtb 9,170040,10000
24: wtb 9,170240;wti 0,11;wti 4,20000;rdi 4→Y
25: if Y<100000;-1→Y;jmp 2
26: Y-100000→Y;wtb 9,170040,20000
27: wtb 9,170240;wti 0,11;wti 4,30000;rdi 4→Z
28: if Z<100000;-1→Z;jmp 2
29: Z-100000→Z;wtb 9,170040,30000
30: if X>-1;prt "slot 401 data =",X
31: if Y>-1;prt "slot 402 data =",Y
32: if Z>-1;prt "slot 403 data =",Z
33: mdec;iret
```

Explanation:

Ø: Establishes interrupt linkage between GPIO interface 9 and routine labeled "poll".

1: Control word with ISL and SYE on is sent to multiprogrammer. Cards in slots 401, 402, and 403 are sequentially armed by address words 1ØØØØ, 2ØØØØ, 3ØØØØ, thus initiating the interrupt sequence for each card. Another control word, "17Ø46Ø", establishes the interrupt mode by turning IEN, SYE, and TME on. With TME and IEN on, the service request will not be set until the first card(s) interrupts. Interface 9 is enabled for interrupts.

2Ø: Beginning of interrupt routine.

21: Control word with ISL, SYE on is sent, followed by an address word "1ØØØØ" to start IRQ poll. Write interface function wti 4, 1ØØØØ initiates a read without gate sequence for the first input card in slot 401. Data is read into variable X.

22: Return data from first card is examined for IRQ. If IRQ is not present, the first card has not interrupted. Store —1 in variable X as an indication tht the card had not interrupted; jump to read without gate sequence for second card.

23: IRQ is present in the return data from first card, indicating that the first card had interrupted and its return data is valid. Subtract IRQ from data and store in X. Disarm first card; control word with SYE on establishes output mode followed by data word 1ØØØØ with gate, which disarms (resets) the card.

24-26: Same sequence as 21-23, using second input card in slot 402 and variable Y.

27-29: Same sequence as 21-23, using third input card in slot 403 and variable Z.

3Ø-32: Print out slot number and data of interrupting cards.

33: Interrupt return statement. Main program resumes executing.

Table 3-4. Requirements for Cards with Interrupt Capability

Card Model	Significance Of Data Sent In Output Mode	Arming Requirements (W6 Out)	Arming Requirements (W6 In) *	Recycling Requirements	Disarming Requirements	Interrupting Requirements
69431A Digital Input		ISL on TME off Address true Gate	IEN on TME on Gate	ISL on TME off Address true Gate	ISL off TME off Address true Gate	External device returns trailing edge of flag
69436A Process Interrupt	All logical "1's" initializes card. Interrupting word recycles card. All logical "0's" disarms card.	ISL on TME off Address true Gate	IEN on TME on Gate	ISL off TME off Address true Data = Interrupting word. Gate	** ISL off TME off Address true Data = logical "0's". Gate	Any one or more of twelve paired positive or negative edge detectors tripped.
69600B Timer	Binary representation of pulse duration. All logical "0's" disarms card.	ISL on TME off Address true Gate	IEN on TME on Gate	ISL off TME off Address true Data = Duration of next pulse. Gate	ISL off TME off Address true Data = logical "0's". Gate	End of programmed pulse
69434A Event Sense	Reference word	ISL on TME off Address true Gate	IEN on TME on Gate		ISL off TME off Address true Gate	External Word IS = Ref. word IS ≠ Ref. word IS > Ref. word IS < Ref. word

* As supplied from factory.

** Card must be recycled prior to disarming.

Chapter IV
HP-IB PROGRAMMING FUNDAMENTALS

4-1 This chapter outlines the fundamentals of programming the multiprogrammer system with a 9825A calculator using an HP-IB interface. Included in this chapter are multiprogrammer compatibility with the HP-IB, how to program the basic word formats, and general programming techniques for multiprogrammer output and input cards.

4-2 HP-IB INTERFACE FUNCTIONS

4-3 The multiprogrammer system (Models 59500A and 6940B/6941B) is capable of performing the following functions on the HP-IB:

a. Talk
b. Listen
c. Service Request and Serial Poll

Conversely, the multiprogrammer system cannot act as a controller, extended talker, or extended listener and will not respond to remote-local, parallel poll, device clear, or device trigger commands.

4-4 MULTIPROGRAMMER WORD FORMATS

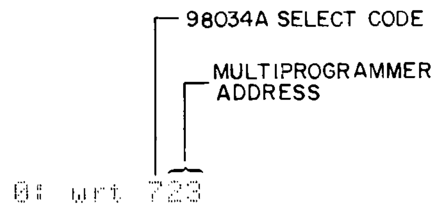
4-5 The next portion of this chapter is devoted to the fundamentals of communicating between the calculator and the multiprogrammer system (59500A/6940B, 6941B). For calculator-to-multiprogrammer exchanges, three different types of codes (words) can be sent and for multiprogrammer-to-calculator exchanges two types of words can be received. The following paragraphs describe the basic construction and purpose of each word type showing how to send the output words or read in the input words. Each word type is treated separately without too much regard for the programming sequences which can be used to accomplish a specific task. Programming examples, using these basic word formats, are included later in this chapter, under general programming techniques.

4-6 Calculator Output Words

4-7 Three types of output words are sent from the calculator to the multiprogrammer system: control words, data words, and address words.

4-8 Multiprogrammer Listen Address. Before any of these words are sent, the multiprogrammer must first be established as the listener. Throughout this chapter it is assumed that the HP-IB interface card (98034A) has been assigned its standard select code of "7" and that the multiprogrammer system has been assigned listen and talk address of "7" and "W" respectively. "7" and "W" correspond to an address of "23" as defined in the 9825A General I/O ROM manual (09825-90024) chapter 4. Using these addresses, the addressing sequence would be as follows:

Example 1: Addressing the Multiprogrammer to Listen



```
0: wrt 723
```

4-9 Control Word. Usually, the first word sent to a multiprogrammer system will be a control word. It is used to specify a unit address and the operating mode of the multiprogrammer system. The unit address portion selects which mainframe in the chain (of one 6940B Multiprogrammer and up to fifteen 6941B Extenders) will respond to subsequent data or address words from the calculator. The mode portion of the word selects one of several output or input modes of multiprogrammer operation (System Enable, Timing Mode Enable, Input Select, etc.). How the multiprogrammer system responds to these various control word modes is described in Chapter VII of this guide, as well as in the 6940B Service Manual. A good understanding of these modes is essential for effective programming involving specific applications.

4-10 Figure 4-1 illustrates the structure of a control word as transmitted on the HP-IB. The six control word characters are transmitted and obeyed in the order written, from left-to-right. The address character "0", is the control word tag that tells the 6940B unit that the data characters that follow are to be interpreted as control mode and unit address information.

4-11 The data field consists of four octal numbers. The most significant digit, D_4, conveys no information in a control word and can be sent as a "Ø" or not transmitted at all (see note on following page). Digits D_3 and D_2 specify the operating mode as indicated by the codes shown in Table 4-1. Only the most commonly used codes are shown in Table 4-1; but combinations for other codes are possible and become obvious through inspection.

NOTE

Because of the way the multiprogrammer system processes the octal characters in a data field, any leading "Ø"'s need not be sent. Of course, trailing "Ø" 's must be transmitted.

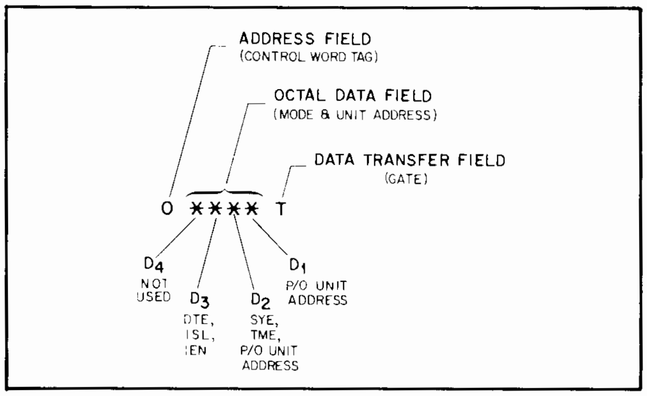


Figure 4-1. Control Word

Table 4-1. Operating Mode Codes

Multiprogrammer Operating Mode		**Data Field ***
Output Mode	**Input Mode**	
SYE Off		ØØØØ
SYE On		ØØ4Ø
DTE, SYE On		Ø14Ø
DTE, SYE, TME On		Ø16Ø
	ISL, SYE On	Ø24Ø
	ISL, SYE, TME On	Ø26Ø
	IEN, SYE, TME On	Ø46Ø

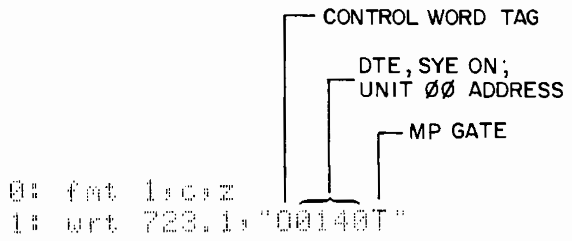
* Data field codes shown are valid for a unit address of ØØ only.

4-12 Digits D_2 and D_1 specify the unit address. The first unit in the chain (6940B, Master) has an address of ØØ, the second unit (6941B Extender) has an address of Ø1, and so forth, up to a unit address of 15. The two unit address digits are simply added (in octal) to the last two control mode digits shown in Table 4-1 to obtain the final data field code. For example, if the user wants to select unit number one (second unit in chain) and program the DTE, SYE, and TME modes, he would send a code of Ø161 in the data field. To select unit number 15 (last unit) and the same modes, he must send an Ø177 ($Ø16Ø_8 + 17_8$).

4-13 The last character in a control word must be a "T". The multiprogrammer system interprets this character as a "gate" which initiates the gate/flag handshake sequence within the 6940B multiprogrammer.

4-14 A typical control word, with DTE and SYE on, would be sent as follows:

Example 2: Typical Control Word



```
0: fmt 1,c,z
1: wrt 723.1,"00140T"
```

4-15 Note the format statement used in Example 2. The "c" specifies a character field, while "z" is used to suppress the carriage return/line feed codes at the end of the write statement. Suppression of the carriage return/line feed codes is essential when using the multiprogrammer in the timing mode. The reason for this is that the calculator must receive handshake flags with the carriage return/line feed codes in order to continue with the program. Thus, by suppressing the carriage return/line feed codes, the calculator can continue to the next statement in the program while the multiprogrammer is still timing out. Note that all write statements shown in this manual will reference a format containing "z", even those used in non-timing mode (TME off). It does no harm to do this, and it helps avoid potential problems.

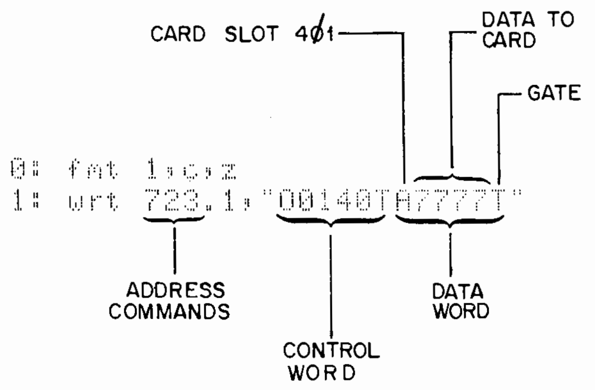
4-16 **Data Word.** Data words select and control output cards and a few "special purpose" input cards. The data word selects an individual card within the mainframe previously specified by a control word.

4-17 Figure 4-2 shows the basic construction of a data word sent from the calculator. The address character selects the particular card (one of 15 slots in a mainframe) that will receive the four "control" digits in the data field portion of the word. Table 4-2 shows the card slot address that corresponds to each ASCII address character. The data field consists of four octal digits that can be of any value between ØØØØ and 7777 (Appendix A contains an explanation on converting numbers from octal to decimal and vice-versa). As in a control word, leading Ø's need not be sent in the data field. A data word with ØØØØ in the data field could be sent with just an address and a gate (T) character; e. g., to send all Ø's to slot 401, simply transmit "AT". Of course, this word could also be sent as "AØØØØT."

4-18 The data transfer character must always be a "T" (gate) in a data word, the same as in a control word.

4-19 Example 3 shows a write (output) statement using a typical data word preceded by the same sample control word used in Example 2.

Example 3: Typical Data Word



```
0: fmt 1,c,z
1: wrt 723.1,"00140TA7777T"
```

In this example, an output card (in slot 401, Unit ØØ) is being directed to output data represented by the octal number "7777."

4-20 **Address Word.** An address word selects the input card that is to send data (a return data word) back to the calculator. Before an address word can be sent, a control word must have first selected an input operating mode (and the unit address). Notice also that before the addressed input card can send a return data word back to the calculator, the multiprogrammer system must be placed in the talk mode (refer to Paragraph 4-23; calculator input words).

4-21 An address word contains just two characters (Figure 4-3). The address character specifies one of 15 possible input card slot addresses which are identical to those given in Table 4-2 for a data word. The data transfer field character can be a "T" (read with gate) an "X" (read without gate), or a "z" (read live data). Which character to use depends on the specific application and the input card type (refer to Chapters V and VII). A "T" always results in the generation of a "gate" to the addressed input card, and the data from the input card (return data word) will be stored in the 59500A on the trailing edge of the flag from the multiprogrammer. Thus, a "T" character is used when it is desired to wait for the completion of a gate/flag sequence before storing the return data word in the 59500A. As previously mentioned, to obtain the return data information after an address word is sent, the user must address the multiprogrammer to talk and read the data into the calculator.

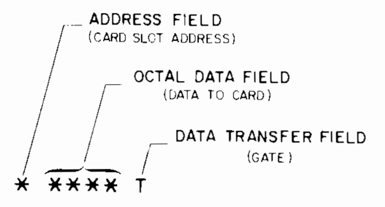


Figure 4-2. Data Word

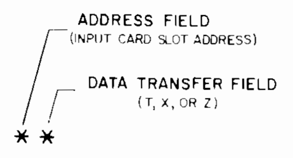


Figure 4-3. Address Word

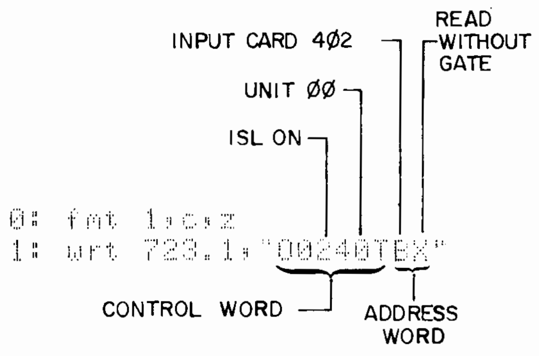
Table 4-2. Card Slot Addresses

ASCII ADDRESS CHARACTER	ASCII CODE	MULTIPROGRAMMER CARD SLOT ADDRESS
@	1ØØØØØØ	Slot 400
A	1ØØØØØ1	Slot 401
B	1ØØØØ1Ø	Slot 402
C	1ØØØØ11	Slot 403
D	1ØØØ1ØØ	Slot 404
E	1ØØØ1Ø1	Slot 405
F	1ØØØ11Ø	Slot 406
G	1ØØØ111	Slot 407
H	1ØØ1ØØØ	Slot 408
I	1ØØ1ØØ1	Slot 409
J	1ØØ1Ø1Ø	Slot 410
K	1ØØ1Ø11	Slot 411
L	1ØØ11ØØ	Slot 412
M	1ØØ11Ø1	Slot 413
N	1ØØ111Ø	Slot 414

4-22 An "X" character is sent when it is necessary to store the return data word from the addressed card immediately, without initiating a multiprogrammer gate/flag cycle. The "X" character itself, stores the return data in the 59500A as soon as it is received from the input card. The "Z" character allows the return data from the addressed card to continuously pass through the 59500A to its output encoder, without being stored. When using the "Z" character, the user must keep in mind that the return data may be continuously changing and can therefore be difficult to interpret.

4-23 Example 4 shows a typical address word sent to an input card installed in multiprogrammer slot 402 (address character B). Notice that the preceding control word calls for an input mode of operation (ISL on), and the address word "BX", specifies a read without gate sequence.

Example 4: Typical Address Word



```
Ø: fmt 1,c,z
1: wrt 723.1,"0024ØTBX"
```

4-24 Calculator Input Words

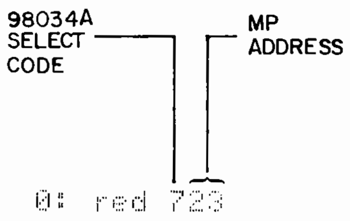
4-25 Two types of input words are sent to the calculator from the multiprogrammer system: Return data words and status characters (bytes).

4-26 **Multiprogrammer Talk Address.** Before the multiprogrammer can send a return data word, a statement must be sent directing the multiprogrammer to talk. Example 5 shows the necessary address commands.

4-27 **Return Data Word.** Return data words contain information from an input card previously selected by an address word. Each word (Figure 4-4) contains an input request character, four octal data characters, and a carriage return/line feed (not shown on Figure 4-4).

4-28 The input request (IRQ) character can be either a "1" or a "Ø". A "1" indicates that the input card has returned a flag and its input data is valid. Moreover, during an interrupt sequence (TME and IEN on), a "1" in the IRQ field indicates that this card has generated an interrupt. A "Ø" indicates that: (1) the input card has not returned a flag (card is still busy or has not interrupted) and its' data is not

Example 5: Addressing the Multiprogrammer to Talk



```
Ø: red 723
```

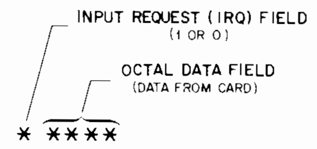


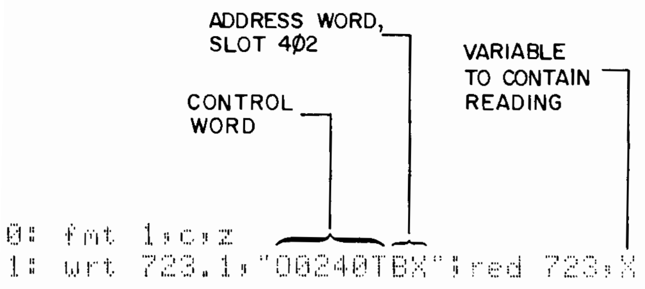
Figure 4-4. Return Data Word

valid; or (2) the card does not contain an IRQ function. Some input cards; such as the voltage monitor card (69421A), do not contain an IRQ function and a "Ø" in the IRQ field can be ignored.

4-29 The four data characters, representing the input information from the multiprogrammer card, can be any octal number, between ØØØØ and 7777. Hence, a return data word with all "7" 's in the data field and a "1" in the IRQ field would appear as a 17777 on the calculator display. Example 6 shows how to read in a return data word and store it in variable "X". First, a write statement is sent containing a control word with ISL on. Next, an address word is sent to select the input card in slot 402 (Unit ØØ) to return its' data. Finally, a read statement directs the multiprogrammer to talk and allows the calculator to read the data into variable "X". The return data in variable "X" can be utilized directly unless it is obtained from a card that contains the IRQ function. In these cases, the return data must first be checked to determine if there is a "1" in the IRQ field. If the IRQ character is a "1", it must then be subtracted from the input data before continuing with the program. How to check for IRQ and subtract it, if necessary, is described under general programming techniques for input cards, in this chapter, as well as in the applicable input card programs of the next chapter.

4-30 **Status Bytes.** The 59500A of the multiprogrammer system returns an 8-bit status byte to the calculator in response to a serial poll routine. Serial polling is the method used by the calculator to determine which bus device has requested service (set SRQ true). The multiprogrammer will request service (set SRQ line) only in the timing mode

Example 6: Addressing an Input Card and Entering its Return Data



```
0: fmt 1,c,z
1: wrt 723.1,"00240TBX";red 723,X
```

(TME on) upon completion of a multiprogrammer flag. Both input and output card types can set SRQ. Chapter VIII lists the specific operating conditions required for a service request to occur (see Paragraph 8-28).

4-31 The user has two options available to monitor the service request line (SRQ) of the bus. He may set up his program to interrupt on SRQ (see Extended I/O ROM manual, chapter 5, HP part no. 09825-90025) or use bus status checks (see General I/O ROM manual, chapter 4, HP part no. 09825-90024) to determine when SRQ is set. In either case, when a bus service request is detected, a serial poll of the bus can identify the device that has requested service. Briefly, doing a serial poll on the multiprogrammer system places the multiprogrammer in the serial poll mode, addresses it to talk, and then reads its status byte into a variable in the calculator. The status byte, stored in the variable, will have a decimal value of 64 if the multiprogrammer requested service or a value of 0 if it had not requested service. Doing a serial poll also clears the SRQ line so that the calculator can detect the presence of new service requests. For this reason, a serial poll should be done at the beginning of any program to ensure that the service request line starts out in a clear state.

4-32 Example 7 illustrates serial polling the multiprogrammer and storing the status byte in variable "C".

Example 7: Multiprogrammer Serial Poll

```
0: rds(723)→C
```

4-33 GENERAL PROGRAMMING INFORMATION

4-34 The next portion of this chapter describes the programming sequences associated with the various plug-in cards of the multiprogrammer system. If a detailed description of multiprogrammer operation is desired, see chapter VII.

4-35 Programming Aids

4-36 **Number System Theory.** Appendix A contains a description of the number systems applicable to the multiprogrammer system. Included are explanations of the decimal, octal, binary, and BCD systems, how to convert numbers from one system to another, and how the number systems relate to some of the plug-in cards.

4-37 **Control Mode Reference Table.** Table 4-3 lists all of the plug-in cards available at the time of this writing and which operating mode bits (in a control word) will affect the operation of each card. Also included is an IRQ column which indicates the cards that contain IRQ circuitry.

4-38 The following two examples show how to use Table 4-3:

1. If the user has a 69321B D/A output card, he can see at a glance that it will be affected by the DTE and SYE control mode bits. He can also determine that if TME is on (extended handshake), it will require 30μsec (nominal) for the card to return a CTF flag.

2. If the user has a 69431A digital input card, he can deduce that ISL must be on to read-in data from the card and that the card has interrupt capability (IEN and IRQ). The user can also determine from the table that this card is capable of providing a gate/flag data transfer cycle with its external device when TME is programmed on. In this case, the length of the flag period is, of course, determined by the external device (Note 1 of table).

4-39 Output Cards, General Programming Techniques

4-40 Three multiprogrammer control mode bits (SYE, DTE, and TME) are used when programming output cards. Table 4-3 lists the cards affected by these bits and Table 4-1 shows the four control mode codes most often used for output card programming. Explanations of sending constants and variables to output cards as well as descriptions of each control mode (SYE, DTE, TME) are given in the following paragraphs. Programming examples are included where applicable.

NOTE

A unit address of ØØ is used in all control words shown in the programming examples.

4-41 **Sending Constants and Variables.** As a rule, output cards are programmed with control words and data words. (There are some output cards that can also be operated in the input mode as will be pointed out in the individual card programs (Chapter V). The data field portion of a data word may contain a constant (explicit value) or a variable (value stored in calculator). Write statements are used to send data to an output card.

4-42 Example 8 illustrates sending a data word containing a constant data value to an output card. In this example, a data value of "7777" is sent to the output card in slot 402 (B).

4-43 Example 9 shows how to send a variable data value to an output card. This example, includes the same address commands and control word as in Example 1. Data is represented by variable A and can be any positive octal value between "ØØØØ" and "7777". Note that if the variable were equal to "7777", Example 9 would perform the same operation as Example 8.

Example 8: Sending a Constant Data Value

```
0: fmt 1,c,z
```

$$\texttt{1: wrt }\underbrace{\texttt{723.1}}_{\text{ADDRESS COMMANDS}}\texttt{,"}\underbrace{\texttt{00140T}}_{\text{CONTROL WORD}}\underbrace{\texttt{B7777T}}_{\text{DATA WORD}}\texttt{"}$$

4-44 Line 10 in example 9 allows the user to work with decimal values when calculating variables. In this case, the decimal to octal function (dto) of the calculator is used to convert a decimal value in variable A to octal. In addition, line 10 checks to see if the octal value exceeds "7777" and,

Table 4-3. Control Mode Summary for Plug-In Cards

MODEL	DESCRIPTION	IEN	ISL	DTE	SYE	Flag (CTF) From Card (4)	IRQ
69321B	D/A Voltage	–	–	Yes	Yes	30μsec	–
69325A	BPSA Voltage Cont.	–	–	–	Yes	9.5msec	–
69326A	BPSA Current Cont.	–	Yes	–	Yes	4.3msec	–
69327A	BPSA Current Cont.	–	Yes	–	Yes	4.3msec	–
69328A	BPSA Gain Cont.	–	–	–	Yes	9.5msec	–
69330A	Relay Output	–	–	Yes	Yes	12msec (1)	–
69331A	Digital Output	–	–	Yes	Yes	(1)	–
69332A	Open Collector Output	–	–	–	–	(5)	–
69335A	Stepping Motor	–	–	–	Yes	(2)	–
69370A	D/A Current	–	–	Yes	Yes	30μsec	–
69380A	Breadboard Output	–	–	–	–	(5)	–
69421A	Voltage Monitor	–	Yes	–	–	6msec	–
69430A	Iso. Digital Input	–	Yes	–	–	(5)	–
69431A	Digital Input	Yes	Yes	–	–	(1)	Yes
69433A	Relay Output/Readback	–	Yes	–	Yes	4msec	–
69434A	Event Sense	Yes	Yes	–	–	(3)	Yes
69435A	Pulse Counter	–	Yes	–	–	(5)	–
69436A	Process Interrupt	Yes	Yes	–	–	(3)	Yes
69480A	Breadboard Input	–	Yes	–	–	(5)	–
69500-06A	12 Bit Resistance	–	–	–	Yes	6msec	–
69510-13A	Dual 6 Bit Resistance	–	–	–	–	6msec	–
69600B	Programmable Timer	Yes	Yes	Yes (7)	–	(2)	Yes
69601B (Note 6)	Frequency Reference (6)	–	–	–	–	(5)	–

(1) Max time depends on flag from external device.

(2) Time depends on number of pulses and pulse frequency.

(3) Max time depends on external inputs to detectors.

(4) CTF flag is ignored unless TME has been programmed on.

(5) These cards have no CTF circuitry and must not be sent a data or address word with a gate ("T") while TME is on. If this is done, no flag will be returned from the card causing the system to hang-up.

(6) Free-running frequency generator (not programmable).

(7) With jumper W4 installed.

Example 9: Sending a Variable Data Value

```
10: dtoA→A;if A>7777;gto "error"
11: fmt 2,c,f4.0,c,z
12: wrt 723.2,"00140TB",A,"T"
```

Explanation:

1Ø: Conversion of variable "A" from decimal to octal (optional).
11: Format statement.
12: Control word (DTE, SYE on) is sent to multiprogrammer, then variable data word (formatted as specified) is sent to output card in slot 402(B) with gate (T).

if it does, goes to a user provided error routine. Of course, if variable A is already an octal value, the dto function must be omitted from line 10.

4-45 Note the format statement used in Example 9. The first c specifies the character string preceding variable A. Format f 4.0 specifies a fixed point format, 4 characters wide, for the variable. The second c specifies the second character string, in this case T, and z suppresses carriage return/line feed codes as previously described. Always use this type of format statement when outputting variables. If additional variables are to be included in each line, additional c and f specs must also be included. A variable exceeding the format width (f 4.Ø) will cause the addressed card to be programmed to zero, which is why the octal value was tested in line 1Ø of the program.

4-46 System Enable (SYE). The SYE control mode is used to control the outputs of most output cards (see Table 4-3). To enable the outputs, SYE must be programmed on. When SYE is programmed off, the outputs of all cards affected by SYE are disabled; i. e., are placed in a predetermined condition. For example, resistance outputs are shorted voltage outputs are held at 0 volts and digital outputs are held in the high or zero state, depending on the card.

4-47 As shown in previous examples, SYE can be programmed on in a control word prior to sending the data word to the output card. In some instances, the user may wish to take advantage of the SYE off condition which removes the outputs from all affected output cards simultaneously. In this case, transmitting a control word with SYE off will cause all affected cards to go to the predetermined condition. Example 10 shows a control word with SYE off. Since leading zeros need not be sent, the control word consists only of "O" (control word tag) and "T" (multiprogrammer gate code). When Example 10 is executed, all output cards responding to SYE will go to a predetermined condition.

Example 10: Transmitting a Control Word With SYE Off

```
0: fmt 1,c,z
1: wrt 723.1,"OT"
```

4-48 Thus, all output cards affected by SYE are in a "safe" condition when SYE is off and will output data when SYE is on, whether or not the card is addressed.

4-49 Data Transfer Enable (DTE). The DTE control mode affects output cards having either external gate/flag capability or dual rank storage. DTE need not be programmed on if a particular card does not have either of these capabilities, but no harm is done if it is programmed on. When using the timing mode, it is generally good practice to program both TME and DTE on together. The reason for this is that the four cards affected by DTE (69321B, 69330A, 69331A, and 69370A) cannot drive the CTF flag line unless DTE is on. Thus, if any of these cards are used in the timing mode, with DTE off, a hang-up will occur. When DTE is on and one of the affected cards is addressed and gated ("T" character) it will immediately return the leading edge of the CTF flag. Depending upon the card model, the trailing edge of the flag is either controlled by circuits on the card itself or by an external circuit connected to the card. For models 69321B and 69370A (D/A cards), the trailing edge of the flag is controlled by circuits on the card itself. For models 69330A (Relay Output) and 69331A (Digital Output), the trailing edge of the flag is normally controlled by an external device.

4-50 Cards With External Gate/Flag Capability. The digital output card (69331A) and the relay output card

(69330A) have external gate/flag capability. This feature allows these cards to transfer data to an external device which requires a gate input to receive data.

4-51 Normally DTE will be programmed on in a control word, before sending a data word to the card, as shown in Examples 8 and 9. As each data word is received, it is loaded into the addressed card and immediately strobed (gated) into an external device.

4-52 Some multiple card applications may require data to be loaded into several output cards, then simultaneously gated into external devices. In this case, a control word may be sent with DTE off, followed by appropriate data words, then another control word with DTE on. As each data word is received, the data will be immediately transferred to the output of the addressed card. However, the gate from the output card will not be enabled until a control word containing DTE is received.

4-53 Example 11 illustrates loading data into two output cards with DTE off, then turning DTE on to simultaneously generate gates to external devices.

4-54 In Example 11, output cards located in multiprogrammer slots 4Ø1 and 4Ø2 (A and B) are sequentially loaded with data. Notice that SYE is on, allowing the data to be immediately transferred to the output of the cards. After both cards have been loaded with data, DTE is turned on by the control word, "ØØ14ØT", resulting in simultaneous generation of gate signals to external devices.

4-55 If it is not required to simultaneously gate external devices from multiple cards, DTE can be included in the first control word of Example 11, (i. e. ØØ14ØT) and the control word following the data words can be eliminated. Each card would then generate a gate as it receives a data word, as previously described.

4-56 Cards With Dual Rank Storage. The D/A voltage (69321B) and current (69370A) converter output cards have dual rank storage capability. This feature allows the outputs of any number of D/A cards to be changed to new values simultaneously. In this case, a control word is sent with DTE off, followed by appropraite data words, then another control word with DTE on. As each data word is received, the data will be loaded into the first rank storage of the addressed card. When a subsequent control word with DTE is received, all previously addressed D/A cards will simultaneously transfer the contents of the first rank storage into second rank storage and to the output D/A conversion circuits.

4-57 Example 11, which is shown for programming DTE for digital output and relay output cards, can also be used for D/A cards. In this case, the first rank storage of D/A cards located in multiprogrammer slots 4Ø1 and 4Ø2 (A and B) would be loaded with new data. After all cards have received new data, DTE is turned on by the control word "ØØ14ØT", resulting in both cards transferring data from first rank to second rank storage and converting the data to analog outputs.

4-58 As was the case for the 69330A and 69331A cards, DTE can also be programmed on first, before sending data words to the D/A cards. As each data word is received, it will be loaded into the addressed card and converted to an analog voltage/current that will be present at the output of the card.

4-59 Timing Mode (Extended Handshake). The TME control mode allows output cards with CTF circuits to control the multiprogrammer flag. Table 4-3 lists the cards which may be used in the TME mode. When programming output cards with TME on, the receipt of a CTF card flag will set the service request (SRQ) bus line to notify the calculator that the output card(s) has timed out (completed data transfer). A serial poll is then performed to confirm that the multiprogrammer system had requested service and to reset the SRQ line.

4-60 Serial or parallel methods may be used when programming output cards in the timing mode. The serial method programs one output card at a time while the parallel method allows programming one or more output cards at once.

4-61 Serial Method. Example 12 illustrates programming two output cards serially (one at a time) in the timing mode. The interrupt capability of the 9825A extended I/O ROM will be utilized to detect service requests from HP-IB interface 7. First, a linkage is established between HP-IB select code 7 and a service routine called "next". Then a control word with both TME and DTE

Example 11: Programming Multiple Cards With DTE

```
0: fmt 1,c,z
1: wrt 723.1,"00040TA7777TB7777T00140T"
```

on is transmitted, placing the multiprogrammer in timing mode. Since certain output cards cannot drive CTF unless DTE is on, DTE and TME should be programmed on together (see paragraph 4-49). With TME on, it is then necessary to do serial poll before addressing the first output card with a data word. This clears the service request line, which was set by the trailing edge of the CTF handshake flag when the control word with TME was transmitted. From this point on, the service request will be sent only as a function of an output card timing out. After a data word is sent to the first card, HP-IB interface 7 is enabled for interrupts.

4-62 When the first card times out, the service request (SRQ) line is set. This causes a jump to service routine "next". Service routine "next" does a serial poll to determine if the multiprogrammer system generated the service request. Assuming that the multiprogrammer was the device requesting service, the second data word is then transmitted. Notice that interface 7 was not re-enabled for interrupt after data was sent to the second card. In this example no action will be taken after the second card times out even though service request will be set upon completion of the data transfer. If the user desires his program to take further action, a new service routine could be designated and interface 7 could be re-enabled for interrupt after transmitting data to the second card.

4-63 If service routine "next" finds that the multiprogrammer was not the bus device requesting service, as indicated by the status byte not being equal to 64, it takes no action and returns. However, since an interrupt had occurred, interface 7 will no longer be enabled for interrupt. It is up to the user's program to determine the device that requested service and clear the service request line before re-enabling interface 7 to interrupt. If interface 7 is not re-enabled for interrupts, the second card will not be programmed.

4-64 Parallel Method. Example 13 illustrates parallel (simultaneous) programming of two output cards in the timing mode. A control word with DTE and SYE, followed

Example 12: Timing Mode of Operation, Serial Method

```
0: rds(723)→C
1: oni 7,"next"
2: fmt 1,c,z
3: wrt 723.1,"00160T";rds(723)→C
4: wrt 723.1,"B5252T";eir 7

             •

             •

             •

20: "next":if rds(723)#64;gto "iret"
21: wrt 723.1,"C2525T"
22: "iret":iret
```

Explanation:

ɸ: Clear service request line from multiprogrammer at beginning of program.
1: Establish interrupt linkage between HP-IB interface 7 and routine labeled "next".
2: Format statement.
3: Send control word with TME, then reset service request caused by control word with TME on.
4: Data word is sent to the output card in multiprogrammer slot 402 (B). Interrupt is enabled after character (T) initiates timing sequence with first card. Main program continues from this point until an interrupt occurs.
2ɸ: Beginning of interrupt routine. If status byte of multiprogrammer is not equal to 64, first card has not completed transfer. Return to main program.
21: Status byte of multiprogrammer is equal to 64. Data word is sent to the output card in multiprogrammer slot 403 (C) to initiate timing sequence with second card.
22: Interrupt return statement.

by data words addressing the two cards, and then another control word with DTE, SYE, and TME are sent to the multiprogrammer. Each output card will begin its timing sequence upon receipt of the "T" in its data word. Since TME is turned on after the timing sequences have started, a trailing edge of the CTF flag will not be returned to set the multiprogrammer's service request until both cards have timed out.

4-65 When both cards time out, the service request (SRQ) line is set. This causes a jump to service routine "done". Service routine "done" does a serial poll to determine if the multiprogrammer system generated the service request. Assuming that the multiprogrammer was the device requesting service, "done" is printed by the calculator as an indication that both cards have timed out. After printing "done", program execution returns to the main program. Of course, the user's service routine does not have to print "done", but could take any action desired.

4-66 If the multiprogrammer was not the bus device requesting service, the same action described in paragraph 4-63 must be taken or the calculator will never notify the user upon completion of the data transfers.

4-67 When programming one output card in the timing mode, the user has the option of using either the serial or the parallel method. The user will probably find the parallel method more convenient since it eliminates the necessity for the initial serial poll required in the serial method.

4-68 Although the previous examples used the interrupt system to monitor service requests, the user has the option of monitoring the status of the bus to determine if a service request is present. If a status check determines that a service request is present, the user can do a serial poll to determine if the multiprogrammer was the device requesting service and take whatever action desired.

4-69 Notice that the previous two examples did not read the device status into a variable when doing a serial poll of the multiprogrammer in the interrupt service routine. Again, this is optional depending on whether the user desires to save the status byte.

4-70 One final note on using the timing mode with output cards. It is not necessary to use the 9825A interrupt system or bus status checks when using timing mode. If the user is not concerned with tying up the calculator for the duration of data transfers, he may send a control word containing

Example 13: Timing Mode of Operation, Parallel Method

```
0: rds(723)→C
1: oni 7,"done"
2: fmt 1,c,z
3: wrt 723.1,"00140TB5252TC2525T00160T";eir 7
                    •
                    •
                    •
20: "done":if rds(723)#64;gto "iret"
21: prt "done"
22: "iret":iret
```

Explanation:

Ø: Clear service request line from multiprogrammer at beginning of program.
1: Establish interrupt linkage between HP-IB interface 7 and routine labeled "done".
2: Format statement.
3: Control word with DTE, SYE is sent to multiprogrammer. Data words are sent to output cards in slots 402 (B) and 403 (C). The "T" in each data word initiates the timing sequence for each card. Another control word with DTE, SYE, and TME on is sent to multiprogrammer. Interface 7 is enabled for interrupt after sending second control word. Main program continues from this point until an interrupt occurs.
2Ø: Beginning of interrupt routine. If status byte of multiprogrammer is not equal to 64, cards have not completed data transfer. Return to main program.
21: Status byte of multiprogrammer is equal to 64. Calculator prints "done".
22: Interrupt return statement. Main program resumes executing.

TME followed by a data word to the appropriate card, then another control word with TME off (e. g. wrt 723.1, "ØØ16ØTB2525TØØØ4ØT"). In this case, the calculator will pause until the data transfer with the card has been completed before sending the second control word. Of course, the service request will be set when the output card times out and should be cleared by doing a serial poll.

4-71 Avoid Hang-Up Conditions in the Timing Mode. During the TME mode, there is a wait-for-flag interval that starts from the time that the output card is addressed with a gate ("T") and ends when the card returns a CTF flag and thus sets the service request line. During the wait-for-flag period, the calculator is said to "hang-up" if another output or input operation is attempted. That is, any additional output words sent to the multiprogrammer system will not be obeyed until the flag from the programmed card is returned. If the calculator is in hang-up because of a user error, it can remain there indefinitely; waiting for a service request (CTF flag) that will never occur. For output card programming in the TME mode, hang-up can be caused by one of three possible problems:

(1) A card without CTF circuitry (Note 5, Table 4-3) was addressed and gated (with a data word) while TME was on.

(2) A card with external gate/flag capability (69330A or 69331A) was addressed and gated, as is normal, *but* the gate/flag terminals on the card are open (CTF flag cannot be terminated).

(3) DTE was not turned on with TME and a card affected by DTE (Table 4-3) was addressed and gated.

4-72 The hang-up of problem (1) results in the initiation of a gate within the multiprogrammer which cannot be terminated. This hang-up can be cleared by sending an "X" prior to the desired control word, as shown in Example 17 later in this chapter. To avoid this problem, the user must bear in mind that cards without CTF circuits are not intended for timing mode operation and should never be addressed and gated while TME is on. Problem (2) of the previous paragraph results in an unterminated CTF flag and cannot be cleared by sending the statement of Example 17. The only way to clear this hang-up is to remove input power from the 6940B unit (toggle the LINE switch off and then on). To avoid this problem, the user must simply remember to jumper the gate and flag terminals together on any 69330A or 69331A cards that do not use an external timing circuit. Problem number (3) results in an unterminated gate within the multiprogrammer, the same as problem (1), and can be cleared by sending an "X" as indicated in Example 17. As described in Paragraph 4-49, problem (3) can be avoided by programming TME and DTE on together.

4-73 Input Cards, General Programming Techniques

4-74 Control Mode Bits for Input Card Operations. As indicated in Table 4-3, three control mode bits are applicable to input card programming; ISL, TME, and IEN. The input select (ISL) bit must be programmed on prior to any basic read sequence in which return data is sent back to the calculator from an input card. The TME and IEN bits are used to establish the timing and interrupt modes of operation, as will be explained subsequently. Table 4-1 lists the three control mode codes that are most commonly used for input card operations.

NOTE

All control words used in this section assume a unit address of ØØ.

4-75 System Enable Bit. Notice that the SYE bit is on in all three of the input mode codes shown in Table 4-1. Although SYE does not affect input cards, it is usually good practice to have SYE on in control words specifying an input mode of operation. If SYE is turned off, and output cards are included in your system, the outputs of these cards will be disabled creating what may be an undesirable situation. Hence, all of the control word examples shown in the input portion of this chapter show SYE on.

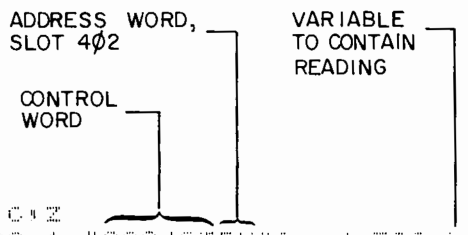
4-76 Basic Input Operating Modes. There are two basic programming modes for input cards; serial and parallel. A serial operation can involve the programming of a single input card or a group of cards, programmed one at a time. A parallel operation, involves the simultaneous programming of a group of cards.

4-77 Within the boundaries of serial operation there are two programming sequences which are often used with input cards. These are the simple direct read without gate sequence and the timing mode (extended handshake). In the direct read sequence the input card is addressed, without a gate, and the existing data on the card is returned immediately to the calculator. In the timing mode sequence (TME on) the card is addressed with a gate which initiates a gate/flag transfer cycle between the card and its' external device. After the transfer is completed, the fresh data on the input card is read into the calculator.

4-78 Parallel operation for input cards is synonymous with the interrupt mode of operation (TME and IEN on). The interrupt mode allows the user to activate (arm) a group of input cards and then provides an indication (in the form of a service request) when the first card is ready with data or the first external process has been completed. Sample programs for the read without gate sequence, timing mode and interrupt mode are provided in the following paragraphs.

4-79 Reading in a Return Data Word. At some point(s) in any input card program, the user will want to read the return data available on the card into the calculator. An example of a simple read without gate sequence was shown in Example 6 and described in Paragraph 4-29. Because this operation is used many times when programming input cards, this example is shown again for convenience. To summarize the read without gate sequence shown in Example 14, a control word is sent with ISL on, followed by an address word directing that the input card in slot 402 return its data. The read (red) statement allows the return data to be stored in the variable "X". Notice that the read without gate sequence of Example 14 is the only recommended way to obtain data from input cards that do not have CTF flag circuits (see Note 5 of Table 4-3). If the user desires to employ a direct read without gate sequence for a second input card, he can simply follow line 1 of Example 14 with a similar line, omitting the control word and changing the card slot address and variable as desired. It is not necessary to retransmit the control word since it has not been changed elsewhere in the program.

Example 14: Addressing an Input Card and Entering Its Return Data



```
0: fmt 1,c,z
1: wrt 723.1,"00240TBX";red 723,X
```

4-80 Checking for IRQ in the Return Data Word. As indicated previously, the return data stored in the variable of Example 14 can be used directly if it is obtained from an input card that does not contain IRQ (see Table 4-3). If the card does contain the IRQ function, the return data must first be examined to determine if there is a "1" in the IRQ field. If the IRQ character is a "1", it must be subtracted from the return data before the data can be utilized. Example 15 shows how to check for IRQ and subtract it, if necessary. Example 15 would immediately follow Example 14 for any cards that have IRQ. In line 2, the return data in variable "X" is checked for the presence of IRQ. If IRQ is present, the number 1ØØØØ is subtracted

Example 15: Checking for IRQ and Subtracting it

```
2: if X<10000;-1→X;gto "continue"
3: X-10000→X
4: "continue":
```

from the return data in "X". If IRQ is not present, –1 is stored in "X" as an indication that the data was not current data, and the program continues execution. Of course, –1 (used here for explanation purposes) is optional.

4-81 Octal to Decimal Conversion. For some input cards, such as the 69431A Digital Input, the user will find it convenient to work with the octal data returned to the variable "X" in the previous examples. For other cards, such as the 69421A Voltage Monitor, it will be more convenient to work with decimal values. In these cases, the octal to decimal function of the calculator can be utilized. To use this function, simply add the following line to your program: otd X→X. The placement of this program line depends on whether or not the card used IRQ. If IRQ is not used, it can be inserted immediately after line 1 of Example 14. If IRQ is used, the line must be inserted after line 3 (subtraction of IRQ) in Example 15.

4-82 Timing Mode (Extended Handshake). The TME control mode permits a data input transfer cycle between an input card and its external device. This is accomplished via the gate/flag method in which input cards with CTF circuits control the flag line. (Table 4-3 shows the input cards that have CTF circuits.) During the TME mode, the trailing edge of the CTF flag will set the service request (SRQ) line to notify the calculator that the input card has completed its data transfer. A serial poll is then done to confirm that the multiprogrammer had requested service and to reset the SRQ line.

4-83 Example 16 shows how to program a single input card in the timing mode. As shown previously with output cards, the interrupt capability of the 9825A calculator will be used to detect service requests from HP-IB interface 7. First a linkage is established between HP-IB select code 7 and a service routine called "read". Then a control word with ISL and TME is sent, placing the multiprogrammer in the input timing mode. A serial poll must be done before sending an address word to the input card to clear the service request line, which was caused by sending the control word containing TME. From this point on, the service request will be set only as a function of an input card timing out.

4-84 When the input card times out, the service request

(SRQ) line is set, causing a jump to service routine "read". Service routine "read" does a serial poll to determine if the multiprogrammer system generated the service request. Assuming that the multiprogrammer was the device requesting service, data is read into variable X. IRQ is then subtracted from the value in X. The calculator then prints the value read from card slot 402 (B) and returns program control to the main program. The user has the option of taking any action desired within the service routine rather than merely printing out the data value.

4-85 If service routine "read" finds that the multiprogrammer was not the bus device requesting service, as indicated by the status byte not being equal to 64, it takes no action and returns. However, since an interrupt had occurred, interface 7 will no longer be enabled for interrupt. It is up to the user's program to determine the device that requested service and clear the service request line before re-enabling interface 7 to interrupt. Of course, if interface 7 is not re-enabled for interrupt, the card will never be read.

4-86 If more than one input card is to be read in the timing mode, they must be read in a serial fashion. That is, the first card must complete its data transfer before the second card begins. In this case, line 4 could be re-transmitted using the next card address. It is not necessary to transmit the control word (with ISL, TME) again, unless a different control word was subsequently sent to the multiprogrammer.

4-87 As mentioned previously for output cards, the user has the option of using bus status checks to monitor SRQ, rather than using the calculator interrupt system. However, the calculator interrupt system allows a faster response time to bus service requests.

4-88 One final note on using the timing mode. It is not necessary to use the 9825A interrupt system or bus status checks when using the timing mode. If the user is not concerned with tying up the calculator for the duration of data transfers, he may send a control word containing

Example 16: Timing Mode for One Input Card

