STIMULUS PROGRAM NAME: RAMSELECT1
DESCRIPTION: SIZE: 267 BYTES

			Response Data				
Node Signal Src	Learned With	SIG	Async LVL	Clk LVL	Counter Mode	Counter Range	Priority Pin
U58-3	I/O MODULE	024F	1 0		TRANS		
U58-6	I/O MODULE	01B6	1 0		TRANS		
U59-6	I/O MODULE	01B6	1 0		TRANS		
U61-11	I/O MODULE	03F9	1 0		TRANS		
U60-2	I/O MODULE	024F	1 0		TRANS		
U60-7	I/O MODULE	01B6	1 0		TRANS		
U60-14	I/O MODULE	01B6	1 0		TRANS		
U59-9	I/O MODULE	03F9	1 0		TRANS		
U63-8	I/O MODULE	01B6	1 0		TRANS		
U19-6	I/O MODULE	024F	1 0		TRANS		
U24-6	I/O MODULE	01B6	1 0		TRANS		
U64-10	I/O MODULE	024F	1 0		TRANS		
U59-10	I/O MODULE	0000	1 0		TRANS		

Figure 4-35: Response File *(ramselect1)*

```
program ramselect2

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! STIMULUS PROGRAM characterizes RAM select logic.                      !
!                                                                       !
! Stimulus programs and response files are used by GFI to backtrace     !
! from a failing node.  The stimulus program must create repeatable UUT !
! activity and the response file contains the known-good responses for  !
! the outputs in the UUT that are stimulated by the stimulus program.   !
!                                                                       !
! Ramselect2 is used to stimulate the RAM select circuitry after the    !
! decoders.  The stimulus is a combination of reads that will ensure    !
! the decoder and related circuitry is working properly.  Ramselect2    !
! differs for ramselect1 because setoffset is required to delay the     !
! data due to signal propogation though the number of parts in the      !
! ram decode circuitry.                                                 !
!                                                                       !
! TEST PROGRAMS CALLED:                                                 !
!     (none)                                                            !
!                                                                       !
! GRAPHICS PROGRAMS CALLED:                                             !
!     (none)                                                            !
!                                                                       !
! Global Variables Modified:                                            !
!     (none)                                                            !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Main Declarations                                                     !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
    declare numeric bias = 999957

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!   FAULT HANDLERS:                                                     !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

handle pod_timeout_enabled_line
   recover()
end handle
handle pod_timeout_recovered
   recover()
end handle
handle pod_timeout_no_clk
end handle

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!   Main part of STIMULUS PROGRAM                                       !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

! Let GFI determine the measurement device.

    if (gfi control) = "yes" then
        devname = gfi device
```

(continued on the next page)

Figure 4-36: Stimulus Program *(ramselect2)*

```
    else
        devname = "/modl"
    end if
    print "Stimulus Program RAMSELECT2"

! Set addressing mode and setup measurement device.

    mem_word = getspace space "memory", size "word"
    mem_byte = getspace space "memory", size "byte"
    reset device devname
    sync device devname, mode "pod"
    sync device "/pod", mode "data"

! Store calibration offset, set new offset
! Display warning message if setting new offset fails

    cal_offset = getoffset device devname
    if (setoffset device devname, offset bias) = 0 then
       fault 'setoffset returned a bad status, fatal error'
    end if

! Present stimulus to UUT.

    arm device devname
       setspace (mem_word)
       read addr $1A5A4
       read addr $F0000
       read addr $F0000
       read addr $5A5A
       read addr $F0000
       read addr $F0000
       write addr $7BDE, data $1234
       read addr $F0000
       write addr $15A5A, data $9876
       read addr $F0000

       setspace (mem_byte)
       read addr 1
       read addr 2
       read addr 3
       write addr 4, data 0
       write addr 5, data $12
       read addr $1111
       read addr $11111
       read addr $AAAA
    readout device devname

! Restore original calibration offset

    setoffset device devname, offset cal_offset
end program
```

Figure 4-36: Stimulus Program *(ramselect2) - continued*

```
STIMULUS PROGRAM NAME: RAMSELECT2
DESCRIPTION:                                          SIZE:            114 BYTES

                 ------------------- Response Data --------------------
  Node          Learned            Async Clk Counter                          Priority
Signal Src       With        SIG    LVL  LVL  Mode     Counter Range             Pin

u58-8         I/O MODULE  B6FD    1 0        TRANS
u58-11        I/O MODULE  B603    1 0        TRANS
u62-8         I/O MODULE  F963    1 0        TRANS
u57-12        I/O MODULE  F99D    1 0        TRANS
```

Figure 4-37: Response File *(ramselect2)*

```
program refsh_addr

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! STIMULUS PROGRAM characterizes the refresh circuitry.                  !
!                                                                        !
! Stimulus programs and response files are used by GFI to backtrace      !
! from a failing node.  The stimulus program must create repeatable UUT !
! activity and the response file contains the known-good responses for  !
! the outputs in the UUT that are stimulated by the stimulus program.   !
!                                                                        !
! TEST PROGRAMS CALLED:                                                  !
!    check_meas (device, start, stop, clock, enable)                     !
!                                         Checks to see if the measure- !
!                                         ment is complete using the    !
!                                         TL/1 checkstatus command.  If !
!                                         the measurement times out then!
!                                         redisplay connect locations.  !
!                                                                        !
! GRAPHICS PROGRAMS CALLED:                                              !
!    (none)                                                              !
!                                                                        !
! Local Variables Modified:                                              !
!    done                                 returned from check_meas()    !
!    devname                              Measurement device            !
!                                                                        !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Main Declarations                                                      !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

   declare numeric done = 0

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!   Main part of STIMULUS PROGRAM                                        !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

! Let GFI determine the measurement device.

   if (gfi control) = "yes" then
      devname = gfi device
   else
      devname = "/mod1"
   end if
   print "Stimulus Program REFSH_ADDR"
```

(continued on the next page)

Figure 4-38: Stimulus Program *(refsh_addr)*

```
! Set addressing mode and setup measurement device.

   setspace space (getspace space "memory", size "word")
   reset device devname
   sync device devname, mode "ext"
   enable device devname, mode "always"
   edge device devname, start "+", stop "-", clock "-"

! Prompt user to connect external lines.

   connect device devname, start "U67-9", stop "U67-9", clock "U63-8", common "gnd"

! External lines determine measurement.
! check_meas times out and reprompts if external lines aren't connected

   loop until done = 1
      arm device devname
         done = check_meas(devname, "U67-9", "U67-9", "U63-8", "*")
      readout device devname
   end loop

end program
```

Figure 4-38: Stimulus Program *(refsh_addr) - continued*

STIMULUS PROGRAM NAME: REFSH_ADDR
DESCRIPTION: SIZE: 182 BYTES

Node Signal Src	Learned With	SIG	Async LVL	Clk LVL	Counter Mode	Counter Range	Priority Pin
			Response Data				
u67-15	I/O MODULE	96EC	1	0	TRANS		
u67-1	I/O MODULE	AFC1	1	0	TRANS		
u67-2	I/O MODULE	4A2C	1	0	TRANS		
u67-3	I/O MODULE	25AF	1	0	TRANS		
u67-4	I/O MODULE	ACDE	1	0	TRANS		
u67-5	I/O MODULE	122D	1	0	TRANS		
u67-6	I/O MODULE	EEA6	1	0	TRANS		
u67-7	I/O MODULE	68F8	1	0	TRANS		

Figure 4-39: Response File *(refsh_addr)*

```
program refsh_time

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! STIMULUS PROGRAM characterizes the refresh timing.                          !
!                                                                             !
! Stimulus programs and response files are used by GFI to backtrace           !
! from a failing node.  The stimulus program must create repeatable UUT       !
! activity and the response file contains the known-good responses for       !
! the outputs in the UUT that are stimulated by the stimulus program.         !
!                                                                             !
! TEST PROGRAMS CALLED:                                                       !
!    check_meas (device, start, stop, clock, enable)                          !
!                                        Checks to see if the measure-        !
!                                        ment is complete using the           !
!                                        TL/1 checkstatus command.  If        !
!                                        the measurement times out then       !
!                                        redisplay connect locations.         !
!                                                                             !
! GRAPHICS PROGRAMS CALLED:                                                   !
!    (none)                                                                   !
!                                                                             !
! Local Variables Modified:                                                   !
!    done                                returned from check_meas()           !
!    devname                             Measurement device                   !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Main Declarations                                                           !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

   declare numeric done = 0

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!   Main part of STIMULUS PROGRAM                                             !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

! Let GFI determine the measurement device.

   if (gfi control) = "yes" then
      devname = gfi device
   else
      devname = "/mod1"
   end if
   print "Stimulus Program REFSH_TIME"
```

(continued on the next page)

Figure 4-40: Stimulus Program *(refsh_time)*

```
! Set addressing mode and setup measurement device.

   setspace space (getspace space "memory", size "word")
   reset device devname
   sync device devname, mode "ext"
   enable device devname, mode "always"
   edge device devname, start "+", stop "count", clock "-"
   stopcount device devname, count 48

! Prompt user to connect external lines.

      connect device devname, start "U67-13", clock "U13-1", common "gnd"

! External lines determine measurement.
! check_meas times out and reprompts if external lines aren't connected.

   loop until done = 1
      arm device devname
         done = check_meas(devname, "U67-13", "*", "U13-1", "*")
      readout device devname
   end loop

end program
```

Figure 4-40: Stimulus Program *(refsh_time) - continued*

STIMULUS PROGRAM NAME: REFSH_TIME
DESCRIPTION: SIZE: 195 BYTES

Node Signal Src	Learned With	SIG	Async LVL	Clk LVL	Counter Mode	Counter Range	Priority Pin
u59-9	I/O MODULE	159A	1 0		TRANS		
u64-13	I/O MODULE	909A	1 0		TRANS		
u44-5	I/O MODULE	87E6	1 0		TRANS		
U44-6	PROBE	DE42	1 0		TRANS		
u44-6	I/O MODULE	DE42	1 0		TRANS		
u59-10	I/O MODULE	4C3E	1 0		TRANS		
U44-9	PROBE	43F3	1 0		TRANS		
u44-9	I/O MODULE	43F3	1 0		TRANS		
u44-8	I/O MODULE	1A57	1 0		TRANS		
u61-11	I/O MODULE		1 0		TRANS		
u43-11	I/O MODULE		1 0		TRANS		

Figure 4-41: Response File *(refsh_time)*

```
program refsh_u56

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! STIMULUS PROGRAM characterizes the refresh circuitry.                     !
!                                                                           !
! Stimulus programs and response files are used by GFI to backtrace        !
! from a failing node.  The stimulus program must create repeatable UUT    !
! activity and the response file contains the known-good responses for     !
! the outputs in the UUT that are stimulated by the stimulus program.       !
!                                                                           !
! TEST PROGRAMS CALLED:                                                     !
!    check_meas (device, start, stop, clock, enable)                        !
!                                         Checks to see if the measure-    !
!                                         ment is complete using the        !
!                                         TL/1 checkstatus command.  If    !
!                                         the measurement times out then!
!                                         redisplay connect locations.     !
!                                                                           !
! GRAPHICS PROGRAMS CALLED:                                                 !
!    (none)                                                                 !
!                                                                           !
! Local Variables Modified:                                                 !
!    done                                 returned from check_meas()        !
!    devname                              Measurement device                !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Main Declarations                                                         !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

   declare numeric done = 0

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!   Main part of STIMULUS PROGRAM                                           !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

! Let GFI determine the measurement device.

   if (gfi control) = "yes" then
      devname = gfi device
   else
      devname = "/mod1"
   end if
   print "Stimulus Program REFSH_U56"
```

(continued on the next page)

Figure 4-42: Stimulus Program *(refsh_u56)*

```
! Set addressing mode and setup measurement device.

   setspace space (getspace space "memory", size "word")
   reset device devname
   sync device devname, mode "ext"
   enable device devname, mode "always"
   edge device devname, start "+", stop "count", clock "+"
   stopcount device devname, count 48

! Prompt user to connect external lines.

   connect device "/mod1", start "U67-13", clock "U13-1", common "gnd"

! External lines determine measurement.
! check_meas times out and reprompts if external lines aren't connected.

   loop until done = 1
      arm device devname
         done = check_meas(devname, "U67-13", "*", "U13-1", "*")
      readout device devname
   end loop

end program
```

Figure 4-42: Stimulus Program *(refsh_u56) - continued*

STIMULUS PROGRAM NAME: REFSH_U56
DESCRIPTION: SIZE: 63 BYTES

Node Signal Src	Learned With	SIG	Async LVL	Clk LVL	Counter Mode	Counter Range	Priority Pin
	------------------- Response Data				--------------------		
U56-12	PROBE		1	0	TRANS	1	
u56-12	I/O MODULE		1	0	TRANS	1	

Figure 4-43: Response File *(refsh_u56)*

Summary of Complete Solution for Dynamic RAM Timing 4.4.7.