```
0: rds(723)→C
1: oni 7,"read"
2: fmt 1,c,z
3: wrt 723.1,"00260T";rds(723)→C
4: wrt 723.1,"BT";eir 7

                •

                •

                •

20: "read":if rds(723)#64;gto "iret"
21: red 723,X;X-10000→X
22: fxd 0;prt "data=",X
23: "iret":iret
```

Explanation:

0: Clear service request line from multiprogrammer at beginning of program.
1: Establish interrupt linkage between HP-IB interface 7 and routine labeled "read".
2: Format statement.
3: Send control word with ISL and TME, then reset service request caused by control word with TME on.
4: Address word is sent to the input card in multiprogrammer slot 402 (B). Interrupt is enabled after character "T". initiates timing sequence with card. Main program continues from this point until an interrupt occurs.
20: Beginning of interrupt routine. If status byte of multiprogrammer is not equal to 64, the first card has not completed transfer. Return to main program.
21: Status byte of multiprogrammer is equal to 64. Return data word is read into variable X and IRQ is subtracted.
22: Data read from card in slot 402 (B) is printed on calculator.
23: Interrupt return statement. Main program resumes executing.

ISL and TME followed by an address word containing a gate code "T" and another address word containing a gate code "X" (wrt 723.1, "00260TBTBX"; red 723,X). In this case, the calculator will pause until the data transfer is complete before reading the return data word into "X". Of course, the service request will be set when the input card times out and should be cleared by doing a serial poll after reading the return data word.

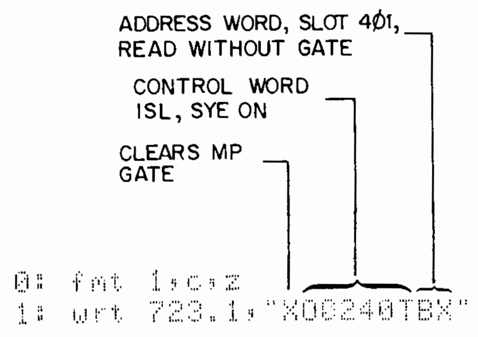
4-89 Avoiding "Hang-up" Conditions in the Timing Mode. During the TME mode, there is a wait-for-flag interval that starts from the time that the input card is addressed with a gate ("T") and ends when the card returns a CTF flag and thus sets the service request line. During the wait-for-flag period, the calculator is said to "hang-up" if another input or output operation is attempted. That is, any additional output words sent to the multiprogrammer system will not be obeyed until the flag from the programmed card is returned. If the calculator is in hang-up because of a user error, it can remain there indefinitely waiting for a service request (CTF flag) that will never occur. For input card programming in the TME mode, hang-up can be caused by one of two possible problems.

(1) A card without CTF circuitry (see Note 5, Table 4-3) was addressed and gated ("T" sent in an address word) while TME was on.

(2) A card with external gate/flag capability (69431A only) was addressed and gated, as is normal, but the gate/flag terminals on the card are open (CTF flag cannot be terminated).

4-90 The hang-up of problem number (1) results in the initiation of a gate within the multiprogrammer which cannot be terminated. Cards without CTF circuits are not intended for timing mode operation and should never be addressed and gated while TME is on. Data from these cards should be obtained only by reading without a gate, as previously recommended in Paragraph 4-79. If you think that you have improperly gated one of these cards, the following example shows a method of clearing the hang-up condition.

Example 17: Clearing a Hang-Up Caused by an Unterminated Gate



```
0: fmt 1,c,z
1: wrt 723.1,"X00240TBX"
```

The "X" before the control word of Example 17 clears the multiprogrammer system by terminating the gate signal, allowing a control word to be sent. The control word turns ISL on and the address word ("BX") initiates a read without gate sequence for the input card in slot 4Ø1 which is assumed to have been improperly addressed. Problem number (2) of the previous paragraph is caused by an unterminated CTF flag and cannot be cleared by sending the statement of Example 17. The only remedy for an unterminated flag is to remove input power from the 6940B unit (toggle the LINE switch off and then on). To avoid this problem the user must simply remember to jumper the gate and flag terminals together on any 69431A cards that do not use an external timing circuit.

4-91 Interrupt Mode of Operation. As indicated in Table 4-3, the interrupt mode can be used only with 69431A, 69434A, 69436A, or 69600B cards. These cards must be activated (armed) before they are able to cause an interrupt. As is described in the interrupt mode description of Chapter VII, (Paragraphs 7-52 through 7-59), there are two ways of arming the cards; individually or as a group. The most commonly used method is individual arming which is permitted only when jumper W6 is removed from the card. For this method, a control word is sent first, with ISL on. The cards are then armed, one at a time, by sending each card an address word with a gate ("T"). The interrupt mode is then established by sending a control word with TME and IEN on. The first armed card that interrupts (receives a data ready indication from its external device), returns a CTF flag setting the service request line. After a serial poll to verify that the multiprogrammer interrupted, TME and IEN are turned off to prevent further interrupts and an IRQ poll (interrupt search) is performed. During the IRQ poll, a read without gate sequence is done for each card and the return data is examined for the presence of IRQ. If the IRQ character is a "1", it indicates that the addressed card had interrupted and its' return data can be stored for processing. If the IRQ character is a "0", it indicates that the card has not yet received new data or detected a change of state in the external device. Once the cards have been polled, the interrupting card (or cards) must be reset (either disarmed or recycled) or it will cause another interrupt if TME and IEN are programmed on again. As indicated in Table 4-4, the requirements for recycling, disarming, and interrupting are different for each card. Notice also, that most of these cards are recycled or disarmed in the output mode using data words. Interrupt mode programs for these cards are given in the individual card programming chapter. Additional information can be found in the instruction manual for each of these cards.

4-92 An interrupt sequence using the group arming (jumper W6 installed) technique is similar to the procedures just described except for the arming procedures at the beginning of the program. However, in this case, all cards with jumper W6 installed are armed simultaneously when a control word is sent with TME and IEN on. Once an interrupt occurs (SRQ line set), an IRQ poll is done in a manner similar to that described previously in the individual arming sequence. Although the group arming technique saves programming steps, it can be difficult to use unless all input cards in the system are being used in the same manner and the user fully understands all of the implications (read Paragraph 7-59 in Chapter VII). Therefore, it is recommended that the user remove jumper W6 from all cards having an interrupt capability and use the individual arming method.

4-93 Example 18 illustrates an interrupt sequence using three input cards from which jumper W6 has been removed (cards are individually armed). The example assumes that the input cards are located in slots 4Ø1, 4Ø2, and 4Ø3 (A, B, and C).

4-94 As with the timing mode examples, the interrupt capability of the 9825A calculator is used to enable a fast response to a service request. In this example, variables X, Y, and Z are used to store either valid data if the card interrupted or a value of −1 if it did not. Service routine "poll" also prints the data and slot number of the interrupting card before returning to the main program. The user has the option of taking any action desired in the service routine rather than printing the interrupting slot number and data. In addition, any card slots or variables desired can be used.

4-95 The user should keep in mind that an interrupt on service request may be caused by another device on the bus. As mentioned previously in Paragraph 4-85, the service request line must then be cleared and the calculator interrupt on interface 7 re-enabled. If this is not done, the multiprogrammer interrupt will not be detected (unless bus status checks are also made in the program).

4-96 At the completion of the interrupt sequence of Example 18, the user may wish to begin a non-interrupt operation. In this case, it is a good practice to disarm any input cards that are still active (had not interrupted) in order to avoid any possibilities of hang-ups or erroneous results in the other modes. Similarly, the user should include only the four card types listed in Table 4-4 in his interrupt sequence to avoid the same possibilities.

4-97 Once the interrupt mode is established, the user may want to proceed to an unrelated operation, such as programming a digital output card. To leave the interrupt mode before a service request occurs, the user must send an X to the multiprogrammer system as illustrated in Example 17. If this is done, and the 9825A interrupt system is being used to monitor service requests, interface 7 must then be disabled for further interrupts by an (eir 7,Ø) statement before programming any other cards. To return to interrupt mode of operation, the user must first clear (and process if necessary) any service requests present on the bus. The multiprogrammer service request line may be cleared by doing a serial poll. After clearing the service request line, a control word with TME and IEN on is sent and interface 7 is re-enabled for interrupts. (e. g. rds (723)→C; wrt 723.1, "ØØ46ØT"; eir 7). Caution is necessary when using this procedure with 69434A event sense cards, as these cards do not store interrupts that occur when the multiprogrammer is not operating in the interrupt mode.

Example 18: Interrupt Mode of Operation, Individual Arming

```
0: rds(723)→C
1: oni 7,"poll"
2: fmt 1,c,z
3: wrt 723.1,"00240TATBTCT00460T";eir 7

                 •

                 •

                 •

20: "poll":fxd 0;if rds(723)#64;gto "iret"
21: wrt 723.1,"00240TAX";red 723,X
22: if X<10000;-1→X;jmp 2
23: X-10000→X;wrt 723.1,"00040TAT"
24: wrt 723.1,"00240TBX";red 723,Y
25: if Y<10000;-1→Y;jmp 2
26: Y-10000→Y;wrt 723.1,"00040TBT"
27: wrt 723.1,"00240TCX";red 723,Z
28: if Z<10000;-1→Z;jmp 2
29: Z-10000→Z;wrt 723.1,"00040TCT"
30: if X>-1;prt "slot A data=",X
31: if Y>-1;prt "slot B data =",Y
32: if Z>-1;prt "slot C data =",Z
33: "iret":iret
```

Explanation:

0: Clear service request line from multiprogrammer at beginning of program.

1: Establish interrupt linkage between HP-IB interface 7 and routine labeled "poll".

2: Format statement.

3: Control word with ISL and SYE on is sent to multiprogrammer. Cards in slots 401 (A), 402 (B), and 403 (C), are sequentially armed by address words "AT", "BT", and "CT" thus initiating the interrupt sequence for each card. Another control word "00460T" establishes the interrupt mode by turning IEN, SYE, and TME on. With TME and IEN on, the service request will not be set until the first card(s) interrupts. Interface 7 is enabled for interrupts.

20: Beginning of interrupt routine. If status byte of multiprogrammer is not equal to 64, multiprogrammer has not interrupted. Return to main program.

21: Status byte of multiprogrammer is equal to 64. Control word with ISL, SYE on is sent, followed by an address word ("AX") to start IRQ poll. "AX" permits a read without gate sequence for first input card in slot 401. Data is read into variable "X".

22: Return data from first card is examined for IRQ. If IRQ is not present, first card has not interrupted. Store −1 in variable "X" as an indication card had not interrupted; jump to read without gate sequence for second card.

23: IRQ is present in return data from first card, indicating first card had interrupted and its return data is valid. Subtract IRQ from data and store in "X". Disarm first card; control word with SYE on establishes output mode followed by data word "AT" to disarm (reset) the card.

24-26: Same sequence as 21-23, using second input card in slot 402 (B) and variable "Y".

27-29: Same sequence as 21-23, using third input card in slot 403 (C) and variable "Z".

30-32: Print out slot number and data of interrupting cards.

33: Interrupt return statement. Main program resumes executing.

Table 4-4. Requirements for Cards with Interrupt Capability

Card Model	Significance Of Data Sent In Output Mode	Arming Requirements (W6 Out)	Arming Requirements (W6 IN) *	Recycling Requirements	Disarming Requirements	Interrupting Requirements
69431A Digital Input		ISL on TME off Address true Gate	IEN on TME on Gate	ISL on TME off Address true Gate	ISL off TME off Address true Gate	External device returns trailing edge of flag
69436A Process Interrupt	All logical "1's" initializes card. Interrupting word recycles card. All logical "0's" disarms card.	ISL on TME off Address true Gate	IEN on TME on Gate	ISL off TME off Address true Data = Interrupting word. Gate	** ISL off TME off Address true Data = logical "0's". Gate	Any one or more of twelve paired positive or negative edge detectors tripped.
69600B Timer	Binary representation of pulse duration. All logical "0's" disarms card.	ISL on TME off Address true Gate	IEN on TME on Gate	ISL off TME off Address true Data = Duration of next pulse. Gate	ISL off TME off Address true Data = logical "0's". Gate	End of programmed pulse
69434A Event Sense	Reference word	ISL on TME off Address true Gate	IEN on TME on Gate		ISL off TME off Address true Gate	External Word IS = Ref. word IS ≠ Ref. word IS > Ref. word IS < Ref. word

* As shipped from factory.
** Card must be recycled prior to disarming.

Chapter V
PLUG-IN CARD DESCRIPTIONS AND PROGRAMS

5-1 This chapter provides GPIO and HP-IB interface programming examples for each type of plug-in card that can be presently used in a multiprogrammer system. The cards are presented in numerical sequence according to the card model number (69321 through 69601). Also, a brief functional description and a sample test program are provided for each type of card. Block diagrams and input/output connector diagrams (if applicable) are provided as programming aids. In addition, several programming examples of typical measurement applications using certain card types are provided after the individual card programs. Any subroutines that may be used in the sample programs can be found in Appendix A.

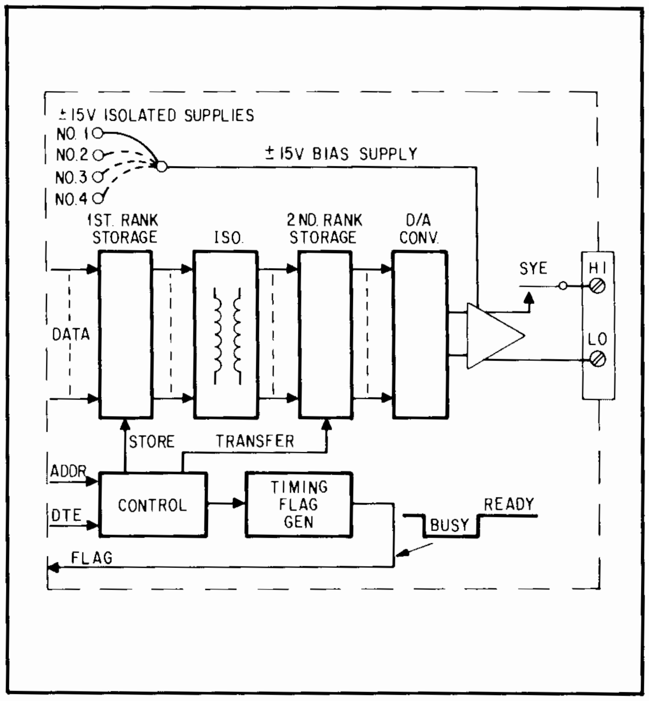
5-2 INDIVIDUAL CARD PROGRAMS

5-3 D/A Voltage Converter Card, 69321B

5-4 This card is a twelve bit bipolar (two's complement) digital to analog converter which may be programmed to output a voltage in the range of −10.240 through +10.235 volts. An explanation of the relationship between an output voltage and the corresponding twelve bit binary code is given in Paragraph A-32 of Appendix A. The following discussion will familiarize the user with programming the D/A voltage converter card, 69321B. Unless otherwise specified, all examples will assume the D/A voltage card is installed in unit Ø, Slot 4Ø2 (B), and a voltage regulator card 69351B (which supplies the required ±15V bias voltages) is installed in unit Ø, slot 6ØØ.

5-5 **Calculating Data Value.** The following steps will yield the octal data value required to program the 69321B to a desired output voltage. Disregard the sign when making calculations.

1. Divide the desired output voltage in volts by .005 (the LSB).
2. For negative voltages only, subtract the absolute value yielded in step 1 from 4096.



69321B Block Diagram

3. Calculate the octal equivalent of the decimal value yielded in step 1 or 2 (see Paragraph A-21 in Appendix A). For example, to calculate the octal data value required for +5 volts;

   5 ÷ .005 = 1000
   octal equivalent of 1000 = 1750

or, to calculate the octal data value required for −5 volts;

   5 ÷ .005 = 1000
   4096-1000 = 3096
   octal equivalent of 3096 = 6030

5-6 **Programming an Output Voltage.** Example 1 illustrates using a data constant of 6030_8 to program the output voltage of a 69321B card to −5 volts. A control

Example 1. Programming an Output Voltage

GPIO Interface

```
0: moct;wtb 9,170140,26030
```

HP-IB Interface

```
0: fmt 1,c,z
1: wrt 723.1,"00140TB6030T"
```

word is sent containing SYE and DTE, followed by a data word containing the card slot address and the data constant.

5-7 Although a 69321B card returns a CTF flag as a function of internal timing circuits, TME is not included in the control words of the programming examples. This is because the period of the internal timing circuits on this card is shorter than the period of the multiprogrammer handshake flag (without TME). Therefore, TME is not useful when programming this card.

5-8 Changing Output Voltages of Multiple 69321B Cards Simultaneously. The dual rank storage feature of 69321B cards and the DTE control mode allow the output voltages of any number of 69321B cards to be changed simultaneously. A detailed description of this technique is given in Chapter III (Paragraph 3-56) and Chapter IV (Paragraph 4-56).

5-9 In Example 1, DTE and SYE were programmed on in a control word prior to sending a data word to the card. In this case when the data word is received, it is loaded into the addressed card and immediately converted to an analog voltage that is present at the output of the card.

5-10 When it is desired to change the outputs of multiple 69321B cards simultaneously, a control word is sent with DTE off, followed by the appropriate data words, then another control word with DTE on. As each data word is received, the data will be loaded into the first rank storage of the addressed card. When a subsequent control word with DTE is received, all previously addressed D/A cards will simultaneously transfer the contents of first rank storage into second rank storage and the output D/A conversion circuits.

5-11 Example 2 illustrates using DTE with two 69321B cards. In this case, the first rank storage of the cards located in multiprogrammer slots 4Ø2 (B) and 4Ø3 (C) is loaded with new data. After both cards have received new data, DTE is turned on by a second control word, resulting in both cards transferring data from first rank to second rank storage and converting the data to analog outputs.

5-12 69321B Sample Test Program. Example 3 is a sample test program which illustrates the use of variable data values and allows the user to program a 69321B card to any value within its capability. If an output is requested which exceeds the capability of the card, the calculator will display "VOLTAGE OUT OF RANGE" and will not program the card.

5-13 Notice that the four program lines beginning with the line labeled "Vout" can be used as a voltage programming subroutine merely by changing the third line following "Vout" from (gto "end") to (ret). When used this way, the main program would supply the desired voltage (in volts) as variable A whenever the subroutine was called.

5-14 The decimal mode (mdec), used in the GPIO program, is optional, depending on whether the user wants to return the calculator to decimal mode after programming the multiprogrammer.

5-15 Test Procedure. Perform the following steps when using the 69321B sample test program.

1. Load the program into the calculator.
2. Connect a voltmeter capable of measuring bipolar voltages between terminals A and B of the output terminal block on the 69321B card. The high terminal of the voltmeter should be connected to terminal A.
3. Depress RUN on the calculator. The calculator will pause with "ENTER VOLTAGE" displayed. Entering the desired voltage, then depressing CONTINUE will cause the 69321B card to program the desired voltage.

Example 2. Simultaneously Changing Output Voltages of Multiple 69321B Cards

GPIO Interface

```
0: moct;wtb 9,170040,21750,36030,170140
```

HP-IB Interface

```
0: fmt 1,c,z
1: wrt 723.1,"U0040TB1750TC6030T00140T"
```

Example 3. 69321B Sample Test Program

GPIO Interface

```
0: ent "ENTER VOLTAGE",A
1: if A<-10.24 or A>10.235;gto "error"
2: "Vout":A/.005→A;if A<0;A+4096→A
3: dtoA→A
4: moct;wtb 9,170140,20000+A;mdec
5: gto "end"
6: "error":dsp "VOLTAGE OUT OF RANGE"
7: "end":end
```

HP-IB Interface

```
0: ent "ENTER VOLTAGE",A
1: if A<-10.24 or A>10.235;gto "error"
2: fmt 2,c,f4.0,c,z
3: "Vout":A/.005→A;if A<0;A+4096→A
4: dtoA→A
5: wrt 723.2,"00140TB",A,"T"
6: gto "end"
7: "error":dsp "VOLTAGE OUT OF RANGE"
8: "end":end
```

5-16 Power Supply/Amplifier Control Cards, 69325A – 69328A

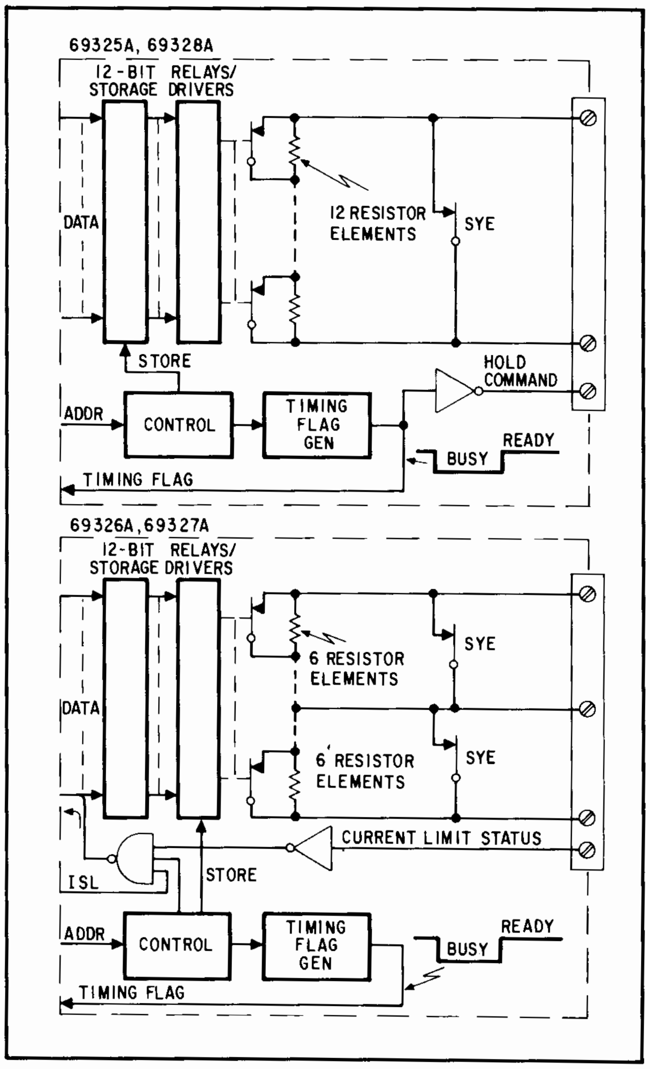
5-17 These resistance output cards are designed to control the gain, voltage, and current limit of Hewlett-Packard Bipolar Power Supply/Amplifiers (BPS/A).

1. Model 69325A controls the magnitude and polarity of the BPS/A output voltage. During the period that the resistance output of the card is changing, a hold command from the card causes the BPS/A voltage to remain constant, and then make a smooth transition to the new output voltage.

2. The 69326A or 69327A card is used to independently control the positive and negative current limits of a BPS/A, and to monitor the status of the BPS/A current limit.

3. The 69328A is used to program the gain of a BPS/A from zero (no gain) to the full gain value of X2, X4, X8,X20, or X40, depending on the particular model of the HP BPS/A being controlled.

5-18 The following discussion will familiarize the user with programming the power supply/amplifier control cards, 69325A-69328A. All examples will assume the card is installed in unit Ø, slot 4Ø2, (B). For 69325A and 69328A cards, the resistance output terminals are A and B. For 69326A-69327A cards, the negative current limit control will be referred to as channel 1, while the positive current limit control will be referred to as channel 2. Channel 1 resistance output terminals are C and B, while channel 2 resistance output terminals are A and B.



69325A/28A and 69326A/27A Block Diagrams

5-19 Resistance Outputs. Octal data is transmitted to program one 12-bit resistance output from a BPS/A control card as follows:

1. Voltage Control Card 69325A: 12-bits in 2's complement form program the resistance output which controls the voltage magnitude and polarity of a BPS/A. (Refer to Paragraph A-27 in Appendix A for an explanation of 2's complement coding.)

2. Current Control Cards 69326A and 69327A: Each card provides two-identical 6-bit resistance channels. Channel 1 controls the negative current limit and channel 2 controls the positive current limit of a BPS/A. The two least significant octal digits in the data field program the negative current limit (channel 1). Two most significant digits program the positive current limit (channel 2).

3. Gain Control Card 69328A: 12-bits program the gain of a BPS/A.

5-20 Table 5-1 is provided as an aid in calculating data values (in decimal) necessary to program the control cards to specific output resistances. The decimal value obtained from using the applicable formula must then be converted to an octal value before being transmitted to the control card.

5-21 Assume that it is desired to program the output resistance of a 69325A card to 12,000 ohms. Using the applicable formula in Table 5-1, the data value is calculated as follows:

$$D = \frac{12{,}000}{5} - 2048$$

$$D = 2400 - 2048$$

$$D = 352$$

Octal equivalent of 352 = 540 (see Paragraph A-21 in Appendix A).

5-22 69325A Voltage Programming Formulas. Table 5-2 is provided as an aid in calculating the data value necessary to program the desired output voltage of BPS/A's. Make calculations as follows:

1. Using Table 5-2, select the formula associated with the applicable BPS/A model, output range, and polarity.

2. Insert the desired voltage output value as V in the formula. The desired output value must not exceed the maximum limits specified in the 69325A card manual.

3. Calculate the required decimal value.

4. Calculate the octal equivalent of the decimal value obtained in step 3.

Table 5-1. 69325A – 69328A Resistance Formulas

BPS/A Control Card	Data Value (D) in Decimal Necessary To Program Desired Output Resistance (R) in Ohms
69325A	If desired resistance (R) is less than 10,240 ohms: $D = (R \div 5) + 2048$ If desired resistance (R) is from 10,240 to 20,475 ohms: $D = (R \div 5) - 2048$
69326A	Channel 1 or Channel 2: $D = R \div 160$
69327A	Channel 1 or Channel 2: $D = R \div 80$
69328A	$D = R \div 5$

Table 5-2. 69325A Voltage Programming Formulas

BPS/A Models	Data Value (D) In Decimal Necessary To Program BPS/A To Desired Output Voltage (V) *			
	BPS/A High Range		BPS/A Low Range	
	Positive	Negative **	Positive	Negative **
6825A/6830A	D = V ÷ .01	D = 4096 – (V ÷ .01)	D = V ÷ .0025	D = 4096 – (V ÷ .0025)
6826A/6831A	D = V ÷ .025	D = 4096 – (V ÷ .025)	D = V ÷ .0025	D = 4096 – (V ÷ .0025)
6827A/6832A	D = V ÷ .05	D = 4096 – (V ÷ .05)	D = V ÷ .005	D = 4096 – (V ÷ .005)

* V is specified in volts

** Use absolute value of V, disregard sign.

5-23 For example, to calculate the octal value required to program + 12.9 volts from a 6825A BPS/A:

1. D = V ÷ .01
2. D = 12.9 ÷ .01
3. D = 1290
4. Octal equivalent of 1290 = 2412

5-24 69326A/69327A Current Limit Formulas. Table 5-3 is provided as an aid in calculating the data value necessary to program the positive and negative current limits of BPS/A's. Make calculations as follows:

1. Using Table 5-3, select the formula associated with the applicable control card and BPS/A model.

2. Insert the desired negative current limit (channel 1) value as I in the selected formulas (use absolute value). The desired value must not exceed the maximum limits specified in the 69326A or 69327A card manual.

3. Calculate the decimal value required.

4. Repeat steps 2 and 3 for the desired positive current limit (channel 2).

5. Calculate the octal equivalent of values obtained in steps 3 and 4.

6. Multiply the octal value obtained for the positive current limit (channel 2) by 100 (decimal).

7. Add the octal values obtained for channel 1 (negative) and channel 2 (positive) together.

5-25 For example, to calculate the octal value required to program a 6826A BPS/A to –160mA (channel 1) and +320mA (channel 2) proceed as follows:

1. D = I ÷ 16
2. D = 160 ÷ 16
3. D = 10 (channel 1)
4. D = 320 ÷ 16
   D = 20 (channel 2)
5. channel 1 octal = 12
   channel 2 octal = 24
6. 24 X 100 = 2400
7. 12 + 2400 = 2412

5-26 69328A Gain Formulas. Table 5-4 is provided as an aid in calculating the data value necessary to program the desired gain of BPS/A's. Make calculations as follows:

1. Using Table 5-4 select the formula associated with the applicable BPS/A model and output range.

Table 5-3. 62326A/69327A Current Limit Formulas

Current Control Card	BPS/A Models	Data Value (D) in Decimal Necessary To Program BPS/A To Desired Positive or Negative Current Limit (I) *
69326A	6825A/6830A	D = I ÷ 32
	6826A/6831A	D = I ÷ 16
69327A	6827A/6832A	D = I ÷ 8

* I is specified in mA. Use absolute value in making calculations, disregard sign.

Table 5-4. 69328A Gain Formulas

BPS/A Models	Data Value (D) in Decimal Necessary To Program BPS/A To Desired Gain (G) High Range	Low Range
6825A/6830A	D = G ÷ (4 ÷ 2048)	D = G ÷ (1 ÷ 2048)
6826A/6831A	D = G ÷ (10 ÷ 2048)	D = G ÷ (1 ÷ 2048)
6827A/6832A	D = G ÷ (20 ÷ 2048)	D = G ÷ (2 ÷ 2048)

2. Insert the desired gain as G in the formula. The desired gain must not exceed the limits specified in the 69328A manual. Note that the maximum programmable gain is 1 LSB less than the maximum gain specified in the 69328A manual.

3. Calculate the required decimal value.

4. Calculate the octal equivalent of the value obtained in step 3.

5-27 Programming Examples. Example 4 illustrates using a data constant of 2412 to program a BPS/A control card located in slot 402 (B). The timing mode of operation used in described in GPIO Chapter 3, paragraph 3-68 and HP-IB Chapter 4, paragraph 4-70.

5-28 Reading Status from 69326A/69327A Cards. The 69326A and 69327A cards examine the current limit status of the BPS/A. Example 5 illustrates the format required to read the status. In Example 5, the return data word is input to variable "X". The return data word will equal 1 if the BPS/A is in current limit or 0 if it is not. Note that the card is always read without a gate. This is necessary to prevent changing the current limit values stored on the card.

5-29 69325A and 69328A Sample Test Program. Example 6 is a sample program which illustrates the use of variable data values and allows the user to program a 69325A or 69328A card to any resistance value within its capability. If an attempt is made to program a resistance value exceeding the capability of the card, the calculator will display "RESISTANCE TOO LARGE" and will not program the card.

5-30 Example 6 can only be used to program a 69328A card. To program a 69325A card, replace the program line beginning with (A/5→A) with the following three lines:

```
If A>20475; gto "error"
If A<10240; A/5+2048→A; dto A→A
If A>=10240;A/5–2048→A; dto A→A
```

NOTE

Since it is only possible to program a resistance which is a multiple of the card LSB, examples 6 and 7 round off resistance values requested by the user to the nearest LSB. This is accomplished by the dto function of the calculator which rounds off fractions.

Example 4. Programming a BPS/A

GPIO Interface

```
0: moct;wtb 9,170160,22412,170040
```

HP-IB Interface

```
0: fmt 1,c,z
1: wrt 723.1,"00160TB2412T00040T"
2: rds(723)→C
```

Example 5. Reading BPS/A Current Limit Status

GPIO Interface

```
0: moct;wtb 9,170240
1: wti 0,11;wti 4,20000;rdi 4→X
```

HP-IB Interface

```
0: fmt 1,c,z
1: wrt 723.1,"00240TBX";red 723,X
```

Example 6. 69325A/69328A Sample Test Program

GPIO Interface

```
0: ent "ENTER RESISTANCE",A
1: A/5→A;dtoA→A;if A>7777;gto "error"
2: moct;wtb 9,170160,20000+A,170040
3: dsp "TEST RESISTANCE";gto "end"
4: "error":dsp "RESISTANCE TOO LARGE"
5: "end":mdec;end
```

HP-IB Interface

```
0: rds(723)→C
1: ent "ENTER RESISTANCE",A
2: A/5→A;dtoA→A;if A>7777;gto "error"
3: fmt 2,c,f4.0,c,z
4: wrt 723.2,"00160TB",A,"T00040T"
5: rds(723)→C
6: dsp "TEST RESISTANCE";gto "end"
7: "error":dsp "RESISTANCE TOO LARGE"
8: "end":end
```

5-31 69325A/69328A Test Procedure. Perform the following steps when using the 69325A/69328A sample test program.

1. Load the program into the calculator.
2. Depress RUN on the calculator. The calculator will pause with "ENTER RESISTANCE" displayed. The user must enter the desired resistance value then depress CONTINUE.
3. After running the program, the calculator will display "TEST RESISTANCE". At this point, the user should use an ohmmeter to measure the resistance output between terminals A and B on the card.

5-32 69326A/69327A Sample Test Program. Example 7 is a sample program which enables the user to verify operation of 69326A and 69327A cards. As shown, the program allows the user to program separate resistance values from channels 1 and 2 of the 69326A card. If an attempt is made to program a resistance exceeding the capability of the card the calculator will display "R1 TOO LARGE", or "R2 TOO LARGE", depending on the channel. In this case, the card will not be programmed.

5-33 To program a 69327A card, variables "A" and "B" should be divided by the resolution (80) of the 69327A card as shown in Table 5-1, instead of 160 which is the resolution of the 69326A card.

5-34 69326A/69327A Test Procedure. Perform the following steps when using the 69326A/69327A sample test program.

1. Load the program into the calculator.
2. Depress RUN on the calculator. The calculator will pause with "ENTER R1" displayed. The user must enter a resistance value for channel 1 then depress CONTINUE.
3. The calculator will pause with "ENTER R2" displayed. The user must enter a resistance value for channel 2 then depress CONTINUE.
4. After running the program, the calculator will display "TEST RESISTANCE". At this point, the user should use an ohmmeter to measure the resistance output as follows:
   a. The resistance output of channel 1 may be measured between card terminals C and B.
   b. The resistance output of channel 2 may be measured between card terminals A and B.

Example 7. 69326A/69327A Sample Test Program

GPIO Interface

```
0: ent "ENTER R1",A
1: A/160→A;dtoA→A;if A>77;gto "error1"
2: ent "ENTER R2",B
3: B/160→B;dtoB→B;if B>77;gto "error2"
4: B*100→B;A+B→D
5: moct;wtb 9,170160,20000+D,170040
6: dsp "TEST RESISTANCE";gto "end"
7: "error1":dsp "R1 TOO LARGE";gto "end"
8: "error2":dsp "R2 TOO LARGE"
9: "end":mdec;end
```

HP-IB Interface

```
0: rds(723)→C
1: ent "ENTER R1",A
2: A/160→A;dtoA→A;if A>77;gto "error1"
3: ent "ENTER R2",B
4: B/160→B;dtoB→B;if B>77;gto "error2"
5: B*100→B;A+B→D
6: fmt 2,c,f4.0,c,z
7: wrt 723.2,"00160TB",D,"T00040T"
8: rds(723)→C
9: dsp "TEST RESISTANCE";gto "end"
10: "error1":dsp "R1 TOO LARGE";gto "end"
11: "error2":dsp "R2 TOO LARGE"
12: "end":end
```

5-35 Relay Output Card, 69330A

5-36 This card provides 12, independent, SPST normally open (Form A) relays. Output data will be in the form of contact closures. Two additional relays are provided for external circuit gate/flag operation. An output connector is provided to route the 12 data outputs and the gate/flag signals to the external devices.

5-37 The following discussion will familiarize the user with programming the relay output card, 69330A. All examples will assume the relay output card is installed in unit 0, slot 402 (B).

5-38 **Operating Modes.** Basically there are two modes of operation which apply to this card:

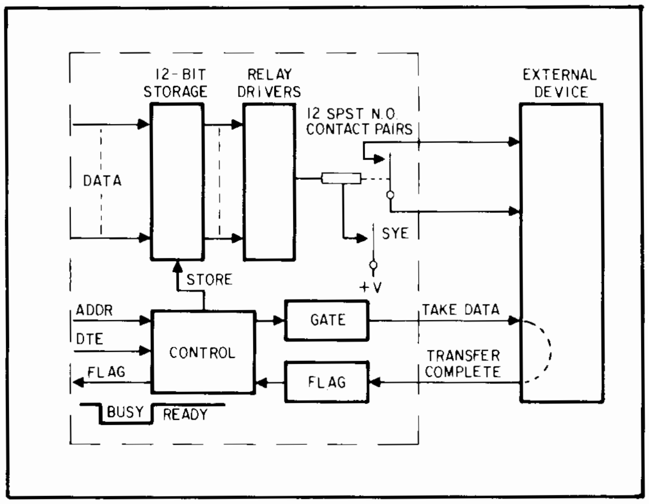
1. Automatic Handshake Mode (TME off): Used to output data to an external device when it is not necessary to handshake with the device.

2. Timing Mode (TME on): Used to output data to an external device when the user's program requires an indication from the device that data has been accepted. The trailing edge of the multiprogrammer flag will be returned upon completion of the data transfer and can be used to generate an interrupt on the GPIO or HP-IB interface, thereby notifying the user's program that the data transfer has been completed. (In the case of the HP-IB, the service request line will be set upon completion of the data transfer).

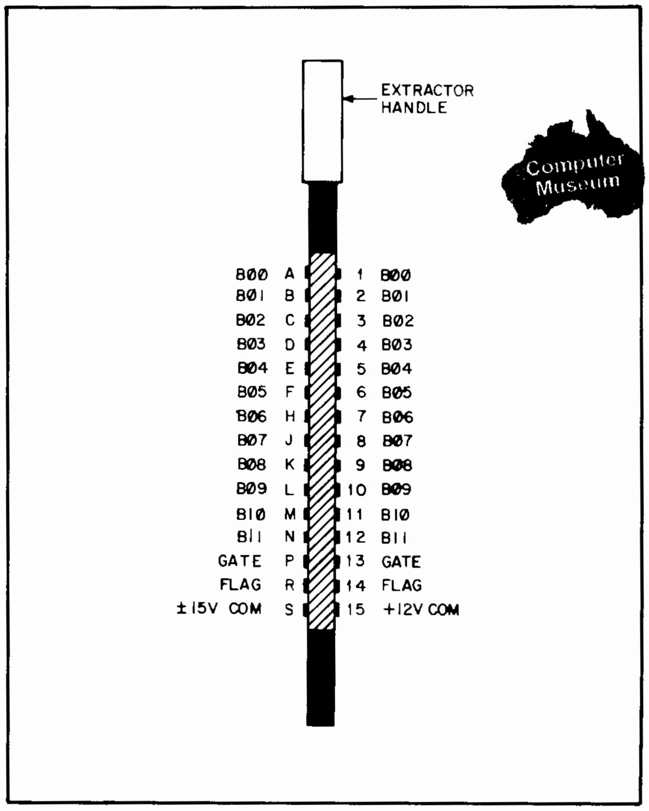
NOTE

When using this card without an external gate/flag timing circuit, the card's gate output must be jumpered to its flag input.

5-39 *Automatic Handshake Mode.* This mode requires sending a control word containing SYE and DTE only, followed by a data word containing the card slot address, and the output data desired. Example 8 illustrates using a data constant of 7777_8 to program the output of a 69330A card, located in slot 402 (B).



69330A Block Diagram



69330A Output Connector

Example 8. Programming a Relay Output Card in Automatic Handshake Mode:

GPIO Interface

```
0: moct;wtb 9,170140,27777
```

HP-IB Interface

```
0: fmt 1,c,z
1: wrt 723.1,"00140TB7777T"
```

5-40 Timing Mode. Example 9 illustrates programming a 69330A card in the timing mode. First, an interrupt linkage is established between the appropriate interface and a service routine "done". Then a control word with DTE, SYE, and TME is sent to the multiprogrammer, followed by a data word which initiates a timing sequence with an external device. The appropriate interface is then enabled to receive interrupts. The main program continues execution from this point until the multiprogrammer completes its data transfer, returning the trailing edge of CTF flag and thereby generating an interrupt to the calculator.

NOTE

DTE and TME must be programmed on together because the 69330A card cannot drive the CTF line unless DTE is on

5-41 When an interrupt occurs, program control passes to service routine "done". Service routine "done" sends a control word to the multiprogrammer, turning off TME, prints "done" to notify the user that the data transfer is complete, then returns program control to the main program. Of course, the actual action to be taken in response to an interrupt will be decided by the user when writing a service routine.

5-42 For a detailed description of timing mode operation as it relates to output cards, read chapter 3 (GPIO) or chapter 4 (HP-IB), paragraphs 3-59 through 3-70 or 4-59 through 4-72, respectively.

5-43 Using DTE to Simultaneously Gate Data Into External Devices. Some applications may require data to be loaded into several 69330A cards, then simultaneously gated into external devices. In this case, a control word may be sent with DTE off, followed by appropriate address words, then another control word with DTE on. As each data word is received, the data will be immediately transferred to the output of the addressed card. However, the gate from the output card will not be enabled until a control word containing DTE is received (see chapter 3 (GPIO) or 4 (HP-IB), paragraphs 3-53 and 4-53, respectively).

5-44 **69330A Sample Test Program.** Example 10 is a sample test program which illustrates programming with variable data values. This example allows the user to verify the operation of 69330A relay output cards by specifying an output pin no. and programming a contact closure with the associated relay. If the user enters a pin number greater than 12, the card will not be programmed.

5-45 **Test Procedure.** Perform the following steps when using the sample test program.

1. Load the program into the calculator.

2. On the output connector of the 69330A card, temporarily install a jumper between pins 13 and 14, and a second jumper between pins P and 15.

3. Depress RUN on the calculator. The calculator will stop with "ENTER PIN NO." displayed. The user must enter a pin number from 1 to 12, then depress CONTINUE.

4. The calculator will stop with "TEST BIT" displayed. At this point, the user should use an ohmmeter to check the contact resistance of the relays. Relay outputs can be measured between a numbered pin, 1 through 12, and a corresponding lettered pin (i.e., 1 and A, 2 and B, 12 and N). All relay contacts should be open except for the contacts corresponding to the pin number that was entered in Step 3.

5. After testing the bit, remove the jumper from pin 13 to 14. Depressing CONTINUE will cause the calculator to pause, waiting for the CTF flag. The red "RUN" light in the display will be lit.

6. Momentarily short pins 13 and 14 together. The CTF flag will be returned and the calculator will display "TEST COMPLETE".

Example 9. Programming a Relay Output Card in Timing Mode

GPIO Interface

```
 0: oni 9,"done"
 1: moct;wtb 9,170160,27777;mdec;eir 9
            •
            •
            •
20: "done":moct;wtb 9,170040;mdec
21: prt "done";iret
```

HP-IB Interface

```
 0: rds(723)→C
 1: oni 7,"done"
 2: fmt 1,c,z
 3: wrt 723.1,"00160T";rds(723)→C
 4: wrt 723.1,"B7777T";eir 7
             •
             •
             •
20: "done":if rds(723)#64;gto "iret"
21: wrt 723.1,"00040T";prt "done"
22: "iret":iret
```

Example 10. 69330A Sample Test Program

GPIO Interface

```
0: ent "ENTER PIN NO.",A
1: 2↑(A-1)→A;dtoA→A;if A>7777;gto "error"
2: moct;wtb 9,170140,20000+A
3: dsp "TEST BIT";stp
4: wtb 9,170160,20000,170040
5: dsp "TEST COMPLETE";gto "end"
6: "error":dsp "PIN NO. GREATER THAN 12"
7: "end":mdec;end
```

HP-IB Interface

```
0: rds(723)→C
1: ent "ENTER PIN NO.",A
2: 2↑(A-1)→A;dtoA→A;if A>7777;gto "error"
3: fmt 2,c,f4.0,c,z
4: wrt 723.2,"00140TB",A,"T"
5: dsp "TEST BIT";stp
6: fmt 1,c,z
7: wrt 723.1,"00160TBT00040T"
8: dsp "TEST COMPLETE";gto "end"
9: "error":dsp "PIN NO. GREATER THAN 12"
10: "end":end
```

5-46 Digital Output Card, 69331A

5-47 This card provides 12 bits with TTL/DTL-compatible logic levels as its output, and uses the gate/flag timing method of digital data transfer with an external device. An output connector is provided to route the 12 output bits and gate/flag signals to the external device.

5-48 The following discussion will familiarize the user with programming the digital output card, 69331A. All examples will assume the output card is installed in unit Ø, slot 4Ø2 (B).

5-49 **Operating Modes.** Basically there are two modes of operation which apply to this card:

1. Automatic Handshake Mode (TME off): Used to output data to an external device when it is not necessary to handshake with the device.

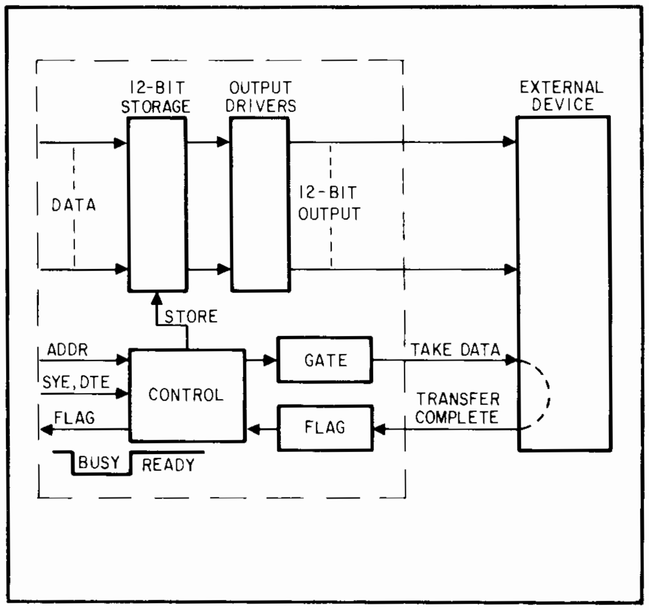
2. Timing Mode (TME on): Used to output data to an external device when the user's program requires an indication from the device that data has been accepted. The trailing edge of the multiprogrammer flag will be returned upon completion of the data transfer and can be used to generate an interrupt on the GPIO or HP-IB interface, thereby notifying the user's program that the data transfer has been completed. (In the case of the HP-IB, the service request line will be set upon completion of the data transfer).

NOTE

When using this card without an external gate/flag timing circuit, the card's gate output must be jumpered to its flag input.

5-50 *Automatic Handshake Mode.* This mode requires sending a control word containing SYE and DTE only, followed by a data word containing the card slot address, and the output data desired. Example 11 illustrates using a data constant of 7777_8 to program the output of a 69331A card located in slot 4Ø2 (B).

5-51 *Timing Mode.* Example 12 illustrates programming a 69331A card in the timing mode. First, an interrupt linkage is established between the appropriate interface and service routine "done". Then a control word with DTE, SYE, and TME is sent to the multiprogrammer, followed by a data word which initiates a timing sequence with an external device. The appropriate interface is then enabled to receive interrupts. The main program continues execution from this point until the multiprogrammer completes its data transfer, returning the trailing edge of CTF flag and thereby generating an interrupt to the calculator.



69331A Block Diagram

Example 11. Programming a Digital Output Card in Automatic Handshake Mode

GPIO Interface