The entire set of programs and files needed to test and GFI troubleshoot the Dynamic RAM Timing functional block is shown below. The format below is similar to a 9100A/9105A UUT directory (you could consider the functional block to be a small UUT), but in addition shows the use of each program and the location in this manual for each file.

UUT DIRECTORY
(Complete File Set for Dynamic RAM Timing)

Programs (PROGRAM):

TST_REFRSH	Functional test	Section 4.4.5
CAS_STIM	Stimulus Program	Figure 4-30
RAS_STIM	Stimulus Program	Figure 4-32
RAMSELECT1	Stimulus Program	Figure 4-34
RAMSELECT2	Stimulus Program	Figure 4-36
REFSH_ADDR	Stimulus Program	Figure 4-38
FREQUENCY	Stimulus Program	Figure 4-117
REFSH_TIME	Stimulus Program	Figure 4-40
REFSH_U56	Stimulus Program	Figure 4-42

Stimulus Program Responses (RESPONSE):

CAS_STIM	Figure 4-31
RAS_STIM	Figure 4-33
RAMSELECT1	Figure 4-35
RAMSELECT2	Figure 4-37
REFSH_ADDR	Figure 4-39
FREQUENCY	Figure 4-118
REFSH_TIME	Figure 4-41
REFSH_U56	Figure 4-43

Node List (NODE):

NODELIST	Appendix A

Text Files (TEXT):

Reference Designator List (REF):

REFLIST	Appendix B

Compiled Database (DATABASE):

GFIDATA	Compiled by the 9100A

(This page is intentionally blank.)

PARALLEL INPUT/OUTPUT FUNCTIONAL BLOCK 4.5.

Introduction to Parallel I/O 4.5.1.

Parallel I/O implementations range in complexity from simple latches to LSI components. This section covers two basic types of parallel I/O circuits, simple discrete I/O circuits, and common LSI components like Programmable Interface Adapters (PIA) and Programmable Interval Timers (PIT).

Parallel I/O is one of a microcomputer's interfaces to the real world. The microcomputers in products like cash registers, copiers, telephone switching equipment, electronic instruments, and personal computers often monitor and control optical or electromechanical components like LEDs, displays, keyboards, optical switches, printers, disk or tape drives. Often, the interface to these components from the microprocessor's perspective is a set of registers to which it can read and write data.

Output lines may be connected to recording or display devices, which can be damaged if random data is written indiscriminately to them. Signals controlled by output ports can produce voltages or actuate devices that can pose a threat to human safety. Care should be taken in designing stimulus programs when the possibility of injury to people or damage to equipment can result.

Considerations for Testing and Troubleshooting 4.5.2.

Programmable LSI Components

Programmable LSI components usually contain internal registers which characterize the component to a particular circuit application. Among the ways in which these components can be programmed are:

- Set internal operating modes.
- Configure I/O ports as inputs or outputs.
- Set edge polarity on edge-sensitive inputs.
- Enable or disable interrupts.
- Establish data exchange protocol.

When testing LSI components, it is necessary to initialize them first. Initialization usually consists of a series of reads from and writes to internal registers. It is useful to create a separate 9100A initialization program which can be called from various stimulus programs, or from the operator's keyboard.

If a component, such as a PIA, does not work properly after initialization, check the inputs that affect its operation, such as chip-select lines, read and write lines, register-select lines, and clocks. Signals that reset, gate, or set outputs to high impedance might also be suspect. If these inputs all appear good, the bus cycles accessing the component may not have the proper number of wait states.

To verify operation of the component, stimulus commands such as *rampdata, read,* and *write* can be used in combination with I/O-module measurements. For troubleshooting both inputs and outputs on devices such as LEDs and keyboards, it is often necessary to prompt the operator to interact with the UUT. Simple commands prompting operator action can be included in stimulus programs and displayed on the operator's display.

Outputs can be tested with *write*, *toggledata*, or *rampdata* commands. Responses can be read as signatures or as asynchronous or clocked level history. Signatures are useful for identifying outputs that are tied to each other. If there is not an appropriate clock available, transition counts or level history can be used.

Inputs can be verified by reading the component. To exercise all states of the input lines, some type of stimulus must be applied. If the circuit allows, the inputs can be overdriven to each logic state with the I/O module. For electromechanical devices such as keys and switches, interaction with the person performing a test may be required. Switch testing can be automated by using solenoids to actuate the switches.

Discrete I/O

Components used for discrete I/O include buffers, latches, addressable latches, and flip-flops. Such components usually have simpler interfaces to the microprocessor than programmable LSI components and they are handled in a similar manner, but their initialization procedures are different, if required at all.

If data does not appear to be reaching I/O latches, or is not read from I/O buffers, it may be necessary to check the address decoding logic to verify that the proper control signals are present. Here are some common problems associated with discrete I/O:

- *Outputs* may be loaded by external devices. Such outputs may work properly when disconnected. The loading problem may be associated with the external device, or with its connector.
- *Inputs* may be damaged by static electricity when they are disconnected from the signal sources and left unprotected.
- *Clocked inputs* on components like latches or flip-flops may be faulty.
- *Reset inputs* may either be stuck, forcing outputs to some state, or open, preventing circuits from being initialized.
- *Pullup or pulldown resistors* that establish static logic levels may be open, creating indeterminate inputs.

Parallel I/O Example 4.5.3.

The Programmable Interface Adapter on the Demo/Trainer UUT (U31) is shown in Figure 4-44. It can be programmed for operation with three ports, each with eight data lines. Each port is addressed for read or write by address lines IA01 and IA02. Ports A (lines PAO-7) and B (lines PB0-7) are used for outputs to the two on-board seven-segment LEDs. Port A corresponds to the upper LED, port B corresponds to the lower LED, and port C (lines PC0-7) is used for inputs from the four push-button switches.

Keystroke Functional Test 4.5.4.

Part A:

1. Initialize the Parallel I/O functional block using the WRITE key with the following commands:

```
WRITE DATA 89 TO ADDR 4006
... (ADDR OPTION: I/O BYTE)
WRITE DATA FF TO ADDR 4000
... (ADDR OPTION: I/O BYTE)
WRITE DATA FF TO ADDR 4002
... (ADDR OPTION: I/O BYTE)
```

2. Use the WRITE key to write values to the PIA chip. Read the resulting numbers on LED A. The values to be written and the results to be displayed are shown in the Response table in Figure 4-44.

```
WRITE DATA <see response table> TO ADDR 4000
... (ADDR OPTION: I/O BYTE)
```

3. Now use the WRITE key to write values to the PIA chip to display numbers on LED B. The values to be written and the results to be displayed are shown in the Response table in Figure 4-44.

```
WRITE DATA <see response table> TO ADDR 4002
... (ADDR OPTION: I/O BYTE)
```

Part B:

1. Use the READ key to read values resulting from pressing the UUT keys 1 through 4. The response table in Figure 4-45 shows the values that should be read for each key pressed.

```
READ ADDR 4004 = <see response table>
... (ADDR OPTION: I/O BYTE)
```

Keystroke Functional Test (Part A)

CONNECTION TABLE

STIMULUS	MEASUREMENT
POD	VISUAL INSPECTION
TEST ACCESS SOCKET	LEDA LEDB

STIMULUS AND RESPONSE TABLE FOR LEDA

DATA WRITTEN TO ADDRESS 4000	LEDA SHOWS
C0	0
79	.1
A4	2
30	.3
99	4
12	.5

STIMULUS AND RESPONSE TABLE FOR LEDB

DATA WRITTEN TO ADDRESS 4002	LEDB SHOWS
C0	0
79	.1
A4	.2
30	.3
99	4
12	.5

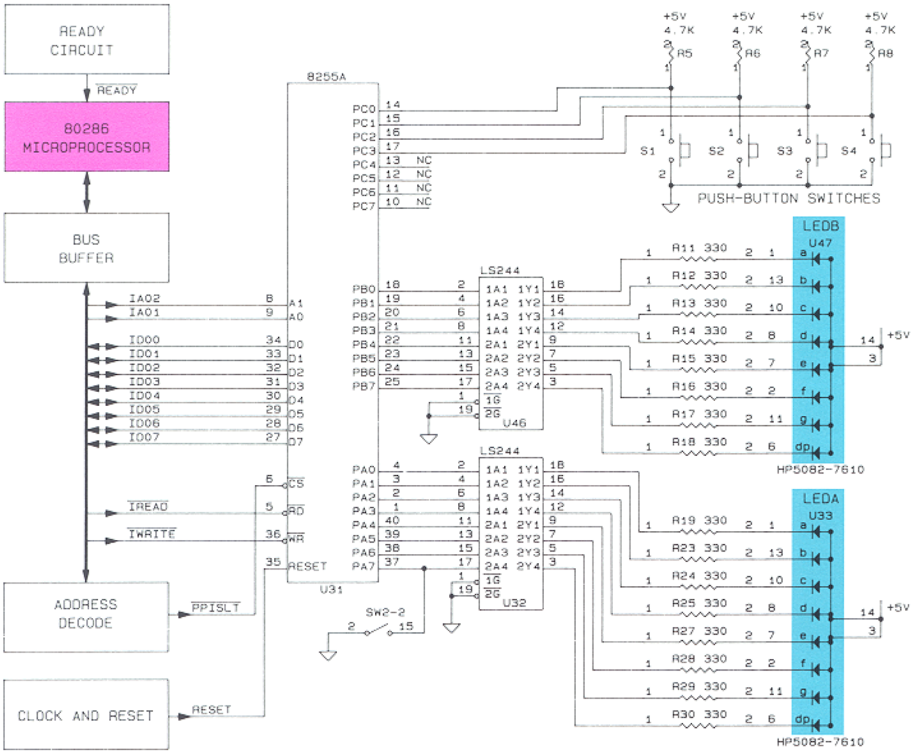


Figure 4-44: Parallel I/O Functional Test (Part A)

Keystroke Functional Test (Part B)