```
0: moct;wtb 9,170140,27777
```

HP-IB Interface

```
0: fmt 1,c,z
1: wrt 723.1,"00140TB7777T"
```

Example 12. Programming a Digital Output Card in Timing Mode

GPIO Interface

```
 0: oni 9,"done"
 1: moct;wtb 9,170160,27777;mdec;eir 9
                 •
                 •
                 •
20: "done":moct;wtb 9,170040;mdec
21: prt "done";iret
```

HP-IB Interface

```
 0: rds(723)→C
 1: oni 7,"done"
 2: fmt 1,c,z
 3: wrt 723.1,"00160T";rds(723)→C
 4: wrt 723.1,"B7777T";eir 7
                •
                •
                •
20: "done":if rds(723)#64;gto "iret"
21: wrt 723.1,"00040T";prt "done"
22: "iret":iret
```

NOTE

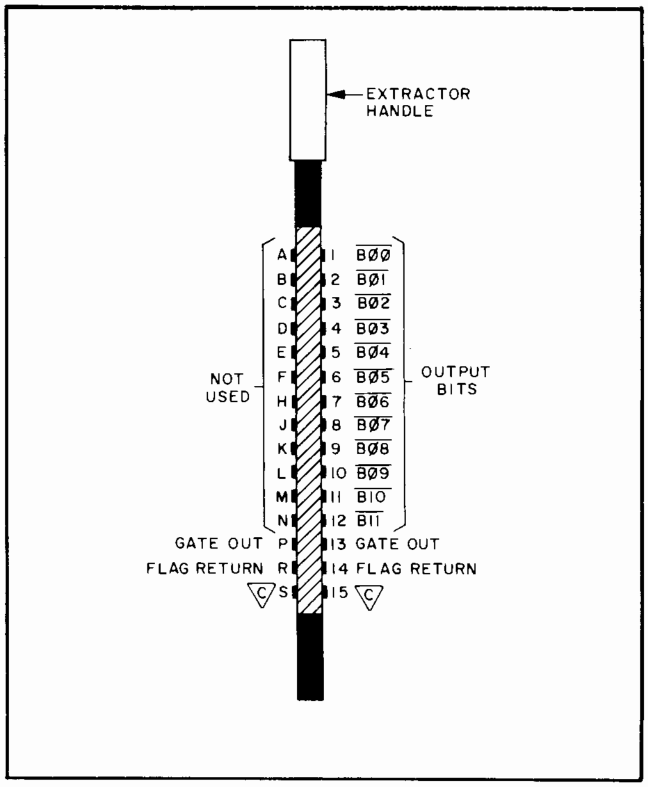
DTE and TME must be programmed on together because the 69331A card cannot drive the CTF line unless DTE is on.

5-52 When an interrupt occurs, program control passes to service routine "done". Service routine "done" sends a control word to the multiprogrammer, turning off TME, prints "done" to notify the user that the data transfer is complete, then returns program control to the main program. Of course,the actual action to be taken in response to an interrupt will be decided by the user when writing a service routine.

5-53 For a detailed description of timing mode operation as it relates to output cards, read chapter 3 (GPIO) or chapter 4 (HP-IB), paragraphs 3-59 through 3-70 or 4-59 through 4-72, respectively.

5-54 Using DTE to Simultaneously Gate Data Into External Devices. Some card applications may require data to be loaded into several 69331A cards, then simultaneously gated into external devices. In this case a control word may be sent with DTE off, followed by appropriate address words, then another control word with DTE on. As each data word is received, the data will be immediately transferred to the output of the addressed card. However, the gate from the output card will not be enabled until a control word containing DTE is received (see chapter 3 (GPIO) or 4 (HP-IB), paragraphs 3-53 and 4-53, respectively).

5-55 69331A Sample Test Program. Example 13 is a sample test program which illustrates programming with variable data values. This example allows the user to verify the operation of 69331A digital output cards by specifying an output pin no. and programming the associated bit output (high or low, depending on the option). If the user enters a pin number greater than 12 the card will not be programmed.



EXTRACTOR HANDLE
A B C D E F H J K L M N
1 2 3 4 5 6 7 8 9 10 11 12
$\overline{B00}$ $\overline{B01}$ $\overline{B02}$ $\overline{B03}$ $\overline{B04}$ $\overline{B05}$ $\overline{B06}$ $\overline{B07}$ $\overline{B08}$ $\overline{B09}$ $\overline{B10}$ $\overline{B11}$
NOT USED
OUTPUT BITS
GATE OUT P
FLAG RETURN R
S
13 GATE OUT
14 FLAG RETURN
15
C

69331A Output Connector

5-56 Test Procedure. Perform the following steps when using the sample test program.

1. Load the program into the calculator.

2. On the output connector of the 69331A Card, temporarily install a jumper between pins 13 and 14.

3. Depress RUN on the calculator. The calculator will stop with "ENTER PIN NO." displayed. The user must enter a pin number from 1 to 12, then depress CONTINUE.

4. The calculator will stop with "TEST BIT" displayed. At this point, the user should use a voltmeter to measure the output voltage on pins 1 through 12. (All voltage readings are referenced to pin 15 (common).
   a. On the standard digital output card, all pins shall read a positive voltage (+5 or +12), except for the pin that the user has previously entered into the program, which shall read between 0 and .3 volts.
   b. On option 073 output cards, all pins shall read between 0 and .3 volts, except for the pin that the user has previously entered into the program, which shall read a positive voltage (+5 or +12).

5. After testing the bit, remove the jumper from pin 13 to 14. Depressing CONTINUE will cause the calculator to pause, waiting for the CTF flag. The red "RUN" light in the display will be lit.

6. Momentarily short pins 13 and 14 together. The CTF flag will be returned and the calculator will display "TEST COMPLETE".

Example 13. 69331A Sample Test Program

<u>GPIO Interface</u>

```
0: ent "ENTER PIN NO.";A
1: 2↑(A-1)→A;dtoA→A;if A>7777;gto "error"
2: moct;wtb 9,170140,20000+A
3: dsp "TEST BIT";stp
4: wtb 9,170160,20000,170040
5: dsp "TEST COMPLETE";gto "end"
6: "error":dsp "PIN NO. GREATER THAN 12"
7: "end":mdec;end
```

<u>HP-IB Interface</u>

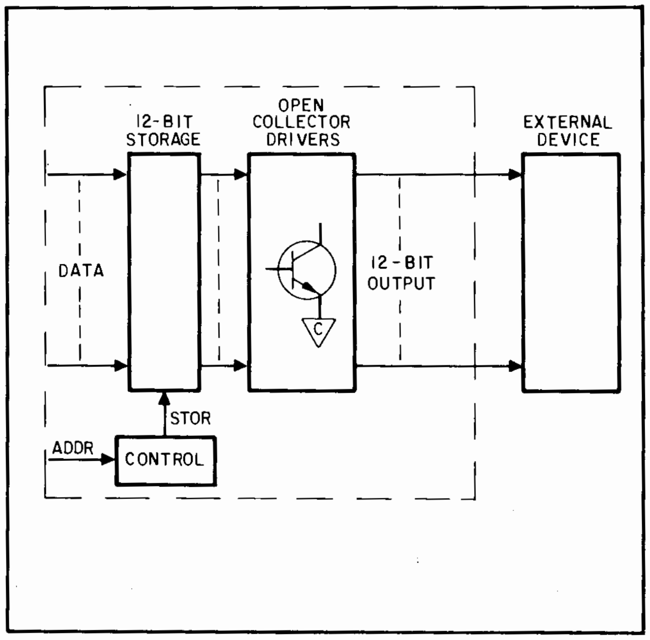
```
0: rds(723)→C
1: ent "ENTER PIN NO.";A
2: 2↑(A-1)→A;dtoA→A;if A>7777;gto "error"
3: fmt 2,c,f4.0,c,z
4: wrt 723.2,"00140TB",A,"T"
5: dsp "TEST BIT";stp
6: fmt 1,c,z
7: wrt 723.1,"00160TBT00040T"
8: dsp "TEST COMPLETE";gto "end"
9: "error":dsp "PIN NO. GREATER THAN 12"
10: "end":end
```

5-57 Open Collector Output Card, 69332A

5-58 This digital output card is designed to drive lamps and relay coils utilizing an external dc power source. Twelve separately programmable outputs are available, each of which is open collector. The following discussion will familiarize the user with programming the open collector output card, 69332A. All examples will assume the output card is installed in unit Ø slot 4Ø2 (B).

5-59 **Programming examples.** Examples 14 and 15 illustrate programming data from an open collector output card.

5-60 Example 14 uses a data constant of 7777 to program the output of a 69332A card located in slot 4Ø2 (B). A control word is sent first, containing SYE only, followed by a data word containing the card slot address and the output data. Although SYE has no effect on this card, it is included in the control word to avoid disabling the outputs of previously programmed output cards.



69332A Block Diagram

5-61 Sample Test Program. Example 15 is a sample test program which illustrates programming with variable data values. This example allows the user to verify the operation of 69332A open collector output cards by specifying an output pin number and toggling the associated bit output high and low. If the user enters a pin number greater than 12, the card will not be programmed.

5-62 Since the data outputs from this card are open collector, it will be necessary for the user to provide a pull-up resistor for this test. Although the resistance value is not critical, the recommended resistance value is between 1K and 10K ohms.

5-63 Test Procedure. Perform the following steps when using the 69332A sample test program.

1. Load the program into the calculator.

2. On the output connector of the open collector output card, temporarily install a pull-up resistor between pin 14 (+5 volts) and any data output pin, 1 through 12, as desired.

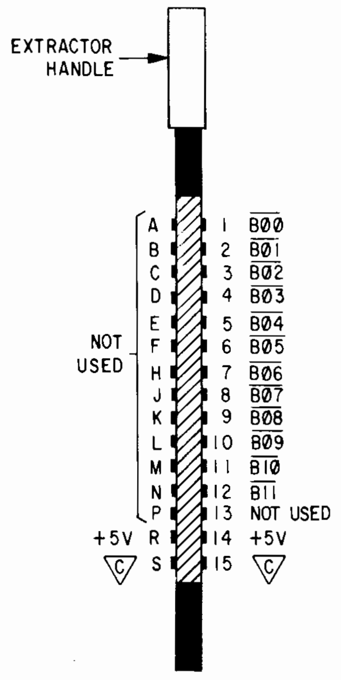
3. Depress RUN on the calculator. The calculator will stop with "ENTER PIN NO." displayed. The user must enter the pin number that had the pull-up resistor installed in step 2, then depress CONTINUE.

4. The calculator will stop with "TEST BIT" displayed. At this point, the user should use a voltmeter to measure the output voltage between the pin selected in step 2 and pin 15 (common).

   a. On the standard open collector output card, the voltmeter will indicate between 0 and .7 volts.

   b. On option 090 output cards, the voltmeter will indicate +5 volts.

5. Depress CONTINUE on the calculator. The calculator will display "TEST COMPLETE", and the voltages measured in step 4 will be reversed. (i.e., a standard card will now indicate +5 volts, while an option 090 card will indicate between 0 and .7 volts on all pins).



69332A Output Connector

Example 14. Programming an Open Collector Digital Output Card

GPIO Interface

```
0: moct;wtb 9,170040,27777
```

HP-IB Interface

```
0: fmt 1,c,z
1: wrt 723.1,"00040TB7777T"
```

Example 15. 69332A Sample Test Program

GPIO Interface

```
0: ent "ENTER PIN NO.",A
1: 2↑(A-1)→A;dtoA→A;if A>7777;gto "error"
2: moct;wtb 9,170040,20000+A
3: dsp "TEST BIT";stp
4: wtb 9,20000
5: dsp "TEST COMPLETE";gto "end"
6: "error":dsp "PIN NO. GREATER THAN 12"
7: "end":mdec;end
```

HP-IB Interface

```
0: rds(723)→C
1: ent "ENTER PIN NO.",A
2: 2↑(A-1)→A;dtoA→A;if A>7777;gto "error"
3: fmt 2,c,f4.0,c,z
4: wrt 723.2,"00040TB",A,"T"
5: dsp "TEST BIT";stp
6: fmt 1,c,z
7: wrt 723.1,"BT"
8: dsp "TEST COMPLETE";gto "end"
9: "error":dsp "PIN NO. GREATER THAN 12"
10: "end":end
```

5-64 Stepping Motor Control Card, 69335A

5-65 This card can be programmed to generate from 1 to 2047 squarewave pulses at either of two output terminals on the card. When applied to a stepping motor translator, these pulses are converted to clockwise and counterclockwise drive pulses for an associated stepping motor. The squarewave outputs of the 69335A can also be used for pulse-train update of supervisory control stations. The following discussion will familiarize the user with programming the stepping motor control card, 69335A. All examples will assume the stepping motor card is installed in unit Ø, slot 4Ø2, (B).

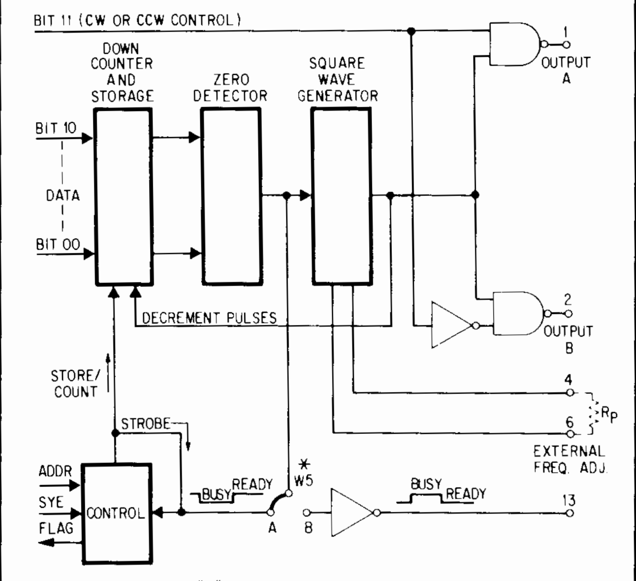
5-66 Selecting Output Terminal and Number of Output Pulses. Data transmitted to this card is a combination of the number of output pulses desired and a user selected output terminal. For values up to octal 3777 (decimal 2047) the output will be from pin 1 of the output connector. If it is desired to obtain the output pulse train from pin 2, an octal 4000 must be added to the number of pulses desired (in octal). For example, 7777_8 would program 2047 pulses from pin 2.

5-67 Operating Modes. Basically there are two modes of operation which apply to this card.

1. Automatic Handshake Mode (TME off): Allows the user to program an output pulse train, consisting of a specified number of pulses, from either of two output circuits. Programmable selection of the appropriate output circuit allows the user to determine the direction of rotation of a stepping motor. Since this mode does not control the trailing edge of the multiprogrammer flag, it does not provide an indication to the user's program that the output pulse train has been completed.

2. Timing Mode (TME on): This mode is used when the user's program requires an indication from the multiprogrammer that the output pulse train has been completed. The trailing edge of the multiprogrammer flag will be returned upon completion of the output pulse train and can be used to generate an interrupt on the GPIO or HP-IB interface, thereby notifying the user's program that the pulse train has been completed. (In the case of the HP-IB, the service request line will be set upon completion of the data transfer).

5-68 Automatic Handshake Mode. This mode requires sending a control word with SYE only, followed by a data word containing the card slot address and output data



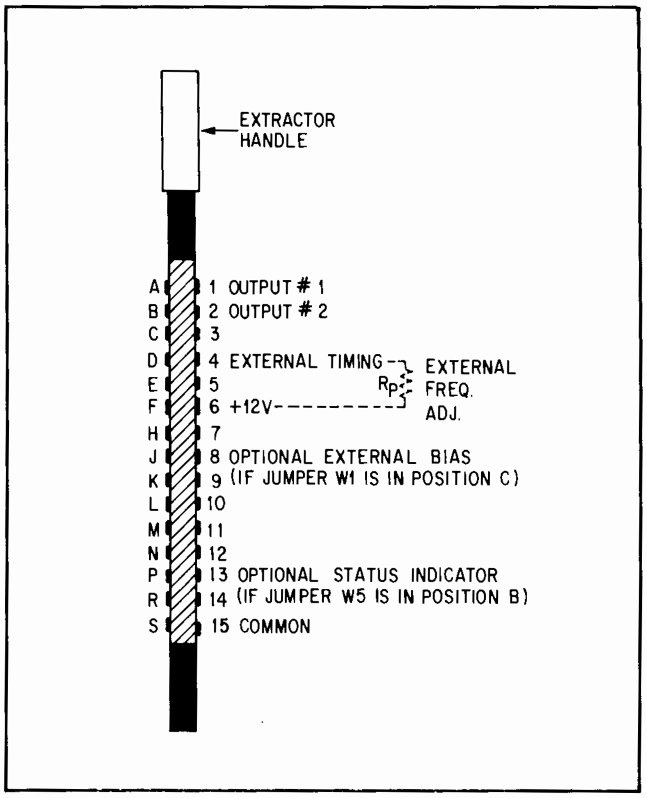
* IN POSITION "A" OF W5, THE TIMING FLAG SIGNAL RETURNED TO THE MULTIPROGRAMMER STAYS "BUSY" FOR THE DURATION OF THE OUTPUT COUNT. IN POSITION "B" OF W5 THE TIMING FLAG IS AVAILABLE AT OUTPUT TERMINAL 13, AND A SHORT DURATION FLAG (≈ 2μSEC) IS RETURNED TO THE MULTIPROGRAMMER.

69335A Block Diagram

(number of pulses and output terminal) desired. Example 16 illustrates using a data constant of 3777_8 to program 2047 output pulses from pin 1 of the 69335A card located in slot 4Ø2 (B).

5-69 *Timing Mode.* Example 17 illustrates programming a 69335A card in the timing mode. First, an interrupt linkage is established between the appropriate interface and service routine "done". Then a control word with DTE, SYE, and TME is sent to the multiprogrammer, followed by a data word specifying 2047 pulses from pin 2 of the 69335A card (7777_8). The appropriate interface is then enabled to receive interrupts. The main program continues execution from this point until the pulse train has been completed. At that time, a trailing edge of multiprogrammer flag will be returned, thereby generating an interrupt to the calculator.

5-70 When an interrupt occurs, program control passes to service routine "done". Service routine "done" sends a control word to the multiprogrammer, turning off TME; prints "done" to notify the user that the pulse train has been completed, then returns program control to the main program. Of course, the actual action to be taken in response to an interrupt will be decided by the user when writing his service routine.



69335A Output Connector

Example 16. Programming an output pulse train in Automatic Handshake Mode

GPIO Interface

```
0: moct;wtb 9,170040,23777
```

HP-IB Interface

```
0: fmt 1,c,z
1: wrt 723.1,"00040TB3777T"
```

Example 17. Programming an output pulse in Timing Mode

GPIO Interface

```
 0: oni 9,"done"
 1: moct;wtb 9,170160,27777;mdec;eir 9
              •
              •
              •
20: "done":moct;wtb 9,170040;mdec
21: prt "done";iret
```

HP-IB Interface

```
 0: rds(723)→C
 1: oni 7,"done"
 2: fmt 1,c,z
 3: wrt 723.1,"00160T";rds(723)→C
 4: wrt 723.1,"B7777T";eir 7
              •
              •
              •
20: "done":if rds(723)#64;gto "iret"
21: wrt 723.1,"00040T";prt "done"
22: "iret":iret
```

5-71 For a detailed description of timing mode operation as it relates to output cards, read chapter 3 (GPIO) or chapter 4 (HP-IB), paragraphs 3-59 through 3-70 or 4-59 through 4-72, respectively.

5-72 69335A Sample Test Program. Example 18 is a sample test program which illustrates programming with variable data values. This example allows the user to verify the operation of 69335A Stepping Motor cards by programming a pulse train between 1 and 2047 pulses long from either output pin. If the user attempts to program a pulse train greater than 2047 pulses the calculator will display "TOO MANY PULSES ENTERED", and will not program the card.

5-73 For this example it will be assumed that an output pulse train from pin 1 will cause clockwise rotation of a stepping motor, while an output pulse train from pin 2 will cause counterclockwise rotation of a stepping motor. It will be up to the user to provide a means of monitoring the pulse train.

5-74 The for/next loop of example 18 merely simulates a mainline program which will run for approximately 30 seconds. This allows sufficient time for the stepping motor card to generate an interrupt with the longest programmable pulse train.

5-75 Notice that the three (GPIO) or four (HP-IB) program lines in example 18 beginning with the line labeled "stepr" could be incorporated as a subroutine in a user's program to program direction and magnitude of stepping motor rotation. When used this way, variable "A" must be supplied to the subroutine as a positive value for clockwise rotation, or a negative value for counterclockwise rotation. Of course a return (ret) statement must be included as the last statement in the subroutine and the user has the option of changing the card slot address as well as using or not using TME and the calculator interrupt system.

5-76 Test Procedure. Perform the following steps when using the 69335A test program.

1. Load the program into the calculator.

2. Depress RUN on the calculator. The calculator will stop with "ENTER NO. OF PULSES" displayed. The user must enter the desired number of pulses (−2047 to +2047), then depress CONTINUE.

3. The stepping motor card will transmit a pulse train approximately 12 volts in amplitude, which may be measured as follows.

   a. If a positive value had been entered in step 2, the output pulse train may be measured between pins 1 and 15 (common).

   b. If a negative value had been entered in step 2, the output pulse train may be measured between pins 2 and 15.

4. Upon completion of the pulse train the calculator will print "done". As shipped from the factory the output pulse rate is 100 HZ. Therefore the time interval between depressing CONTINUE in step 2 and printing "done" will be approximately 1 second for every 100 pulses entered in step 2.

Example 18. 69335A Sample Test Program

GPIO Interface

```
0: oni 9,"done"
1: ent "ENTER NO. OF PULSES",A
2: if A<-2047 or A>2047;gto "error"
3: "stepr":if A<0;2048+abs(A)→A
4: dtoA→A
5: moct;wtb 9,170160,20000+A;mdec;eir 9
6: for J=1 to 50000
7: next J
8: gto "end"
9: "error":dsp "TOO MANY PULSES ENTERED"
10: "end":end
11: "done":moct;wtb 9,170040
12: prt "done";mdec;50000→J;iret
```

HP-IB Interface

```
0: rds(723)→C
1: oni 7,"done"
2: ent "ENTER NO. OF PULSES",A
3: if A<-2047 or A>2047;gto "error"
4: fmt 1,c,z;fmt 2,c,f4.0,c,z
5: "stepr":if A<0;2048+abs(A)→A
6: dtoA→A
7: wrt 723.1,"00160T";rds(723)→C
8: wrt 723.2,"B",A,"T";eir 7
9: for J=1 to 50000
10: next J
11: gto "end"
12: "error":dsp "TOO MANY PULSES ENTERED"
13: "end":end
14: "done":if rds(723)#64;gto "iret"
15: wrt 723.1,"00040T";prt "done"
16: "iret":50000→J;iret
```

5-77 D/A Current Converter Card, 69370A

5-78 This card provides a high speed, constant current output that is the analog equivalent of the digital input data. The output current range is from 0 to 20.475mA with a minimum programmable step change of 5μA. Current output levels are programmed in straight 12-bit binary form.

5-79 The following discussion will familiarize the user with programming the D/A current converter card, 69370A. Unless otherwise specified, all examples will assume the D/A current card is installed in unit Ø, slot 4Ø2 (B), and a voltage regulator card 69351B (which supplies the required ±15V bias voltages) is installed in unit Ø, slot 6ØØ.

5-80 **Calculating Data Value.** The following steps will yield the octal data value required to program the 69370A to the desired output current.

1. Divide the desired output current in milliamps by .005 (the LSB).

2. Calculate the octal equivalent of the value yielded in step 1 (see Paragraph A-21 in Appendix A).

For example, to calculate the octal data value required for 5 milliamps;

$5 \div .005 = 1000$
octal equivalent of 1000 = 1750

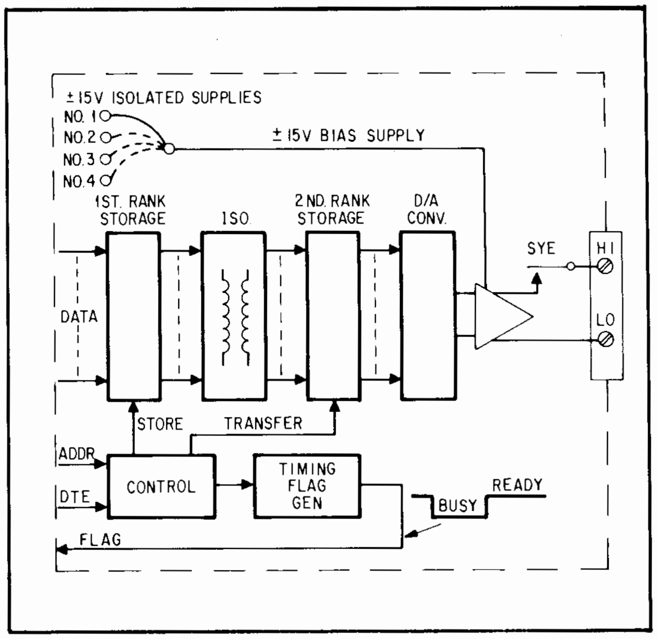
or, to calculate the octal data value required for 15 milliamps;

$15 \div .005 = 3000$
octal equivalent of 3000 = 5670

5-81 **Programming an Output Current.** Example 19 illustrates using a data constant of 1750_8 to program the output current of a 69370A card to 5mA. A control word is sent containing SYE and DTE, followed by a data word containing the card slot address and the data constant.

5-82 Although a 69370A card returns a CTF flag as a function of internal timing circuits, TME is not included in the control words of the programming examples. This is because the period of the internal timing circuits on this card is shorter than the period of the multiprogrammer handshake flag (without TME). Therefore, TME is not useful when programming this card.

5-83 **Changing Output Currents of Multiple 69370A Cards Simultaneously.** The dual rank storage feature of 69370A cards and the DTE control mode allow the output currents of any number of 69370A cards to be changed simultaneously. A detailed description of this technique is given in chapters III (GPIO) and IV (HP-IB), paragraphs 3-56 and 4-56, respectively.



69370A Block Diagram

5-84 In example 19, DTE and SYE were programmed on in a control word prior to sending a data word to the card. In this case when the data word is received, it is loaded into the addressed card and immediately converted to an analog current that is present at the output of the card.

5-85 When it is desired to change the outputs of multiple 69370A cards simultaneously, a control word is sent with DTE off, followed by the appropriate data words, then another control word with DTE on. As each data word is received, the data will be loaded into the first rank storage of the addressed card. When a subsequent control word with DTE is received, all previously addressed D/A cards will simultaneously transfer the contents of first rank storage into second rank storage and the output D/A conversion circuits.

5-86 Example 20 illustrates using DTE with two 69370A cards. In this case, the first rank storage of the cards located in multiprogrammer slots 4Ø2 (B) and 4Ø3 (C) is loaded with new data. After both cards have received new data, DTE is turned on by a second control word, resulting in both cards transferring data from first rank to second rank storage and converting the data to analog outputs.

5-87 **Sample Test Program.** Example 21 is a sample test program which illustrates the use of variable data values and allows the user to program a 69370A card to any value within its capability. If an output current is requested which exceeds the capability of the card, the calculator will display "CURRENT OUT OF RANGE" and will not program the card.

5-88 The user must provide a load resistor from 100 to 500 ohms in order to monitor the output current. The accuracy of this resistor must be taken into account when determining the actual output current. For an absolute accuracy check, refer to the 69370A manual.

5-89 Notice that the three program lines beginning with the line labeled "Iout" can be used as a current programming subroutine merely by changing the second line following "Iout" from (gto "end") to (ret). When used this way, the main program would supply the desired current (in milliamps) as variable "A" whenever the subroutine was called.

5-90 The decimal mode (mdec), used in the GPIO program, is optional, depending on whether the user wants to return the calculator to decimal mode after programming the multiprogrammer.

5-91 Test Procedure. Perform the following steps when using the 69370A sample test program.

1. Connect a load resistor between terminals A and B of the output terminal block on the 69370A card. Connect a voltmeter across the load resistor with the high input of the voltmeter at the end of the resistor connected to terminal A.

2. Load the program into the calculator, then depress RUN. The calculator will pause with "ENTER CURRENT" displayed. Entering the desired current in milliamps, then depressing CONTINUE will program the current D/A card to the desired output.

3. The output current may then be determined by dividing the voltage measured across the load resistor by the value of the load resistor. Notice that, unless the resistor accuracy and the D/A card accuracy are included in the calculations, this should only be considered an approximate value. It will, however, be close enough to the programmed current to ascertain the card is working properly.

Example 19. Programming an Output Current

GPIO Interface

```
0: moct;wtb 9,170140,21750
```

HP-IB Interface

```
0: fmt 1,c,z
1: wrt 723.1,"00140TB1750T"
```

Example 20. Simultaneously Changing Output Currents of Multiple 69370A Cards

GPIO Interface

```
0: moct;wtb 9,170040,21750,35670,170140
```

HP-IB Interface

```
0: fmt 1,c,z
1: wrt 723.1,"00040TB1750TC5670T00140T"
```

Example 21. 69370A Sample Test Program

GPIO Interface

```
0: ent "ENTER CURRENT",A
1: if A<0 or A>20.475;gto "error"
2: "Iout":A/.005→A;dtoA→A
3: moct;wtb 9,170140,20000+A;mdec
4: gto "end"
5: "error":dsp "CURRENT OUT OF RANGE"
6: "end":end
*31734
```

HP-IB Interface

```
0: ent "ENTER CURRENT",A
1: if A<0 or A>20.475;gto "error"
2: fmt 2,c,f4.0,c,z
3: "Iout":A/.005→A;dtoA→A
4: wrt 723.2,"00140TB",A,"T"
5: gto "end"
6: "error":dsp "CURRENT OUT OF RANGE"
7: "end":end
*18613
```

5-92 Breadboard Output Card, 69380A

5-93 The 69380A breadboard output card is a simple multiprogrammer interface card, which allows the user to design, build, and control special output circuits through the multiprogrammer system. The following discussion will familiarize the user with programming the breadboard output card, 69380A. All examples will assume the output card is installed in unit Ø, slot 4Ø2 (B).

5-94 Programming examples. Examples 22 and 23 illustrate programming data from a breadboard output card.

5-95 Example 22 uses a data constant of 7777 to program the output of a 69380A card located in slot 4Ø2 (B). A control word is sent first, containing SYE only, followed by a data word containing the card slot address and the output data. Although SYE has no effect on this card, it is included in the control word to avoid disabling the outputs of previously programmed output cards.

5-96 Sample Test Program. Example 23 is a sample test program which illustrates programming with variable data values. This example allows the user to verify the operation of 69380A breadboard output cards by specifying an output pin number and programming the associated bit output. If the user enters a pin number greater than 12, the card will not be programmed.

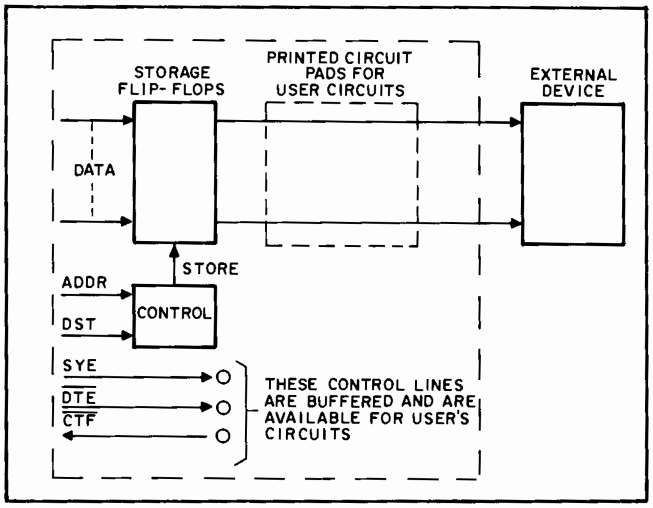
5-97 Test Procedure. Perform the following steps when using the 69380A sample test program.

1. Load the program into the calculator.

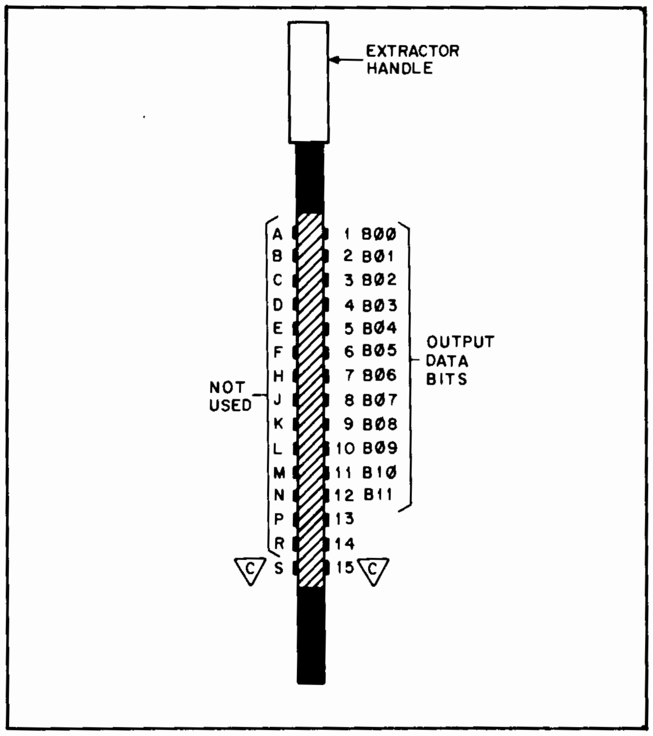
2. Depress RUN on the calculator. The calculator will pause with "ENTER PIN NO." displayed. The user must enter a pin number from 1 to 12, then depress CONTINUE.

3. The calculator will stop with "TEST BIT" displayed. At this point, the user should use a voltmeter to measure the output voltage on pins 1 through 12 (all voltage readings are referenced to pin 15 (common). All pins will read between 0 and .3 volts, except for the pin that the user has previously entered, which will read greater than 2.4 volts.

4. Repeating steps 2 and 3 will enable the user to check each data output line on the card.


STORAGE FLIP-FLOPS
PRINTED CIRCUIT PADS FOR USER CIRCUITS
EXTERNAL DEVICE
DATA
STORE
ADDR
CONTROL
DST
SYE
DTE
CTF
THESE CONTROL LINES ARE BUFFERED AND ARE AVAILABLE FOR USER'S CIRCUITS

69380A Block Diagram



69380A Output Connector

Example 22. Programming a Breadboard Output Card

GPIO Interface

```
0: moct;wtb 9,170040,27777
```

HP-IB Interface

```
0: fmt 1,c,z
1: wrt 723.1,"00040TB7777T"
```

Example 23. 69380A Sample Test Program

GPIO Interface

```
0: ent "ENTER PIN NO.",A
1: 2↑(A-1)→A;dtoA→A;if A>7777;gto "error"
2: moct;wtb 9,170040,20009→A
3: dsp "TEST BIT";gto "end"
4: "error":dsp "PIN NO. GREATER THAN 12"
5: "end":mdec;end
*27332
```

HP-IB Interface

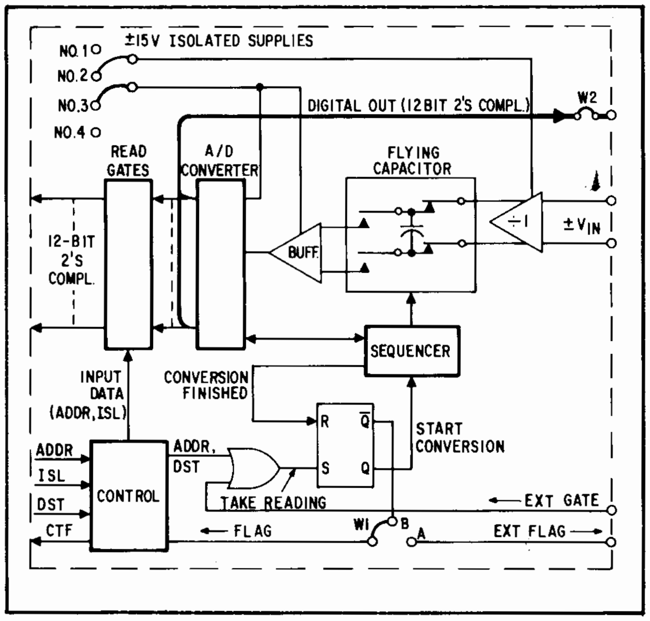
```
0: rds(723)→C
1: ent "ENTER PIN NO.",A
2: 2↑(A-1)→A;dtoA→A;if A>7777;gto "error"
3: fmt 2,c,f4.0,c,z
4: wrt 723.2,"00040TB",A,"T"
5: dsp "TEST BIT";gto "end"
6: "error":dsp "PIN NO. GREATER THAN 12"
7: "end":end
*25573
```

5-98 Voltage Monitor Card, 69421A

5-99 This input card monitors bipolar dc voltages in the range of +10.235 to −10.240 volts, and returns a 12-bit 2's complement digital word to indicate the magnitude and sign of the measured voltage. In addition, options are available which will allow voltage measurements from +1.0235 to −1.0240 and +102.35 to −102.40 volts (options 001 and 102, respectively). An explanation of the relationship between the applied voltages and corresponding return data words is given in Appendix A, Paragraph A-37. The following discussion will familiarize the user with programming the voltage monitor card 69421A. All examples will assume the voltage monitor card is installed in unit Ø, slot 4Ø2, (B), and a voltage regulator card, 69351B (which supplies the required ±15V bias voltages), is installed in unit Ø, slot 6ØØ.

5-100 69421A Sample Test Program. Example 24 is a sample test program that allows the user to measure a voltage and display the value on the calculator. If an option 001 or 102 is being used, .005 in line 3 should be changed to .0005 or .05, respectively.

5-101 The timing mode of operation used in example 24 is described in chapters 3 (GPIO) and 4 (HP-IB), paragraphs 3-84 and 4-88, respectively. Basically, a control word is transmitted specifying the input timing mode (ISL and TME on); then, the card is addressed with a gate. When the voltage monitor card receives a gate in the timing mode, its internal CTF circuits will delay the trailing edge of the multiprogrammer flag until the voltage monitor card converts the analog input voltage into a digital equivalent. Attempting another multiprogrammer operation before the trailing edge of flag is received causes the calculator to wait until the voltage monitor card completes its conversion (6 ms) and returns a CTF flag. When the trailing edge of flag is received by the calculator, the program will continue. The voltage monitor card is then addressed without a gate and the octal equivalent of the analog voltage is read into variable "X".



69421A Block Diagram

Example 24. 69421A Sample Test Program

GPIO Interface

```
0: "meas":moct;wtb 9,170260,20000,170240
1: wti 0,11;wti 4,20000;rdi 4→X;mdec
2: otdX→X;if X>2047;X-4096→X
3: .005X→X
4: fxd 3;dsp X,"VOLTS";end
*31361
```

HP-IB Interface

```
0: "meas":fmt 1,c,z;wrt 723.1,"0260TBTBX"
1: rds(723)→C;red 723,X
2: otdX→X;if X>2047;X-4096→X
3: .005X→X
4: fxd 3;dsp X,"VOLTS";end
*13122
```

5-102 In line number 2, the octal data returned from the voltage monitor card in variable "X" is converted to a decimal value by use of the octal to decimal function of the calculator (otd). If the decimal value exceeds 2047, it indicates a negative voltage and requires a 2's complement. This is done by subtracting 4096 from the decimal value. In either case, (positive or negative), the final decimal value is then converted to a voltage value in line number 3 by multiplying the decimal value stored in "X" by the LSB (in volts) of the voltage monitor card.

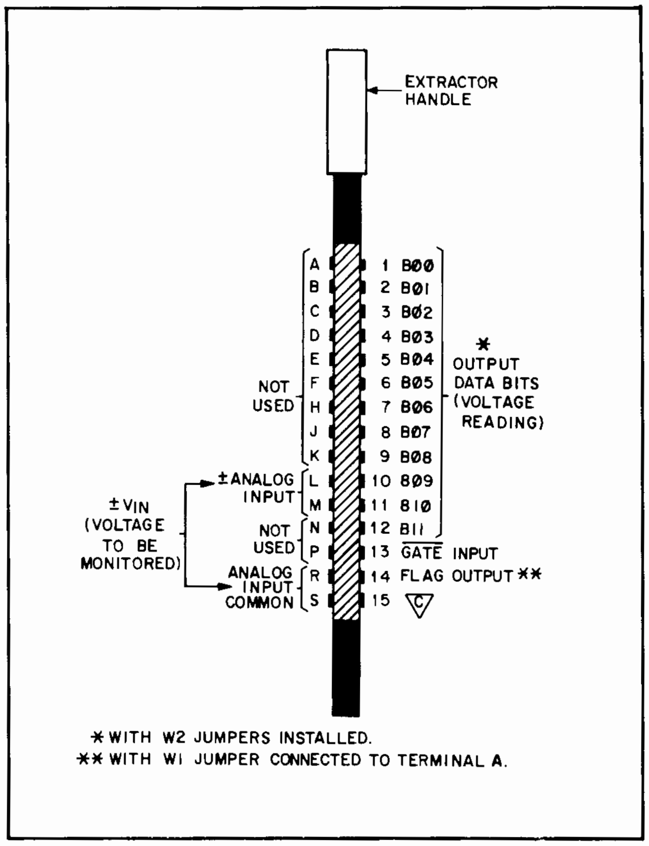
5-103 Notice that the first four lines of example 24 can be incorporated as a voltage measuring subroutine in a user's program merely by adding a return (ret) statement to the fourth line. When used this way, subroutine "meas" would return variable "X" containing the measured voltage in volts.

5-104 Test Procedure. Perform the following steps when using the 69421A sample test program.

1. Load the program into the calculator.

2. Apply the voltage to be measured between pins L and R (common) of the voltage monitor card.

3. Depress RUN on the calculator. The voltage will be measured and displayed.



69421A Connector

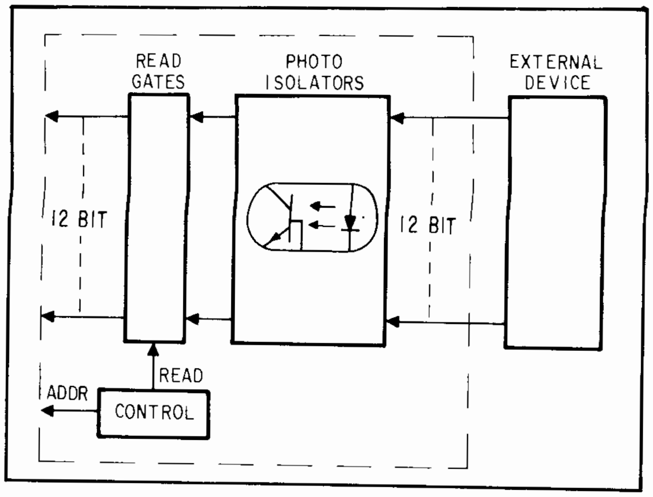
5-105 Isolated Digital Input Card, 69430A

5-106 This card allows the calculator to read 12-bits of logic level data that is isolated from ac earth ground, thereby eliminating ground loop problems in automatic test and control systems. The following discussion will familiarize the user with programming the isolated digital input card, 69430A. All examples will assume the input card is installed in unit Ø, slot 4Ø2, (B).

5-107 69430A Sample Test Program. Example 25 is a sample test program which will allow the user to verify the operation of 69430A cards by reading in and displaying data applied to the 69430A input connector. As written, the program can be used to verify the operation of 69430A cards equipped with options 069, 084, 085, and 086. To verify options 073, 087, 088, or 089, delete the third line (7777–X→X) of the sample test program.

5-108 In example 25, the octal data is read into variable "X" in line number 1. The first two lines of this example are all that is required to return data from this card. The user has the option of using the data in its octal form, or using the octal to decimal function (otd) of the calculator to obtain the equivalent decimal value. The data inversion in line number 2, as well as the (otd) function are only used in this example to satisfy the requirements of subroutine "BIT".

5-109 The isolated digital input card is always read without a gate, as there is no storage or CTF drive capability on this card. The only requirements for reading data from this card are: (1) a control word with ISL on must have been previously sent, and (2) the card must be addressed.



69430A Block Diagram

5-110 Test Procedure. Perform the following steps when using the 69430A test program.

1. Load the program into the calculator, being sure to include subroutine "BIT" from Appendix A.

2. From an external power supply, select a voltage level which corresponds to the high state of the option to be tested as specified in the Isolated Digital Input Card Manual. (e.g., 5 volts would suffice for Options 069 and 073).

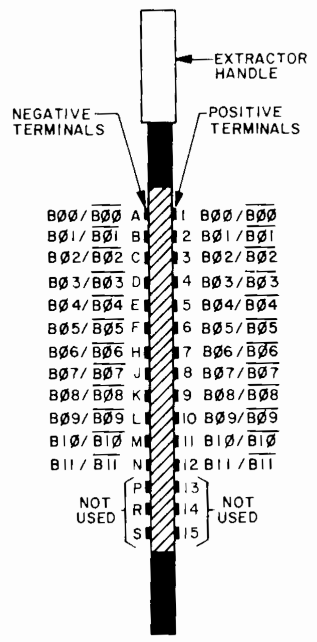
3. Apply an input voltage to the input connector of the isolated input board as follows:

   a. Apply the positive side of the input voltage to any pin, 1 through 12, selected by the user.

   b. Apply the negative side of the input voltage to a lettered pin which corresponds to the numbered pin previously selected. (e.g., pin A corresponds to pin 1 and pin N corresponds to pin 12).

4. Depress RUN on the calculator. The calculator will stop with the input bit and pin number displayed. If no data has been input, the calculator will stop with "NO DATA RECEIVED" displayed.

5. Repeating steps three and four will enable the user to check each data input line on the card.



69430A Input Connector

Example 25. 69430A Sample Test Program

GPIO Interface

```
0: moct;wtb 9,170240
1: wti 0,11;wti 4,20000;rdi 4→X;mdec
2: 7777-X→X
3: otdX→X;gsb "BIT"
4: end
*23576
```

HP-IB Interface