CONNECTION TABLE

STIMULUS	MEASUREMENT
PUSH-BUTTON SWITCHES	POD
51 52 53 54	TEST ACCESS SOCKET

STIMULUS AND RESPONSE TABLE FOR LEDA

PUSH-BUTTON PRESSED	DATA READ AT ADDRESS 4004
NONE	F
S1	E
S2	D
S3	B
S4	7
ALL	0

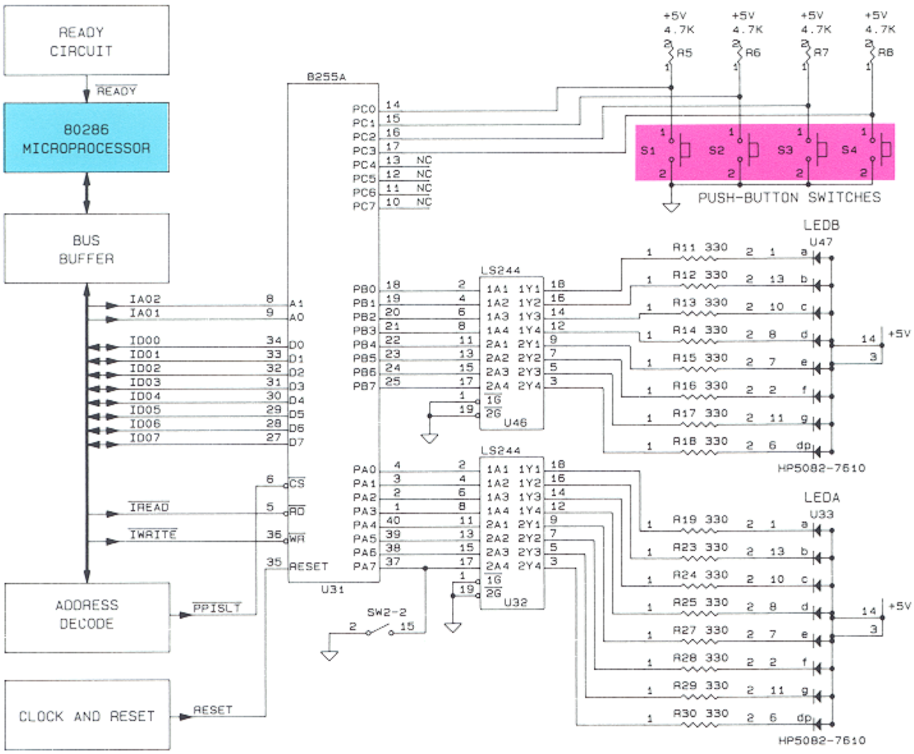


Figure 4-45: Parallel I/O Functional Test (Part B)

Programmed Functional Test 4.5.5.

The *test_pia* program is the programmed functional test for the Parallel I/O functional block. The program asks the test operator to check the visual properties of the LEDs that are driven by the PIA chip and also to check the mechanical operation of the pushbutton switches.

The program displays a message to the operator to watch LED A while the program displays numbers 1 through 9 on it. The operator is prompted to acknowledge proper operation or failing operation. If the LED fails, the *gfi test* command is used to test the LED drivers. If the LED drivers fail, GFI takes control and backtraces to the source of the failure. The same operation is then repeated for LED B.

Next, the operator is prompted to press key 1. The program polls the PIA chip and determines when the operator has pushed the key 1 button (if the key and the PIA are working properly). If the PIA cannot sense that the operator has pressed the key, the operator is instructed to press a 9100A/9105A key to indicate a failure. When the operator indicates a failing key, the *gfi test* command is used to verify correct signal levels at the key output. If a failure exists, GFI takes control and backtraces to the source of the failure. The same operation is repeated for keys 2, 3 and 4.

```
program test_pia

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! FUNCTIONAL TEST of the PARALLEL I/O functional block.                  !
!                                                                        !
! This program tests the PARALLEL I/O functional block of the            !
! Demo/Trainer.  The two LEDs and the four pushbutton switches are       !
! tested.  The test operator is prompted to visually inspect the LEDs    !
! as the LEDs count a series of numbers.                                 !
!                                                                        !
! TEST FUNCTIONS CALLED:                                                 !
!    keys       (key_number)              Test Demo/Trainer pushbutton   !
!                                         key key_number.  Prompt test   !
!                                         operator to push the key.      !
!                                                                        !
!    leds       (led_addr, led_name)      Test Demo/Trainer LED led_name!
!                                         which is driven by the PIA and!
!                                         has the address led_addr.      !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Functions                                                                   !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

function keys(keynum)
   declare numeric keynum                          ! Number of key to test.
   declare string norm = "\1B[0m"                  ! Normal video escape string
   declare string rev  = "\1B[0;7m"                ! Reverse video escape string
   declare string entry
   declare string fail = ""
   declare global numeric tlb
   declare global numeric tli

   mask = setbit(keynum - 1)

   loop until fail = chr($D)                               ! loop until YES key
      print on tlb ,"\nlPress ", rev," UUT KEY ", keynum," ",norm," pushbutton"
      print on tlb ,"Press any 9100 key if test is stuck"
      loop until (poll channel tli, event "input") = 1
         if ((read addr $4004) and mask) = 0 then return
      end loop
      loop until (poll channel tli, event "input") = 0    ! Flush input buffer
         input on tli ,entry
      end loop
      print on tlb ,"\nlPress ",rev," YES ",norm," to fail KEY ",keynum," test,"
      print on tlb ,"Press "+rev+" NO  "+norm+" to continue key test,"
      input on tli ,fail
   end loop
   print on tlb ,"\nl\nl"
   fault                                          ! Fail Key test (set termination
end function                                      ! status of function to fail.

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

function leds(led_addr, led_name)
   declare numeric led_addr
   declare string led_name
   declare string key
   declare string norm = "\1B[0m"
   declare string bold = "\1B[1m"
   declare string rev = "\1B[7m"
   declare string clear_screen = "\1B[2J"
   declare string no_auto_linefeed = "\1B[20h"
   declare global numeric tli
   declare numeric array [0:10] numbers

   numbers [0] = $C0    \    numbers [5] = $92
   numbers [1] = $F9    \    numbers [6] = $82
   numbers [2] = $A4    \    numbers [7] = $F8
   numbers [3] = $B0    \    numbers [8] = $80
   numbers [4] = $99    \    numbers [9] = $98
   NO = chr($7F)        \    YES = chr( $D)

   print norm, clear_screen, "Watch LED ", led_name, " count"
   print "Press ", rev, " ENTER ", norm, " key to start LED counting."
   input  key
   print clear_screen

   for i = 0 to 9
      write addr led_addr, data numbers [i]
      wait time 500
   next
```

```
   write addr led_addr, data $7F
   print  clear_screen, "\1B[201"
   print  "\1B[1;1fDid LED ", led_name, " display ALL segments off, then"
   print  "\1B[2;1fdigits 0 to 9, then only the Decimal Point ?"
   print  "\1B[3;fpress: "+rev+" YES "+norm+" or "+rev+" NO  "+norm
   loop until key = YES or key = NO
      input on tli ,key
      if key = NO then fault
   end loop
   write addr led_addr, data $FF \  print  no_auto_linefeed,clear_screen

end function

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! PARALLEL I/O Test.                                                    !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

   tlb = open device "/term1", as "update", mode "buffered"
   tli = open device "/term1", as "input", mode "unbuffered"
   execute pia_init()

   if leds($4000, "A") fails then fault 'PIA LED A failed' \ return
   if leds($4002, "B") fails then fault 'PIA LED B failed' \ return

   if keys(1) fails then fault 'PIA KEY 1 failed' \ return
   if keys(2) fails then fault 'PIA KEY 2 failed' \ return
   if keys(3) fails then fault 'PIA KEY 3 failed' \ return
   if keys(4) fails then fault 'PIA KEY 4 failed' \ return

end program
```

Stimulus Programs and Responses 4.5.6.

Figure 4-46 is the stimulus program planning diagram for the Parallel I/O functional block. The Parallel I/O stimulus programs only measure the electrical parameters of the Parallel I/O circuit; the visual properties of the LEDs are not measured.

The *ram_data* stimulus program outputs data from the PIA onto the data bus. The *pia_leds* stimulus program exercises outputs going to the LEDs. The *key_1*, *key_2*, *key_3*, and *key_4* stimulus programs monitor the operation of the four numbered pushbutton switches.

All the stimulus programs execute the *pia_init* program before any measurements are made on the PIA circuitry.

(This page is intentionally blank.)

Stimulus Program Planning

PROGRAM: PIA_DATA
EXECUTES PIA_INIT AND READS DATA FROM PIA REGISTERS
MEASUREMENT AT: U31-34,33,32,31,30,29,28,27

PROGRAM: PIA_LEDS
EXECUTES PIA_INIT AND EXERCISES OUTPUTS TO LEDs
MEASUREMENT AT: U31-18,19,20,21,22,23,24,25 U46-18,16,14,12,9,7,5,3 R11-2, R12-2, R13-2, R14-2 R15-2, R16-2, R17-2, R18-2 U31-4,3,2,1,40,39,38,37 U32-18,16,14,12,9,7,5,3 R19-2, R23-2, R24-2, R25-2 R27-2, R28-2, R29-2, R30-2

PROGRAM: KEY_1
EXECUTES PIA_INIT AND MONITORS LEVELS AND TRANSITIONS AFTER PROMPTING THE OPERATOR TO PRESS KEY 1
MEASUREMENT AT: R5-1

PROGRAM: KEY_2
EXECUTES PIA_INIT AND MONITORS LEVELS AND TRANSITIONS AFTER PROMPTING THE OPERATOR TO PRESS KEY 2
MEASUREMENT AT: R6-1

PROGRAM: KEY_3
EXECUTES PIA_INIT AND MONITORS LEVELS AND TRANSITIONS AFTER PROMPTING THE OPERATOR TO PRESS KEY 3
MEASUREMENT AT: R7-1

PROGRAM: KEY_4
EXECUTES PIA_INIT AND MONITORS LEVELS AND TRANSITIONS AFTER PROMPTING THE OPERATOR TO PRESS KEY 4
MEASUREMENT AT: R8-1

INITIALIZATION PROGRAM: PIA_INIT
INITIALIZES THE PIA PORT
MEASUREMENT AT: (NONE)

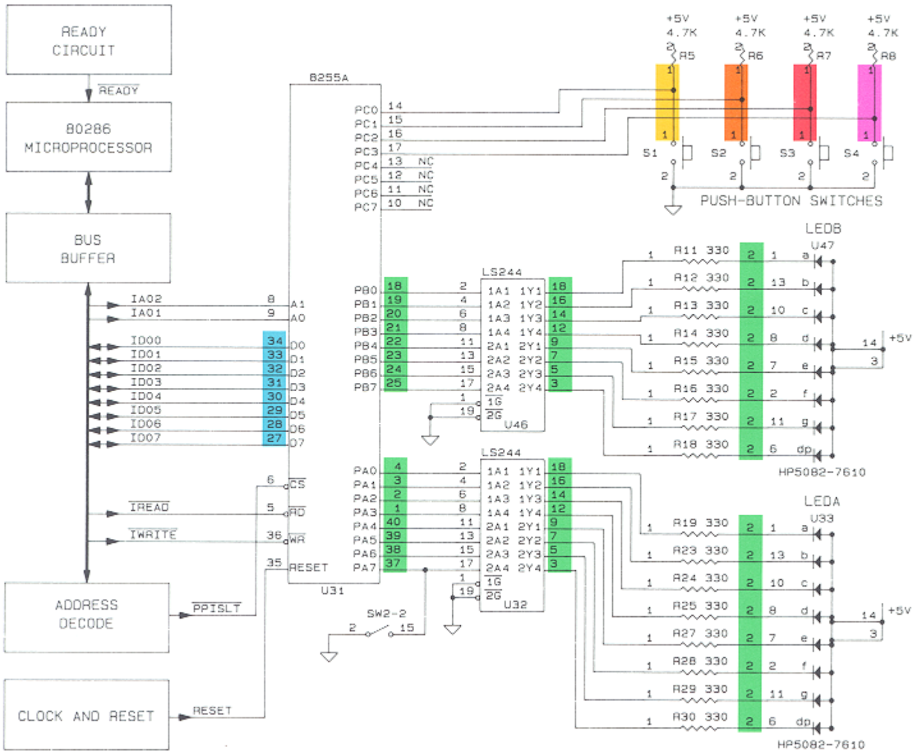


Figure 4-46: Parallel I/O Stimulus Program Planning

```
program key_1

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! STIMULUS PROGRAM checks KEY 1 of PIA circuit.                         !
!                                                                       !
! Stimulus programs and response files are used by GFI to backtrace     !
! from a failing node.  The stimulus program must create repeatable UUT !
! activity and the response file contains the known-good responses for  !
! the outputs in the UUT that are stimulated by the stimulus program.   !
!                                                                       !
! TEST PROGRAMS CALLED:                                                 !
!    pia_init   ()                                                      !
!                                                                       !
! GRAPHICS PROGRAMS CALLED:                                             !
!    (none)                                                             !
!                                                                       !
! Local Constants Modified:                                             !
!    CARRAGE_RETURN                Matches a carrage return input.      !
!                                                                       !
! Local Variables Modified:                                             !
!    devname                       Measurement device                   !
!    input_str                     Input from keypad                    !
!    state                         Level returned from measurement      !
!    finished                      State of loop looking for condition !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Main Declarations                                                     !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

   declare global numeric finished = 0
   declare string CARRAGE_RETURN = ""
   declare string input_str
   declare numeric state = 0
   declare numeric high = 4
   finished = 0

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!   Main part of STIMULUS PROGRAM                                       !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

! Let GFI determine the testing device.

   if (gfi control) = "yes" then
      devname = gfi device
      if (gfi ref) = "U31" then pinnum = 14
   else
      devname = "/probe"
   end if
   print "Stimulus Program KEY_1"
```