```
0: fmt 1,c,z
1: wrt 723.1,"00240TBX";red 723,X
2: 7777-X→X
3: otdX→X;gsb "BIT"
4: end
*21140
```

5-111 Digital Input Card, 69431A

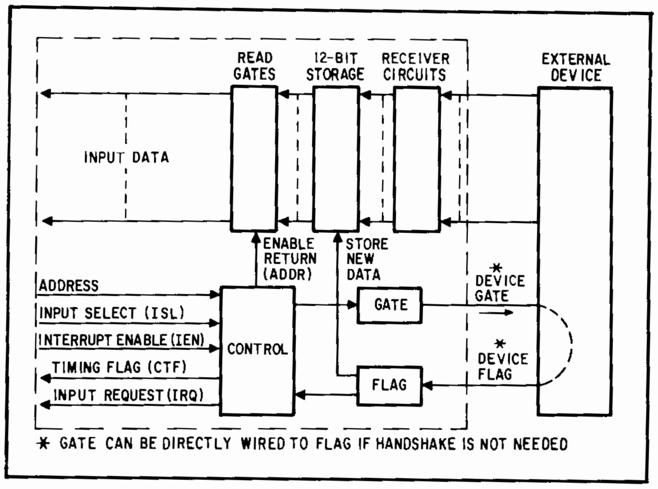
5-112 This card allows the calculator to read 12-bits of logic level or contact closure data that is referenced to the logic common (ac earth ground). Gate/flag circuits are provided on each 69431A to allow external input devices to use the interrupt system of the calculator. The following discussion will familiarize the user with programming the digital input card 69431A. All examples will assume the digital input card is installed in unit Ø, slot 4Ø2 (B).

NOTE

When using this card without an external gate/flag timing circuit, the card's gate output must be jumpered to its flag input.

5-113 Operating Modes. Basically there are three modes of operation which apply to this card:

1. Automatic Handshake Mode (TME off): Used to read data from an input card when it is not necessary to handshake with an external device.



69431A Block Diagram

2. Timing Mode (TME on): Also called the dedicated input mode, used to read data from an input card when it is necessary for the card to handshake with an external device.

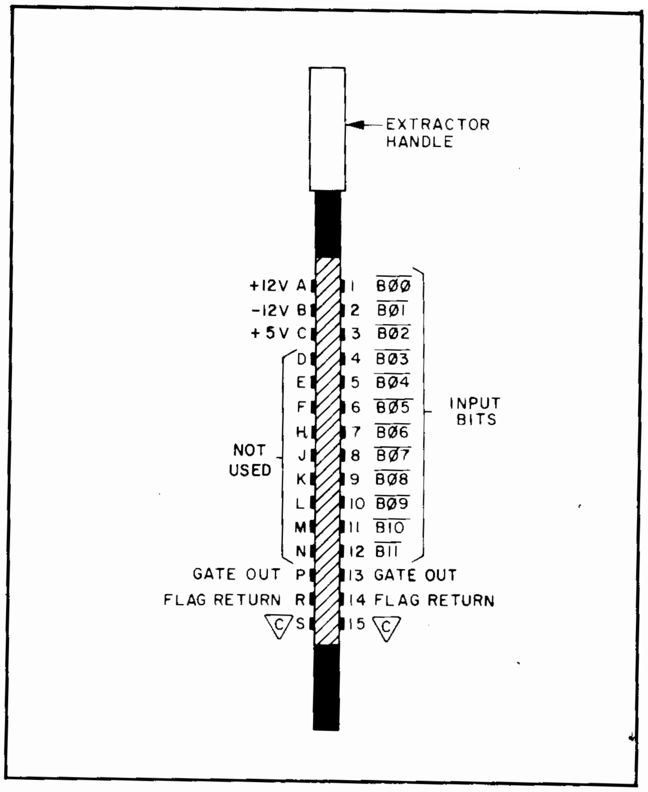
3. Interrupt Mode: Useful for activating many input cards simultaneously, then reading back data after a handshake is completed between an external device and one of the activated input cards.

5-114 Regardless of the mode used, the return data word is input in octal form to variable "X". The data can be used in octal form or can be converted to decimal by using the octal to decimal function (otd) of the calculator.

5-115 Automatic Handshake Mode. Example 26 illustrates reading back data in the automatic handshake mode (TME off). This mode is used when a handshake between the 69431A card and external device (data source) is not required. Pins 13 (gate) and 14 (flag) must be jumpered together.

5-116 Note in Example 26 that IRQ is subtracted from the return data word in the third program line so that the data can be utilized. Checking and subtracting IRQ is explained in chapter 3 (GPIO) and chapter 4 (HP-IB), paragraphs 3-78 and 4-80, respectively.

5-117 Timing Mode. Example 27 illustrates reading back data in the timing mode (TME and ISL on). Pin 13 of the



69431A Input Connector

69431A card should be connected to the gate input and pin 14 connected to the flag output of the external device. First, an interrupt linkage is established between the multiprogrammer interface and service routine "read". Then, a control word with ISL and TME on is sent to the multiprogrammer, followed by an address word (with gate) to the appropriate card slot. The appropriate interface is then enabled to receive interrupts and the main program continues execution from this point until a calculator interrupt occurs.

5-118 When addressed with a gate, the card will initiate a gate to the external device and will not read data until it receives a flag from the device indicating data is ready. When the flag from the external device is returned to the input card, the card will read in the data and allow the multiprogrammer to return a trailing edge of flag to the appropriate interface, thereby generating an interrupt to the calculator.

Example 26. Reading Data in Automatic Handshake Mode (TME off)

GPIO Interface

```
0: moct;wtb 9,170240,20000
1: wti 0,11;rdi 4+X;mdec
2: X-100000+X
```

HP-IB Interface

```
0: fmt 1,c,z
1: wrt 723.1,"00240TBT";red 723,X
2: X-10000+X
```

Example 27. Reading Data in Timing Mode (TME on)

GPIO Interface

```
 0: oni 9,"read"
 1: moct;wtb 9,170260,20000;mdec;eir 9
          •
          •
          •
20: "read":moct;wti 0,11;rdi 4→X
21: X-100000→X
22: fxd 0;prt "data=",X;mdec;iret
```

HP-IB Interface

```
 0: rds(723)→C
 1: oni 7,"read"
 2: fmt 1,c,z
 3: wrt 723.1,"00260T";rds(723)→C
 4: wrt 723.1,"BT";eir 7
                •
                •
                •
20: "read":if rds(723)#64;gto "iret"
21: red 723,X;X-10000→X
22: fxd 0;prt "data=",X
23: "iret":iret
```

Example 28. Programming a Digital Input Card in Interrupt Mode

GPIO Interface

```
 0: oni 9,"interrupt"
 1: moct;wtb 9,170240,20000,170460;mdec
 2: eir 9
          •
          •
          •
20: "interrupt":moct;wtb 9,170240
21: wti 0,11;wti 4,20000;rdi 4→X
22: if X<100000;-1→X;gto "data"
23: X-100000→X;wtb 9,170040,20000
24: "data":prt "interrupt data =",X
25: mdec;iret
```

HP-IB Interface

```
 0: rds(723)→C
 1: oni 7,"interrupt"
 2: fmt 1,c,z
 3: wrt 723.1,"00240TBT00460T";eir 7
                •
                •
                •
20: 'interrupt":if rds(723)#64;gto "iret"
21: wrt 723.1,"00240TBX";red 723,X
22: if X<10000;-1→X;gto ."data"
23: X-10000→X;wrt 723.1,"00040TBT"
24: "data":prt "interrupt data =",X
25: "iret":iret
```

5-119 When an interrupt occurs, program control passes to service routine "read". Service routine "read" will store the data from the input card in variable "X" and print the data on the calculator before returning program control to the main program. Of course, the actual action to be taken in response to an interrupt will be decided upon by the user when writing a service routine.

5-120 For a detailed description of timing mode operation as it relates to input cards, read chapter 3, (GPIO) or chapter 4 (HP-IB), paragraphs 3-80 through 3-87 or 4-82 through 4-90, respectively.

5-121 Interrupt Mode. Example 28 illustrates the interrupt mode of operation. Programming the digital input card in the interrupt mode consists of three basic steps:

1. The card must be armed using either one of the following methods:

   a. Sending a control word with ISL on, followed by an address word (with gate) to the appropriate card slot, then another control word with IEN and TME on.

   b. If jumper W6 is installed, simply send a control word with IEN and TME on.

2. When the external device returns a trailing edge of flag to the digital input card, the digital input card will allow the multiprogrammer to return a trailing edge of flag to the appropriate interface, thereby generating an interrupt to the calculator. (Of course, the interface must have previously been enabled to receive interrupts). The card must then be addressed without a gate in the ISL mode and the return data word read in and examined for the presence of IRQ. If the input data word is equal to or greater than 100000 (GPIO) or 10000 (HP-IB), it indicates that the card has interrupted and has data ready.

3. The card should then be disarmed by sending a control word with ISL off, followed by an address word (with gate) to the appropriate card.

5-122 Example 28 assumes that jumper W6 has been removed from the input card, allowing the user to selectively arm the card. Unless the user is very familiar with the multiprogrammer, it is recommended that W6 be removed from all digital input cards in the system.

5-123 When an interrupt occurs, service routine "interrupt" will read the digital input card in slot 402 (B) (without gate) and examine the state of IRQ to be certain that the card had generated the interrupt. (This is very important when using multiple input cards in interrupt mode). A value of −1 will be stored in "X" if the card in slot 4Ø2 (B) did not generate the interrupt. If the card had interrupted, valid data will be stored in "X" and the card will be disarmed by sending a control word with ISL off and addressing the card with gate. In either case, the calculator will then print the value stored in "X" and return program control to the main program.

5-124 For more information on interrupt mode of operation, including a sample multiple card program, read chapter 3 (GPIO) or chapter 4 (HP-IB) beginning with paragraphs 3-88 and 4-91, respectively.

5-125 69431A Sample Test Program. Example 29 is a sample test program which will allow the user to verify the operation of 69431A digital input cards equipped with options 069 or 070. To verify the operation of option 073 cards which have an opposite logic sense make the following changes in the programs:

1. Change line number 5 of the GPIO program to read: otd(1Ø7777-X)→X;gsb "BIT".

2. Change line number 6 of the HP-IB program to read: otd (17777-X)→X; gsb "BIT".

5-126 Test Procedure. Perform the following steps when using the sample test program.

1. Load the program (including subroutine "BIT" from Appendix A) into the calculator.

2. On the input connector of the 69431A card, temporarily install a jumper between pins 13 and 14, and another jumper between pin 15 and any user selected input pin (1 through 12).

3. Depress RUN on the calculator. If IRQ is not operating properly, or if a flag is not returned, the program will be terminated and the calculator will display "IRQ OR FLAG RETURN ERROR". This problem should be corrected before re-running the test. If IRQ is operating properly and the flag is returned normally, the calculator will stop with the input bit and pin number displayed. If no data line (pins 1-12) has been jumpered to pin 15, the calculator will stop with "NO DATA RECEIVED" displayed.

4. Depressing CONTINUE after the bit has been read back and displayed will cause the calculator to display "TME TEST: REMOVE JUMPERS".

5. Disconnect all jumpers.

6. Depressing CONTINUE will cause the calculator to pause, waiting for the input card to return a CTF flag. The red "RUN" light in the display will be lit.

7. Momentarily short pins 13 and 14 together. The CTF flag will be returned and the calculator will display "TEST COMPLETE".

Example 29. 69431A Sample Test Program

GPIO Interface

```
0: moct;wtb 9,170240,20000
1: wti 0,11;rdi 4→X
2: if X>99999;jmp 3
3: dsp "IRQ OR FLAG RETURN ERROR'
4: gto "END"
5: otd(X-100000)→X;gsb "BIT"
6: stp
7: dsp "TME TEST: REMOVE JUMPERS";stp
8: wtb 9,170260,20000,170240
9: dsp "TEST COMPLETE"
10: "END":mdec;end
*39
```

HP-IB Interface

```
0: rds(723)→C
1: fmt 1,c,z
2: wrt 723.1,"00240TBT";red 723,X
3: if X>9999;jmp 3
4: dsp "IRQ OR FLAG RETURN ERROR"
5: gto "END"
6: otd(X-10000)→X;gsb "BIT"
7: stp
8: dsp "TME TEST: REMOVE JUMPERS";stp
9: wrt 723.1,"00260TBT00240T";rds(723)→C
10: dsp "TEST COMPLETE"
11: "END":end
*27178
```

5-127 Relay Output With Readback, 69433A

5-128 This card provides 12, independent, SPST, normally open (form A) relays. Output data will be in the form of contact closures. In addition, the 69433A relay output card allows the calculator to examine the status of the relay coil drive circuits on the card, before and/or after the contacts are actually changed. The following discussion will familiarize the user with programming the relay output/readback card 69433A. All examples will assume the card is installed in unit Ø, slot 4Ø2 (B).

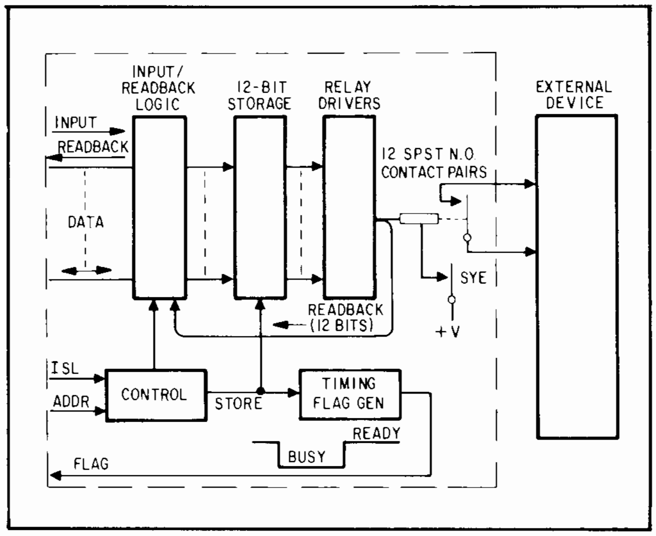
5-129 **Programming Output Data.** Example 30 illustrates using a data constant of 7777_8 to program a 69433A relay output card located in slot 4Ø2 (B). A control word with SYE, DTE, and TME on is sent, followed by a data word containing the card slot address and output data, then another control word with SYE only. Since TME has been turned on by the first control word, the calculator will be forced to wait until the relay card returns a CTF flag (4 msec) before sending the second control word. The timing mode of operation used in example 30 is described in chapters 3 (GPIO) and 4 (HP-IB), paragraphs 3-68 and 4-70, respectively.

5-130 **Reading Relay Status.** The user may examine the status of the relays at any time. Example 31 illustrates reading and storing the status of the relays in variable "X". A control word with ISL on is sent to the multiprogrammer, followed by an address word (without gate) to the appropriate card slot. The return data word is then stored in "X". The 69433A card is always read without a gate.

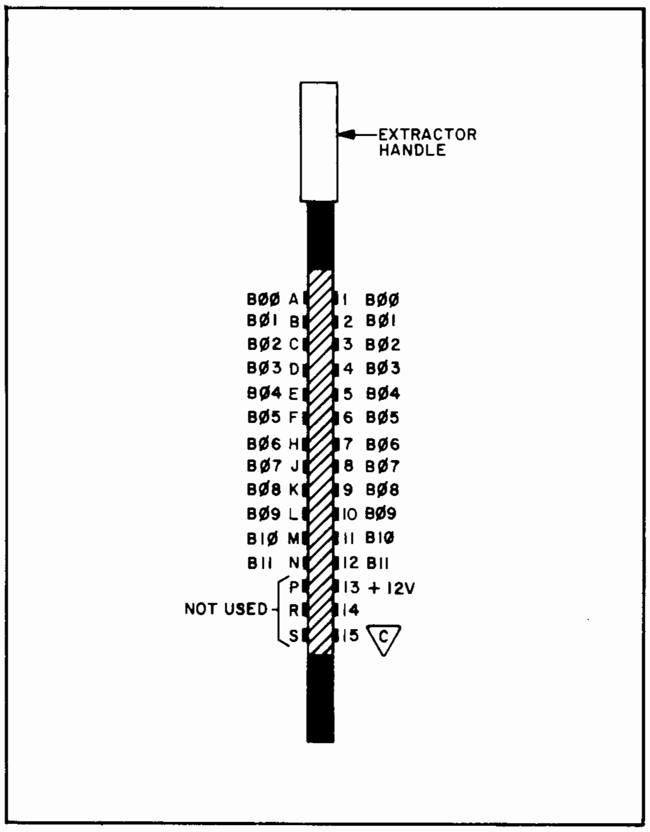
5-131 **69433A Sample Test Program.** Example 32 is a sample test program which illustrates programming variable data values and reading the status of the relays. In this example, a control word with SYE only is used when programming the relay outputs. Since TME is not used, there is no indication to the calculator when the card finishes programming the relays. However, since the programming time is only 4 milliseconds, this approach is acceptable in some applications.

5-132 **Test Procedure.** Perform the following steps when using the sample test program.

1. Load the program (including subroutine "BIT" from Appendix A) into the Calculator.



69433A Block Diagram



69433A Output Connector

Example 30. Programming a Relay Output

GPIO Interface

```
Ø: moct;wtb 9,170160,27777,170040
```

HP-IB Interface

```
0: fmt 1,c,z
1: wrt 723.1,"00160TB7777T00040T"
2: rds(723)→C
```

2. Depress RUN on the calculator.

3. The calculator will stop with "ENTER PIN NO." displayed. The user must enter a pin number from 1 to 12, then depress CONTINUE.

4. The calculator will stop with "TEST BIT" displayed. At this point, the user should use an ohmmeter to check the contact resistance of the relays. Relay outputs can be measured between a numbered Pin, 1 through 12, and the corresponding lettered pin (e.g., 1 and A, 2 and B, 12 and N). All relay contacts should be open except for the contacts corresponding to the pin number that was entered in step 3.

5. On the 6940 Multiprogrammer, place the data source switch to LOCAL. Momentarily depress the CLEAR REGISTER switch. Press switches 15-12 (control word) and switch 5 (SYE). Depress LOAD OUTPUT switch.

6. Depress CLEAR REGISTER switch. Press switch 13, then select and press a data switch from 0 to 11. Depress LOAD OUTPUT switch.

7. Switch multiprogrammer Data Source switch back to REMOTE.

8. Depress CONTINUE on the calculator. The calculator will display the bit and corresponding output pin number that was manually programmed in step 6.

Example 31. Reading Relay Status

GPIO Interface

```
0: moct;wtb 9,170240
1: wti 0,11;wti 4,20000;rdi 4→X
```

HP-IB Interface

```
0: fmt 1,c,z
1: wrt 723.1,"00240TBX";red 723,X
```

Example 32. 69433A Sample Test Program

GPIO Interface

```
0: ent "ENTER PIN NO.",A
1: dto(2↑(A-1))→A;if A>7777;gto "error"
2: moct;wtb 9,170040,20000+A
3: dsp "TEST BIT";stp
4: wtb 9,170240
5: wti 0,11;wti 4,20000;rdi 4→X
6: otdX→X;gsb "BIT"
7: gto "end"
8: "error":dsp "PIN NO. GREATER THAN 12"
9: "end":mdec;end
*6541
```

HP-IB Interface

```
0: rds(723)→C
1: ent "ENTER PIN NO.",A
2: dto(2↑(A-1))→A;if A>7777;gto "error"
3: fmt 2,c,f4.0,c,z
4: wrt 723.2,"00040TB",A,"T"
5: dsp "TEST BIT";stp
6: fmt 1,c,z
7: wrt 723.1,"00240TBX"
8: red 723,X;otdX→X;gsb "BIT"
9: gto "end"
10: "error":dsp "PIN NO. GREATER THAN 12"
11: "end":end
*3013
```

5-133 Event Sense Card, 69434A

5-134 This card monitors up to 12 external contact closures and can be used to interrupt the calculator when one or more contacts change for 20 milliseconds or longer with respect to 12 reference bits stored on the card. The following discussion will familiarize the user with programming the event sense card 69434A. All examples will assume the event sense card is installed in unit Ø, slot 4Ø2 (B).

5-135 **Return Data Word.** The return data word will be input in octal form to variable "X". It will then be up to the user to process the data. For example, the data could be used in octal form or converted to decimal by the octal to decimal function (otd) of the calculator

5-136 **Interrupt Mode.** Basically, this card is only used in the interrupt mode, either singly or with other cards having interrupt capability, and will generate an interrupt when any one of four conditions prevails. The interrupting condition is selected by jumper (W3) as follows:

*W3 Position	Interrupting Condition
A	Input Word = Reference Word
B	Input Word > Reference Word
C	Input Word < Reference Word
D	Input Word ≠ Reference Word

* The card is shipped from the factory with the W3 jumper in the D position.

5-137 The event sense card continually compares the input word with a reference word while the system is in the interrupt mode and stores the entire input word at the moment the event (interrupt) occurs. Both the interrupting input word and the reference word can be read back, after the event has occurred. As shipped from the factory, the event sense card will not store input data unless the multiprogrammer is in the interrupt mode with the gate set.

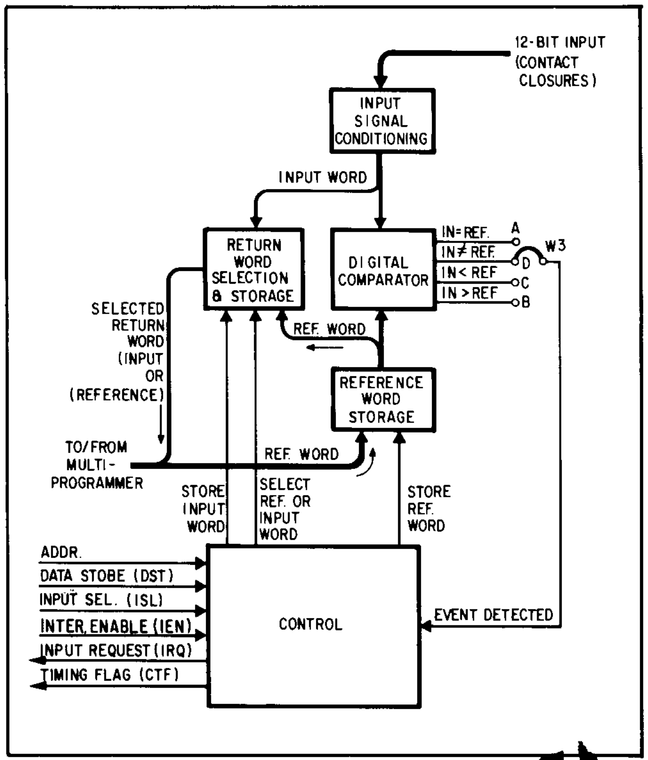
5-138 Programming the Event Sense Card consists of four basic steps.

1. Loading the reference word to initialize the card.

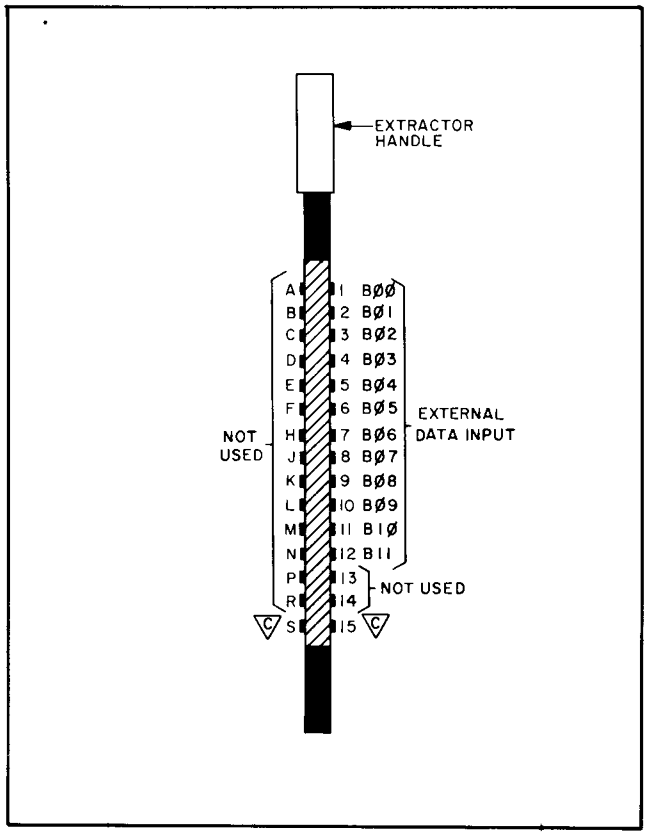
2. Arming the card. This permits the card to control the multiprogrammer flag and thereby generate an interrupt to the calculator.

3. Reading the card without a gate to return the interrupting word.

4. Disarming the card.



69434A Block Diagram



69434A Input Connector

5-139 Sample Test Program. Example 33 is a sample test program which illustrates programming a 69434A event sense card.

5-140 The first step in running a program using event sense cards is to establish an interrupt linkage between a user provided service routine and the interface that will be interrupting. Line 0 of example 33 establishes the service routine labeled "interrupt" as the routine to be executed when an interrupt occurs on the appropriate interface.

5-141 Loading the Reference Word. Line Number 1 of the GPIO example and line number 2 of the HP-IB example illustrate loading a fixed reference word of 7777 into storage on the event sense card. A control word containing SYE only is sent first, followed by a data word containing the card slot address and reference data.

5-142 Arming the Card. Line number 2 of the GPIO example and line number 3 of the HP-IB example illustrate arming the event sense card. The card must be armed in one of two ways before it can generate an interrupt.

1. Send a control word with ISL on, address the card with a gate, then send another control word with IEN and TME,

or

2. If the W6 jumper is installed, simply send a control word with IEN and TME on.

5-143 Example 33 assumes that jumper W6 has been removed from the event sense card, allowing the user to selectively arm the card. Unless the user is very familiar with the multiprogrammer, it is recommended that W6 be removed from all event sense cards in the system.

5-144 After arming the card, the appropriate interface is enabled for interrupts. The main program would normally continue from this point until an interrupt occurs. The for/next loop in example 33 simulates a main program which will run for approximately five minutes or until an interrupt occurs.

5-145 Reading the Card. When an interrupt occurs, program control will pass to the service routine labeled "interrupt". Lines 7 and 8 of the GPIO example and line 8 of the HP-IB example illustrate sending a control word with ISL on, addressing the card without a gate, and storing the return data word in "X".

5-146 Normally the user's program will keep track of the reference data sent to the event sense card. In the event the program failed to keep track of the data, the reference data sent to the card before it interrupted can be retrieved by addressing and reading the card with a gate. However, the card must not be read with a gate until it has first been read without a gate and the interrupting word stored in the calculator. Reading the card with a gate causes the interrupting word to be lost from storage. A jumper (W2) permits the user to modify the manner in which the input word is stored. Consult the 69434A operating and service manual for details.

5-147 The return data word is then checked for IRQ. If IRQ is not present, it means another card in the system has interrupted (This is a very important test when using multiple cards in interrupt mode). In this case, a value of −1 is stored in "X" and printed on the calculator as an indication that another card in the system generated the interrupt.

5-148 Disarming the Card. Once the card has generated an interrupt (as indicated by IRQ being set) and been read, it should be disarmed. The card is disarmed by addressing the card in the output mode, which is the same method used for loading the reference word. Generally, the user

Example 33. 69434A Sample Test Program

GPIO Interface

```
0: oni 9,"interrupt"
1: moct;wtb 9,170040,27777
2: wtb 9,170240,20000,170460;mdec
3: eir 9
4: for J=1 to 500000
5: next J
6: end
7: "interrupt":moct;wtb 9,170240
8: wti 0,11;wti 4,20000;rdi 4→X
9: if X<100000;-1→X;prt "X=-1";gto "iret"
10: X-100000→X;wtb 9,170040,20000+X
11: otd(7777-X)→X;gsb "BIT"
12: "iret":mdec;500000→J;iret
*21685
```

HP-IB Interface

```
0: rds(723)→C;oni 7,"interrupt"
1: fmt 1,c,z;fmt 2,c,f4.0,c,z
2: wrt 723.1,"00040TB7777T"
3: wrt 723.1,"00240TBT00460T";eir 7
4: for J=1 to 500000
5: next J
6: end
7: "interrupt":if rds(723)#64;gto "iret"
8: wrt 723.1,"00240TBX";red 723,X
9: if X<10000;-1→X;prt "X=-1";gto "iret"
10: X-10000→X;wrt 723.2,"00040TB",X,"T"
11: otd(7777-X)→X;gsb "BIT"
12: "iret":500000→J;iret
*6383
```

will simultaneously disarm the card and load in a new reference word, after which the card may be rearmed at the user's convenience. Line number 10 illustrates subtracting IRQ from the return data word, storing the interrupting word in "X", and disarming the card by sending a new reference word containing "X" as the reference data.

5-149 The octal to decimal function and data inversion, used in line 11 of service routine "interrupt", are only to satisfy the requirements of subroutine "BIT" which prints out the interrupting bit and pin number. Upon returning from subroutine "BIT", the service routine increments the 5 minute counter to the end and returns program control to the simulated main program.

5-150 For more information on the interrupt mode of operation, including a sample multiple card program, read chapter 3 (GPIO) or chapter 4 (HP-IB) beginning with paragraphs 3-88 and 4-91, respectively.

5-151 Test Procedure. Perform the following steps when using the 69434A sample test program.

1. Load the program (including subroutine "BIT" from Appendix A) into the calculator.

2. Depress RUN on the calculator. The red RUN light in the calculator display will be lit.

3. Connect a jumper wire between pin 15 and any input pin (1 through 12) on the card.

4. The calculator will stop with the interrupting bit and pin number displayed.

5. Disconnect the jumper wire connected in step 3 before re-running the test.

5-152 Pulse Counter Card, 69435A

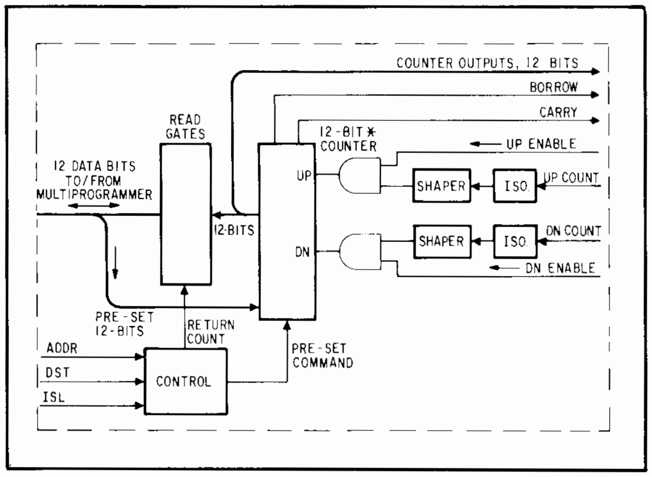
5-153 This card will count pulses, up or down, in the range of 0 to 4095. A carry or borrow pulse is generated as the count goes above 4095 or below 0. These pulses allow multiple counter cards to be cascaded for greater counting capability or they can serve as alarm signals. The card can also be used as a pre-set counter. The following discussion will familiarize the user with programming the pulse counter card 69435A. All examples will assume the pulse counter card is installed in unit Ø, slot 4Ø2 (B).

5-154 Programming the Card. Programming the pulse counter card consists of two basic steps.

1. The pulse counter card must first be preset by sending a control word to the multiprogrammer with ISL off, followed by a data word containing the card slot address and the data value which the counter is to be preset to. When the data word is gated into the multiprogrammer the counter will be preset.

2. The pulse counter card can be read by sending a control word with ISL on, then addressing the card without a gate and storing the return data in the calculator. It will then be up to the user to process the data. For example, the data can be used in octal form or converted to decimal by using the octal to decimal (otd) function of the calculator

5-155 Sample Test Program. Example 34 is a sample test program which illustrates programming a 69435A pulse counter card. Jumpers W1 and W2 must not be installed on the pulse counter card when running this program.



69435A Block Diagram

5-156 Presetting the Card. Line Ø of example 34 illustrates presetting the pulse counter card to zero. A control word with SYE only is sent first, followed by a data word containing the card slot address and a data value of zero.

5-157 The data used to preset the card must be either zero or an octal value. If necessary, the decimal to octal (dto) function of the calculator may be used to convert decimal values to octal before being inserted in the data word. Data may be in the form of a constant, as shown in example 34 or a variable. For an example of transmitting variable data values, see chapter 3 (GPIO) or 4 (HP-IB) paragraphs 3-43 or 4-43, respectively.

5-158 Once the card has been preset, the user's program is free to continue with other tasks, then come back at any time and read the data contained in the card. In example 34 the program stops after presetting the card to allow the user to enter data into the card before being read.

5-159 Reading the Card. Program lines numbered 2 and 3 illustrate reading the data contained in the card, then storing it in variable "X". A control word with ISL on is sent first. The card is then addressed without a gate and the return data word is stored in "X". The pulse counter card must always be read without a gate.

5-160 As mentioned previously, it is up to the user to process the octal data stored in variable "X". In example 34, this data is converted to decimal and the count is displayed on the calculator.

5-161 Test Procedure. Perform the following steps when using the 69435A sample test program.

1. Load the program into the calculator.

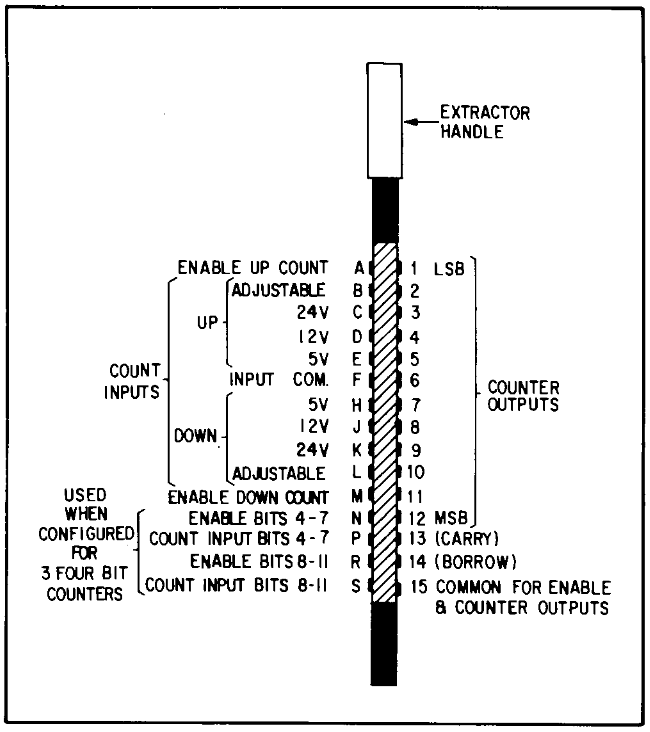
2. On the input connector of the pulse counter card, install a jumper between pins F and 15.

3. Connect a second jumper as follows:
   a. For a count-up test: connect a jumper between pin E of the pulse counter card and test point GAT 300 located on the logic and timing card in slot 3ØØ.
   or
   b. For a count-down test: connect a jumper between pin H of the pulse counter card and test point GAT 300 located on the logic and timing card in slot 3ØØ.

4. Depress RUN on the calculator. The calculator will display "BEGIN COUNTING".

5. On 6940 multiprogrammer, place DATA SOURCE switch to LOCAL. Momentarily depress CLEAR REGISTER switch. Press (light) switches 15-12 (control word) and switch 5 (SYE).

6. Press LOAD OUTPUT switch as many times as desired. (Pressing the LOAD OUTPUT switch will generate a data strobe pulse which we will count).



69435A Input Connector

7. Switch DATA SOURCE switch back to REMOTE.

8. Depress CONTINUE on the calculator.
   a. If a count up test is being run the calculator will display: count = (the number of times the LOAD OUTPUT switch was depressed in step 6, ± 1 count).
   b. If a count down test is being run the calculator will display count = (4094 minus the number of times the LOAD OUTPUT switch was depressed in step 6, ± 1 count).

NOTE

A tolerance of 1 count is allowed in this test to provide for any transients that may be coupled into the jumper wire as a result of switching the DATA SOURCE switch.

Example 34. 69435A Sample Test Program

GPIO Interface

```
0: moct;wtb 9,170040,20000;mdec
1: dsp "BEGIN COUNTING";stp
2: moct;wtb 9,170240
3: wti 0,11;wti 4,20000;rdi 4→X;mdec
4: otdX→X
5: fxd 0;dsp "count=",X-1;end
*10800
```

HP-IB Interface

```
0: fmt 1,c,z;wrt 723.1,"00040TBT"
1: dsp "BEGIN COUNTING";stp
2: wrt 723.1,"00240TBX"
3: red 723,X
4: otdX→X
5: fxd 0;dsp "count=",X-1;end
*6753
```

5-162 Process Interrupt Card, 69436A

5-163 This card provides TTL and open collector compatible edge detectors; one positive and one negative for each of 12 storage latches. Logic transitions lasting 100 nanoseconds or longer are detected, stored and can be used to interrupt the calculator. The following discussion will familiarize the user with programming the process interrupt card 69436A. All examples will assume the process interrupt card is installed in unit 0, slot 402 (B).

5-164 **Return Data Word.** The return data word will be input in octal form to variable "X". It will then be up to the user to process the data. For example, the data could be used in octal form or converted to decimal by the octal to decimal (otd) function of the calculator.

5-165 **Interrupt Mode.** Basically this card is only used in the interrupt mode, either singly or with other cards having interrupt capability, and will generate as interrupt when any one of the 24 edge detectors is toggled.

5-166 Programming the Process Interrupt Card consists of five basic steps.

1. Initializing all edge detectors to a known (reset) state.

2. Arming the card. This permits the card to control the multiprogrammer flag and thereby generate an interrupt to the calculator.

3. Reading the card without a gate to return the interrupting bit(s) to the calculator.

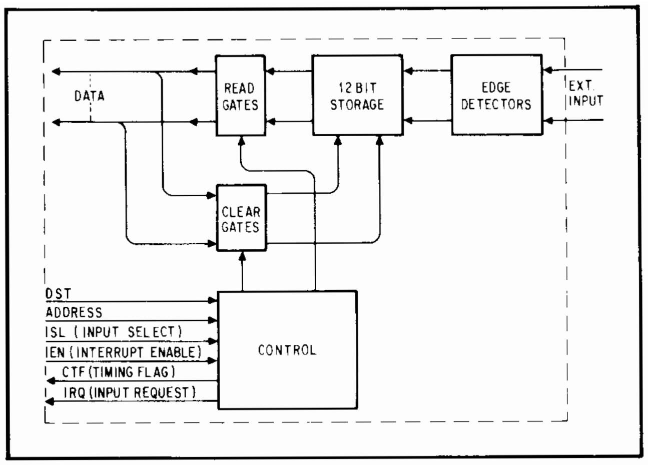
4. Resetting the card to clear the bit(s) that interrupted and prepare the card for generating a new interrupt.

5. Disarming the card to prevent further interrupts.

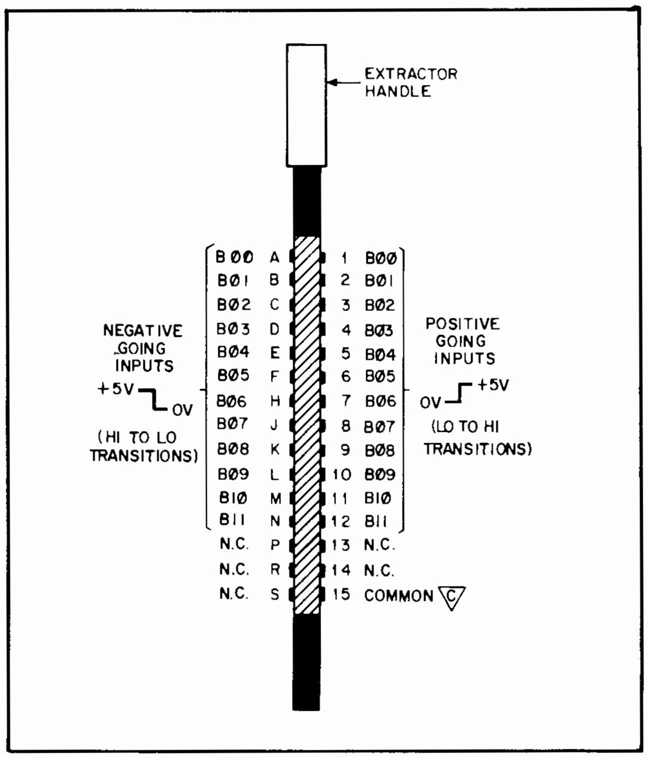
5-167 **Sample Test Program.** Example 35 is a sample test program which illustrates programming a 69436A Process Interrupt Card.

5-168 The first step in running a program using process interrupt cards is to establish an interrupt linkage between a user provided service routine and the interface that will be interrupting. Line 0 of Example 35 establishes the service routine labeled "interrupt" as the routine to be executed when an interrupt occurs on the appropriate interface.

5-169 *Initializing.* Line number 1 of the GPIO example and line number 2 of the HP-IB example illustrate initializing the card by sending a data word with logical ones in bit positions BØØ through B11 (i. e., octal 7777). Notice that a control word containing SYE only is sent first. Although



69436A Block Diagram



69436A Input Connector

SYE has no effect on this card, it is included in the control word to avoid disabling the outputs of previously programmed cards.

5-170 The card may also be initialized by first reading a return data word from the card, subtracting IRQ (only if it is contained in the return data word) and then sending the same word back. In this case the output data will be a variable. The method of transmitting variable data to initialize the card is the same as the method described in paragraph 5-176 for resetting the card.

5-171 Arming the Card. Line number 2 of the GPIO example and line number 3 of the HP-IB example illustrate arming the process interrupt card. The card must be armed in one of two ways before it can generate an interrupt.

1. Send a control word with ISL on, address the card with a gate, then send another control word with IEN and TME,

or

2. If the W6 jumper is installed, simply send a control word with IEN and TME on.

5-172 Example 35 assumes that jumper W6 has been removed from the process interrupt card allowing the user to selectively arm the card. Unless the user is very familiar with the multiprogrammer, it is recommended that W6 be removed from all process interrupt cards in the system.

5-173 After arming the card, the appropriate interface is enabled for interrupts. The main program would normally continue from this point until an interrupt occurs. The for/next loop in Example 35 simulates a main program which will run for approximately five minutes or until an interrupt occurs.

5-174 Reading the Card. When an interrupt occurs, program control will pass to the service routine labeled "interrupt". Lines 7 and 8 of the GPIO example and line 8 of the HP-IB example illustrate sending a control word with ISL on, addressing the card without a gate, and storing the return data word in "X".

5-175 The return data word is then checked for IRQ. If IRQ is not present, it means another card in the system has interrupted (this is a very important test when using multiple cards in interrupt mode). In this case, a value of −1 is stored in "X" and printed on the calculator as an indication that another card in the system generated the interrupt.

5-176 Resetting the Card. Once the card has generated an interrupt and been read, it must be reset. The card is reset when the return data word (minus the IRQ bit) is sent back to the card in the same manner given in paragraph 5-170 for initializing the card. Line number 1Ø of Example 35 illustrates subtracting IRQ from the return data word, storing the data (minus IRQ) in "X", and resetting the card by sending a data word containing the data value stored in variable "X" to the card. Since the reset operation clears only those bits which were detected prior to the read operation, other bit transitions which occur during the read operation or afterwards are not lost, but are detected and stored, and may be sent to the calculator during the next input cycle.

5-177 In order to receive further interrupts after the card has been reset, the card must be re-armed (paragraph 5-171) and the appropriate calculator interface must be re-enabled to receive interrupts.

NOTE

It is important that the process interrupt card not be read and reset if no interrupt has occurred, since the reset data word (all zeroes in this case) will disarm the card.

5-178 Disarming the Card. If it is not desired to receive further interrupts, the card should be disarmed after it has been read and reset. Once disarmed, the card cannot interrupt, but the edge detectors and storage registers are not disabled, allowing any further bit transitions to be detected and stored for possible future evaluation (when program returns to interrupt mode). The process interrupt card is disarmed whenever it receives a data word containing all zeros in the output mode (ISL off). To disarm the card, change line 1Ø of the GPIO and HP-IB programs in Example 35 as follows:

GPIO – add, 2ØØØØ at end of line 1Ø.
HP-IB – change "T" to ":TBT" at end of line 1Ø.

5-179 For more information on interrupt mode of operation, including a sample multiple card program, read Chapter 3 (GPIO) or Chapter 4 (HP-IB) beginning with paragraphs 3-88 and 4-91, respectively.

5-180 Test Procedure. Perform the following steps when using the 69436A sample test program.

1. Load the program (including subroutine "BIT" from Appendix A) into the calculator.

2. Connect a jumper wire between pin 15 and any input pin (1 through 12) on the card.

3. Depress RUN on the calculator. The red RUN light in the calculator display will be lit.

4. Remove the jumper wire connected in step number 2.

5. The calculator will stop with the interrupting bit and pin number displayed.

Example 35. 69436A Sample Test Program

GPIO Interface