(continued on the next page)

Figure 4-47: Stimulus Program *(key_1)*

```
! Setup measurement device and prompt operator.

   podsetup 'report power' "off"
   podsetup 'report forcing' "off"
   podsetup 'report intr' "off"
   podsetup 'report address' "off"
   podsetup 'report data' "off"
   podsetup 'report control' "off"
   reset device devname
   execute pia_init()
   setspace space (getspace space "i/o", size "byte")
   sync device devname, mode "int"
   tlup = open device "/term1", as "update"

! Wait for a high.  Leave program if <ENTER> key is pressed.

   loop until state = high
      arm device devname \ readout device devname
      if devname = "/probe" then
         state = level device devname, type "async"
      else
         state = level device devname, pin pinnum, type "async"
      end if
      if (poll channel tlup, event "input") = 1 then
         input on tlup ,input_str
         if input_str = CARRAGE_RETURN then return
      end if
   end loop

! Start response capture.  End when POD detects reset.
   arm device devname
      strobeclock device devname
      print on tlup ,"WHILE MEASURING, Press \1B[7mDemo UUT KEY 1\1B[0m"
      print on tlup ,"Press 9100 ENTER key if test is stuck."
      loop until finished = 1
         if ((read addr $4004) and 1) = 0 then
            wait time 2                     ! De-bounce.
            strobeclock device devname
            finished = 1
         else if (poll channel tlup, event "input") = 1 then
            input on tlup ,input_str
            if input_str = CARRAGE_RETURN then finished = 1
         end if
      end loop
   readout device devname

   print "\nl\nl"
end program
```

Figure 4-47: Stimulus Program *(key_1) - continued*

```
STIMULUS PROGRAM: KEY_1
DESCRIPTION:                                          SIZE:          78 BYTES

            ------------------- Response Data --------------------
  Node        Learned            Async Clk Counter                        Priority
Signal Src     With       SIG     LVL  LVL  Mode     Counter Range          Pin

R5-1        PROBE       0002           1 0 TRANS
R5-1        I/O MODULE  0002           1 0 TRANS
```

Figure 4-48: Response File *(key_1)*

```
program key_2

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! STIMULUS PROGRAM checks KEY 2 of PIA circuit.                         !
!                                                                       !
! Stimulus programs and response files are used by GFI to backtrace     !
! from a failing node.  The stimulus program must create repeatable UUT !
! activity and the response file contains the known-good responses for  !
! the outputs in the UUT that are stimulated by the stimulus program.   !
!                                                                       !
! TEST PROGRAMS CALLED:                                                 !
!    pia_init  ()                                                       !
!                                                                       !
! GRAPHICS PROGRAMS CALLED:                                             !
!    (none)                                                             !
!                                                                       !
! Local Constants Modified:                                             !
!    CARRAGE_RETURN                Matches a carrage return input.      !
!                                                                       !
! Local Variables Modified:                                             !
!    devname                       Measurement device                   !
!    input_str                     Input from keypad                    !
!    state                         Level returned from measurement      !
!    finished                      State of loop looking for condition !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Main Declarations                                                     !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

   declare global numeric finished = 0
   declare string carrage_return = ""
   declare string str
   declare numeric state = 0
   declare numeric high = 4
   finished = 0

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!  Main part of STIMULUS PROGRAM                                        !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

! Let GFI determine the testing device.

   if (gfi control) = "yes" then
      devname = gfi device
      if (gfi ref) = "U31" then pinnum = 15
   else
      devname = "/probe"
   end if
   print "Stimulus Program KEY_2"
```

(continued on the next page)

Figure 4-49: Stimulus Program *(key_2)*

```
! Setup measurement device and prompt operator.

   reset device devname
   execute pia_init()
   setspace space (getspace space "i/o", size "byte")
   sync device devname, mode "int"
   tlup = open device "/term1", as "update"

! Wait for a high.  Leave program if <ENTER> key is pressed.

   loop until state = high
      arm device devname \ readout device devname
      if devname = "/probe" then
         state = level device devname, type "async"
      else
         state = level device devname, pin pinnum, type "async"
      end if
      if (poll channel tlup, event "input") = 1 then
         input on tlup ,str
         if str = carrage_return then return
      end if
   end loop

! Start response capture.  End when PIA detects line low.

   arm device devname
      strobeclock device devname
      print on tlup ,"WHILE MEASURING, Press \1B[7mDemo UUT KEY 2\1B[0m"
      print on tlup ,"Press 9100 ENTER key if test is stuck."
      loop until finished = 1
         if ((read addr $4004) and 2) = 0 then
            wait time 2                    ! De-bounce.
            strobeclock device devname
            finished = 1
         else if (poll channel tlup, event "input") = 1 then
            input on tlup ,str
            if str = carrage_return then finished = 1
         end if
      end loop
   readout device devname

   print "\nl\nl"
end program
```

Figure 4-49: Stimulus Program *(key_2) - continued*

```
STIMULUS PROGRAM: KEY_2
DESCRIPTION:                                          SIZE:            78 BYTES

           ------------------- Response Data --------------------
  Node       Learned          Async Clk Counter                       Priority
Signal Src    With       SIG   LVL  LVL  Mode     Counter Range         Pin

R6-1       PROBE       0002          1  0 TRANS
R6-1       I/O MODULE  0002          1  0 TRANS
```