```
0: oni 9,"interrupt"
1: moct;wtb 9,170040,27777
2: wtb 9,170240,20000,170460;mdec
3: eir 9
4: for J=1 to 500000
5: next J
6: end
7: "interrupt":moct;wtb 9,170240
8: wti 0,11;wti 4,20000;rdi 4→X
9: if X<100000;-1→X;prt "X=-1";gto "iret"
10: X-100000→X;wtb 9,170040,20000+X
11: otdX→X;gsb "BIT"
12: "iret":mdec;500000→J;iret
*2445
```

HP-IB Interface

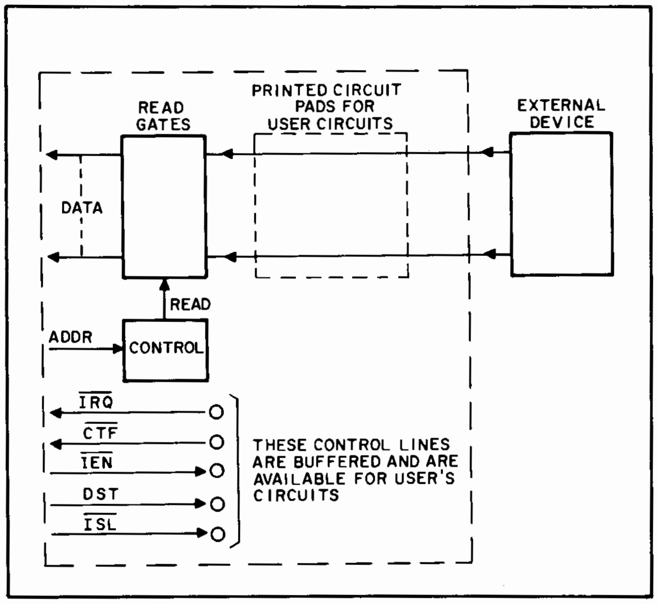
```
0: rds(723)→C;oni 7;"interrupt"
1: fmt 1,c,z;fmt 2,c,f4.0,c,z
2: wrt 723.1,"00040TB7777T"
3: wrt 723.1,"00240TBT00460T";eir 7
4: for J=1 to 500000
5: next J
6: end
7: "interrupt":if rds(723)#64;gto "iret"
8: wrt 723.1,"00240TBX";red 723,X
9: if X<10000;-1→X;prt "X=-1";gto "iret"
10: X-10000→X;wrt 723.2,"00040TB",X,"T"
11: otdX→X;gsb "BIT"
12: "iret":500000→J;iret
*19911
```

5-181 Breadboard Input Card, 69480A

5-182 The 69480A breadboard input card is a simple multiprogrammer interface card, which allows the user to design, build, and control special input circuits through the multiprogrammer system. The following discussion will familiarize the user with programming the breadboard input card 69480A. All examples will assume the input card is installed in unit Ø, slot 402 (B).

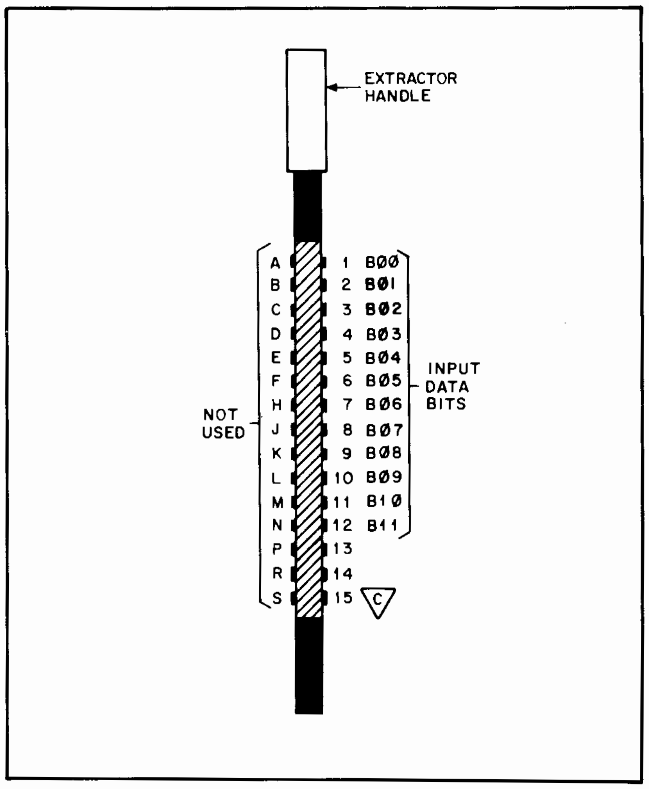
5-183 **69480A Sample Test Program.** Example 36 is a sample test program which will allow the user to verify the operation of 69480A cards by reading in and displaying data applied to the 69480A input connector.

5-184 In Example 36, the octal data is read into variable "X" in line number 1. The first two lines of this example are all that is required to return data from this card. The user has the option of using the data in its octal form, or using the octal to decimal function (otd) of the calculator to obtain the equivalent decimal value. The data inversion in line number 2, as well as the (otd) function, are only used in this example to satisfy the requirements of subroutine "BIT".



PRINTED CIRCUIT PADS FOR USER CIRCUITS
READ GATES
EXTERNAL DEVICE
DATA
READ
ADDR
CONTROL
IRQ
CTF
IEN
DST
ISL
THESE CONTROL LINES ARE BUFFERED AND ARE AVAILABLE FOR USER'S CIRCUITS

69480A Block Diagram



69480A Input Connector

5-185 The breadboard input card is always read without a gate, as there is no storage or CTF drive capability on this card. The only requirements for reading data from this card are: (1) A control word with ISL on must have been previously sent and (2) the card must be addressed.

5-186 Test Procedure. Perform the following steps when using the 69480A sample test program.

1. Load the program (including subroutine "BIT" from Appendix A) into the calculator.

2. On the input connector of the breadboard input card, install a jumper between pin 15 and any user selected input pin (1 through 12).

3. Depress RUN on the calculator. The calculator will stop with the input bit and pin number displayed. If no data has been input, the calculator will stop with "NO DATA RECEIVED" displayed.

4. Repeating steps 2 and 3 will enable the user to check each data input line on the card.

Example 36. 69480A Sample Test Program

GPIO Interface

```
0: moct;wtb 9,170240
1: wti 0,11;wti 4,20000;rdi 4→X;mdec
2: 7777-X→X
3: otdX→X;gsb "BIT"
4: end
*23576
```

HP-IB Interface

```
0: fmt 1,c,z
1: wrt 723.1,"00240TBX";red 723,X
2: 7777-X→X
3: otdX→X;gsb "BIT"
4: end
*21140
```

5-187 Resistance Output Cards 69500A—69513A

5-188 These cards provide a programmed value of resistance as their output. Twelve magnetically shielded, mercury-wetted, reed relays select the resistance values by modifying the value of a series string of high-accuracy, binary weighted resistors.

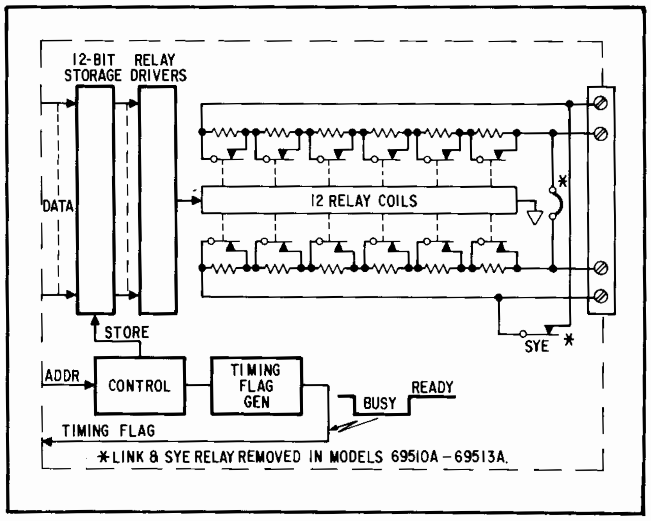
1. Model 69500A is supplied without output resistors so that customers may select and load their own resistors if desired.

2. Models 69501A—69506A are single output, 12-bit resolution cards designed to program the voltage output of a single HP power supply equipped with Option 040.

3. Models 69510A—69513A are dual output, 6-bit resolution cards designed to program the current outputs of two HP power supplies equipped with Option 040.

5-189 The following discussion will familiarize the user with programming the resistance output cards 69500A — 69513A. All examples will assume the card is installed in unit Ø, slot 402 (B). For 69510A—69513A cards, channel 1 refers to the output programmed by the lower order bits, BØØ — BØ5, and channel 2 refers to the output programmed by the higher order bits, BØ6—B11. Channel 1 resistance output terminals are C and D while channel 2 resistance output terminals are A and B.

5-190 Resistance Outputs. Octal data transmitted to program a resistance card will program one 12-bit resistance output from 69500A—69506A resistance cards, or two 6-bit resistance outputs from 69510A—69513A resistance cards. When programming a 69510A—69513A card, the two least significant digits in the data field will program channel 1 (BØØ —BØ5), and the two most significant digits will program channel 2 (BØ5—B11). Table 5-5 lists the maximum resistance outputs and resolution of each card.



69500A — 69513A Block Diagram

Table 5-5. Maximum Output Resistance and Resolution

Model		Maximum Resistance (Ohms)	Resolution (Ohms)
69500A	*		
69501A		8,190	2
69502A		30,712.5	7.5
69503A		61,425	15
69504A		40,950	10
69505A		81,900	20
69506A		204,750	50
69510A	**	252	4
69511A	**	945	15
69512A	**	1,260	20
69513A	**	1,890	30

* Resistance values on the 69500A are determined by the user.

** Maximum resistance and resolution are specified per channel.

5-191 The following steps will yield the data value necessary to program a resistance output card to the desired resistance. For 69500–69506 cards, or single channel (channel 1) programming of 69510–69513 cards, it is only necessary to perform steps 1 and 2. For dual channel programming of 69510A–69513A cards, perform steps 1 through 4.

1. Divide the required output resistance in ohms by the resolution of the card for each channel.

2. Calculate the octal equivalent of the value yielded in step 1.

3. Multiply the octal value obtained for channel 2 by 100 (decimal).

4. Add the values obtained for channels 1 and 2 together.

For example, to calculate the octal data value required to program 4128 ohms from a 69501A resistance card;

1. 4128/2 = 2064
2. Octal equivalent of 2064 = 4020

Or, to calculate the octal data value required to program a 69510A card to 64 ohms on channel 1 and 128 ohms on channel 2;

1. 64/4 = 16
2. Octal equivalent of 16 = 20 (channel 1)
3. 128/4 = 32
4. Octal equivalent of 32 = 40 (channel 2)
5. 40 x 100 = 4000
6. 20 + 4000 = 4020

5-192 Programming a Resistance Output. Example 37 illustrates using a data constant of 4020_8 to program a resistance from a resistance output card. A control word is sent first, containing SYE, DTE, and TME, followed by a data word containing the card slot address and the output data desired.

5-193 The control word following the data word in Example 37 will cause the calculator to pause until the data transfer with the multiprogrammer is complete (6ms) before continuing the program. This technique of timing mode operation is also used in Examples 38 through 41 and is described in Chapters 3 (GPIO) and 4 (HP-IB), paragraphs 3-68 and 4-70, respectively. The user has the option of not using TME (and the second control word) if desired. In this case, there would be no indication to the calculator when the data transfer was completed.

5-194 Programming Power Supplies Equipped With Option 040. Up to this point the discussion has focused on programming a resistance output from a resistance output card. However, it is not necessary to calculate the resistance to program an Option 040 power supply to a desired voltage or current. Instead, selection of a resistance card corresponding to a specific Option 040 power supply changes the calculations to the following steps.

1. Divide the required output voltage or current by the LSB (see Paragraph 5-195) of the power supply/resistance card combination.

2. Convert the decimal value to octal.

3. If a dual channel resistance card is being used, multiply the octal value obtained for channel two by 100, then add channels one and two together, as shown previously when programming resistance outputs.

Example 37. Programming a Resistance Output

GPIO Interface

```
0: moct;wtb 9,170160,24020,170040
```

HP-IB Interface

```
0: fmt 1,c,z
1: wrt 723.1,"00160TB4020T00040T"
2: rds(723)→C
```

5-195 The LSB of the power supply/resistance card combination may be determined as follows.

1. For resistance cards 69500A–69506A, the LSB is .01 volts for power supplies up to 40 volts, and .025 volts for power supplies 41-100 volts.

2. For resistance cards 69510A–69513A, the LSB is equal to 2% of the maximum rated current of the power supply, with the exception of the 6105A, where the LSB is equal to 1.875% of the maximum rated current.

5-196 Examples 38 and 39 are subroutines which will allow the user to program Option 040 power supplies to any voltage or current desired, within the capability of the supply. Since it is not possible to program voltage or current values with greater accuracy than the LSB of the power supply/resistance card combination, any voltage (or current) value which is not a multiple of the LSB will be rounded off to the nearest value which is a multiple of the LSB, by the decimal to octal (dto) function of the calculator.

5-197 Example 38 is a subroutine which allows the user to program a voltage output from an Option 040 power supply with a maximum rating of 10 volts. The subroutine must be entered with a variable "A", equal to the output voltage desired. If an attempt is made to program either a negative voltage, or a voltage exceeding the maximum rated voltage of the power supply, the calculator will stop the program and display VOLTAGE ERROR. To continue the program, the user must depress CONTINUE on the calculator. In this case, the program will be continued but the subroutine will not process the requested output.

5-198 In order to use the subroutine in Example 38 for programming power supplies with outputs other than 10 volts, the following changes must be made.

1. Change the maximum limit in the first program line to reflect the maximum rated output voltage of the power supply. For example, if it is desired to program a 40 volt power supply, change line number Ø to read as follows:

"VØ4Ø": if A> –.ØØ1 and A< = 4Ø; jmp 2

2. If the maximum rated output of the power supply is greater than 40 volts, change line number 2 in the GPIO example or the first statement of line number 2 in the HP-IB example to read as follows:

dto (A/.025)→A

5-199 Example 39 is a subroutine which allows the user to program current outputs from two Option 040 power supplies with a maximum rating of 1 amp each. The subroutine must be entered with two variables, "A" and "B", equal to the output current, in amps, desired from each supply. Variable "A" specifies the desired output current from the power supply connected to channel 1, while variable "B" specifies the desired output current from the power supply connected to channel 2.

Example 38. Programming Voltage from an Option 040 Power Supply

GPIO Interface

```
Ø: "VØ4Ø":if A>-.ØØ1 and A<=1Ø;jmp 2
1: dsp "VOLTAGE ERROR";stp ;gto "ret"
2: dto(A/.Ø1)→A
3: moct;wtb 9,17Ø16Ø,2ØØØØ+A,17ØØ4Ø;mdec
4: "ret":ret
```

HP-IB Interface

```
Ø: "VØ4Ø":if A>-.ØØ1 and A<=1Ø;jmp 2
1: dsp "VOLTAGE ERROR";stp ;gto "ret"
2: dto(A/.Ø1)→A;fmt 2,c,f4.Ø,c,z
3: wrt 723.2,"ØØ16ØTB",A,"TØØØ4ØT"
4: rds(723)→C
5: "ret":ret
```

Example 39. Programming Current from an Option 040 Power Supply

GPIO Interface

```
Ø: "IØ4Ø":A/(1*.Ø2)→A
1: if A>-.ØØ1 and A<=5Ø;dtoA→A;jmp 3
2: dsp "CHANNEL 1 CURRENT ERROR";stp
3: gto "ret"
4: B/(1*.Ø2)→B
5: if B>-.ØØ1 and B<=5Ø;dtoB→B;jmp 3
6: dsp "CHANNEL 2 CURRENT ERROR";stp
7: gto "ret"
8: B*1ØØ+A→D
9: moct;wtb 9,17Ø16Ø,2ØØØØ+D,17ØØ4Ø;mdec
1Ø: "ret":ret
```

HP-IB Interface

```
Ø: "IØ4Ø":A/(1*.Ø2)→A
1: if A>-.ØØ1 and A<=5Ø;dtoA→A;jmp 3
2: dsp "CHANNEL 1 CURRENT ERROR";stp
3: gto "ret"
4: B/(1*.Ø2)→B
5: if B>-.ØØ1 and B<=5Ø;dtoB→B;jmp 3
6: dsp "CHANNEL 2 CURRENT ERROR";stp
7: gto "ret"
8: B*1ØØ+A→D;fmt 2,c,f4.Ø,c,z
9: wrt 723.2,"ØØ16ØTB",D,"TØØØ4ØT"
1Ø: rds(723)→C
11: "ret":ret
```

5-200 If an attempt is made to program either a negative current, or a current exceeding the maximum rated current of either power supply, the calculator will stop the program and display "CHANNEL 1 CURRENT ERROR" or "CHANNEL 2 CURRENT ERROR", corresponding to the channel that has been overprogrammed. Depressing CONTINUE after a stop occurs will allow the program to continue, however, the subroutine will not process the requested current(s). If channel 1 has been properly programmed and channel 2 has been overprogrammed, variable "A", the entry variable for channel 1, will now represent the octal equivalent of the desired current. In this case, when variable "B" is corrected, variable "A" must also be reset to the desired current value before re-entering the subroutine.

5-201 In order to use the subroutine in Example 39 to program a single power supply, or power supplies with outputs other than one ampere, the following changes must be made.

1. To program the current from a single power supply, delete lines numbered 4-8 from the GPIO example or lines numbered 4-7 and the first statement in line number 8 from the HP-IB example. Variable "D" in line number 9 should then be changed to "A". Variable "B" then becomes unnecessary and the output will be from channel 1.

NOTE

If user selection of a single output channel is desired, the subroutine may be used as is. In this case, the variable corresponding to the channel not being programmed must be set to equal zero.

2. For power supplies with outputs other than one ampere, lines numbered 0 and 4 must be changed so the maximum current outputs of the power supplies are multiplied by .02. For example, if it is desired to program a 1.5 amp power supply, change lines numbered 0 and 4 to read as follows:

```
0: "I040": A/(1.5* .02)→A
4: B/(1.5* .02)→B
```

NOTE

If a 6105A power supply is being programmed, the multiplication factor becomes .01875. This allows 54 steps to be programmed instead of the 50 steps available with other supplies. Therefore, the step-limit value of 50 in line number 1 should be changed to 54, and the multiplication factor in lines 0 and 4 should be changed to .01875.

5-202 **69500A–69506A Sample Test Program.** Example 40 is a sample test program which illustrates programming with variable data values. It allows the user to verify the operation of resistance output cards 69500A–69506A by programming any resistance value within the capability of the card. If an attempt is made to program a resistance value exceeding the capability of the card, the card will not be programmed.

5-203 As shown, Example 40 can only be used to program a 69501A card. To program 69500A, 69502A–69506A cards, line number 1 of the GPIO example or line number 2 of the HP-IB example must be changed so variable "A" is divided by the resolution of the card as given in Table 5-5.

NOTE

Since resistance values programmed must always be a multiple of the resolution (LSB) of the card, any value which is not a multiple of the card LSB will be rounded off to the nearest value which is a multiple of the LSB.

Example 40. 69500A/69506A Sample Test Program

GPIO Interface

```
0: ent "ENTER RESISTANCE",A
1: A/2→A;dtoA→A;if A>7777;gto "error"
2: noct;wtb 9,170160,20000+A,170040;ndec
3: dsp "TEST RESISTANCE";gto "end"
4: "error":dsp "RESISTANCE TOO LARGE"
5: "end":end
*6724
```

HP-IB Interface

```
0: rds(723)→C
1: ent "ENTER RESISTANCE",A
2: A/2→A;dtoA→A;if A>7777;gto "error"
3: fmt 2,c,f4.0,c,z
4: wrt 723.2,"00160TB",A,"T00040T"
5: rds(723)→C
6: dsp "TEST RESISTANCE";gto "end"
7: "error":dsp "RESISTANCE TOO LARGE"
8: "end":end
*5529
```

5-204 69500A–69506A Test Procedure. Perform the following steps when using the 69500A–69506A sample test program.

1. Load the program into the calculator.

2. Depress RUN on the calculator. The calculator will stop with ENTER RESISTANCE displayed. The user must enter a resistance value, no greater than the maximum capability of the card (as listed in Table 5-5), then depress CONTINUE.

3. After running the program, the calculator will display "TEST RESISTANCE". At this point, the user should use an ohmmeter to measure the resistance output between terminals C and D on the card. Note that a 10 ohm zero calibration resistor is in series with the output resistance of the 69501A–69506A cards.

5-205 69510A–69513A Sample Test Program. Example 41 is a sample test program which will allow the user to verify the operation of resistance output cards 69510A–69513A by allowing the user to program separate resistance values from each channel. If an attempt is made to program resistance values exceeding the capability of the card, the card will not be programmed.

5-206 As shown, Example 41 can only be used to program a 69510A card. To program 69511A–69513A cards, lines numbered 1 and 3 of the GPIO example or lines numbered 2 and 4 of the HP-IB example must be changed so variables "A" and "B" are divided by the resolution of the card as given in Table 5-5.

5-207 69510A–69513A Test Procedure. Perform the following steps when using the 69510A–69513A sample test program.

1. Load the program into the calculator.

2. Depress RUN on the calculator. The calculator will stop with "ENTER R1" displayed. The user must enter a resistance value for channel 1, no greater than the maximum capability of the card (as listed in Table 5-5), then depress CONTINUE.

3. The calculator will stop with "ENTER R2" displayed. The user must enter a resistance value for channel 2, no greater than the maximum capability of the card (as listed in Table 5-5), then depress CONTINUE.

4. After running the program, the calculator will display "TEST RESISTANCE". At this point, the user should use an ohmmeter to measure the resistance output as follows.

   a. The resistance output of channel 1 may be measured between terminals C and D.

   b. The resistance output of channel 2 may be measured between terminals A and B.

Example 41. 69510A/69513A Sample Test Program

GPIO Interface

```
0: ent "ENTER R1",A
1: A/4→A;dtoA→A;if A>77;gto "error1"
2: ent "ENTER R2",B
3: B/4→B;dtoB→B;if B>77;gto "error2"
4: B*100→B;A+B→D
5: moct;wtb 9,170160,20000+D,170040;mdec
6: dsp "TEST RESISTANCE";gto "end"
7: "error1":dsp "R1 TOO LARGE";gto "end"
8: "error2":dsp "R2 TOO LARGE"
9: "end":end
*31903
```

HP-IB Interface

```
0: rds(723)→C
1: ent "ENTER R1",A
2: A/4→A;dtoA→A;if A>77;gto "error1"
3: ent "ENTER R2",B
4: B/4→B;dtoB→B;if B>77;gto "error2"
5: B*100→B;A+B→D
6: fmt 2,c,f4.0,c,z
7: wrt 723.2,"00160TB",D,"T00040T"
8: rds(723)→C
9: dsp "TEST RESISTANCE";gto "end"
10: "error1":dsp "R1 TOO LARGE";gto "end"
11: "error2":dsp "R2 TOO LARGE"
12: "end":end
*16296
```

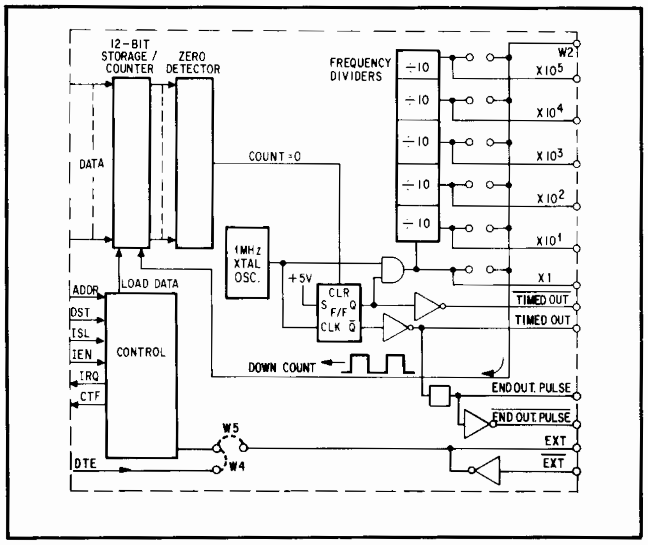
5-208 Programmable Timer Card, 69600B

5-209 This card generates a crystal-controlled one-shot pulse each time it is commanded by the program. The duration of the pulse is determined by the combination of two factors: (1) the number of programmed time increments, from 1 to 4095; and (2) the selected period of the increments, from 1 microsecond to .1 second. Period selection can be accomplished by fixed jumpers on the 69600B card or brought under program control by using a 69330A relay output card. The following discussion will familiarize the user with programming timer card 69600B. All examples will assume the timer card is installed in unit Ø, slot 402 (B).

5-210 At the option of the user, jumpers may be installed on this card to allow DTE or an external trigger signal to control the output from the card. When used this way, either DTE or an external trigger (depending on the jumper installed) may be used to simultaneously generate timed output pulses from multiple cards. Use of DTE to simultaneously generate outputs from multiple cards is described in Chapters 3 (GPIO) and 4 (HP-IB), beginning with Paragraphs 3-49 and 4-49, respectively. However, instead of generating gates, voltage outputs, or current outputs (described in Chapters 3 and 4), 69600B cards generate timed output pulses. Using an external trigger is similar to using DTE to control the card output. In this case, an external pulse, rather than a control word with DTE initiates the output pulse(s).

5-211 Operating Modes. Basically there are three operating modes which apply to this card.

1. Automatic Handshake Mode: Allows the user to program an output pulse of a specified duration, but does not provide an indication to the calculator when the pulse is complete.



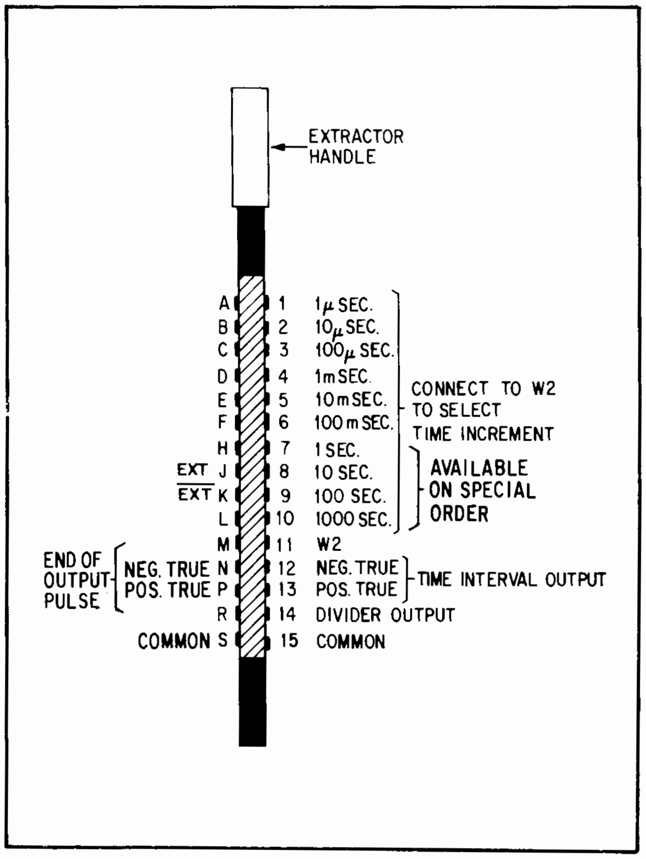
69600B Block Diagram

2. Timing Mode (TME on): Used when the user's program requires an indication that the output pulse is complete. The trailing edge of the multiprogrammer flag will be returned upon completion of the output pulse and can be used to generate an interrupt on the GPIO or HP-IB interface, thereby notifying the user's program that the pulse is complete. (In the case of the HP-IB, the service request line will be set upon completion of the output pulse). Jumper W3 must be installed on the 69600B card for this mode of operation.

3. Interrupt Mode: Allows the user to program output pulses from multiple cards and notifies the calculator, by means of an interrupt, when the first card has completed its output pulse. Jumper W3 is omitted on the 69600B for this mode of operation.

NOTE

Throughout the following discussion, the user should keep in mind that the number of time increments is programmable, but the actual period of the increments is selected by installing appropriate jumpers on the card.



69600B Output Connector

5-212 *Automatic Handshake Mode.* This mode requires sending a control word with SYE only followed by a data word containing the card slot address and the number of time increments desired (in octal). Example 42 illustrates using a data constant of 144_8 to specify the number of time increments (duration of output pulse).

5-213 *Timing Mode.* Example 43 illustrates programming a 69600B card in the timing mode. First, an interrupt linkage is established between the multiprogrammer interface and service routine "done". Then a control word with DTE, SYE, and TME is sent to the multiprogrammer, followed by a data word containing the card slot address and number of time increments desired. The appropriate interface is then enabled to receive interrupts. The main program continues execution from this point until the output pulse is complete. At that time, the multiprogrammer will return a trailing edge of flag, thereby generating an interrupt to the calculator.

5-214 When an interrupt occurs, program control passes to service routine "done". Service routine "done" sends a control word to the multiprogrammer, turning off TME, prints "done" to notify the user that the data transfer is complete, then returns program control to the main program. Of course, the actual action to be taken in response to an interrupt will be decided by the user when writing a service routine.

5-215 For a detailed description of timing mode operation as it relates to output cards, read Chapter 3, Paragraphs 3-59 through 3-70 for the GPIO interface, or Chapter 4, Paragraphs 4-59 through 4-72 for the HP-IB interface.

5-216 *Interrupt Mode.* Example 44 illustrates the interrupt mode of operation. Programming the timer card in the interrupt mode consists of four basic steps:

1. The timer card must first be loaded by sending a control word to the multiprogrammer with ISL off, followed by a data word containing the slot address of the card and the number of time increments desired.

2. The card must be armed using either one of the following methods:

   a. Sending a control word with ISL on, followed by an address word (with gate) to the appropriate card slot, then another control word with IEN and TME on.

   or

   b. If jumer W6 is installed, simply send a control word with IEN and TME on.

3. When the card completes its output pulse the multiprogrammer will return a trailing edge of flag to the appropriate interface, thereby generating an interrupt to the calculator. (Of course, the interface must have previously been enabled to receive interrupts). The card must then be addressed without a gate in the ISL mode and the return data word read in and examined for the presence of IRQ. If the return data word is equal to or greater than 100000 (GPIO) or 10000 (HP-IB) it indicates that the card has interrupted and has data ready.

4. The card should then be disarmed by sending a control word with ISL off, followed by a data word containing zero data to the appropriate card.

Example 42. Programming an Output Pulse in Automatic Handshake Mode

GPIO Interface

```
0: moct;wtb 9,170040,20144
```

HP-IB Interface

```
0: fmt 1,c,z
1: wrt 723.1,"00040TB144T"
```

Example 43. Programming an Output Pulse in Timing Mode

GPIO Interface

```
 0: oni 9,"done"
 1: moct;wtb 9,170160,20144;ndec;eir 9
               •
               •
               •
20: "done":moct;wtb 9,170040;ndec
21: prt "done";iret
```

HP-IB Interface

```
 0: rds(723)→C
 1: oni 7,"done"
 2: fmt 1,c,z
 3: wrt 723.1,"00160T";rds(723)→C
 4: wrt 723.1,"B144T";eir 7
               •
               •
               •
20: "done":if rds(723)#64;gto "iret"
21: wrt 723.1,"00040T";prt "done"
22:-"iret":iret
```

5-217 Example 44 assumes that jumper W6 has been removed from the timer card, allowing the user to selectively arm the card. Unless the user is very familiar with the multiprogrammer, it is recommended that W6 be removed from all 69600B timer cards in the system.

5-218 When an interrupt occurs, service routine "interrupt" will read the timer card in slot 402 (B) (without gate), and examine the state of IRQ to be certain that the card had generated the interrupt. (This is very important when using multiple timer cards in interrupt mode). This example only uses one card, and no action is taken if the card in slot 402 (B) did not interrupt. In actual practice the next card would be read to determine if it was the one that interrupted. If the card had interrupted, it is then disarmed by sending a control word with ISL off followed by a data word containing zero data. The calculator would then print "PULSE COMPLETE" and return program control to the main program.

5-219 For more information on interrupt mode of operation, including a sample multiple card program, read Chapter 3 (GPIO) or Chapter 4 (HP-IB) beginning with Paragraphs 3-88 and 4-91), respectively.

5-220 69600B Sample Test Program. Example 45 is a sample test program which uses the interrupt mode of operation and illustrates programming with variable data values. This example allows the user to verify the operation of 69600B Programmable Timer Cards by programming a pulse between .01 and 40.95 seconds long. If the user attempts to program a negative pulse or a pulse longer than 40.95 seconds the calculator will display "PULSE OUT OF RANGE" and will not program the card.

5-221 The for/next loop of Example 45 merely simulates a mainline program which will run for approximately 60 seconds. This allows sufficient time for the programmable timer card to generate an interrupt with the longest programmable pulse, as long as the jumper is properly installed in step number 2 of Paragraph 5-222.

5-222 Test Procedure. Perform the following steps when using the 69600B test program.

1. Load the program into the calculator.

2. On the output connector of the timer card, temporarily install a jumper between pin 5 and pin 11. If desired, the user may connect a voltmeter between pins 13 and 15 to observe the output pulse. In this case, pin 15 is common.

3. Depress RUN on the calculator. The calculator will stop with "ENTER PULSE LENGTH" displayed. The user must enter the desired pulse length, in seconds, then depress CONTINUE.

4. When the number of seconds entered in step 3 have elapsed, the calculator will display "PULSE COMPLETE". If a voltmeter was connected in step **2**, the output pulse will go positive for the number of seconds entered in step 3.

Example 44. Programming a Timer Card in Interrupt Mode

GPIO Interface

```
 0: oni 9,"interrupt"
 1: moct;wtb 9,170040,20144
 2: wtb 9,170240,20000,170460;mdec
 3: eir 9
              •
              •
              •
20: "interrupt":moct;wtb 9,170240
21: wti 0,11;wti 4,20000;rdi 4→X
22: if X<100000;gto "iret"
23: wtb 9,170040,20000
24: prt "PULSE COMPLETE"
25: "iret":mdec;iret
```

HP-IB Interface

```
 0: rds(723)→C
 1: oni 7,"interrupt"
 2: fmt 1,c,z
 3: wrt 723.1,"00040TB144T"
 4: wrt 723.1,"00240TBT00460T";eir 7
              •
              •
              •
20: "interrupt":if rds(723)#64;gto "iret"
21: wrt 723.1,"00240TBX";red 723,X
22: if X<10000;gto "iret"
23: wrt 723.1,"00040TBT"
24: prt "PULSE COMPLETE"
25: "iret":iret
```

Example 45. 69600B Sample Test Program

GPIO Interface

```
0: oni 9,"done"
1: ent "ENTER PULSE LENGTH",A
2: if A<0 or A>40.95;gto "error"
3: dto(A*100)→A
4: moct;wtb 9,170040,20000+A
5: wtb 9,170240,20000,170460;mdec;eir 9
6: for J=1 to 100000
7: next J
8: gto "end"
9: "error":dsp "PULSE OUT OF RANGE"
10: "end":end
11: "done":moct;wtb 9,170040,20000
12: dsp "PULSE COMPLETE"
13: mdec;100000→J;iret
*22054
```

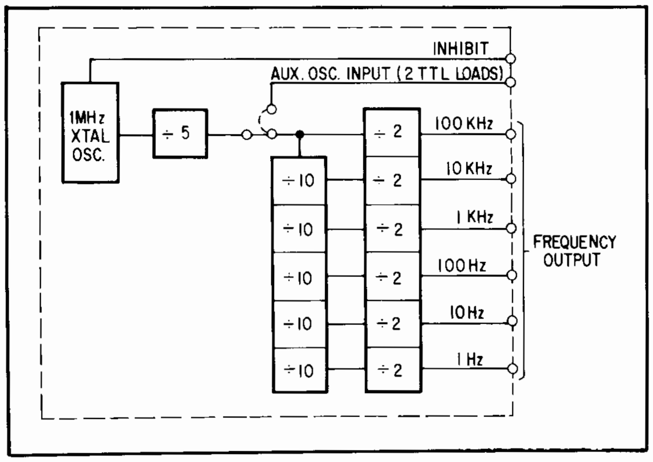
HP-IB Interface

```
0: rds(723)→C
1: oni 7,"done"
2: ent "ENTER PULSE LENGTH",A
3: if A<0 or A>40.95;gto "error"
4: fmt 1,c,z;fmt 2,c,f4.0,c,z
5: dto(A*100)→A
6: wrt 723.2,"00040TB",A,"T"
7: wrt 723.1,"00240TBT00460T";eir 7
8: for J=1 to 100000
9: next J
10: gto "end"
11: "error":dsp "PULSE OUT OF RANGE"
12: "end":end
13: "done":if rds(723)#64;gto "iret"
14: wrt 723.1,"00040TBT"
15: dsp "PULSE COMPLETE"
16: "iret":100000→J;iret
*13092
```

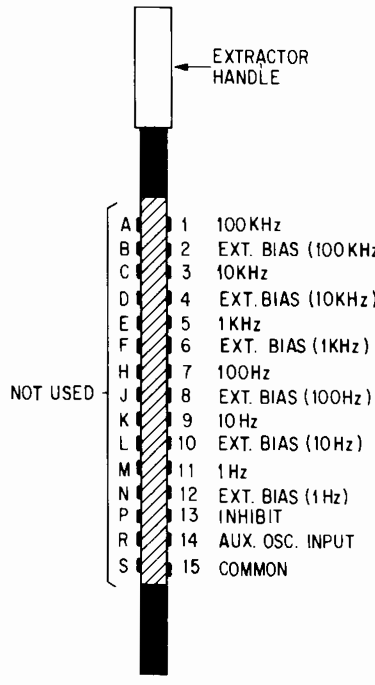
5-223 Frequency Reference Card, 69601B

5-224 This card is not programmable but can be used in conjunction with other plug-in cards. For example, the 69601B card can be used with pulse counter card 69435A for making time interval measurements (see Paragraph 5-238). In addition, the frequency reference card can be used as a divider for TTL signals through use of the auxiliary oscillator input.

5-225 The frequency reference card provides six square-wave outputs at fixed frequencies of 1Hz, 10Hz, 100Hz, 1kHz, 10kHz, and 100kHz. The output frequencies are derived from an internal 1MHz crystal oscillator and can be turned off by the low state of an external TTL logic gate or external contact closure between the inhibit and common pins on the card.



69601B Block Diagram



69601B Output Connector

5-226 MULTIPLE CARD PROGRAMS

5-227 The programs and explanations in this section provide examples of typical measurement applications and methods using several types of multiprogrammer cards together. The techniques shown in this section are merely examples of how to solve certain measurement problems and should not be considered as an exhaustive treatment of the subject. The user is encouraged to expand or change the methods shown here, to suit his particular requirements. The examples given below (1) voltage scanning, (2) frequency measurement and (3), time interval measurement), may be used as stand alone programs or modified and incorporated as subroutines in a larger program.

5-228 Voltage Scanning and Measurement Using the 69421A and 69433A

5-229 Example 46 illustrates using a 69421A voltage monitor card in conjunction with a 69433A relay output with readback card to measure six isolated voltages, store the measured values, and then print the values. If the voltages to be measured share a common reference, it is possible to use this card combination to measure twelve different voltages. Additional 69433A cards and appropriate modifications to the program may be used to expand the scanner array if desired.

5-230 Example 46 assumes that the relay card is installed in slot 4Ø1 (A) and the voltage monitor card is installed in slot 4Ø2 (B). It is further assumed that the voltages to be measured are in the range of −10.240 through +10.235 volts, permitting use of a standard voltage monitor card. The voltage measurement program described in Paragraphs 5-100 through 5-103 has been slightly modified (moct, mdec, and fmt 1 have been removed, since they are in the main program) and incorporated as a subroutine (i. e. gsb "meas") in this example.

5-231 As shown in Figure 5-1, the numbered edge connector pins on the relay card are connected in pairs across the voltages being measured (i. e. 1-2, 3-4, etc), with the even numbered pin connected to the positive side of the voltage to be sampled. Relay card pins, A, C, E, H, K, and M are jumpered together and connected to pin R of the voltage monitor card and pins B, D, F, J, L, and N are connected to pin L of the voltage monitor card.

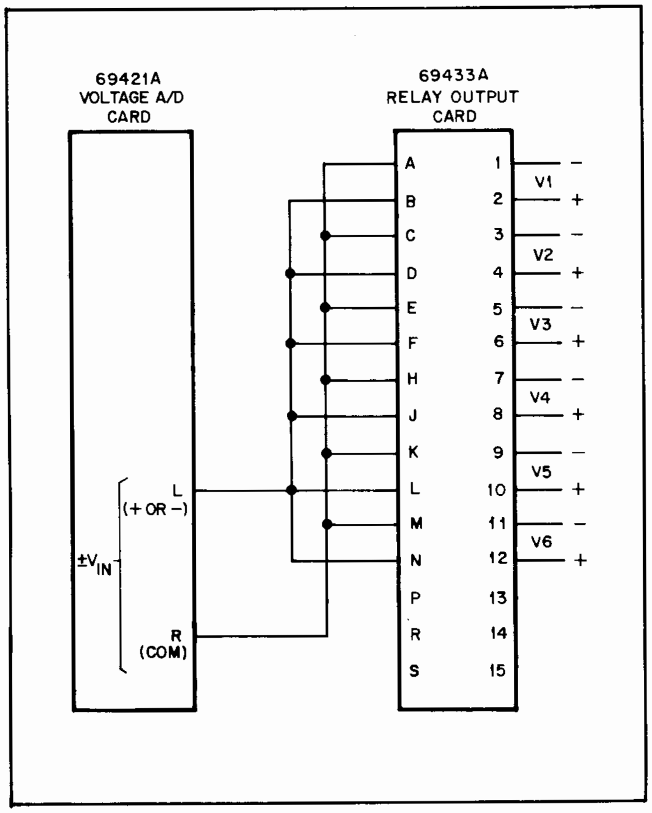


Figure 5-1. Voltage Scanning Connections

Example 46. Voltage Measurement Using 69421A and 69433A Cards

GPIO Interface

```
0: for J=0 to 10 by 2
1: dto(2↑J+2↑(J+1))→A
2: moct;wtb 9,170160,10000,10000+A
3: gsb "meas"
4: X→r((J+2)/2);next J
5: for J=1 to 6
6: fxd 3;prt "VOLTAGE =",rJ;next J
7: mdec;end
8: "meas":wtb 9,170260,20000,170240
9: wti 0,11;wti 4,20000;rdi 4→X
10: otdX→X;if X>2047;X-4096→X
11: .005X→X;ret
*21526
```

HP-IB Interface

```
0: rds(723)→C
1: fmt 1,c,z;fmt 2,c,f4.0,c,z
2: for J=0 to 10 by 2
3: dto(2↑J+2↑(J+1))→A
4: wrt 723.2,"00160TATA",A,"T"
5: gsb "meas"
6: X→r((J+2)/2);next J
7: for J=1 to 6
8: fxd 3;prt "VOLTAGE =",rJ;next J
9: end
10: "meas":wrt 723.1,"0260TBTBX"
11: rds(723)→C;red 723,X
12: otdX→X;if X>2047;X-4096→X
13: .005X→X;ret
*8478
```

Explanation:

GPIO

Ø: Establishes a "for/next" loop with line 4 in order to calculate and program the relay closures required to take the six voltage measurements.

1: Octal value representing appropriate relay closures is calculated and stored in variable A.

2: Control word with DTE, SYE, TME establishes output timing mode. Data word 1ØØØØ opens previously closed relays before programming next set of contact closures, thereby preventing possibility of shorting voltages together. Data word 1ØØØØ + A programs appropriate relay closures for each of six voltage measurements.

3: Calls voltage measurement subroutine (lines 8-11) which returns to line 4 with decimal value in volts stored in variable X.

4: Decimal value in volts is successively stored in variables r1 through r6 (corresponding to voltage being measured).

5-6: Another "for/next" loop is established to print the six voltage values stored in r1 through r6.

7: After the six voltage values are printed, the calculator is placed in the decimal mode and the program ends.

8-11: Voltage measurement subroutine; see explanation in paragraphs 5-100 through 5-103.

HP-IB

Ø: Clears service request line from multiprogrammer at beginning of program.

1: Format statements for write statements in lines 4 and 10.

2: Establishes a "for/next" loop with line 6 in order to calculate and program the relay closures required to take the six voltage measurements.

3: Octal value representing appropriate relay closures is calculated and stored in variable A.

4: Control word with DTE, SYE, TME establishes output timing mode. Data word (AT) opens previously closed relays before programming next set of contact closures, thereby preventing possibility of shorting voltages together. Second data word with variable A programs appropriate relay closures for each of six voltage measurements.

5: Calls voltage measurement subroutine (lines 10-13) which returns to line 6 with a decimal value in volts stored in variable X.

6: Decimal value in volts is successively stored in variables r1 through r6 (corresponding to voltage being measured).

7-8: Another "for/next" loop is established to print the six voltage values stored in r1 through r6.

9: End of program.

10-13: Voltage measurement subroutine; see explanation in paragraphs 5-100 through 5-103. Note that in line 11, the service request line which was set in line 10 (TME on) is cleared.

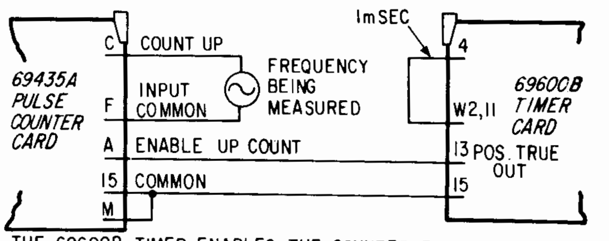
5-232 Frequency Measurement Using the 69435A and 69600B Cards

5-233 Example 47 illustrates using a 69435A pulse counter card and a 69600B programmable timer card to measure the frequency of a pulse train, and then display the frequency on the calculator.

5-234 Example 47 assumes the pulse counter card is installed in slot 4Ø1 (A) and the timer card is installed in slot 4Ø2 (B). The pulse train frequency being measured should not exceed 200kHz.

5-235 A primary consideration when measuring frequency with the technique described here is the capacity of the pulse counter card. The timer card output is connected to the count enable input of the counter card and is programmed to select the gate time for the frequency measurement. The user must select gate times so that the maximum count capacity of 4095 is not exceeded during the measurement period. This particular program automatically selects gate times as follows:

1. The initial gate time is 10 milliseconds, which produces 2000 counts at the maximum input frequency of 200kHz.



THE 69600B TIMER ENABLES THE COUNTER FOR A PROGRAMMED PERIOD OF TIME. THE CALCULATOR DIVIDES THE COUNT BY THE TIME TO DETERMINE FREQUENCY.

Figure 5-2. Frequency Measurement Connections

2. Whenever the total count is less than 100, the time interval is increased by a factor of ten and a new measurement is made. The maximum gate time in this program is one second.

5-236 The timer card will be used in the timing mode, requiring that jumper W3 be installed on the card. Instead of using the calculator interrupt system as an indication that the gate pulse is completed, however, the data word containing the pulse length is followed by a control word which will turn TME off and ISL on prior to reading data from the counter card. The calculator will not send the control word with ISL on until the output pulse is complete and the timer card returns a CTF flag. This technique of timing mode operation is described in Chapters 3 (GPIO) and 4 (HP-IB) Paragraphs 3-68 and 4-70, respectively.

5-237 Figure 5-2 shows the connections appropriate for this particular example. Timer card pins 11 (W2) and 4 should be jumpered together and pin 13 should be connected to pulse counter card pin A. Pulse counter card pins M and 15 should be jumpered together. The pulse train to be measured should be applied between the appropriate count-up input (as explained in the pulse counter card manual) and pin F (common) of the pulse counter card.

Example 47. Frequency Measurement Using 69435A and 69600B Cards

GPIO Interface

```
0: 10→B;1000→D
1: dtoB→A;moct;wtb 9,170040,10000
2: wtb 9,170160,20000+A,170240
3: wti 0,11;wti 4,10000;rdi 4→X
4: B*10→B;D/10→D
5: if X<144 and D>1;jmp -4
6: otdX*D→X
7: dsp "FREQUENCY=",X,"HZ"
8: mdec;end
*23573
```

HP-IB Interface

```
0: rds(723)→C
1: fmt 1,c,z;fmt 2,c,f4.0,c,z
2: 10→B;1000→D
3: dtoB→A;wrt 723.1,"00040TAT"
4: wrt 723.2,"00160TB",A,"T00240TAX"
5: red 723,X
6: B*10→B;D/10→D
7: if X<144 and D>1;jmp -4
8: otdX*D→X
9: dsp "FREQUENCY=",X,"HZ"
10: rds(723)→C;end
*36
```

Explanation

GPIO

0: Initial gate time of 10 milliseconds is stored in variable B; initial frequency conversion multiplier (1000), used to convert pulse count to Hz in line number 6, is stored in variable D.

1: Variable B is converted to octal and stored in variable A; calculator is placed in octal mode; control word with SYE establishes output mode; data word 10000 presents pulse counter card to zero.

2: Control word with DTE, SYE, and TME on establishes output timing mode; initial gate time is sent to the timer card; control word with ISL on will establish input mode upon completion of the gate pulse from the timer card.

3: Pulse counter card is read without a gate and the pulse count is stored in variable X.

4: Gate time is increased by a factor of 10 and frequency conversion multiplier is reduced by a factor of 10 to prepare for next reading.

5: The total pulse count is tested for 100 (144_8) and the frequency multiplier is tested for a value greater than 1 (i. e. 10 or 100); if the count is less than 100 and the frequency multiplier is greater than 1 (indicating the previous gate time was less than 1 second), the program returns to line 1 and a new measurement is made.

6-7: Pulse count is converted to a decimal value and multiplied by frequency multiplier to determine frequency in Hz; frequency is displayed.

8: Calculator is returned to decimal mode and program ends.

HP-IB

0: Clears service request line from multiprogrammer at beginning of program.

1: Format statements for write statements in lines 3 and 4.

2: Initial gate time of 10 milliseconds is stored in variable B; initial frequency conversion multiplier (10000), used to convert pulse count to Hz in line number 8, is stored in variable D.

3: Variable B is converted to octal and stored in variable A; control word with SYE establishes output mode; data word "AT" presets pulse counter card to zero.

4: Control word with DTE, SYE, and TME on establishes output timing mode; initial gate time is sent to the timer card; control word with ISL on will establish input mode upon completion of the gate pulse from the timer card.

5: Pulse counter card is read without a gate and the pulse count is stored in variable X.

6: Gate time is increased by a factor of 1Ø and frequency conversion multiplier is reduced by a factor of 1Ø to prepare for next reading.

7: The total pulse count is tested for 1ØØ (144_8) and the frequency multiplier is tested for a value greater than 1 (i. e. 1Ø or 1ØØ); if the count is less than 1ØØ and the frequency multiplier is greater than 1 (indicating the previous gate time was less than 1 second), the program returns to line 3 and a new measurement is made.

8-9: Pulse count is converted to a decimal value and multiplied by frequency multiplier to determine frequency in Hz; frequency is displayed.

10: Service request line from multiprogrammer is cleared; program ends.

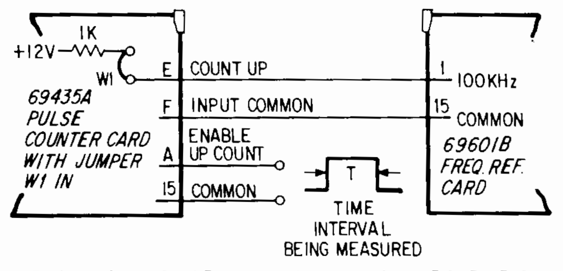
5-238 Time Interval Measurement Using the 69435A and 69601B Cards

5-239 The following program illustrates using a 69435A pulse counter card and a 69601B frequency reference card to measure the period of a single positive pulse, and then display the period (or time interval) on the calculator.

5-240 Example 48 assumes that the pulse counter card is installed in slot 4Ø1 (A) and the frequency reference card is installed in slot 4Ø2 (B). For the connections and calculations used in this particular example, the pulse duration should not exceed 40 milliseconds.

5-241 A primary consideration when measuring time intervals with the technique described here is the capacity of the pulse counter card. The user must select a reference frequency so that the maximum count capacity of 4095 is not exceeded during the longest time interval measurement. Capability varies inversely with the reference frequency. For example, selection of a 1kHz reference frequency would permit measurement of time intervals with durations to approximately 4 seconds. Lower reference frequencies reduce the resolution of the measurement, however, so that some compromise is necessary either way. Longer intervals may of course be measured at high resolution by cascading counter cards in order to eliminate the overflow problem.

5-242 Figure 5-3 shows the connections appropriate for the particular example described. The frequency reference card edge connector pins 1 and 15 should be connected to pulse counter card pins E and F respectively. Pulse counter card pins M and 15 should be jumpered together and the pulse to be measured should be applied between pins A and 15 of the pulse counter card. When the program in example 64 is run it will stop with "ENTER PULSE" displayed on the calculator. The user may now apply a positive pulse up to 40 milliseconds long to the pulse counter card. Depressing CONTINUE on the calculator will cause the calculator to display the time interval of the pulse.



THE CALCULATOR DIVIDES THE TOTAL COUNT BY THE FREQUENCY OUTPUT OF THE 69601B TO DETERMINE THE TIME INTERVAL

Figure 5-3. Time Interval Measurement Connections

Example 48. Time Interval Measurement Using 69435A and 69601B Cards

GPIO Interface

```
0: moct;wtb 9,170040,10000
1: dsp "ENTER PULSE";stp
2: wtb 9,170240
3: wti 0,11;wti 4,10000;rdi 4→X
4: otdX→X;X/100000→X;fxd 6
5: dsp "PULSE TIME =",X,"SECONDS"
6: mdec;end
*12429
```

HP-IB Interface

```
0: rds(723)→C
1: fmt 1,c,z;wrt 723.1,"00040TAT"
2: dsp "ENTER PULSE";stp
3: wrt 723.1,"00240TAX";red 723,X
4: otdX→X;X/100000→X;fxd 6
5: dsp "PULSE TIME =",X,"SECONDS"
6: end
*6801
```

Explanation

GPIO

0: Calculator is placed in octal mode. Control word with SYE on establishes output mode. Data word 10000 presets the pulse counter card to zero.

1: After calculator stops, pulse is applied to the enable up count input of pulse counter card.

2: Control word with ISL establishes input mode.

3: Pulse counter card is read without a gate and the pulse count is stored in variable X.

4: Pulse count is converted to decimal value and divided by reference frequency to obtain time interval in seconds.

5: Pulse time interval is displayed.

6: Calculator is returned to decimal mode and program ends.

HP-IB

0: Clears service request line from multiprogrammer at beginning of program.

1: Control word with SYE on establishes output mode. Data word "AT" presets the pulse counter card to zero.

2: After calculator stops, pulse is applied to the enable up count input of pulse counter card.

3: Control word with ISL establishes input mode. Pulse counter card is read without a gate and the pulse count is stored in variable X.

4: Pulse count is converted to decimal value and divided by reference frequency to obtain time interval in seconds.

5: Pulse time interval is displayed.

6: Program ends.

Chapter VI
OTHER PROGRAMMING TECHNIQUES

6-1 The programming techniques described in Chapters III, IV, and V will satisfy the vast majority of multiprogrammer programming needs. This chapter provides additional techniques which may prove to be helpful for certain multiprogrammer applications. The additional techniques covered in this chapter are:

1. String variables (using 98034A HP-IB interface card).
2. Equating names with device codes.
3. Call statements.
4. Buffered I/O commands.

6-2 STRING VARIABLES

6-3 General

6-4 Before using string variables with the multiprogrammer system, the user should read and understand the String Variable Programming Manual (HP Part No. 09825-90020). In general, for multiprogrammer applications, string variables are useful only with the 98034A HP-IB interface card. Before attempting to use string variables check that the calculator is equipped with the String Variables ROM.

6-5 Programming Examples

6-6 In a system that requires a variety of setup conditions based on factors such as which device is being tested, string variables are useful. For example, a system used to test digital I.C.'s and linear I. C.'s would need 5 volts bias for the digital I. C.'s and ±15 volts for the linear I. C.'s. A program can be written such that the setup conditions for the digital I. C.'s are stored in variable D$, while those for linear I. C.'s are stored in L$. Program example 1 illustrates such a program. This example assumes that 69501A resistance cards are used to program a 6227B dual output power supply to 5 volts and 0 volts for digital I. C.'s and to ±15 volts for linear I. C.'s.

6-7 String variables can be used for frequently used data strings to save programming typing time and reduce program storage requirements. For example, if data string "ØØ14ØT DT A1Ø15T BT CØ1ØØT ØØ16ØT DØØ14T CØ21ØT" is used 10 times in a program. a string variable can be used to save considerable memory space. This string could be used to change the bias voltage for a power supply such as the one used in example 1. The data string could also be used to program a 69600B Programmable Timer card for a delay (CØ1ØØT) of sufficient duration to permit the power supply to reach steady state. Relays on a 69433A Relay Readback card could then be programmed closed (DØØ14T) and another delay (CØ21ØT) programmed to permit the device under test to reach steady state before a measurement is made.

6-8 String variables can also be used when reading input cards. Input cards are often read one after the other, in sequence. The reading sequence does not have to be orderly (e. g. card "A", card "B", etc.); it can be random. In fact, the user can change the sequence as required. Example 2 shows how input cards with card addresses A, C, J, K can be randomly sequenced.

6-9 Another use of string variables could be in updating output cards. Example 3 illustrates how output cards with card addresses A, B, C, and D are updated. Line Ø dimensions and loads string variable A$ with the desired card addresses and data transfer field codes (T in this case). Line 2 contains the required format statements. The output data (variable A) can be the result of computations based on input data taken in lines 3 through 13. Line 15 contains the output statement.

Example 1. Using String Variables for Test Setup Conditions

```
0: dim D$[18],L$[18],B$[1],A$[18]
1: "00040TA0764TBT"→D$
2: "00040TA2734TB2734T"→L$
3: ent "digital=D;linear=L",B$
4: if cap(B$)="D";D$→A$;jmp 2
5: L$→A$
6: wrt 723,A$
*14321
```

Explanation:

Ø:	Dimensions the necessary string variables.
1, 2:	Places the required data into D$ and L$.
3:	Asks the user to decide which unit will be tested.
4:	Tests B$ to determine which string (D$ or L$) is to be output.
5:	Output string A$ is made equal to L$ for linear I. C. testing.
6:	Outputs setup conditions.

Example 2. Using String Variables to Randomly Sequence Input Cards

```
0: dim A$[8],X[8]
1: "JXAXCXKX"→A$;fmt 1,c,z
2: wrt 723.1,"00260TATCTJTKT00240T";rds(723)→C
        •
        •
        •
10: for I=1 to 7 by 2
11: wrt 723.1,A$[I,I+1];red 723,X[I]
12: next I
```

Explanation:

Ø: Dimensions string variables and array variable.

1: Assigns the reading sequence for the cards and stores data string in A$.

2: Turns on the TME, then sends a gate to each input card (such as 69421A Voltage Monitor Cards). Each card must return a CTF flag before next card is gated. Clear service request.

10-12: Addresses cards in specified order and stores data in array X [1], [3], [5], [7].

Example 3. Using String Variables to Update Output Cards

```
0: dim A$[8];"ATBTCTDT"→A$
1: for I=1 to 7 by 2
2: fmt 1,c,z;fmt 2,c,f4.0,c,z
        •
        •
        •
14: wrt 723.1,"00140T"
15: wrt 723.2,A$[I,I],A,A$[I+1,I+1]
16: next I
```

6-10 EQUATING NAMES WITH DEVICE CODES

6-11 General

6-12 Two statements available with the Extended I/O ROM can be used to make programs more "readable". These statements are the device statement (dev) and the equate statement (equ). A full description of the required syntax is given in the Extended I/O Programming ROM manual (HP Part No. 09825-90025).

6-13 Device Statement

6-14 Device statements (dev) can use descriptive alpha labels in place of numeric select codes to identify devices. This allows the user to quickly identify which device is being addressed. Example 4 shows how to use the device statement and includes sample programs for both the GPIO and HP-IB interfaces. Line Ø uses the device statement (dev) to identify the "Multiprogrammer". Line 2 clearly indicates that the multiprogrammer is the device being addressed. To simplify the typing, the alpha label "Multiprogrammer" could be shortened to "MP".

6-15 Equate Statement

6-16 Inclusion of the Equate Statement (equ) increases program "readability" even further. Example 5 illustrates both dev and equ statements for an HP-IB interface program. In line 2, the Multiprogrammer "MP" is addressed with format 1 "MP.1" and given a data string "INPUT" that places the Multiprogrammer in the input mode as well as addressing the card in slot E.

6-17 The dev and equ statements not only improve readability of a program, but they also cut down on the time necessary to enter a program. In addition, updating a program can be accomplished by changing one dev or equ statement. For example, the program (Example 5) can be changed to read the input card in the timing mode (ISL and TME on) by changing line Ø to read as follows:

Ø: dev "MP", 723; equ "INPUT", "ØØ26ØTET"

Example 4. Using Device Statement

GPIO Interface

```
0: dev "Multiprogrammer",9
1: moct;fmt 3,2b,z
2: wrt "Multiprogrammer.3",170240,50000
```

HP-IB Interface

```
0: dev "Multiprogrammer",723
1: fmt 1,c,z
2: wrt "Multiprogrammer.1","00240TET"
```

Example 5. Using Device (dev) and Equate (equ) Statements

```
0: dev "MP",723;equ "INPUT","00240TET"
1: fmt 1,c,z
2: cmd "MP","INPUT"
*19132
```

6-18 CALL STATEMENT

6-19 General

6-20 Programming multiple cards of the same model number (e. g. six each 69335A stepping motor control cards) is made easier by using subroutines or call (cll) statements. Call statements are preferred, because parameters can be passed to and from the main program. Full syntax and programming details are given in the Advanced Programming ROM manual (HP Part No. 09825-90021).

6-21 Programming Examples

6-22 Call statements can be used to pass the following parameters: card slot addresses, control mode bits, and output data. Example 6 shows how to use call statements and includes sample programs for both the GPIO and HP-IB interfaces.

6-23 In line 1, the user indicates how many cards will be addressed via the call statements. Lines 2 through 7 set up a loop that permits the user to enter those parameters that are passed to the call statement subroutine. Lines 8 and 9 are written by the user to satisfy his particular needs. Line 10 is the first time that the call statement is used. Note that the data passed is that previously entered for card one. Lines 13, 14, and 15 are used to call additional cards. These cards are called randomly. The actual call subroutine starts on line 2Ø.

6-24 In the GPIO program, line 21 first sends a control word (17ØØ4Ø) with TME off. Next, the data (p3) is sent with the appropriate card address (1ØØØØ * dto p1). Finally, a control word with or without TME (17ØØØØ + p2) is sent. In this example, p1 = A [ ], p2 = T [ ], and p3 = D [ ].

6-25 In the HP-IB program, line 21 defines B$ as a control word (ØØØ4ØT) with TME off. Next the character that represents the numeric card slot address is generated and stored in B$. Use of the character statement (char) is required, since call statements cannot pass string variables or alpha characters. Line 22 outputs B$ (which sends both a control word and the alpha card address), data (p3) as well as the alpha "T" (B$ [6,6]. Line 23 converts the TME code stored in T [I, I] (or p2) into the appropriate ASCII character and stores it in B$ [4,4]. The appropriate control word (TME on/off) is then sent as B$ [1,6].

Example 6. Using Call (cll) Statements

GPIO Interface

```
0: dim A[15],T[15],D[15];fxd 0
1: ent "No. of 69335A cards",N
2: for I=1 to N
3: ent "Card address",A[I]
4: ent "TME:on=60;off=40",T[I]
5: dsp "Data for card",I;wait 1000;ent D[I]
6: fmt c7,f4.0;wrt 16,"Address",A[I];wrt 16,"TME",T[I]
7: wrt 16,"Data",D[I];spc 2;next I
8:          •
9:          •
10: cll '69335A'(A[1],T[1],D[1])
11:         •
12:         •
13: cll '69335A'(A[3],T[3],D[3])
14: cll '69335A'(A[2],T[2],D[2])
15: cll '69335A'(A[4],T[4],D[4])
16:         •
17:         •
18:         •
19: end
20: "69335A":noct
21: wtb 9,170040,p3+10000*dtop1,170000+p2
22: mdec;ret p2
```

HP-IB Interface

```
0: dim A[15],T[15],D[15],B$[15];fxd 0
1: ent "No. of 69335A cards",N;fxd 0
2: for I=1 to N;prt "Card",I
3: ent "Card address",A[I]
4: ent "TME:on=60;off=40",T[I]
5: dsp "Data for card",I;wait 1000;ent D[I]
6: fmt c7,f4.0;wrt 16,"Address",A[I];wrt 16,"TME",T[I]
7: wrt 16,"Data",D[I];spc 2;next I
8:          •
9:          •
10: cll '69335A'(A[1],T[1],D[1])
11:         •
12:         •
13: cll '69335A'(A[3],T[3],D[3])
14: cll '69335A'(A[2],T[2],D[2])
15: cll '69335A'(A[4],T[4],D[4])
16:         •
17:         •
18:         •
19: end
20: "69335A":fmt 1,c,z;fmt 2,c,f4.0,c,z
21: "00040T"→B$;char(p1+64)→B$[7,7]
22: wrt 723.2,B$,p3,B$[6,6]
23: char(p2/10+48)→B$[4,4];wrt 723.1,B$[1,6];ret p2
```

6-26 BUFFERED I/O

6-27 General

6-28 Multiprogrammer I/O speeds (up to 20,000 words/sec) are typical of instrumentation speeds (500 to 50,000 data transfers/sec). Normal read/write statements will efficiently handle these data transfers. However, if large blocks of data must be transferred quickly, buffered I/O should be used. Total program running time with buffered I/O will be longer because the buffer must be filled or emptied before/after it is used. Nevertheless, the actual data transfer between the buffer and the multiprogrammer will be faster. A full description of buffered I/O syntax is given in the Extended I/O Programming manual (HP Part No. 09825-90025).

6-29 For the GPIO/multiprogrammer interface, buffered I/O operations are useful for both output and input data transfers. However, buffered I/O operations on the HP-IB with the multiprogrammer are basically limited to output data transfers. Input data transfers on the HP-IB (except for the single reading buffered I/O case) are not possible because the buffer transfer only occurs between the 59500A multiprogrammer interface unit and the calculator buffer. Use of the buffer transfer on the HP-IB does not cause data to be transferred from the appropriate multiprogrammer plug-in card. Table 6-1 lists the buffer types which can be used with the multiprogrammer in input/output transfers (GPIO interface) or output transfers (HP-IB interface).

6-30 Output Data Transfers

6-31 Example 7 illustrates using buffered I/O to transfer data to the multiprogrammer. Sample programs are included for both the GPIO and HP-IB interfaces. In example 7, line Ø designates the interrupt buffer. For this case, the buffer name is "out". Note that the buffer used with the HP-IB is six times larger than that used with the GPIO. With HP-IB, six characters must be stored for each output data word, while with GPIO one entire 16-bit word is stored. Line 1 contains the necessary formatting information used in lines 2 and 4. Line 2 outputs a control word and line 4 loads the buffer. Lines 3, 4, and 5 form a loop that fills the buffer. The time used to fill the buffer is exactly the same amount of time that would be required to output to the multiprogrammer without a buffer. Because of this "overhead time" (i. e. time to fill the buffer), buffers are not advantageous when a continuous stream of data must take place. The actual transfer of data is accomplished by line 6. While the data is being transferred, lines 7 and 8 are executed. When the transfer is completed, lines 9 and 10 are executed. Note that line 10 is used to ensure that the buffer is cleared.

6-32 To transfer output data using a fast read/write buffer, only line Ø in example 7 must be changed as follows:

For GPIO change line Ø to:

```
Ø buf "out" , 1ØØØ,2
```

For HP-IB, change line Ø to:

```
Ø buf "out" , 6ØØØ,3
```

6-33 Operation with fast read/write would be the same as with an interrupt buffer until line 6. If the multiprogrammer card is slow (e. g. a relay output card), line 7 may be executed. However, once the data transfer has begun, lines 7 and 8 will not be executed until the transfer is completed. Fast multiprogrammer cards (e. g. those that return a flag in microseconds, like the 69321B) will keep lines 7 and 8 from being executed until all data has been transferred.

6-34 Input Data Transfers

6-35 Data input with buffered I/O is only feasible with GPIO for the reason previously given (see Paragraph 6-29). Example 8 illustrates data input (GPIO interface) using an interrupt buffer. The buffer is defined in line Ø. Line 1 gives the necessary formats, while line 2 sends a

Table 6-1. I/O Buffer Types

Buffer Type	Buffer No.	GPIO Data Format	Data Transfer	Buffer No.	HP-IB Data Format	Data Transfer
Interrupt	Ø	words	Input/Output	1	bytes	Output
Fast Read/Write	2	words	Input/Output	3	bytes	Output
DMA *	4	words	Input/Output	—	—	—

* With jumper 7 added to the configuration board of the 98032A Option 040 interface card.

control word with ISL, SYE, and TME on (17Ø26Ø) and addresses (without a gate) the appropriate card (wti Ø, 11; wti 4,5ØØØ). Data is transferred into the buffer in line 3, while line 4 is executed. Buffer status, rds ("in"), will be −1 while the buffer is busy. When the buffer is finished, the buffer status will be 1ØØØ at which time lines 5 and 6 are executed.

6-36 To transfer input data using a fast read/write, only line Ø in example 8 must be changed as follows:

Ø buf "in", 1ØØØ,2

6-37 Operation would be similar to that in example 8. However, using fast read/write, line 4 would not be executed until after the data transfer of line 2, unless the multiprogrammer input card was very slow.

6-38 Use of DMA Buffer

6-39 The DMA (direct memory access) buffer can be used with the multiprogrammer if jumper 7 is added to the configuration board of the 98032A/Option 040 interface card. Discretion should be used in assigning the DMA channel. If DMA is assigned to the 98032A/040 for the multiprogrammer, tape drive or disk operations are not possible during DMA data transfers to the multiprogrammer.

6-40 Example 7 can be easily changed to provide DMA output on GPIO. All that need be done is to change line Ø as follows: Ø: buf "out" , 1ØØØ,4

6-41 Operation will appear to be identical to that when an interrupt buffer is used. However, the data transfer will be as fast as with the fast read/write buffer because of "cycle stealing". DMA will permit data transfer to occur while performing other tasks.

6-42 DMA input can be accomplished by changing line Ø in example 8 to "Ø: buf "in", 1ØØØ,4". Operation of DMA for input offers the same benefits as DMA for output.

Example 7. Data Output Using an Interrupt Buffer

GPIO Interface

```
0: buf "out",1000,0
1: moct;fmt 3,b,z
2: wtb 9,170160
3: for N=1 to 1000
4: wtb "out.3",dtoN+50000
5: next N
6: tfr "out",9,1000
7: for N=1 to 100;dsp N;1+A→A
8: next N;if rds("out")#0;jmp -1
9: dsp "READY.   A=",A;beep
10: buf "out"
*11668
```

HP-IB Interface

```
0: buf "out",6000,1
1: fmt 1,c,z;fmt 4,"E",f4.0,"T"
2: wrt 723.1,"00160T"
3: for N=1 to 1000
4: wrt "out.4",dtoN
5: next N
6: tfr "out",723,6000
7: for N=1 to 100;dsp N;1+A→A
8: next N;if rds("out")#0;jmp -1
9: dsp "READY.   A=",A;beep
10: buf "out"
*4989
```

Example 8. Data Input Using an Interrupt Buffer (GPIO Interface Only)

```
0: buf "in",1000,0
1: moct;fmt 3,b,z
2: wtb 9,170260;wti 0,11;wti 4,50000
3: tfr 9.3,"in",1000
4: if rds("in")#1000;for N=1 to 100;dsp N;1+A→A;next N;jmp rds("in")=0
5: dsp "READY   A=",A;beep
6: buf "in"
*29072
```

Chapter VII
MULTIPROGRAMMER 6940B/6941B DESCRIPTION

7-1 INTRODUCTION

7-2 In the most general sense, the 6940B/6941B multiprogrammer units function as a multi-channel bi-directional interface between a controller and the "real world" environment to which command signals are sent, and from which status signals are received. For the purposes of this discussion, the controller can be a 9825A calculator equipped with a GPIO or an HP-IB interface card. The GPIO interface provides a sixteen bit word directly to the 6940B. The HP-IB interface provides an eight bit ASCII coded output which must be converted to the sixteen bit format required by the 6940B. Thus, when operating on the HP-IB, a 59500A interface unit must be used to convert the ASCII code to the 6940B format. Operation of the 59500A multiprogrammer interface unit is described in Chapter VIII.

7-3 Communications between the multiprogrammer and the external environment is realized via plug-in input/output (I/O) cards, of which more than thirty different types now exist. Data transfers between the multiprogrammer mainframe and the I/O cards are digital (twelve data bits); transfers between the cards and external devices can be either digital (again, twelve data bits) or analog (voltages, currents, etc.). Cards with analog I/O capability have on-card converters that permit them to transmit data to and from the mainframe in digital form.

7-4 Slot receptacles for fifteen cards are provided in both the 6940B and 6941B mainframes. As many as sixteen mainframes (a 6940B master and up to fifteen 6941B slaves) may be chained together so as to provide a maximum possible total of 240 I/O channels, any one of which may be selectively accessed by the controller.

7-5 Data Transfers

7-6 Data transfers between the 6940B and the controller take the form of a sixteen bit output word (from the controller to the multiprogrammer) and a thirteen bit (twelve data bits plus one status bit) return data word (from the multiprogrammer to the controller). From the standpoint of the I/O card-mainframe interface all data transfers are twelve bit; the controller, however, is required to output a sixteen bit word in order to bring about the twelve bit communication. This reduction in word length provides the mechanism by which the multiprogrammer functions as a multiplexer, effectively expanding a single duplex I/O channel into many bi-directional I/O channels. The cards themselves generally act in one capacity (input or output) to the exclusion of the other, but either card type may be used in any I/O channel.

7-7 Data Transfer Synchronization

7-8 Two additional signal lines ($\overline{\text{GTE}}$ and $\overline{\text{FLA}}$) facilitate a "handshake", permitting synchronization of data transfers. The gate signal ($\overline{\text{GTE}}$) is sent by the controller to indicate that it has valid data for the multiprogrammer to accept, while the flag signal ($\overline{\text{FLA}}$) permits the multiprogrammer to indicate that it (1) is processing data; and (2) has completed processing. Figure 7-1 shows the sequence of a handshake cycle. A data transfer cycle is not finished unless the handshake is also completed, in which case the controller may begin a new transaction. In some operating modes, completion of the handshake may take considerable time, particularly when compared to the period required by the controller to execute a program step. If this situation is known to exist, the controller may be directed to proceed with other portions of the program (data reduction, communication with other peripherals, etc.), returning periodically to test the progress of the handshake. Once a transfer cycle is begun by means of the gate signal, failure of the system to provide both edges of flag will always create a "hang-up" situation in which the controller will be unable to initiate new communications with the multiprogrammer unless the handshake is terminated by special programming commands.

7-9 Handshake capability may be extended through the multiprogrammer mainframe to the I/O card in certain modes of operation thereby altering the period (but not the sequence) of the handshake to suit the timing requirements of the particular card. In addition some cards permit the handshake to be extended to the external device in order to ensure that data transfers are accurate and complete. In either case, the mainframe no longer automatically returns the flag signal, but instead relinquishes control to the card and/or the external device. Figure 7-2 shows an example of this technique.

7-10 Extender Units

7-11 Any number (none at all or up to fifteen total) of 6941B extender units may be included in a system as required. It is not possible however, to use a 6941B without

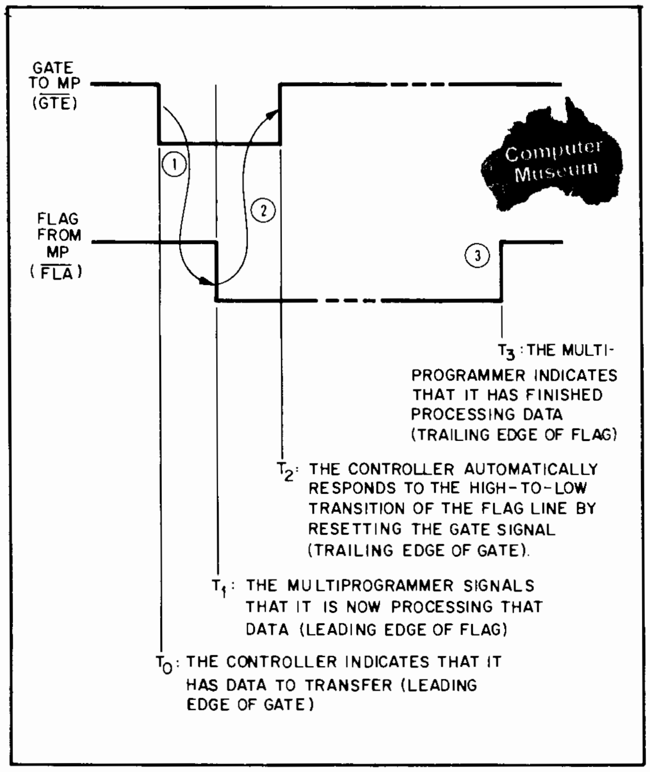
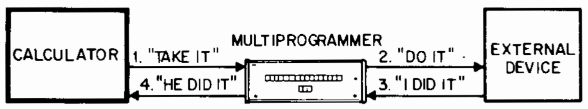

GATE TO MP ($\overline{\text{GTE}}$)
FLAG FROM MP ($\overline{\text{FLA}}$)
1
2
3
T_3: THE MULTIPROGRAMMER INDICATES THAT IT HAS FINISHED PROCESSING DATA (TRAILING EDGE OF FLAG)
T_2: THE CONTROLLER AUTOMATICALLY RESPONDS TO THE HIGH-TO-LOW TRANSITION OF THE FLAG LINE BY RESETTING THE GATE SIGNAL (TRAILING EDGE OF GATE).
T_1: THE MULTIPROGRAMMER SIGNALS THAT IT IS NOW PROCESSING THAT DATA (LEADING EDGE OF FLAG)
T_0: THE CONTROLLER INDICATES THAT IT HAS DATA TO TRANSFER (LEADING EDGE OF GATE)

Figure 7-1. Handshake Sequence



THE CONTROLLER (CALCULATOR) USES ITS GATE TO TELL THE MULTIPROGRAMMER TO TAKE DATA; THE MULTIPROGRAMMER USES A GATE ON ONE OF ITS PLUG-INS TO TELL AN EXTERNAL INSTRUMENT SUCH AS A PRINTER OR DIGITALLY CONTROLLED VALVE TO TAKE THE DATA; WHEN THE EXTERNAL DEVICE IS DONE, IT USES THE FLAG ON THE PLUG-IN CARD TO TELL THE MULTIPROGRAMMER IT IS FINISHED, AND THE MULTIPROGRAMMER TELLS THE CALCULATOR THE OPERATION IS COMPLETE.

Figure 7-2. Extended Handshake

a 6940B in the system. In addition, it is necessary that the 6940B be the first unit in the chain. A cabling diagram showing how a 6940B and one or more 6941B's are chained together in a system is provided in Chapter II.

7-12 Slot and Unit Addresses

7-13 Whether or not 6941B extender units are used each card slot automatically assumes a two part address which uniquely specifies its location within the system. A card slot within either type mainframe has a fixed address (slot ØØ through slot 14) and may be accessed by means of a four bit binary code ($\overline{\text{B12}}$ through $\overline{\text{B15}}$) as shown in Table 7-1. The sixteenth address (Control Word) accesses circuitry which is present in the 6940B mainframe only, and permits the controller to specify a unit select address, thereby providing the second part of the total address needed to access a particular I/O channel.

7-14 Like the slot address, the unit select address is specified by a four bit binary code, ($\overline{\text{BØØ}}$ through $\overline{\text{BØ3}}$), but in this case all sixteen possible combinations are interpreted as unit addresses. The address of a particular unit is automatically established by its position in the chain; the first 6941B becomes unit Ø1, the second becomes unit Ø2, etc. The 6940B mainframe is always unit ØØ and is specified when $\overline{\text{BØØ}}$ through $\overline{\text{BØ3}}$ are logical "0's".

7-15 Once a data transfer cycle with a control word is completed, both the mode of operation and the unit select address have been stored by the 6940B mainframe, and subsequent operations will be directed towards the particular unit addressed. Slot addresses, designated in succeeding words output by the controller, cannot access card slots except in the previously selected unit unless a new control word is sent specifying a different unit address. Similarly, if it is desired that the operating mode be changed but not the unit address, the new control word must re-specify the particular unit address.

7-16 BLOCK DIAGRAM DESCRIPTION

7-17 At this point it is convenient to introduce both the functional block diagram as shown in Figure 7-3, and the concept of word format. Figure 7-3 shows the 6940B mainframe logic, the controller, a typical I/O card, and the significant connections between these segments of the total system. Although several signal lines between the mainframe and the I/O card have been omitted in order to simplify the block diagram, it remains faithful to the functional mechanisms of data transfer and it may be assumed that the card could be located in any I/O channel in the system. The signal lines shown between the multiprogrammer and the controller correspond exactly to the hardware with the single exception that the SYE (system enable) jumper connection is not included. This feature serves only to provide an interlock on the SYE control line such that a cable must be connected between the multiprogrammer and the controller before SYE can be sent to the cards. Certain cards are de-activated to a "safe" state in the absense of the SYE signal.

7-18 Figure 7-4 shows the format, or structure, of a data word. The data word is the most general of three formats used for the sixteen bit output word sent from the controller to the multiprogrammer. $\overline{\text{B12}}$ through $\overline{\text{B15}}$ specify the card slot address, while the remaining twelve bits are used to transfer data. Note that the hardware organization as shown in the block diagram complements

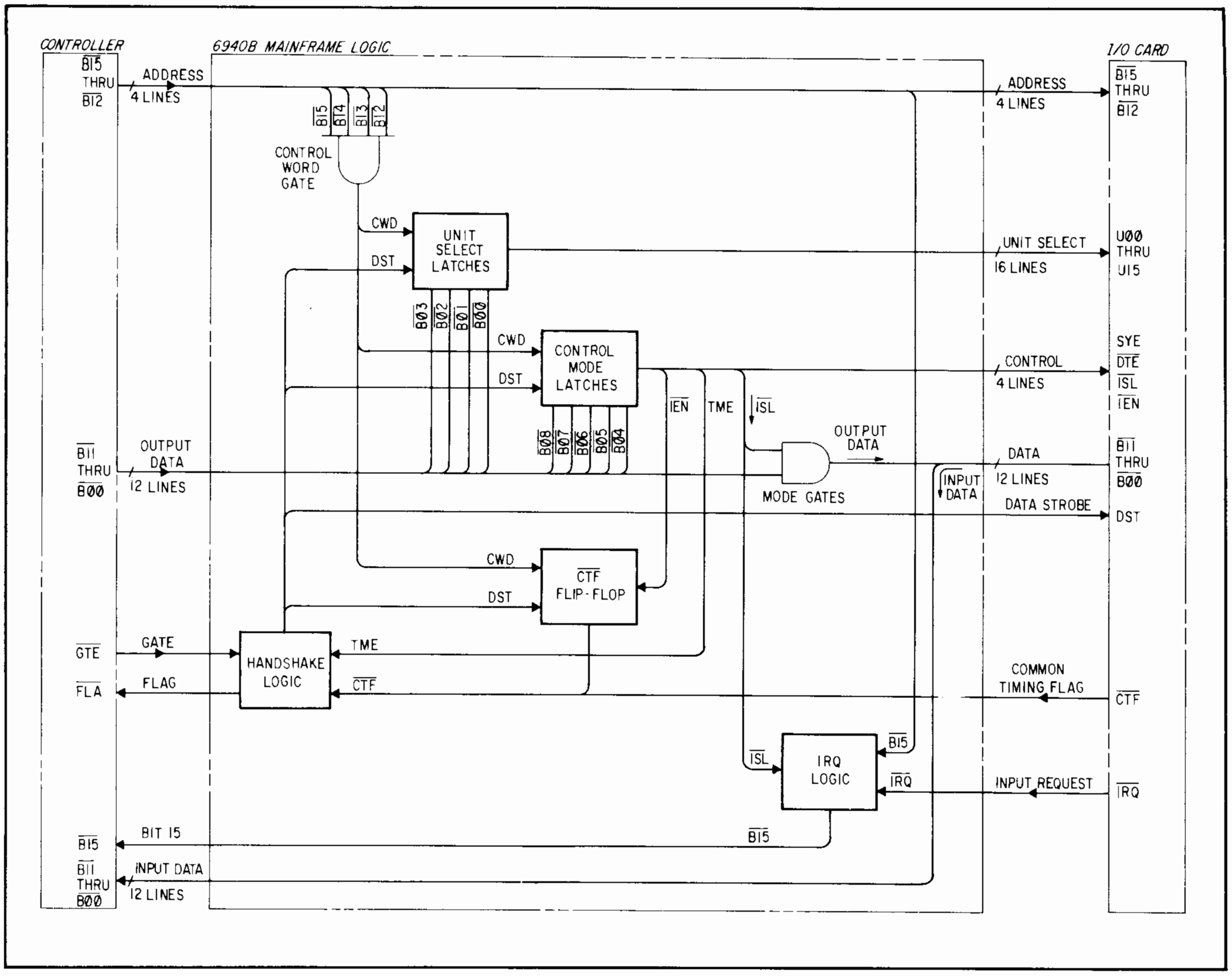


Figure 7-3. Multiprogrammer 6940B, Block Diagram

Table 7-1. Slot Address Codes

ADDRESS	$\overline{B15}$	$\overline{B14}$	$\overline{B13}$	$\overline{B12}$
Slot ØØ	Ø	Ø	Ø	Ø
Slot Ø1	Ø	Ø	Ø	1
Slot Ø2	Ø	Ø	1	Ø
Slot Ø3	Ø	Ø	1	1
Slot Ø4	Ø	1	Ø	Ø
Slot Ø5	Ø	1	Ø	1
Slot Ø6	Ø	1	1	Ø
Slot Ø7	Ø	1	1	1
Slot Ø8	1	Ø	Ø	Ø
Slot Ø9	1	Ø	Ø	1
Slot 10	1	Ø	1	Ø
Slot 11	1	Ø	1	1
Slot 12	1	1	Ø	Ø
Slot 13	1	1	Ø	1
Slot 14	1	1	1	Ø
Control Word	1	1	1	1

the format of the data word; that is, the address and data portions of the word are processed separately by the mainframe logic.

7-19 As shown in Figure 7-3 the data signal lines are distributed to the unit select latches ($\overline{BØØ}$ through $\overline{BØ3}$), the control mode latches ($\overline{BØ4}$ through $\overline{BØ8}$), and the mode gates (all twelve bits) which, depending on the mode of operation, either terminate the signals or transmit them in parallel to every card slot in the system. Data signals sent to the unit select and control mode latches are ignored except when a control word is programmed and detected by the control word gate. As was previously explained, a control word is specified by the presence of logical "1's" on all four address lines. Therefore, a control word is a special case of the data word and has the format shown in Figure 7-5.

7-20 The address signal lines are sent to the control word gate and to each of the I/O card slots where a

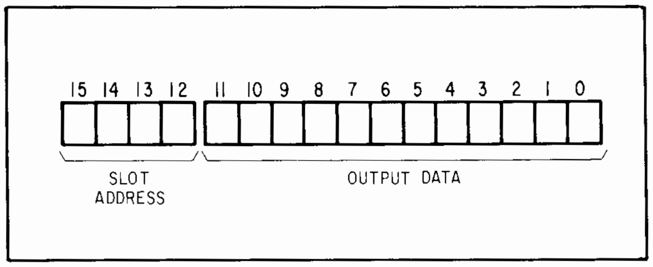


Figure 7-4. Data Word Format

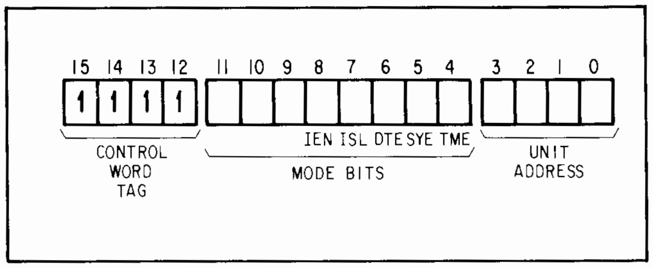


Figure 7-5. Control Word Format

combination of hardwired logic in the mainframe and address decode gates (similar to the control word gate) on the cards permit the addressed card, and only the addressed card, to be partially enabled. A card is completely enabled when, (1) the card slot address sent by the controller matches the address of the slot in which the card is located, (2) the unit address is similarly correct, and (3) the appropriate control signals are received by the card. One additional signal (data strobe or DST) is required whenever data is transferred from the controller to the multiprogrammer. Depending on the card model and the system operating mode, the start of the data strobe initiates one of several possible responses to data present on lines $\overline{\text{B00}}$ through $\overline{\text{B11}}$.

7-21 Basic Data Transfer Cycle

7-22 Data strobe is a function of the gate signal sent by the controller and is developed by the handshake logic. Referring back to Figure 7-1, the most basic form of the data transfer cycle takes place as follows:

1. The controller transfers data to its output register, placing that data on the sixteen lines (four address and twelve data) that connect to the 6940B input circuitry. To indicate that it wishes to initiate a data transfer, the controller drives the gate ($\overline{\text{GTE}}$) line low.

2. About two μsecs after the leading edge of $\overline{\text{GTE}}$, handshake logic generates the data strobe. At some point in time after the beginning of DST, the handshake logic returns the leading edge of flag ($\overline{\text{FLA}}$), indicating that processing has begun.

3. The controller responds to the leading edge of flag by resetting the gate line to its high state.

4. The handshake logic detects the resetting of the gate signal and terminates DST. When the multiprogrammer has finished processing, the handshake logic returns the trailing edge of flag, thereby concluding the transfer cycle.

7-23 The sequence of events outlined above is typical of all data transfers in the output and control modes. Information transferred in this manner (either to a card or to the unit select and control mode latches) remains stored until the particular address is again specified and a transfer cycle initiated by the controller. Input operations (transfer of data from multiprogrammer to the controller) do not necessarily require use of the gate signal and subsequent handshake cycle. Note also that the sequence outlined above is general enough to describe either the automatic handshake or the extended handshake. Detailed descriptions of both handshake sequences are found in Paragraphs 7-33 through 7-39.

7-24 Control Mode Bits

7-25 Table 7-2 identifies the various control mode bits and summarizes their effect on the operation of the multiprogrammer system. Control signals are distributed throughout the entire system and are totally independent of the slot and unit select address functions. The action of the unit select signals has been explained previously and will not be discussed again here.

7-26 System Enable (SYE). The SYE signal effects the operation of output cards only. With the exception of a few cards (unaffected by SYE), the SYE signal provides a system wide control of output cards which acts independently of the address function; in other words, the SYE signal influences the operation of cards regardless of whether or not they are addressed. When SYE is on, output cards are enabled to transmit any data currently latched in local (on card) storage registers. If SYE is off, data connections between output cards and the external environment are open-circuited or otherwise disabled. The SYE circuitry on individual cards has no control over the address and storage functions, meaning that card outputs may be inhibited without effecting the controller's ability to address and load data into local registers. If SYE is programmed on after a number of cards have been loaded with data in this manner, the cards will simultaneously output that data to external devices when the control word is gated.

7-27 Preset circuitry in the 6940B mainframe ensures that SYE is held off when the system is first powered up. Additional preset circuitry on the individual cards disables outputs (even if SYE is programmed on) until the card is addressed and strobed for the first time. Also, as explained previously, the SYE signal is used to protect against possible damage in the event that a cable is accidently disconnected.

Table 7-2. Control Bits

BIT POSITION	CONTROL BIT	FUNCTION
4	TME (Timing Mode Enable)	Modifies action of the handshake logic by suppressing the automatic flag and causing the $\overline{\text{FLA}}$ signal to follow the state of the $\overline{\text{CTF}}$ (common timing flag) signal.
5	SYE (System Enable)	Absence of this signal automatically deactivates certain cards to a "safe" state.
6	$\overline{\text{DTE}}$ (Data Transfer Enable)	Controls data transfer and timing signals (extended handshake) on certain output cards.
7	$\overline{\text{ISL}}$ (Input Select)	Terminates output data in the mode gates, thereby permitting input data to be returned to the controller from an addressed input card.
8	$\overline{\text{IEN}}$ (Interrupt Enable)	Used in conjunction with TME to modify the action of the handshake logic during input operations.

If the connection is broken in the middle of a chain of mainframes, all units downstream from the break are disabled. Cards within these units will retain previously programmed data, however, and will output that data once the connection is restored. If a mainframe in the midst of a chain is powered down, the SYE signal will again be disabled, inhibiting outputs from that unit and from any units downstream. Once power is restored, all outputs will resume as previously programmed except in the unit that was turned off. Cards in that mainframe will be preset, requiring that they be individually addressed with data and strobed before the system is fully returned to the state that existed just prior to the power off condition. In all cases, absence of the system enable signal serves to prevent uncontrolled outputs, thus reducing the possibility of stimulating external devices with undesired contact closures, excessive voltages, etc.

7-28 **Data Transfer Enable ($\overline{\text{DTE}}$).** The $\overline{\text{DTE}}$ signal provides a mechanism for synchronizing outputs from several cards. The $\overline{\text{DTE}}$ signal influences the operation of four card models, all of which function as output devices. Two of these, the voltage and current D/A cards, have dual rank storage registers as shown in Figure 7-6A. The first register accepts data from the mainframe when the card is addressed and gated with the DST signal. The second register receives data from the first and outputs it to the D/A converter, but does not accept this new data unless $\overline{\text{DTE}}$ is programmed on. Since the converter is connected directly to the external device (if SYE is on), outputs from several D/A cards can be held constant while new data is loaded into first rank storage on each of the cards. $\overline{\text{DTE}}$ can then be programmed on, simultaneously transferring data on each of the cards from first to second rank storage, thereby changing the outputs of all such cards in unison. This feature can be defeated by programming $\overline{\text{DTE}}$ on prior to loading data into the first register. Data sent and gated under these circumstances immediately passes through first rank storage to the second rank and then to the converter.

7-29 The two other cards effected by $\overline{\text{DTE}}$ do not have dual rank storage, but instead have a gate/flag interface with the external environment, (see Figure 7-6B). These cards inhibit this gate signal unless $\overline{\text{DTE}}$ is on. The controller may sequentially address cards and load data into local storage with the DST signal. If SYE is on, the data will be transferred to the external device, but the gate signal to the external device will be suppressed until $\overline{\text{DTE}}$ is programmed on and latched with DST, at which time all loaded cards will simultaneously generate gate signals. If $\overline{\text{DTE}}$ is programed on first, each card will generate its gate signal when addressed and strobed. The gate signals may be used to enable storage circuits in the external device. In this case, the combination of the external storage circuits and the storage circuits on the cards provides a dual rank storage system similar to that described in Paragraph 7-28.

7-30 **Input Select ($\overline{\text{ISL}}$).** The $\overline{\text{ISL}}$ signal causes the most fundamental change in system operating mode. By terminating output data in the mode gates, the $\overline{\text{ISL}}$ signal effectively converts the mainframe from an output multiplexer to an input multiplexer. The address portion of the output word remains functional and when combined with $\overline{\text{ISL}}$ on an input card, enables the addressed card to return data to the

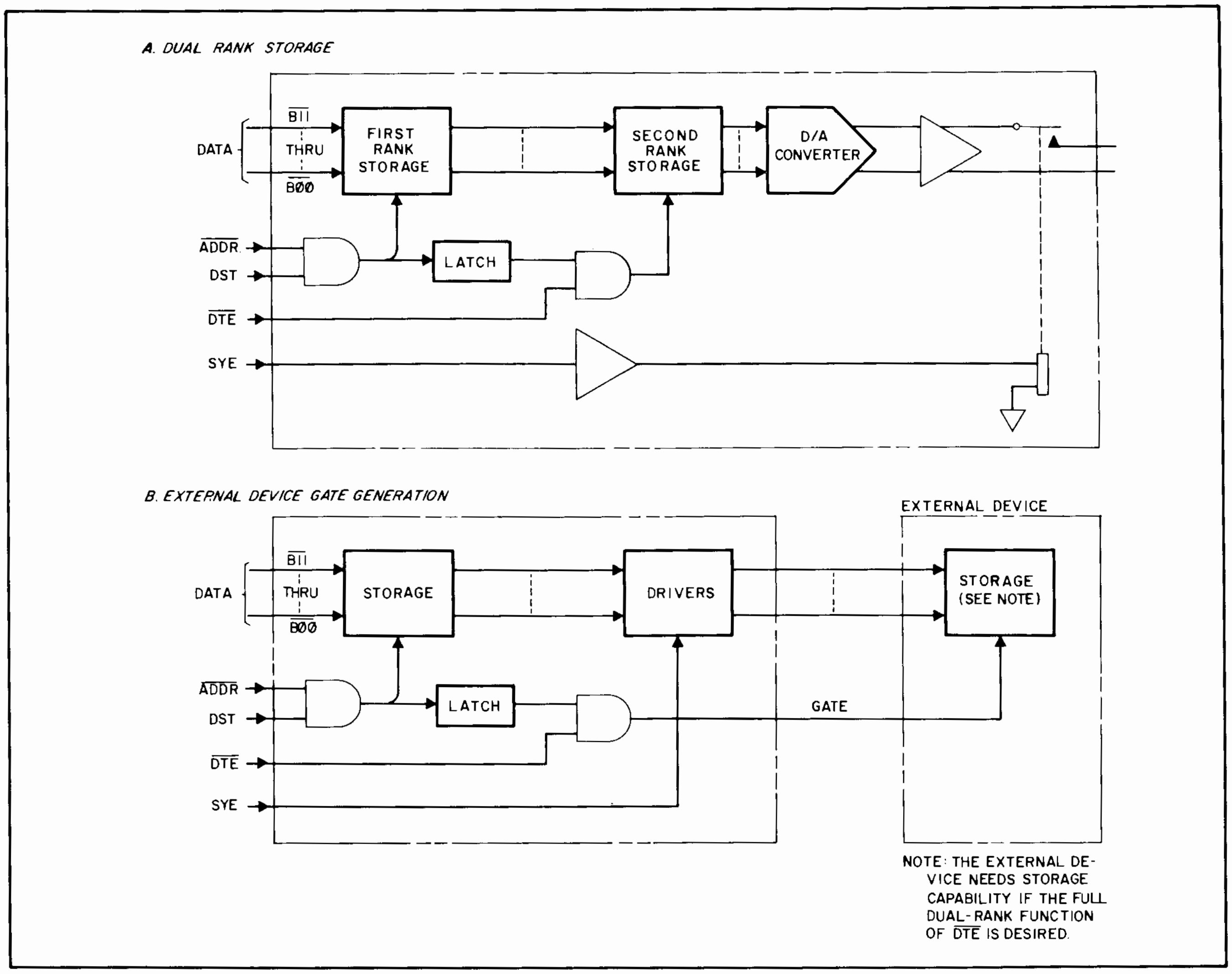


Figure 7-6. Data Transfer Enable ($\overline{\text{DTE}}$) circuits

controller. The output word sent by the controller when the system is in the input mode is called an address word and has the format shown in Figure 7-7.

7-31 Figure 7-8 shows the format of the return data word. The twelve data bit lines ($\overline{\text{B00}}$ through $\overline{\text{B11}}$) follow the state of the corresponding output data bits so long as $\overline{\text{ISL}}$ is off; when $\overline{\text{ISL}}$ is on, they are driven by the addressed card. Bits $\overline{\text{B12}}$ through $\overline{\text{B14}}$ are not used in the return data word; but $\overline{\text{B15}}$ is, and functions in a manner similar to the

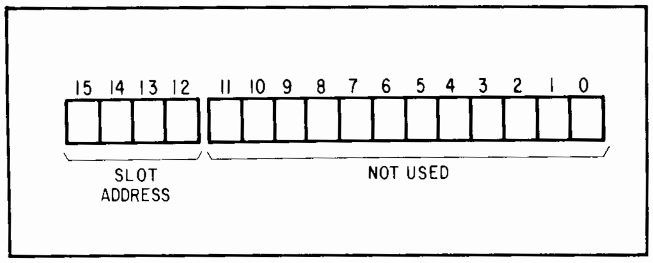


Figure 7-7. Address Word Format

twelve data bits. Referring back to Figure 7-3, the IRQ (Input Request) logic selects the source of the $\overline{\text{B15}}$ signal according to the state of $\overline{\text{ISL}}$. If $\overline{\text{ISL}}$ is off, $\overline{\text{B15}}$ of the output word is returned to the controller. If $\overline{\text{ISL}}$ is on, however, $\overline{\text{B15}}$ of the return data word no longer follows the state of the output bit, but instead reflects the state of the $\overline{\text{IRQ}}$ signal line. The $\overline{\text{IRQ}}$ line is returned from every I/O slot in the system and may be driven by the currently addressed card provided that card is one of the models equipped with $\overline{\text{IRQ}}$ drive circuitry. This signal indicates whether or not the addressed card has completed a data transfer cycle with the external device and is used as the basis for a status polling routine which is part of the interrupt input mode programming sequence (see Paragraphs 7-52 through 7-59).

7-32 Timing Mode Enable (TME) and Interrupt Enable ($\overline{\text{IEN}}$).

Up to this point, explanations of the handshake cycle have stressed the sequence of events without much reference to the factors controlling those events.

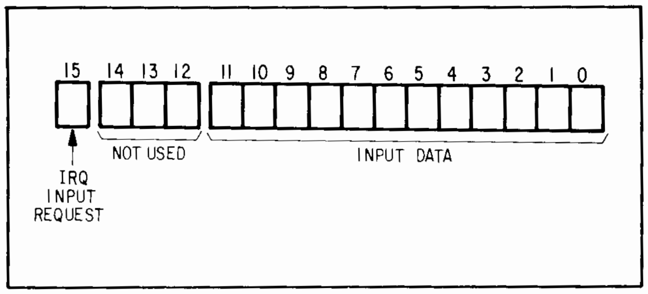


Figure 7-8. Return Data Word Format

An expanded discussion of these factors is necessary if operation of the TME (Timing Mode Enable) and the $\overline{\text{IEN}}$ (Interrupt Enable) control signals is to be understood. Figure 7-9 shows a portion of the functional block diagram given in Figure 7-3. Only those elements effecting the handshake are shown, and have been slightly re-arranged in order to clarify the following discussion.

7-33 Automatic Handshake. The TME signal determines whether the multiprogrammer operates in the automatic or extended handshake mode. When TME is off (automatic handshake), the handshake logic is the source of the flag signal returned to the controller. The DST signal, which is generated by the handshake logic in response to the controller's gate ($\overline{\text{GTE}}$), causes the handshake logic to return the leading edge of flag ($\overline{\text{FLA}}$). When the controller resets $\overline{\text{GTE}}$, DST is terminated and the handshake logic returns the trailing edge of flag, completing the transfer cycle. In this mode of operation both edges of $\overline{\text{FLA}}$ are an automatic response to the controller's gate. The handshake logic has absolute control of the flag signal, and together with the controller forms the complete circuit necessary to synchronize a data transfer.

7-34 Extended Handshake. When TME is on, the automatic handshake just described is suppressed, and the circuit elements shown to the right of the handshake logic in Figure 7-9 assume control of the flag signal. The data strobe (DST) and the gate signal sent to the external device ($\overline{\text{GAT}}$) act as extensions of the controller's gate ($\overline{\text{GTE}}$). In a corresponding manner, the flag returned from the external device ($\overline{\text{FLG}}$) and the common timing flag ($\overline{\text{CTF}}$) become the source of the flag returned to the controller ($\overline{\text{FLA}}$). These six signals, then, are used to implement the extended handshake. Since a number of card models do not have a gate/flag interface with the external device, the complete circuit is not always used; the $\overline{\text{FLA}}$ signal, however, always follows the state of the common timing flag when TME is on.

7-35 The $\overline{\text{CTF}}$ signal line is returned to the handshake logic from the $\overline{\text{CTF}}$ flip-flop and from all I/O slots in the system. The entire circuit is arranged in a wire-OR'ed configuration such that any single source (a card or the $\overline{\text{CTF}}$ flip-flop) can drive the $\overline{\text{CTF}}$ line to a low state. It is not unusual for $\overline{\text{CTF}}$ to be driven by several sources at the same time, in which case it will be held low as long as it is driven by at least one source. As long as $\overline{\text{CTF}}$ is low, $\overline{\text{FLA}}$ will also be held low. The controller will interpret the low state of the $\overline{\text{FLA}}$ signal as an indication that the multiprogrammer is processing data and therefore is "busy".

7-36 Except in the unique situation when the $\overline{\text{IEN}}$ control signal is used to arm input cards that have the W6 jumper option installed, (arming operations are explained in Paragraphs 7-52 through 7-54), no source can drive $\overline{\text{CTF}}$ unless it has been addressed and gated with DST. In most cases, additional conditions (which vary from one card model to another) must be satisfied before the source is fully activated. Since the $\overline{\text{FLA}}$ signal returned to the controller is dependent upon the state of the $\overline{\text{CTF}}$ line when the multiprogrammer is in the extended handshake mode, care must be taken to ensure that one or more sources are or can be activated whenever the controller gates the system.

7-37 The $\overline{\text{CTF}}$ flip-flop is always activated when a control word in which $\overline{\text{IEN}}$ is off is programmed and gated. Its only function is to guarantee that a flag is returned to the controller at the time a control word with TME on is gated

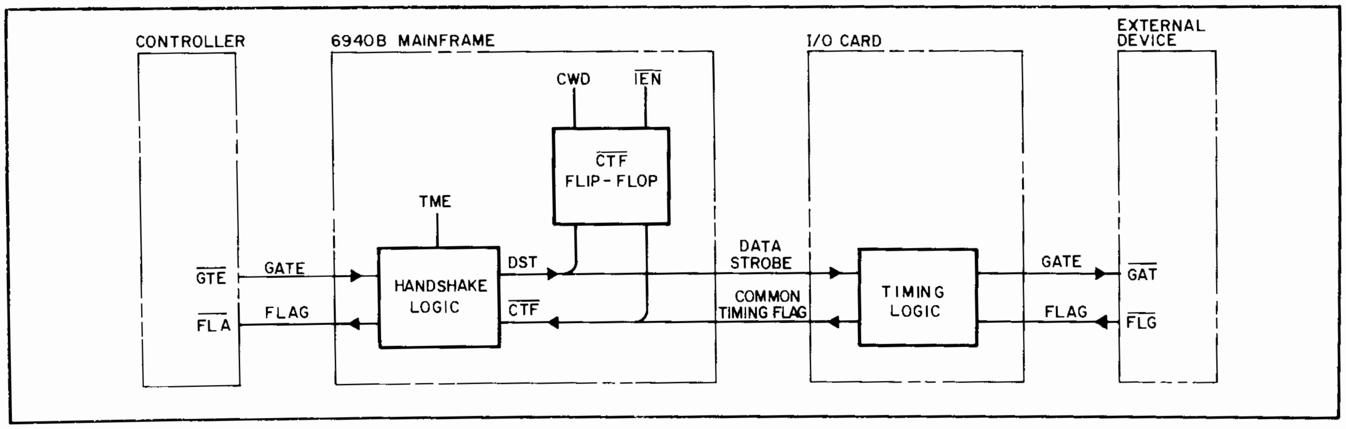


Figure 7-9. Extended Handshake Circuitry

and either of two situations exists:

1. No card has previously been activated to drive $\overline{\text{CTF}}$; or
2. All previously activated cards have completed their transfer cycles and have returned to a de-activated state.

7-38 When a control word with $\overline{\text{IEN}}$ on is gated, the action of the $\overline{\text{CTF}}$ flip-flop is suppressed. This control state is established as part of a specific sequence used during the interrupt input mode of operation and should not be programmed except as part of that sequence. The interrupt mode is described in Paragraphs 7-52 through 7-59.

7-39 Table 7-3 lists the cards with $\overline{\text{CTF}}$ drive circuitry and indicates which control signals must be present for the card to be fully activated. Cards not included have no means for driving $\overline{\text{CTF}}$ and must not be addressed and gated when $\overline{\text{TME}}$ is on since a flag signal will not be returned to the controller.

7-40 PROGRAMMING SEQUENCES

7-41 Although the exact order in which control modes are established and cards activated depends in part upon the particular application, certain generalized programming sequences can be identified and explained. Figure 7-10 shows the sequences applicable to the basic classes of operation in the TME mode. These classes may be distinguished from one another by the answers to three questions:

1. Is the transaction an input or an output operation?
2. Is one card used in the transaction or are several cards used?
3. If several cards are used, do complete transactions occur sequentially (serially) or do they occur concurrently (in parallel)?

7-42 For the purposes of the following explanations, it is assumed that SYE has been programmed on previously and that no cards are activated when the programming sequence is started. Note that the source of the $\overline{\text{CTF}}$ signal used to control $\overline{\text{FLA}}$ is given in brackets whenever Figure 7-10 indicates that a gate is sent by the controller. Timing diagrams showing the transitions of the $\overline{\text{GTE}}$ and $\overline{\text{FLA}}$ signals in both the automatic and the extended handshake modes are provided in Figure 7-11. Arrows indicate that a given transition automatically brings about the one immediately following it.

7-43 Output Operations (Figure 7-10)

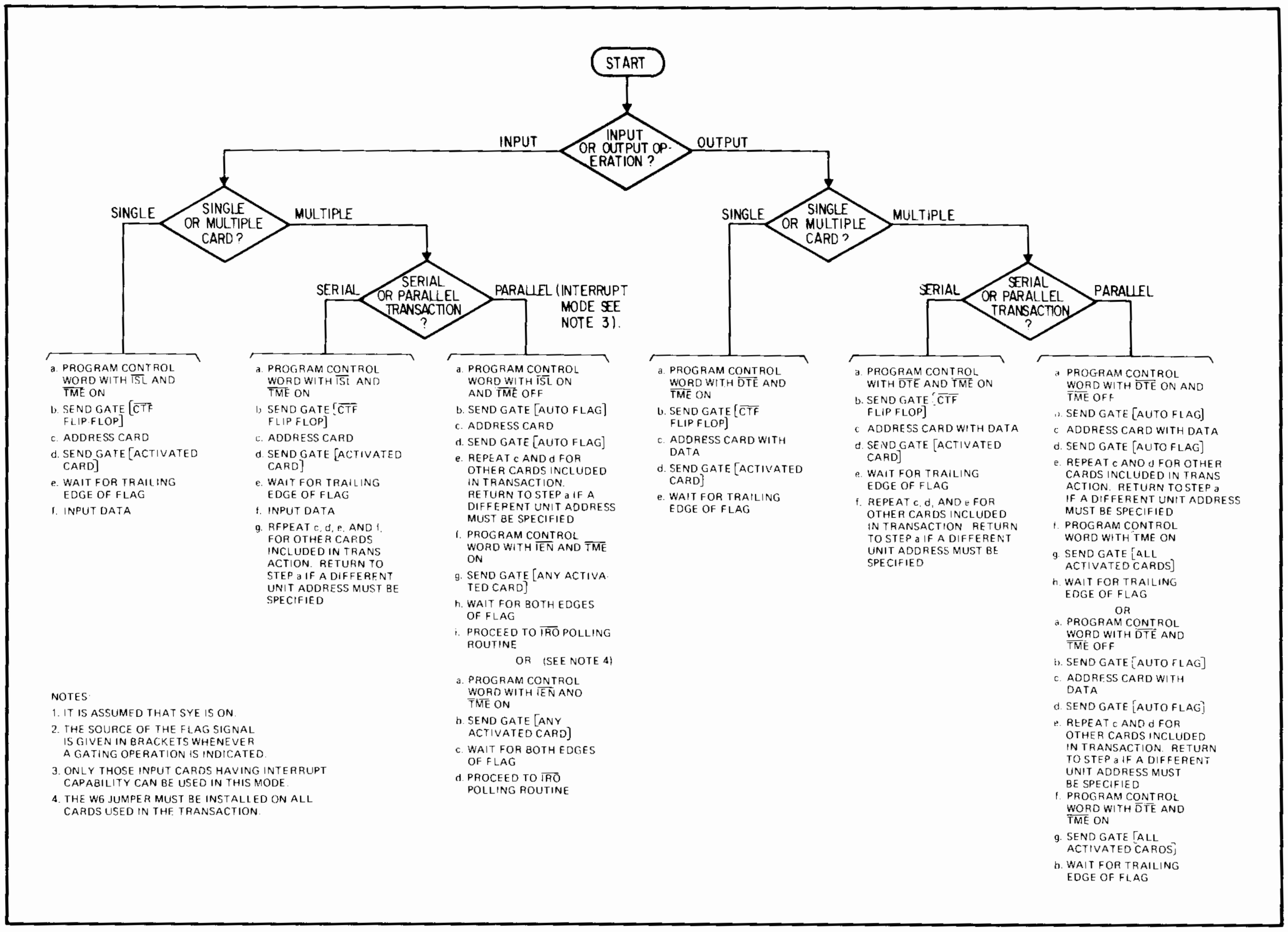
7-44 **Single Transaction.** An output transaction with a single card is the least complicated data transfer operation. The output mode is first established by programming and gating a control word with $\overline{\text{DTE}}$ and TME on. $\overline{\text{ISL}}$ and $\overline{\text{IEN}}$ are off. $\overline{\text{DTE}}$ need not be on if the particular card is not controlled by it, but no harm is done if it is on, and it is generally good practice to program TME and $\overline{\text{DTE}}$ on together during output operations since many output cards

Table 7-3. $\overline{\text{CTF}}$ Activation Requirements

MODEL	DESCRIPTION	DST	$\overline{\text{DTE}}$	$\overline{\text{ISL}}$	$\overline{\text{IEN}}$
69321B	Voltage D/A	Yes	Yes	— —	— —
69325A	Power Supply	Yes	— —	— —	— —
69326A	Amplifier	Yes	— —	— —	— —
69327A	Programming	Yes	— —	— —	— —
69328A	Cards	Yes	— —	— —	— —
69330A	Relay Output	Yes	Yes	— —	— —
69331A	TTL Output	Yes	Yes	— —	— —
69335A	Stepper Motor	Yes	— —	— —	— —
69370A	Current D/A	Yes	Yes	— —	— —
69421A	Voltage Monitor	Yes	— —	— —	— —
69431A	Digital Input	Yes	— —	See Note	
69433A	Relay Readback	Yes	— —	— —	— —
69434A	Event Sense	Yes	— —	See Note	
69436A	Process Interrupt	Yes	— —	See Note	
69500A thru 69513A	Resistance Programming	Yes	— —	— —	— —
69600B	Timer	Yes	Yes *	See Note	

Note: The $\overline{\text{ISL}}$ signal is sufficient to activate these cards. If W6 is installed, $\overline{\text{IEN}}$ is sufficient.

* With jumper W4 installed.


START
INPUT OR OUTPUT OPERATION ?
INPUT
OUTPUT
SINGLE OR MULTIPLE CARD ?
SINGLE
MULTIPLE
SERIAL OR PARALLEL TRANSACTION ?
SERIAL
PARALLEL (INTERRUPT MODE SEE NOTE 3).
SINGLE OR MULTIPLE CARD ?
SINGLE
MULTIPLE
SERIAL OR PARALLEL TRANSACTION ?
SERIAL
PARALLEL

a. PROGRAM CONTROL WORD WITH $\overline{\text{ISL}}$ AND $\overline{\text{TME}}$ ON
b. SEND GATE [$\overline{\text{CTF}}$ FLIP-FLOP]
c. ADDRESS CARD
d. SEND GATE [ACTIVATED CARD]
e. WAIT FOR TRAILING EDGE OF FLAG
f. INPUT DATA

a. PROGRAM CONTROL WORD WITH $\overline{\text{ISL}}$ AND $\overline{\text{TME}}$ ON
b. SEND GATE [$\overline{\text{CTF}}$ FLIP FLOP]
c. ADDRESS CARD
d. SEND GATE [ACTIVATED CARD]
e. WAIT FOR TRAILING EDGE OF FLAG
f. INPUT DATA
g. REPEAT c, d, e, AND f, FOR OTHER CARDS INCLUDED IN TRANSACTION. RETURN TO STEP a IF A DIFFERENT UNIT ADDRESS MUST BE SPECIFIED

a. PROGRAM CONTROL WORD WITH $\overline{\text{ISL}}$ ON AND $\overline{\text{TME}}$ OFF
b. SEND GATE [AUTO FLAG]
c. ADDRESS CARD
d. SEND GATE [AUTO FLAG]
e. REPEAT c AND d FOR OTHER CARDS INCLUDED IN TRANSACTION. RETURN TO STEP a IF A DIFFERENT UNIT ADDRESS MUST BE SPECIFIED
f. PROGRAM CONTROL WORD WITH $\overline{\text{IEN}}$ AND $\overline{\text{TME}}$ ON
g. SEND GATE [ANY ACTIVATED CARD]
h. WAIT FOR BOTH EDGES OF FLAG
i. PROCEED TO $\overline{\text{IRO}}$ POLLING ROUTINE

OR (SEE NOTE 4)

a. PROGRAM CONTROL WORD WITH $\overline{\text{IEN}}$ AND $\overline{\text{TME}}$ ON
b. SEND GATE [ANY ACTIVATED CARD]
c. WAIT FOR BOTH EDGES OF FLAG
d. PROCEED TO $\overline{\text{IRO}}$ POLLING ROUTINE

a. PROGRAM CONTROL WORD WITH $\overline{\text{DTE}}$ AND $\overline{\text{TME}}$ ON
b. SEND GATE [$\overline{\text{CTF}}$ FLIP FLOP]
c. ADDRESS CARD WITH DATA
d. SEND GATE [ACTIVATED CARD]
e. WAIT FOR TRAILING EDGE OF FLAG

a. PROGRAM CONTROL WITH $\overline{\text{DTE}}$ AND $\overline{\text{TME}}$ ON
b. SEND GATE [$\overline{\text{CTF}}$ FLIP FLOP]
c. ADDRESS CARD WITH DATA
d. SEND GATE [ACTIVATED CARD]
e. WAIT FOR TRAILING EDGE OF FLAG
f. REPEAT c, d, AND e FOR OTHER CARDS INCLUDED IN TRANSACTION. RETURN TO STEP a IF A DIFFERENT UNIT ADDRESS MUST BE SPECIFIED

a. PROGRAM CONTROL WORD WITH $\overline{\text{DTE}}$ ON AND $\overline{\text{TME}}$ OFF
b. SEND GATE [AUTO FLAG]
c. ADDRESS CARD WITH DATA
d. SEND GATE [AUTO FLAG]
e. REPEAT c AND d FOR OTHER CARDS INCLUDED IN TRANSACTION. RETURN TO STEP a IF A DIFFERENT UNIT ADDRESS MUST BE SPECIFIED
f. PROGRAM CONTROL WORD WITH TME ON
g. SEND GATE [ALL ACTIVATED CARDS]
h. WAIT FOR TRAILING EDGE OF FLAG

OR

a. PROGRAM CONTROL WORD WITH $\overline{\text{DTE}}$ AND $\overline{\text{TME}}$ OFF
b. SEND GATE [AUTO FLAG]
c. ADDRESS CARD WITH DATA
d. SEND GATE [AUTO FLAG]
e. REPEAT c AND d FOR OTHER CARDS INCLUDED IN TRANSACTION. RETURN TO STEP a IF A DIFFERENT UNIT ADDRESS MUST BE SPECIFIED
f. PROGRAM CONTROL WORD WITH $\overline{\text{DTE}}$ AND $\overline{\text{TME}}$ ON
g. SEND GATE [ALL ACTIVATED CARDS]
h. WAIT FOR TRAILING EDGE OF FLAG

NOTES:
1. IT IS ASSUMED THAT SYE IS ON.
2. THE SOURCE OF THE FLAG SIGNAL IS GIVEN IN BRACKETS WHENEVER A GATING OPERATION IS INDICATED.
3. ONLY THOSE INPUT CARDS HAVING INTERRUPT CAPABILITY CAN BE USED IN THIS MODE.
4. THE W6 JUMPER MUST BE INSTALLED ON ALL CARDS USED IN THE TRANSACTION.

Figure 7-10. Programming Sequences in the Timing Mode

cannot drive $\overline{\text{CTF}}$ unless $\overline{\text{DTE}}$ is on. The card is then addressed and gated, and immediately returns the leading edge of $\overline{\text{CTF}}$. Depending on the card model, the trailing edge of $\overline{\text{CTF}}$ is controlled by circuits on the card itself, or by the external device. In either case, the transaction is concluded when the trailing edge of $\overline{\text{CTF}}$ is detected by the handshake logic and returned to the controller as the trailing edge of $\overline{\text{FLA}}$.

7-45 Serial Transactions. Serial transactions with output cards may be programmed by simply repeating the sequence just described for each card in turn. The serial method is disadvantageous when the number of cards is large and the periods of the successive handshake cycles are long. The controller must wait for each card to return the trailing edge of flag before it can proceed to the next card, with the result that the total operation consumes an unnecessarily large amount of time and makes inefficient use of the controller's processing capability. Consequently, the serial method should not be used unless a strict sequential operation is required by a particular application.

7-46 Parallel Transactions. The two programming sequences for parallel output transactions shown in Figure 7-10 solve this problem by not programming TME on until all of the cards have been activated in the automatic handshake mode. This high speed approach is possible since TME has no effect on the activation of cards and their subsequent control of $\overline{\text{CTF}}$, but only determines whether or not the $\overline{\text{FLA}}$ signal returned to the controller by the handshake logic will follow the state of the common timing flag. Once all cards have been activated, a control word with TME on is programmed and gated. The $\overline{\text{CTF}}$ flip-flop returns the leading edge of flag when the control word is gated, but the wire-OR'ed nature of the $\overline{\text{CTF}}$ circuitry will suppress the trailing edge until all of the activated cards have "timed out".

7-47 The two parallel output sequences differ from one another in the manner in which $\overline{\text{DTE}}$ is used. The first programming sequence is somewhat faster in execution because each card begins its transfer cycle as soon as it is addressed and gated. Cards with long transfer cycles can be activated early in the sequence, while cards with short cycles can be activated last, thus minimizing the time used for the complete transaction. The second sequence

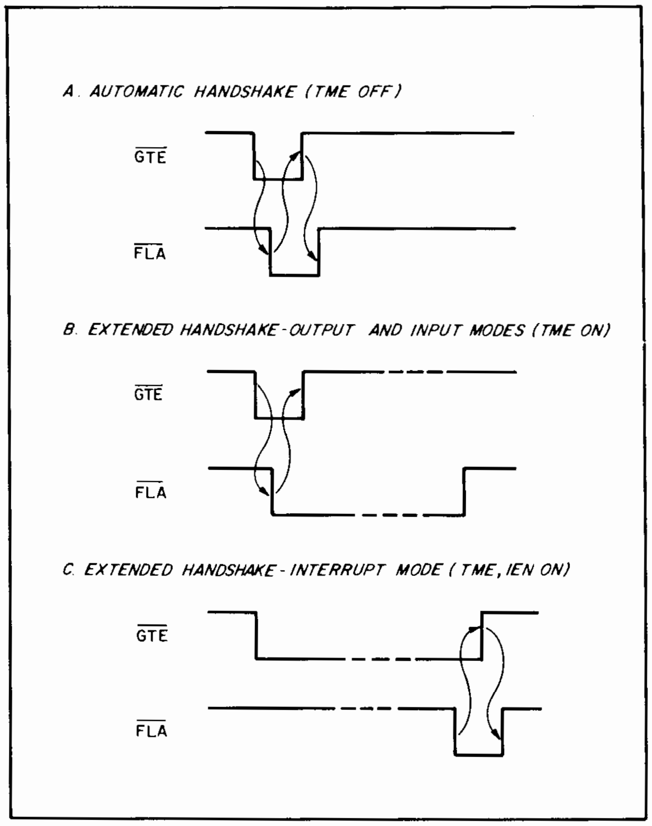

A. AUTOMATIC HANDSHAKE (TME OFF)
GTE
FLA
B. EXTENDED HANDSHAKE-OUTPUT AND INPUT MODES (TME ON)
GTE
FLA
C. EXTENDED HANDSHAKE-INTERRUPT MODE (TME, IEN ON)
GTE
FLA

Figure 7-11. Handshake Timing

sacrifices speed in order to provide full parallel output capability. This method uses only those card models controlled by the $\overline{\text{DTE}}$ signal and is identical to the first except that neither TME nor $\overline{\text{DTE}}$ is programmed on until all cards have been addressed and gated. Since $\overline{\text{DTE}}$ must be on for these card models to be fully activated, they are not able to drive the $\overline{\text{CTF}}$ line, but do remain in a partially activated state even when they are no longer addressed. When the control word with TME and $\overline{\text{DTE}}$ on is programmed and gated, all cards are simultaneously enabled to output new data and fully activated to drive the $\overline{\text{CTF}}$ line.

7-48 If TME is not programmed on, all output cards (including those without $\overline{\text{CTF}}$ drive circuitry) may be operated in any of the four programming sequences discussed so far. The distinction between the serial sequence and the first parallel sequence is no longer significant, however, since the flag is now returned automatically by the handshake logic. The second parallel sequence is different only because it makes use of the $\overline{\text{DTE}}$ function to synchronize outputs to the external environment.

7-49 Input Operations (Figure 7-10)

7-50 Certain input cards may also be operated in the output mode, and are generally programmed using the single or serial sequences with TME off. Just as there are input cards which may be used in the output mode, so there are output cards which have a secondary function that is used in the input mode.

NOTE

Unlike output operations where a gate signal is always required to strobe data into local storage registers, use of the gate depends on the particular card model during input transactions.

7-51 Cards with input capability have circuitry for driving either the $\overline{\text{IRQ}}$ signal line, some or all of the twelve return data lines, or for driving all thirteen lines. Any card in this group is enabled to drive the return data bus if it is addressed with $\overline{\text{ISL}}$ on, regardless of whether or not it is gated. Cards without $\overline{\text{CTF}}$ circuitry must not be gated if TME is also on, and are usually operated without a gate in the automatic handshake mode where TME is off and the possibility of a system "hang-up" is eliminated. The remaining cards with input capability may be used in this mode or in one or more of the input programming sequences given in Figure 7-10.

7-52 **Parallel (Interrupt Mode) Transactions.** Cards which function in the parallel input mode (interrupt mode) are the only cards able to drive the $\overline{\text{IRQ}}$ signal line. These cards also have $\overline{\text{CTF}}$ drive circuitry and, with the exception of the 69431A digital input card, respond to and act upon data when addressed and gated in the output mode. Only one card (the 69600A timer card) actually transmits data to the external environment, however, the others use data received in the output mode to modify the manner in which they react to input signals. All cards with interrupt capability must be armed before they are enabled to drive either the $\overline{\text{IRQ}}$ or the $\overline{\text{CTF}}$ signal lines in the interrupt mode.

7-53 Individual arming is accomplished by addressing and gating the card when the system is in the input mode (a control word with $\overline{\text{ISL}}$ on must have been programmed and gated previously). Group arming, on the other hand, is made possible by a hardware option available on all cards with $\overline{\text{IRQ}}$ circuitry. A jumper (W6), when installed, permits the controller to arm cards by programming the interrupt mode (a control word with $\overline{\text{IEN}}$ and TME on is programmed and gated). As they are armed, the cards are enabled to begin a data transfer cycle with the external device and to drive the $\overline{\text{IRQ}}$ signal line.

7-54 As shown in Figure 7-10, steps (a) through (g) given in the first parallel input programming sequence and steps (a) and (b) given in the second arm the cards and place the system in the interrupt mode. Since the $\overline{\text{IEN}}$ control signal suppresses the $\overline{\text{CTF}}$ flip-flop and TME suppresses the automatic handshake, the controller depends on the armed cards to return the leading edge of flag. When operated in the interrupt mode, cards do not return a leading edge of flag until they have completed a data transfer cycle with the external device. The first card to complete transfer drives $\overline{\text{CTF}}$ low and causes the handshake logic to return the leading edge of flag, whereupon the controller resets $\overline{\text{GTE}}$ and the handshake logic responds by terminating DST. Once the data strobe is terminated, the $\overline{\text{CTF}}$ circuitry on all armed cards is disabled and the trailing edge of flag is returned to the controller indicating that an interrupt has occurred. Unlike the parallel output mode where the trailing edge of flag is controlled by the last card to complete its operation, the interrupt mode flag is controlled by the first card.

7-55 $\overline{\text{IRQ}}$ Poll. At this point the controller polls all armed cards by sequentially addressing them without a gate in the input mode ($\overline{\text{ISL}}$ is on and TME is off) and examining the $\overline{\text{IRQ}}$ bit (bit fifteen of the return data word) returned from each card as it is addressed in order to determine which card completed its transfer cycle and generated the interrupt. If the addressed card has completed a transfer cycle, the $\overline{\text{IRQ}}$ signal is returned to the controller low (true), otherwise it is returned high (false). Depending on the card model, a high state on the $\overline{\text{IRQ}}$ line simply means that the data transaction is still in process, in other cases it indicates that the addressed card has not detected a change of state in the external device. Since it is possible that several cards have completed their transfer cycles, the order in which cards are polled establishes an interrupt priority system in software.

7-56 When an interrupting card is found, the controller may process the return data word according to the demands of the particular program. Once this is completed, the controller must either recycle or disarm the interrupting card since it will generate an interrupt as soon as the system is again gated with $\overline{\text{IEN}}$ on unless it is reset (the requirements for interrupting, recycling and disarming are different for each card model and are explained in detail in Chapter V). If a card is recycles, it remains armed but will not generate an interrupt until it has completed another transfer cycle; if it is disarmed, it becomes inoperative in the interrupt mode until it is again armed. Recycling or disarming should be considered part of any gated data input sequence since cards with interrupt capability must be reset in one way or the other whenever they are used, regardless of whether the system is in the interrupt mode or not.

7-57 Once the interrupting card is reset, the controller may continue with the poll or it may re-establish the interrupt mode by gating the system with $\overline{\text{IEN}}$ on. If the first interrupt was generated by several cards at the same time or a card completed its transfer cycle while the polling routine was conducted, a new interrupt will be generated immediately;* if not, the controller will again wait for both edges of flag. The controller may also re-establish the interrupt mode, proceed to a totally unrelated operation, and then return to the polling sequence at some later time, either as the interrupt occurs (if the controller has a hardware interrupt system) or when directed to do so by the program.

7-58 The entire interrupt sequence may be terminated after an interrupt has occurred provided that the system has not been gated again with $\overline{\text{IEN}}$ on. Even if the gate has been sent, the mode may still be aborted, but the controller's gate must now be reset by a program command. In either case it is a good policy to disarm any active cards.

7-59 The W6 jumper option should be used with caution. All cards in the system which have W6 installed will be armed whenever IEN is programmed on and gated. This situation can create difficulties if the programmer is not fully aware of its implications. For instance: the $\overline{\text{IRQ}}$ polling routine must be conducted with $\overline{\text{ISL}}$ on and TME off, but $\overline{\text{IEN}}$ has no influence during this operation and may either be left on or turned off. If $\overline{\text{IEN}}$ is turned off, however, and the interrupting card is disarmed, the interrupt mode cannot be programmed back on without re-arming the card. The problem may be solved by leaving $\overline{\text{IEN}}$ on during the polling and disarming operation. Similarly, cards with W6 installed may not be individually armed, but must instead be selectively disarmed after $\overline{\text{IEN}}$ is programmed on.

7-60 Single and Serial Transactions. Returning to Figure 7-10, it may be seen that single card and serial input transactions are related in the same sense that those two classes of output transactions are; the serial process is nothing more than a repetition of the single card operation. Assuming that all cards involved are able to drive the $\overline{\text{CTF}}$ line, the input and extended handshake modes are established by programming and gating a control word with $\overline{\text{ISL}}$ and TME on. The first card is then addressed, gated, and immediately drives $\overline{\text{CTF}}$ low, returning the leading edge of flag. Once the data transfer between the external device and the card is complete, the card stops driving $\overline{\text{CTF}}$ and the trailing edge of flag is detected and returned to the controller by the handshake logic. Thus informed that the data transfer has taken place, the controller maintains the card address and reads back the input word on the return data bus. Card models having the necessary drive circuitry will also return $\overline{\text{IRQ}}$ while addressed, but the controller need not test that bit since the trailing edge of flag is only returned when the data transfer is complete and therefore

* The 69434A event sense card is an exception since it cannot detect or store an event unless both DST and $\overline{\text{IEN}}$ are on.

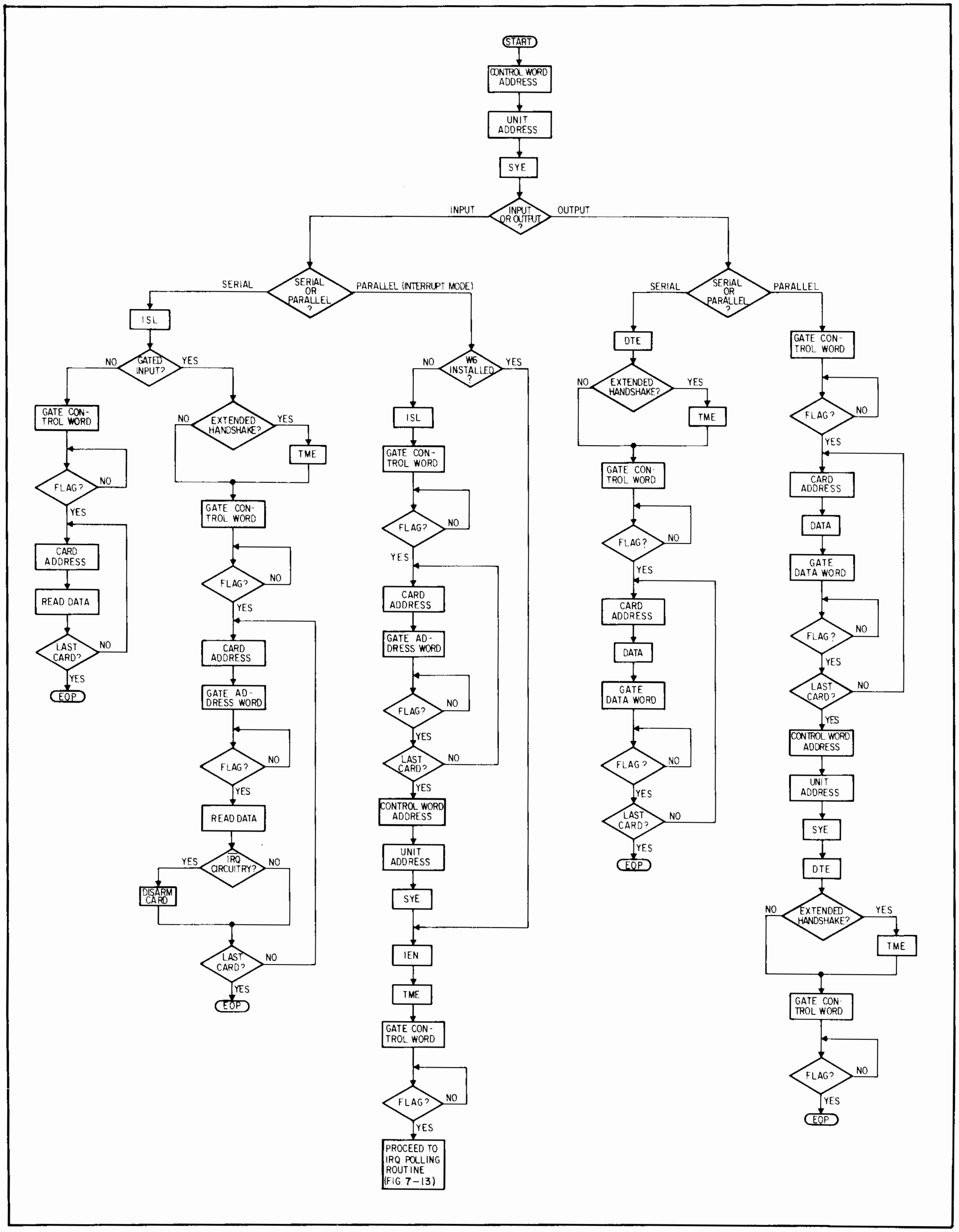


Figure 7-12. Programming Sequences, Flow Chart

is synonomous with $\overline{\text{IRQ}}$ true. Regardless of whether the total operation involves one card or many, transactions are completed with a single card at a time and the $\overline{\text{IRQ}}$ function is not used. As previously indicated, cards with $\overline{\text{IRQ}}$ circuitry are armed when gated with $\overline{\text{ISL}}$ on and should be disarmed once the transaction is completed.

7-61 Programming Sequences Flow Charts

7-62 Main Programming Sequences. Figure 7-12 presents, in flow chart form, the programming sequences given in Figure 7-10. The automatic handshake modes are also shown. Note that the relationship between single card and serial transactions is made more apparent by combining the two modes into a single sequence which incorporates a "last card?" decision box. When the transaction involves a single card only, the "yes" condition is satisfied on the first pass through the sequence and the flow proceeds directly to the end of operation (EOP). The "no" condition is satisfied during serial transactions until the last card has been programmed.

7-63 The "last card?" decision box also implies the possibility of a different unit address. In such cases, the flow must return to "start" and proceed from there back to the same branch sequence after a control word with the new unit address and appropriate control bits is programmed and gated.

7-64 A "flag?" decision box is shown immediately after each gating operation. The trailing edge of flag is required to satisfy the "yes" condition; the "no" condition is thus sustained until the handshake is complete and provides a "wait for flag" loop.

7-65 IRQ Poll Sequence. As shown in Figure 7-12, the interrupt mode programming sequence does not conclude with an end of operation box, but proceeds to the $\overline{\text{IRQ}}$ polling routine. A flow chart for that sequence is given in Figure 7-13, and is arranged as a self-contained subroutine.

7-66 FRONT PANEL CONTROLS

7-67 Local Operation

7-68 The 6940B mainframe is provided with a front panel switch register which is enabled when the data source switch is set to the local position. The switches (with the exception of the data source and power switches) are proximity types actuated by body capacitance and have integral indicator lamps. These lamps monitor the state of the various signal lines (the lamp is on when the signal is in the true state) in both the local and remote operating modes.

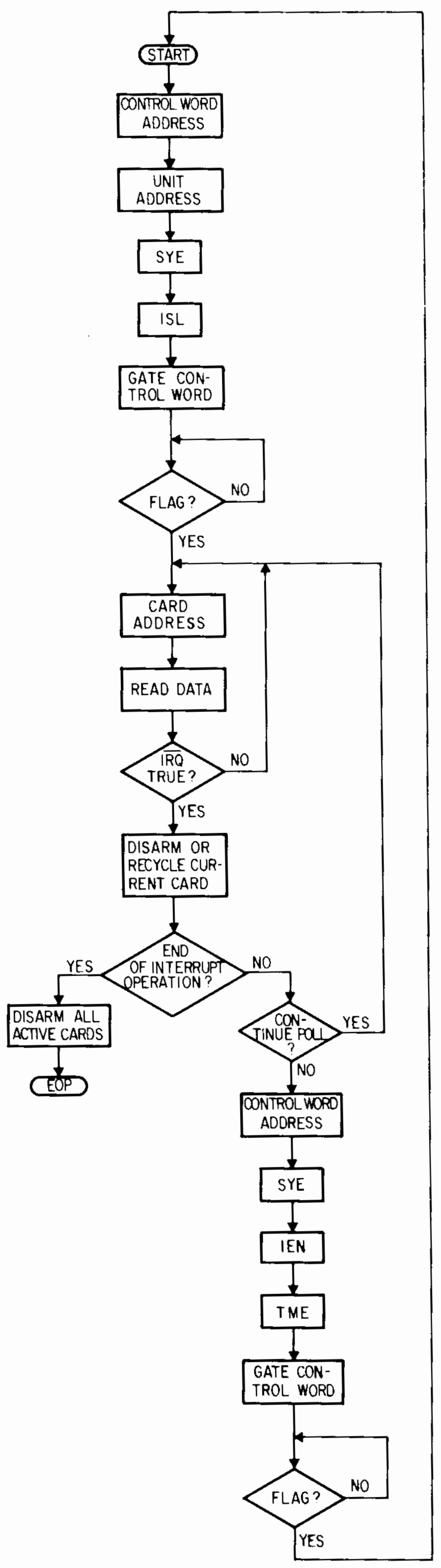


Figure 7-13. IRQ Polling Sequence, Flow Chart

7-69 The sixteen switch/indicators in the top row monitor and/or drive the data signal lines that connect between the multiprogrammer and the controller. If the mainframe is in the output mode ($\overline{ISL}$ is off), the switch register displays the data word either as sent by the controller or as manually programmed on the switches (assuming that the data source switch is set to the local mode). If the input mode is operative ($\overline{ISL}$ is on and a control word address is not specified), the display is a combination of the address word and the return data word. The switches corresponding to bits $\overline{B00}$ through $\overline{B11}$ are not enabled when local operation is selected in the input mode, but the switches for bits $\overline{B12}$ through $\overline{B15}$ are enabled and may be used to specify the card address.

7-70 Return Data Circuit

7-71 The $\overline{IRQ}$ signal is not displayed as bit $\overline{B15}$ on the switch register, but is instead displayed on the return data indicator. The return data switch/indicator also displays and/or drives the $\overline{FLA}$ signal line. Figure 7-14 shows the circuitry used to implement these functions. The return data indicator displays the state of $\overline{FLA}$ regardless of the system operating mode (the lamp is on when $\overline{FLA}$ is low). It will also display the state of the $\overline{IRQ}$ line if $\overline{ISL}$ is also on (the lamp is on if $\overline{IRQ}$ is true). The switch function is enabled only in the local mode, in which case the handshake logic is prevented from driving the $\overline{FLA}$ line and the return data switch becomes the only source for $\overline{FLA}$. The exact reverse situation is in effect when remote operation is selected.

7-72 The load output indicator monitors the controller's gate (the lamp is on when $\overline{GTE}$ is low). The associated switch substitutes drives the $\overline{GTE}$ line when the local operating mode is selected. The clear register switch returns all data switches to their off state and is provided as an operator convenience.

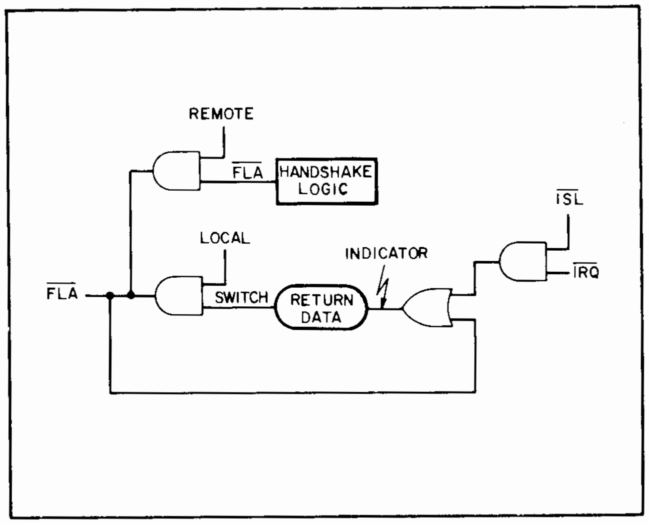


Figure 7-14. Return Data Circuit

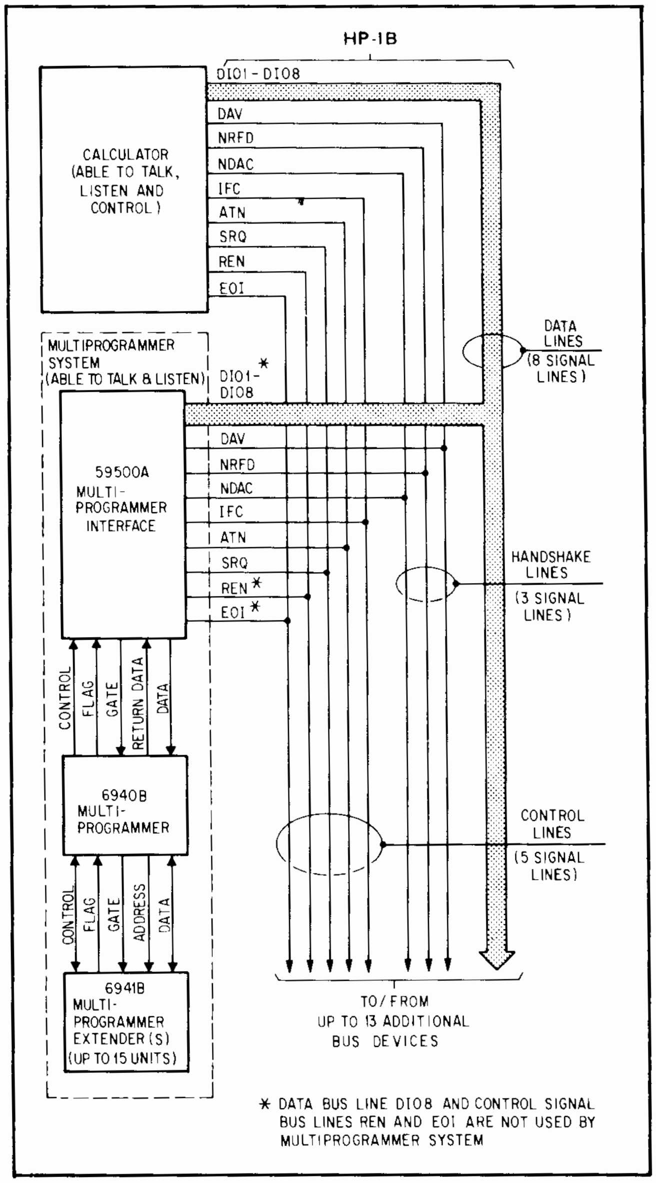
Chapter VIII
MULTIPROGRAMMER INTERFACE 59500A DESCRIPTION

8-1 INTRODUCTION

8-2 The 59500A Multiprogrammer Interface permits bi-directional operation of the 6940B Multiprogrammer and any associated 6941B Multiprogrammer Extenders on the Hewlett-Packard Interface Bus (HP-IB). This chapter provides a basic description of the operations and functions performed by the 59500A and its relationship to the HP-IB and 6940B/6941B.

8-3 Figure 8-1 illustrates the signal flow on the HP-IB between the controller (calculator) and the 59500A Multiprogrammer Interface. The major signal flow between the units (59500A, 6940B, and 6941B) of the multiprogrammer system is also shown.

8-4 Data is transferred over the bus on data input/output lines DIO1-DIO8 utilizing the byte serial technique. Each byte (character) of information, 8-bits in parallel, is transferred onto the bus in serial fashion. Lines DIO1-DIO7 accommodate the 7-bits (1 character) of the ASCII code. Line DIO8 is not used by the multiprogrammer system but can be used by other bus instruments. The state of the attention (ATN) line, controlled by the calculator, determines how the data lines are interpreted. The ATN line is constantly monitored by the 59500A and all other bus devices. When ATN is true, the bus devices (in this case the 59500A) interpret the data on lines DIO1-DIO8 as instructions (commands) from the calculator. Assume that a command is sent which enables the 59500A to listen in preparation to receive data from a talker (in this case the calculator). Multiprogrammer operations can only be initiated when the 59500A is listening. When the ATN line is false, data is transferred from the calculator to the 59500A. The 59500A converts the serial ASCII characters from the calculator into the 16-bit word format required by the multiprogrammer. Depending upon the data received, the multiprogrammer will perform either an output or an input operation as described in Chapter VII. If the timing mode is not used, the calculator can send the 59500A additional data. However, if it is used, the calculator is prevented from sending the 59500A additional data until the multiprogrammer has completed operations. When an input operation is performed by the multiprogrammer, a return data word is stored in the 59500A. In order for the calculator to receive this data, the calculator must set ATN true, command the 59500A to talk and the calculator to listen. When ATN is false, the 59500A converts the multiprogrammer return



HP-IB
DIO1 - DIO8
CALCULATOR (ABLE TO TALK, LISTEN AND CONTROL)
DAV
NRFD
NDAC
IFC
ATN
SRQ
REN
EOI
MULTIPROGRAMMER SYSTEM (ABLE TO TALK & LISTEN)
DIO1 - DIO8 *
59500A MULTI-PROGRAMMER INTERFACE
REN *
EOI *
DATA LINES (8 SIGNAL LINES)
HANDSHAKE LINES (3 SIGNAL LINES)
CONTROL LINES (5 SIGNAL LINES)
CONTROL
FLAG
GATE
RETURN DATA
DATA
6940B MULTI-PROGRAMMER
ADDRESS
6941B MULTI-PROGRAMMER EXTENDER(S) (UP TO 15 UNITS)
TO/FROM UP TO 13 ADDITIONAL BUS DEVICES
* DATA BUS LINE DIO8 AND CONTROL SIGNAL BUS LINES REN AND EOI ARE NOT USED BY MULTIPROGRAMMER SYSTEM

Figure 8-1. HP-IB Connections

data word into serial ASCII characters for processing by the calculator.

8-5 The service request (SRQ) line is used by the 59500A to indicate that the multiprogrammer requires service. The SRQ line is enabled whenever the multiprogrammer is operating in the timing mode and will be set when the multiprogrammer completes an operation(s) or requests an interruption of the current programming sequence. Control signals from the 6940B to the 59500A implement the service request function. The 59500A also responds to the interface clear (IFC) control signal from the calculator. The IFC signal is used by the calculator to terminate activity on the HP-IB. The 59500A does not respond to the REN (remote enable) and EOI (end or identify) control bus lines.

8-6 A 3-wire handshake process occurs with each character transferred on the HP-IB. A 3-wire handshake cycle takes place when the 59500A is receiving a character and when it is sending a character. The 59500A adapts the 3-wire handshake process used on the HP-IB to the 2-wire method (Gate/Flag) employed by the multiprogrammer. The 3-wire handshake is implemented by the DAV (data valid), NRFD (not ready for data), and NDAC (data not accepted) bus lines.

8-7 BLOCK DIAGRAM DESCRIPTION

8-8 Figure 8-2 illustrates the major circuits and signal flow through the 59500A. All signal abbreviations (i. e. DAV, NDAC, etc.) on the block diagram refer to the true state of the corresponding signal line. The following paragraphs describe operation of the 59500A in the command mode and in the listen and talk modes. The 3-wire hand-

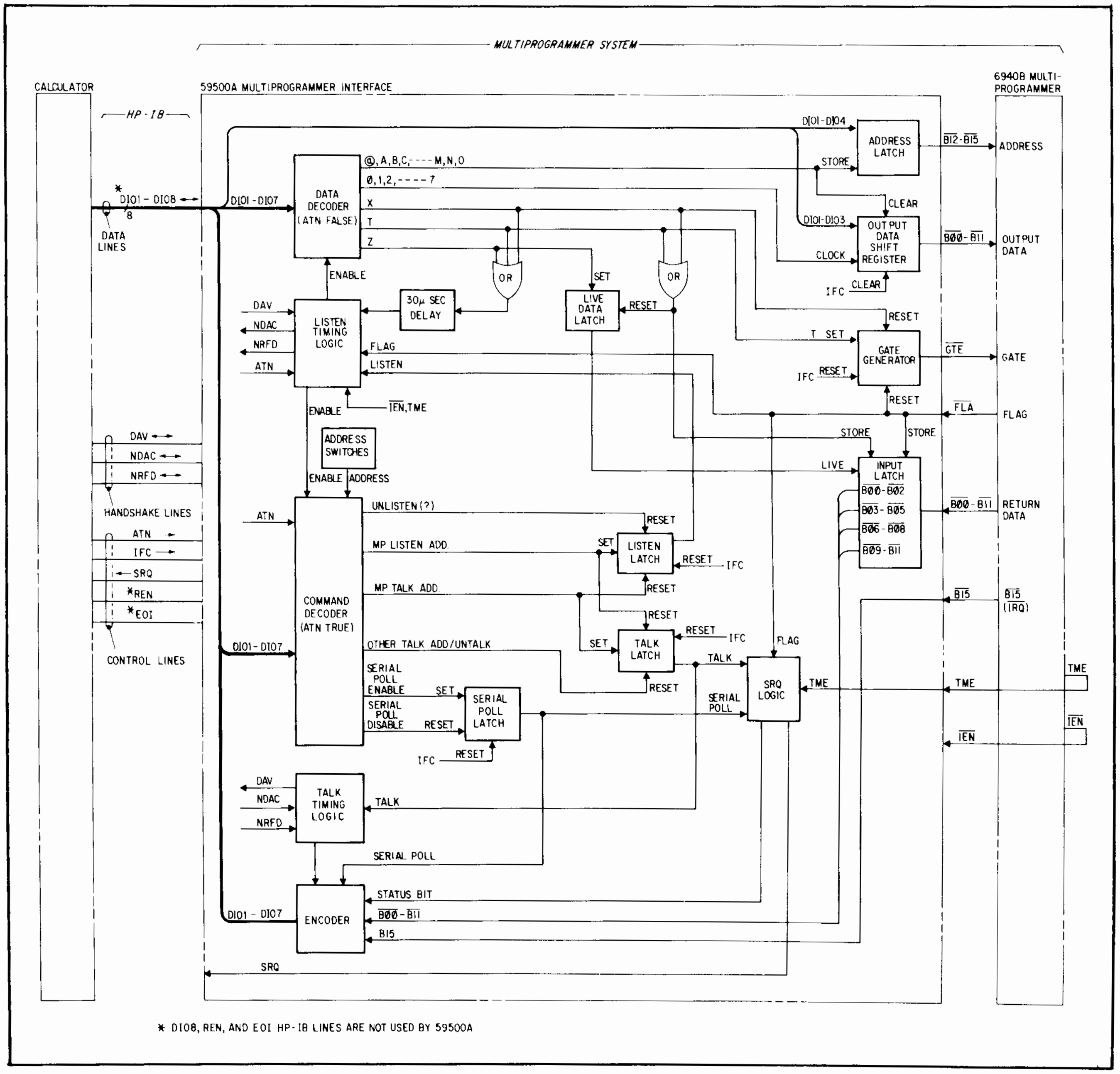


Figure 8-2. Multiprogrammer Interface 59500A, Block Diagram

shake cycle, service request operation and associated serial poll operation are also described.

8-9 Command Mode

8-10 When the ATN line goes true, the command decoder is enabled. The command decoder consists of a talk/listen decoder and a serial poll decoder. The talk/listen decoder receives data lines DIO7 (MSB) through DIO1 (LSB) from the HP-IB and the address code from the address switches (slide switches located on rear of 59500A unit). When the ATN line is true, the talk/listen decoder controls the talk and/or listen latches when any one of the following ASCII characters is decoded:

a. Unlisten command ("?"): Resets the listen latch inhibiting the 59500A from functioning as a listener.

b. Multiprogrammer Listen Address (suggested listen address is "7"): Sets the listen latch and resets the talk latch. The 59500A can function as a listener but not as a talker.

c. Multiprogrammer Talk Address (suggested talk address is "W"): Sets the talk latch and resets the listen latch. The 59500A can function as a talker but not as a listener. The address switches select talk and listen addresses in pairs; a talk address of "W" corresponds to a listen address of "7".

d. Another bus device's talk address or an untalk command ("—"): Resets the talk latch inhibiting the multiprogrammer system from functioning as a talker.

8-11 After one of the above commands or addresses is decoded, the 59500A is enabled to function as a listener or as a talker; or is inhibited from functioning as either. The talk and listen latches will remain either both reset or one set and the other reset until another command is decoded. The talk or listen latch (whichever is set) is reset when the interface clear (IFC) bus line is true. The IFC line is used by the calculator to clear all devices on the HP-IB.

8-12 The serial poll decoder also receives data lines DIO7-DIO1 from the HP-IB. When the ATN line is true, the serial poll decoder controls the serial poll latch if one of the following commands is decoded:

a. Serial Poll Enable: Sets the serial poll latch enabling the 59500A to operate in the serial poll mode.

b. Serial Poll Disable: Resets the serial poll latch inhibiting the 59500A from operating in the serial poll mode.

8-13 Serial polling is a method of sequentially determining which device connected to the HP-IB has requested service. Service request and serial polling operations are described in Paragraphs 8-30 through 8-34.

8-14 A 3-wire handshake cycle occurs with each command character that is transferred from the calculator to every device on the HP-IB. In the command mode (ATN line true), the handshake cycle is automatically implemented by the timing and logic (listen) circuits in the 59500A. The 3-wire handshake process is described in Paragraph 8-35.

8-15 Listen Mode

8-16 When the 59500A listen address has been received (listen latch is set) and the ATN line goes false, the 59500A will interpret alpha and numeric ASCII characters from the HP-IB as data and process them accordingly. The data decoder (5 separate decoders) decodes ASCII characters on data bus lines DIO7-DIO1 as follows:

Letter Decoder – decodes "@" sign and letters from "A" through "O".

Number Decoder – decodes numbers from "0" through "7".

"T", "X", "Z" Decoders – decode gate codes "T", "X", and "Z" respectively.

8-17 When any one of the above characters is received, it is decoded and a handshake cycle between the 59500A and the calculator takes place. Unrecognized characters will be ignored but a handshake cycle between the 59500A and calculator will occur anyway (see Paragraph 8-35).

8-18 **Letter Decoder.** When the three MSB's (DIO7, 6, 5) specify the ASCII code for an "@" sign or a letter from "A" through "O", the letter decoder stores the remaining four bits (DIO4-DIO1) in the address latch. In addition, a clear signal is generated which presets the output data shift register to zero. The bits stored in the address latch represent a multiprogrammer card slot address or designate a multiprogrammer control word. The bits derived from the ASCII codes for "@" and "A" through "N" represent multiprogrammer card slot addresses 400 through 414, respectively. The bits derived from the ASCII code for "O" designate a multiprogrammer control word. If another "A" through "O" or "@" character is received, it replaces the previous character stored in the address latch.

8-19 **Number Decoder.** When the first four MSB's (DIO7-DIO4) specify the ASCII code for a number in the range from "0" through "7", the number decoder generates a clock pulse which loads bits DIO3-DIO1 into the bottom three positions of the shift register. As succeeding numbers are received by the number decoder, the data present in the data shift register is shifted up, three bits at a time, as the new data (number) is loaded in. Numbers are shifted in from the bottom up as they are received. Thus, the first three bits (octal number) received are the most significant output bits in the register. The data shift register holds 12-bits of data, $\overline{\text{B11}}$ (MSB) through $\overline{\text{BØØ}}$ (LSB) representing data input to the multiprogrammer. Thus, after four numbers (3-bits each) are received, the data shift register is filled. If less than four numbers are received by the decoder, the most significant bits of the data shift register will be logical zeros because of the prior clearing of the shift register (see Paragraph 8-18). Since some calculators do not transmit leading zeros, this feature of the shift register makes the transmission of leading zeros optional as they will have no effect. If more than four numbers are received, they will continue to be shifted into the data shift register, however, since only 12-bits can be retained, only the last four decoded numbers will be present and all previous numbers will be lost. The shift register is cleared when the IFC line is true.

8-20 "T", "X", and "Z" Decoders. An ASCII "T", "X", or "Z" gate code is decoded on data bus lines DIO7-DIO1 by the applicable decoder. The "T" code, included in all control and data words, generates the multiprogrammer gate signal. The control or data word bits $\overline{\text{B15}}$-$\overline{\text{BØØ}}$ (contents of the address latch and output data shift register) are sent to the multiprogrammer along with the gate. The gate signal initiates the multiprogrammer's data storage and/or processing functions specified by the associated control and data words. The "T" code can also be used in an address word to initiate a data strobe to activate the addressed input card, and then store return data bits $\overline{\text{B11}}$-$\overline{\text{BØØ}}$ from the addressed input card in the 59500A's input latch. Thus, when the "T" code is used in an address word, it designates a "read with gate" function. The "X" code (read without gate) is used in an address word when it is desired to store data from the addressed input card without gating (strobing) the card. When the "X" code is received, the return data ($\overline{\text{B11}}$-$\overline{\text{BØØ}}$) from an addressed input card is stored in the input latch. The "Z" code (read live data) is used in an address word to allow return data ($\overline{\text{B11}}$-$\overline{\text{BØØ}}$) from the addressed card to continually pass through the input latch to the encoder. When the "Z" code is received, the input latch is enabled and will remain enabled until a "T" or "X" is received. Thus, the "Z" code allows the contents of the input latch (encoder input) to be continually updated to reflect the return data from the addressed multiprogrammer input card. Control, data, and address words and the uses of the "T", "X", and "Z" gate codes are described in detail in Chapter IV.

8-21 The 3-wire handshake cycle for a "T", "X", or "Z" character is delayed 30μsec. When a "T", "X", or "Z" is decoded, the timing logic holds the NRFD bus line true for 30μsec preventing the calculator from sending another character until NRFD goes false. The delay allows the multiprogrammer time to process the data before the calculator sends new data. Note that if a "T" is received, the receipt of another character could be delayed for more than 30μsec since the multiprogrammer flag will override the 30μsec delay (see Paragraph 8-35).

8-22 If a "T" is decoded, the gate generator is set and the live data latch is reset. When set, the gate generator produces the multiprogrammer gate signal which strobes bits $\overline{\text{B15}}$-$\overline{\text{BØØ}}$ into the multiprogrammer. The multiprogrammer gate can only be generated when a "T" is decoded. The leading edge of the multiprogrammer flag resets the gate generator and the trailing edge stores the return data bits ($\overline{\text{B11}}$-$\overline{\text{BØØ}}$) in the input latch. The flag is also applied to the timing logic (listen) circuit to control the NRFD bus signal. In effect, when a "T" is decoded, the HP-IB 3-wire handshake is converted to the multiprogrammer 2-wire handshake (see Paragraph 8-35).

8-23 If an "X" is decoded, it will reset the gate generator, store data present on return data bit lines ($\overline{\text{B11}}$-$\overline{\text{BØØ}}$) in the input latch, and reset the live data latch. The ability to reset the gate generator with an "X" becomes necessary if the gate generator remains set when the multiprogrammer is operating in the interrupt mode. In the interrupt mode, the trailing edge of the flag signal, indicating an interrupt condition, resets the gate generator and activates the service request (SRQ) bus line. If the trailing edge of the flag is not received, the 59500A will "hang up" waiting for an interrupt indication with the gate generator set and the SRQ line false. Consequently, if the interrupt indication (SRQ line goes true) does not occur as anticipated, the calculator can send an "X" to reset the gate generator removing the "hang up" condition. Conditions for setting the SRQ line true are described in Paragraph 8-28.

8-24 When a "Z" is received, the live data latch is set causing the input latch to be continually enabled. For this condition the input latch continually reflects the status of return data bits ($\overline{\text{B11}}$-$\overline{\text{BØØ}}$) from the addressed multiprogrammer input card.

8-25 Talk Mode

8-26 Before the 59500A is placed in the talk mode, a control word with a "T" gate code must have previously been received defining the input mode of operation (ISL on). Also, an address word specifying a particular input card's slot address and an accompanying "T", "X", or "Z" gate code must have been previously sent to the 59500A. After the multiprogrammer system is in the input mode (ISL on) and the address word and gate code have been received, the data stored in the 59500A's input latch reflects the data returned ($\overline{\text{B11}}$-$\overline{\text{BØØ}}$) from the addressed input card. This information, along with bit $\overline{\text{B15}}$, the multiprogrammer $\overline{\text{IRQ}}$ bit, is sent to the encoder. At this point, the direction of the data flow on the HP-IB must be reversed by commanding the calculator to listen and the 59500A to talk.

8-27 When the 59500A talk address is received (talk latch is set) and the ATN line goes false, the encoder will translate bit 15 into one octal digit and bits $\overline{\text{B11}}$-$\overline{\text{BØØ}}$into four octal digits. Bit $\overline{\text{B15}}$ is encoded a 1 if $\overline{\text{IRQ}}$ is set and Ø if $\overline{\text{IRQ}}$ is not set. The $\overline{\text{IRQ}}$ bit is used by certain multiprogrammer input cards to indicate that they contain valid data. IRQ is also used to identify the interrupting card when multiple input cards are used in the interrupt mode. Bits $\overline{\text{B11}}$-$\overline{\text{BØ9}}$, $\overline{\text{BØ8}}$-$\overline{\text{BØ6}}$, $\overline{\text{BØ5}}$-$\overline{\text{BØ3}}$, and $\overline{\text{BØ2}}$-$\overline{\text{BØØ}}$ are encoded into ASCII codes for numbers ranging from "1" to "7". Data is transmitted from the encoder onto the HP-IB in the order given above starting with the octal digit representing bit $\overline{\text{B15}}$ and ending with the octal digit representing bits $\overline{\text{BØ2}}$-$\overline{\text{BØØ}}$. Carriage return and line feed are added after the last multiprogrammer octal digit ($\overline{\text{BØ2}}$-$\overline{\text{BØØ}}$) as an "end of record" indication to the calculator. As each character is transmitted, a 3-wire handshake occurs between the 59500A and the calculator (see Paragraph 8-35).

8-28 Service Request

8-29 The service request (SRQ) line is used by the 59500A to notify the calculator that the multiprogrammer requires service. Whenever the multiprogrammer is operating in the timing mode (TME on), the SRQ logic circuit in the 59500A is enabled and will set the SRQ line true upon receipt of the trailing edge of the multiprogrammer flag. As stated previously a multiprogrammer flag is only obtained when a "T" gate code is received by the 59500A. The three possible conditions which determine when the trailing edge of the flag will be received (with TME on) are as follows:

1. When using certain multiprogrammer input or output cards in serial (sequential) timing mode, TME must be programmed on (control word with a gate) before addressing a card with a gate. When the control word is received, the multiprogrammer will generate a handshake flag which will cause the 59500A to set the SRQ bus line true. Since this service request is caused by the control word, it must be cleared (see Paragraph 8-33) before addressing the card with a gate. Once the card has been addressed with a gate, it will control the flag line to the 59500A. When the 59500A receives the trailing edge of this flag, it sets SRQ true as an indication that the card has completed a data transfer.

2. When using multiple output cards in parallel (simultaneously) TME mode, the trailing edge of the multiprogrammer flag will not be returned to the 59500A until the last output card completes its data transfer operation.

3. When the multiprogrammer is in the interrupt mode (TME, $\overline{\text{IEN}}$ on), the multiprogrammer flag will not be received until a multiprogrammer card generates an interrupt. Note that for this condition (TME, $\overline{\text{IEN}}$ on), the NRFD line is not a function of the multiprogrammer flag. The TME and $\overline{\text{IEN}}$ control signals are sent to the timing logic (listen) to prevent the multiprogrammer flag from controlling the NRFD line in the interrupt mode. Thus, when cards interrupt, the resulting flag(s) will not affect the NRFD line, however, the trailing edge of the flag from the first interrupting card sets the SRQ bus line true.

8-30 Serial Poll

8-31 Serial poll is a method used by the calculator to determine which bus device has requested service (set SRQ bus line true). Essentially it consists of interrogating bus devices in sequence and reading back a status byte from each device which is used to identify the device(s) requesting service. When addressed to talk in the serial poll mode, the 59500A will return a status byte equal to decimal 64 if it is requesting service or a status byte of zero if it is not.

8-32 When a serial poll enable (SPE) command is received, it is latched in (see Paragraph 8-12) and sent to the encoder and the SRQ logic circuits. The SRQ logic circuit will send a status bit which indicates the state of the service request to the encoder. A logic 1 status bit, indicating SRQ is true, is translated into a decimal 64 by the encoder. A logic Ø status bit, indicating SRQ is false, is translated into a decimal Ø by the encoder.

8-33 When the serial poll enable latch is set and the 59500A talk address is received, the 59500A is placed in the serial poll active state (SPAS). For this condition, the SRQ logic circuit is reset and further transmission of SRQ on the HP-IB is inhibited. In addition, the 59500A transmits the status byte (decimal 64 or Ø) on data bus input/output lines DIO7-DIO1.

8-34 When the 59500A receives a serial poll disable command (SPD), an untalk command its listen address, or another devices talk address, the serial poll active state is terminated, preventing further transmission of the status byte.

8-35 3-Wire Handshake Process

8-36 As stated previously, a 3-wire handshake cycle occurs with each character transferred over the HP-IB data lines. The following paragraphs describe the 3-wire handshake process in greater detail. Figure 8-3 illustrates the 59500A interface with the DAV, NDAC, and NRFD HP-IB handshake lines. The handshake signal flow through the multiprogrammer system is also shown. A 3-wire handshake cycle occurs when the 59500A is receiving characters

Appendix A
NUMBER THEORY AND UTILITY SUBROUTINES

A-1 This appendix is divided into two major sections: (1) a brief discussion of number theory as it relates to use of the multiprogrammer system; and (2) a listing of utility subroutines which may be called from a main program to perform frequently required conversions or operations.

A-2 NUMBER THEORY

A-3 The decimal, the octal, and the binary number systems are all used to define, in numerical fashion, the data transferred within a calculator based multiprogrammer system. In addition, the user may desire to interface the multiprogrammer to external devices which send or receive data in binary coded decimal (BCD) form. Subroutines listed in Paragraph A-47 of this appendix will perform the various conversions required as part of a typical software system, but do not provide a substitute for a thorough understanding of the data code environment within which the system operates.

A-4 Decimal Numbers

A-5 Most of us are so familiar with the decimal number system that we have become largely unconscious of the underlying principles; at least so far as they relate the decimal system to other systems such as binary or octal. Any number system has what is called a "base", which is the number of unique symbols used in the particular system. Since it has ten unique symbols (0, 1, 2, - - - 9), the decimal system is the base 10 number system. Although it is often omitted since the number system is usually implied by the context, the base may be specified by a subscript:

$$235_{10} \tag{1}$$

A-6 This subscript simply tells us that the preceding number is a unique quantitative value presented in decimal form. If we examine the individual digits that make up a decimal number, it becomes clear that the number can be represented as follows:

$$\begin{aligned} 235_{10} = 2 \times 10^2 &= 2 \times 100 = 200 \\ 3 \times 10^1 &= 3 \times 10 \;\;= \;\;30 \\ 5 \times 10^0 &= 5 \times 1 \;\;\;= \;\;\underline{\;\;5} \\ && 235_{10} \end{aligned} \tag{2}$$

A-7 From (2) it may be seen that each digit is weighted by a power of the base (in this case 10), and that the power increases by one for each successive digit to the left. This relationship holds true for number systems to any base, and provides a longhand method for converting to base 10 from any other base.

A-8 Binary Numbers

A-9 The binary system, which is the system of machine language, is the base 2 number system, and has two unique symbols; "0" and "1".

$$11101011_2 \tag{3}$$

A-10 If we change (3) to the form used in (2), we get:

$$\begin{aligned} 11101011_2 = 1 \times 2^7 &= 1 \times 128 = 128 \\ 1 \times 2^6 &= 1 \times \;\;64 = \;\;64 \\ 1 \times 2^5 &= 1 \times \;\;32 = \;\;32 \\ 0 \times 2^4 &= 0 \times \;\;16 = \;\;\;\;0 \\ 1 \times 2^3 &= 1 \times \;\;\;\;8 = \;\;\;\;8 \\ 0 \times 2^2 &= 0 \times \;\;\;\;4 = \;\;\;\;0 \\ 1 \times 2^1 &= 1 \times \;\;\;\;2 = \;\;\;\;2 \\ 1 \times 2^0 &= 1 \times \;\;\;\;1 = \;\;\underline{\;\;\;\;1} \\ && 235_{10} \end{aligned} \tag{4}$$

A-11 Octal Numbers

A-12 The octal number system has eight unique symbols (0 through 7) and functions in a similar fashion:

$$353_8 \tag{5}$$

and:

$$\begin{aligned} 353_8 = 3 \times 8^2 &= 3 \times 64 = 192 \\ 5 \times 8^1 &= 5 \times \;\;8 = \;\;40 \\ 3 \times 8^0 &= 3 \times \;\;1 = \;\;\underline{\;\;\;3} \\ && 235_{10} \end{aligned} \tag{6}$$

A-13 The octal number system has another characteristic which makes it particularly useful to the programmer. Returning to binary numbers for a moment, we see that three binary digits allow us to represent eight unique values:

$$\begin{array}{lll}
111 = (1 \times 2^2) + (1 \times 2^1) + (1 \times 2^0) = 7 & & \\
110 = (1 \times 2^2) + (1 \times 2^1) + 0 & = 6 & \\
101 = (1 \times 2^2) + 0 + (1 \times 2^0) & = 5 & \\
100 = (1 \times 2^2) + 0 + 0 & = 4 & (7) \\
011 = 0 + (1 \times 2^1) + (1 \times 2^0) & = 3 & \\
010 = 0 + (1 \times 2^1) + 0 & = 2 & \\
001 = 0 + 0 + (1 \times 2^0) & = 1 & \\
000 = 0 + 0 + 0 & = 0 &
\end{array}$$

A-14 Since a single octal digit may also represent eight unique values, we can use octal digits to represent binary triads for any binary number regardless of its size. Conversion is by direct inspection:

$$\underbrace{101}_{5}\ \underbrace{111}_{7}\ \underbrace{100}_{4}\ \underbrace{011}_{3}{}_{2} \;{}_{8} \qquad (8)$$

The reverse process is just as simple:

$$\overset{7}{\overline{111}}\ \overset{6}{\overline{110}}\ \overset{1}{\overline{001}}\ \overset{5}{\overline{101}}{}_{2} \;{}^{8} \qquad (9)$$

A-15 The octal number is clearly easier to remember than its binary equivalent, yet is also readily converted to or derived from binary and is therefore more closely related to machine language than decimal.

A-16 Conversion from binary to octal by inspection is a particularly useful technique when TTL output or relay cards are used as switches in a calculator based multiprogrammer system. The octal equivalent of the binary value corresponding to a desired switch pattern will produce that pattern when combined with the card slot address and output to the multiprogrammer

A-17 Decimal Conversion Algorithms

A-18 Integer conversions between the decimal and either the binary or the octal systems are less obvious and considerably more tedious. Direct conversions from decimal to binary, and vice-versa, are not usually performed, but instead conversions between decimal and octal are used in conjunction with the conversion by inspection techniques shown in (8) and (9) to achieve the same results indirectly.

A-19 **Octal To Decimal.** Conversion from octal to decimal may be performed as follows:

1. Multiply the most significant octal digit by 8.

2. Add the next most significant octal digit and multiply the sum by 8.

3. Repeat step 2 until the least significant digit is reached.

4. Add the least significant octal digit but <u>do not</u> multiply the sum by 8.

For example:

$$\begin{array}{r}
5743_8 \\
\times \underline{\ 8} \\
40 \\
+ \underline{\ 7} \\
47 \\
\times \underline{\ 8} \\
376 \\
+ \underline{\ 4} \\
380 \\
\times \underline{\ 8} \\
3040 + 3 = 3043_{10}
\end{array} \qquad (10)$$

A-20 If we put (10) in a more general form, its relationship to (6) becomes apparent:

$$\begin{array}{r}
ABCD_Y = \\
\times Y \\
AY \\
+ B \\
AY + B \\
\times Y \\
AY^2 + BY \\
+ C \\
AY^2 + BY + C \\
\times Y \\
AY^3 + BY^2 + CY + D_{10}
\end{array}
\qquad
\begin{array}{l}
(A \times Y^3) + (B \times Y^2) + \\
(C \times Y^1) + (D \times Y^0)_{10} = \\
AY^3 + BY^2 + CY + D_{10}
\end{array} \qquad (11)$$

A-21 **Decimal To Octal.** Conversion from decimal to octal may also be reduced to a simple algorithm:

1. Divide the decimal number by 8, and write down the remainder.

2. Divide the quotient of step 1 by 8 and write down the remainder.

3. Repeat step 2 until the quotient is zero (the last remainder is also retained).

A typical example:

$$\begin{array}{rcl} 8\,\overline{)3981} & & \\ 8\,\overline{)497} & \text{r.} & 5 \\ 8\,\overline{)62} & \text{r.} & 1 \\ 8\,\overline{)7} & \text{r.} & 6 \\ 0 & \text{r.} & 7 \\ & & 7\;\;6\;\;1\;\;5_8 \end{array} \qquad (12)$$

A-22 Several operations are implied in (12) and if shown would appear as follows:

$$\begin{array}{rlr} 8\,\overline{)3981}_{10} & & \\ 8\,\overline{)497} & \longrightarrow 5 \times 10^0 = & 5 \\ 8\,\overline{)62} & \longrightarrow 1 \times 10^1 = & 10 \\ 8\,\overline{)7} & \longrightarrow 6 \times 10^2 = & 600 \\ 0 & \longrightarrow 7 \times 10^3 = & \underline{7000} \\ & & 7615_8 \end{array} \qquad (13)$$

A-23 When used in the form given in (13) this algorithm is fully reversible:

$$\begin{array}{rlr} 10\,\overline{)7615_8} & & \\ 10\,\overline{)761} & \longrightarrow 5 \times 8^0 = & 5 \\ 10\,\overline{)76} & \longrightarrow 1 \times 8^1 = & 8 \\ 10\,\overline{)7} & \longrightarrow 6 \times 8^2 = & 384 \\ 0 & \longrightarrow 7 \times 8^3 = & \underline{3584} \\ & & 3981_{10} \end{array} \qquad (14)$$

A-24 Negative Numbers

A-25 A number system is not really complete unless it makes some provision for negative values. The decimal system does so through the use of two additional symbols: the plus (+) and the minus (–) signs. Obviously, these two symbols serve a dual purpose since they not only indicate polarity, but also act as arithmetic operators. Any value within the range of $-\infty < n < +\infty$ may be more or less conveniently represented by this system of twelve symbols (i. e., the digits 0-9 and the "+" and "–" signs).

A-29 The relationship between the polarity indicating and the operative functions of the "+" and "–" signs is implicit in subtraction:

$$\begin{array}{r} 10 \\ -\;\underline{5} \\ 5 \end{array} \quad \text{or} \quad 10 - 5 = 5 \qquad (15)$$

is the same as:

$$10 + (-5) = 5 \qquad (16)$$

The sum of any number and its true negative is, of course, zero.

A-27 **Two's Complement Numbers.** The binary number system is useful precisely because it has only two symbols, and as a consequence, polarity cannot be indicated as it is in the decimal system without destroying the very property that permits implementation of binary codes in hardware. This reality together with the fact that from a hardware standpoint, it is generally easier to add than to subtract directly has encouraged use of the two's complement system for representing negative numbers in binary. The two's complement of any binary number is formed by:

1. Complementing each digit (changing all "0's" to "1's" and vice-versa).

2. Adding one.

Thus:

$$\begin{array}{rl} 011010011101 & \longleftarrow \text{Original Number} \\ \downarrow\quad & \\ 100101100010 & \longleftarrow \text{One's Complement} \\ +\;\underline{\qquad\qquad 1} & \\ 100101100011 & \longleftarrow \text{Two's Complement} \end{array} \qquad (17)$$

A-28 Subtraction using two's complement numbers is as follows:

$$\begin{array}{rrl} 12_{10} & 1100 & \\ -\;\underline{9_{10}} & +\;\underline{0111} & \longleftarrow \text{(Two's complement of } 1001_2) \\ 3_{10} & 1\,0011 & \\ & \downarrow\quad & \\ \text{(Last carry is ignored)} & 0011_2 & = 3_{10} \end{array} \qquad (18)$$

A-29 Returning to (17), we see that adding the original number and its two's complement produces:

$$\begin{array}{r} 011010011101 \\ +\;\underline{100101100011} \\ 1\;000000000000 \\ \downarrow\qquad \\ 000000000000_2 = \text{zero} \end{array} \qquad (19)$$

A-30 The result is zero and demonstrates the validity of treating the two's complement as a true negative of its root binary number.

A-31 At this writing, there are only two multiprogrammer cards which require manipulation of negative numbers: the 69321B Voltage D/A Card and the 69421A Voltage Monitor A/D Card.

A-32 69321B D/A Card

A-33 The 69321B card is a twelve bit biploar (two's complement) digital to analog converter and may be programmed to output a voltage in the range of −10.240 through +10.235 volts. The twelve bit binary word accepted by the D/A module allows for 4096_{10} (2^{12} = 4096) unique programmed codes. Since zero is one of the possibilities, however, the actual decimal values fall in the range of 0 through 4095. Presented graphically, the range of output voltages and the corresponding decimal, octal, and binary values appear as shown in Figure A-1.

A-34 If we take the absolute value of negative full scale and add it to positive full scale (10.240 + 10.235), we arrive at a value of 20.475 volts as the actual range of the module. This figure may then be divided by 4095_{10} to yield the analog equivalent value of the least significant bit (LSB), which in this case is exactly 0.005 volts or 5mV. All nominal output voltages will be an integer multiple of this 5mV value. Note that we divide by 4095 rather than 4096 since a range of X^n (2^{12}) values has X^n-1 ($2^{12}-1$ or 4095_{10}) intervals.

A-35 Armed with the knowledge that the LSB is 5mV, we may now divide any voltage value within the range of the converter by that figure and obtain the decimal equivalent of the binary code that will produce the desired output. For example:

$$\frac{5.000\text{V}}{.005\text{V}} = 1000_{10} \tag{20}$$

$$-\frac{6.745\text{V}}{.005\text{V}} = -1349_{10} \tag{21}$$

$$\frac{0.123\text{V}}{.005\text{V}} = 24.6 \tag{22}$$

A-36 The decimal equivalent values derived by the method shown in (20), (21), and (22) will always fall within the range of -2048_{10} to $+2047_{10}$ provided that the voltage to be programmed is within the specified range of −10.240 to +10.235 volts. The additional program statements included in the decimal to octal subroutine permit the program to accept negative and/or non-integer values by automatically rounding-off and scaling inputs to integer values within the range of 0_{10} to 4095_{10}. The round-off routine ignores the sign of the number (+1.5 becomes +2 and −1.5 becomes −2) and is accurate to nine decimal places. The scaling routine converts negative values to the positive decimal equivalent of the two's complement code corresponding to the desired negative voltage.

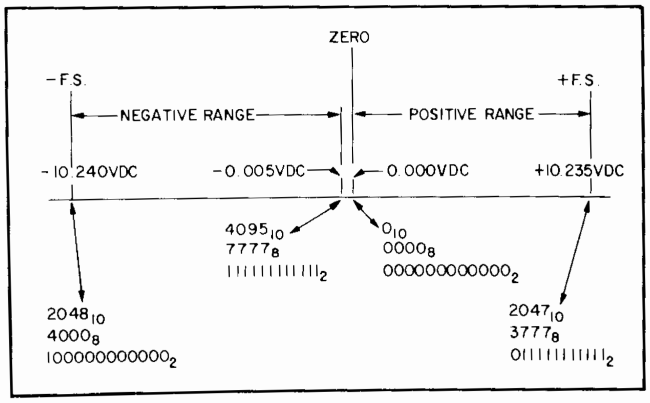


Figure A-1. D/A Voltage Converter Card, Output Range

A-37 69421A A/D Card

A-38 The conversions required by the 69421A voltage monitor card are basically the inverse of those just discussed. The process is simplified, however, since the octal value returned to the calculator by the 69421A card is always an integer in the range of 0_8 through 7777_8. The calculator's octal to decimal function converts this value to a decimal number in the range of 0 through 4095. Since the 69421A is a bipolar device and produces a two's complement code, the decimal number must be scaled and then multiplied by the analog equivalent value of the LSB in order to yield the actual decimal value of the applied voltage. This conversion is accomplished as follows:

1. If the decimal value is 2048_{10} or larger then subtract 4096_{10}; if not, go directly to step 2.

2. Multiply the decimal value from step 1 by the analog equivalent value of the LSB.

For example:

$$\begin{array}{r} 6065_8 \\ \downarrow \\ 3125_{10} \\ -\underline{4096_{10}} \\ -971_{10} \\ \times\ \underline{.005 \text{ volts}} \\ -4.855 \text{ volts} \end{array} \tag{23}$$

A-39 The sample test program for the 69421A (Example 24 in Chapter V) shows the appropriate statements (program lines 2 and 3) for the operations performed in (23) above.

A-40 BCD Conversions

A-41 Included in Paragraph A-47 are subroutines for converting from decimal to BCD - weighted octal and the reverse. These programs are primarily intended to permit transfer of numerical data to and from BCD programmable external devices via the 69331A digital output and the

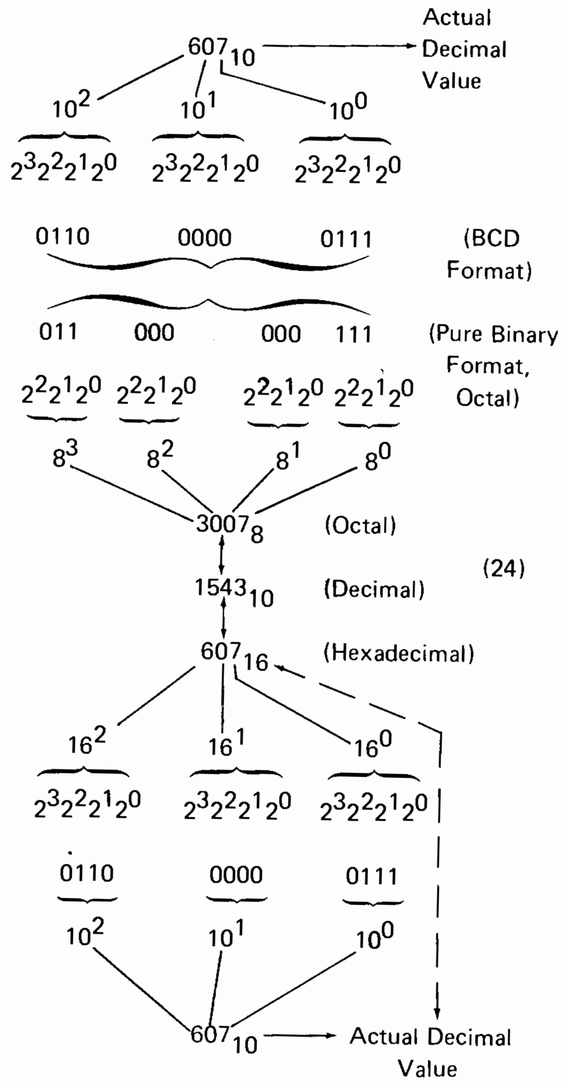
69431A digital input cards. The algorithm for these conversions is similar to that used for the base conversion routines discussed previously.

A-42 BCD To Decimal. For BCD to decimal conversions, the octal value corresponding to the twelve bit pattern produced by three decades of BCD data is first converted to decimal. This decimal value is BCD - weighted, however, and must undergo further conversion before the encoded actual decimal value is recovered. Since decimal equivalents of BCD codes and the hexadecimal (base 16) system are both higher order representations of binary quads (BCD is a truncated version of hexadecimal in the sense that it makes use of just ten of the sixteen unique combinations in a binary quad), the weighted decimal value will yield the actual decimal value when converted to base sixteen. The latter portion of the BCD to decimal subroutine performs this operation.

NOTE

The technique outlined here is based upon the assumption that an octal value representing a valid BCD code is input to the subroutine. Octal representations of full hexadecimal or invalid BCD codes will produce erroneous results.

A-43 Decimal To BCD. The decimal to BCD conversion is the inverse of the subroutine just described. Decimal numbers in the range of 0 through 999 may be converted to a BCD - weighted octal code using this program. The octal value will produce a twelve bit pattern corresponding to three decades of BCD coded numerical data.



(24)

A-44 The relationships between the various number systems are summarized for a decimal value of 607 in (24). Note that the change from BCD to pure binary format is a conversion which takes place in the hardware. It is this change in format that permits us to interpret what is actually a hexadecimal equivalent value (607_{16}) as a decimal value; we can in effect justify an implied conversion as a compensation for the hardware.

A-45 The algorithm as shown in (24) also takes advantage of the noncontiguous nature of BCD coding. Since we have limited the permissible inputs to valid BCD codes, certain octal, and likewise certain decimal, values do not occur. Conveniently, these particular values are the only ones within the twelve bit range of the multiprogrammer which would produce one or more of the hexadecimal characters A through F. The portion of the subroutine which converts from decimal to hexadecimal (or vice-versa) will not handle these values correctly, however, the constraints outlined above eliminate the need to do so.

A-46 Large BCD Numbers. Larger blocks of BCD data may be transferred in three decade slices via multiple cards if appropriate scaling operations are performed. The outline below illustrates the technique for an input transfer of nine decades (N is the recovered actual decimal value).

1. Read data from card #1 and store in X.
2. Go to BCD to decimal "BCDtd" subroutine.
3. F → N
4. Read data from card #2 and store in X.
5. Go to BCD to decimal subroutine.
6. (10 ↑ 3 * F) + N → N
7. Read data from card #3 and store in X.
8. Go to BCD to decimal "BCDtd" subroutine.
9. (10 ↑ 6 * F) + N → N

A-47 UTILITY SUBROUTINES

A-48 Table A-1 lists the utility subroutines, labels their function, and indicates the entry and exit variables used to connect the subroutine to the main program. The actual program listings follow in order.

Table A-1. Subroutine Summary

Subroutine	Function	Entry Variable	Exit Variable
"BIT"	Display Bits for Card Test Programs	X	
"dtBCD"	Decimal To BCD Weighted Octal Conversion	A	E
"BCDtd"	BCD Weighted Octal to Decimal Conversion	X	F

Subroutines

```
0: "BIT":if X#0;jmp 2
1: dsp "NO DATA RECEIVED";jmp 5
2: for Z=11 to 0 by -1
3: 2↑Z→Y;if X-Y>-.5;jmp 2
4: next Z;jmp 2
5: fxd 0;dsp "BIT",Z,"PIN",Z+1;X-Y→X;jmp -1
6: fxd .2;ret
*10052
```

```
0: "dtBCD":dto(A+6int(A/10)+96int(A/100))→E;ret
*23562
```

```
0: "BCDtd":otdX→X;X-6int(X/16)-60int(X/256)→F;ret
*8734
```

HEWLETT hp PACKARD

59500-90005

Printed in U. S. A.