Figure 4-50: Response File *(key_2)*

```
program key_3

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! STIMULUS PROGRAM checks KEY 3 of PIA circuit.                         !
!                                                                       !
! Stimulus programs and response files are used by GFI to backtrace    !
! from a failing node.  The stimulus program must create repeatable UUT !
! activity and the response file contains the known-good responses for  !
! the outputs in the UUT that are stimulated by the stimulus program.   !
!                                                                       !
! TEST PROGRAMS CALLED:                                                 !
!    pia_init  ()                                                       !
!                                                                       !
! GRAPHICS PROGRAMS CALLED:                                             !
!    (none)                                                             !
!                                                                       !
! Local Constants Modified:                                             !
!    CARRAGE_RETURN                Matches a carrage return input.      !
!                                                                       !
! Local Variables Modified:                                             !
!    devname                       Measurement device                   !
!    input_str                     Input from keypad                    !
!    state                         Level returned from measurement      !
!    finished                      State of loop looking for condition !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Main Declarations                                                     !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

   declare global numeric finished = 0
   declare string carrage_return = ""
   declare string str
   declare numeric state = 0
   declare numeric high = 4
   finished = 0

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!   Main part of STIMULUS PROGRAM                                       !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

! Let GFI determine the testing device.

   if (gfi control) = "yes" then
      devname = gfi device
      if (gfi ref) = "U31" then pinnum = 16
   else
      devname = "/probe"
   end if
   print "Stimulus Program KEY_3"
```


(continued on the next page)

Figure 4-51: Stimulus Program *(key_3)*

```
! Setup measurement device and prompt operator.

   reset device devname
   execute pia_init()
   setspace space (getspace space "i/o", size "byte")
   sync device devname, mode "int"
   tlup = open device "/term1", as "update"

! Wait for a high.  Leave program if <ENTER> key is pressed.

   loop until state = high
      arm device devname \ readout device devname
      if devname = "/probe" then
         state = level device devname, type "async"
      else
         state = level device devname, pin pinnum, type "async"
      end if
      if (poll channel tlup, event "input") = 1 then
         input on tlup ,str
         if str = carrage_return then return
      end if
   end loop

! Start response capture.  End when POD detects reset.
   arm device devname
      strobeclock device devname
      print on tlup ,"WHILE MEASURING, Press \1B[7mDemo UUT KEY 3\1B[0m"
      print on tlup ,"Press 9100 ENTER key if test is stuck."
      loop until finished = 1
         if ((read addr $4004) and 4) = 0 then
            wait time 2                    ! De-bounce.
            strobeclock device devname
            finished = 1
         else if (poll channel tlup, event "input") = 1 then
            input on tlup ,str
            if str = carrage_return then finished = 1
         end if
      end loop
   readout device devname

   print "\nl\nl"
end program
```

Figure 4-51: Stimulus Program *(key_3) - continued*

STIMULUS PROGRAM: KEY_3
DESCRIPTION: SIZE: 78 BYTES

	Response Data						
Node Signal Src	Learned With	SIG	Async LVL	Clk LVL	Counter Mode	Counter Range	Priority Pin
R7-1	PROBE	0002	1	0	TRANS		
R7-1	I/O MODULE	0002	1	0	TRANS		

Figure 4-52: Response File *(key_3)*

```
program key_4

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! STIMULUS PROGRAM checks KEY 4 of PIA circuit.                        !
!                                                                      !
! Stimulus programs and response files are used by GFI to backtrace    !
! from a failing node.  The stimulus program must create repeatable UUT !
! activity and the response file contains the known-good responses for  !
! the outputs in the UUT that are stimulated by the stimulus program.   !
!                                                                      !
! TEST PROGRAMS CALLED:                                                !
!    pia_init  ()                                                      !
!                                                                      !
! GRAPHICS PROGRAMS CALLED:                                            !
!    (none)                                                            !
!                                                                      !
! Local Constants Modified:                                            !
!    CARRAGE_RETURN                 Matches a carrage return input.    !
!                                                                      !
! Local Variables Modified:                                            !
!    devname                        Measurement device                 !
!    input_str                      Input from keypad                  !
!    state                          Level returned from measurement    !
!    finished                       State of loop looking for condition !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Main Declarations                                                    !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

   declare global numeric finished = 0
   declare string carrage_return = ""
   declare string str
   declare numeric state = 0
   declare numeric high = 4
   finished = 0

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!   Main part of STIMULUS PROGRAM                                      !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

! Let GFI determine the testing device.

   if (gfi control) = "yes" then
      devname = gfi device
      if (gfi ref) = "U31" then pinnum = 17
   else
      devname = "/probe"
   end if
   print "Stimulus Program KEY_4"
```

(continued on the next page)

Figure 4-53: Stimulus Program *(key_4)*

```
! Setup measurement device and prompt operator.

   reset device devname
   execute pia_init()
   setspace space (getspace space "i/o", size "byte")
   sync device devname, mode "int"
   tlup = open device "/term1", as "update"

! Wait for a high.  Leave program if <ENTER> key is pressed.

   loop until state = high
      arm device devname \ readout device devname
      if devname = "/probe" then
         state = level device devname, type "async"
      else
         state = level device devname, pin pinnum, type "async"
      end if
      if (poll channel tlup, event "input") = 1 then
         input on tlup ,str
         if str = carrage_return then return
      end if
   end loop

! Start response capture.  End when POD detects reset.
   arm device devname
      strobeclock device devname
      print on tlup ,"WHILE MEASURING, Press \1B[7mDemo UUT KEY 4\1B[0m"
      print on tlup ,"Press 9100 ENTER key if test is stuck."
      loop until finished = 1
         if ((read addr $4004) and 8) = 0 then
            wait time 2                        ! De-bounce.
            strobeclock device devname
            finished = 1
         else if (poll channel tlup, event "input") = 1 then
            input on tlup ,str
            if str = carrage_return then finished = 1
         end if
      end loop
   readout device devname

   print "\nl\nl"
end program
```

Figure 4-53: Stimulus Program *(key_4) - continued*

```
 STIMULUS PROGRAM NAME: KEY_4
 DESCRIPTION:                                            SIZE:              78 BYTES

            ------------------- Response Data --------------------
   Node       Learned             Async Clk Counter                          Priority
Signal Src     With        SIG     LVL  LVL  Mode     Counter Range           Pin

R8-1         PROBE        0002          1 0 TRANS
R8-1         I/O MODULE   0002          1 0 TRANS
```

Figure 4-54: Response File *(key_4)*

```
program pia_data

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! STIMULUS PROGRAM outputs data from the PIA onto the bus.              !
!                                                                       !
! Stimulus programs and response files are used by GFI to backtrace     !
! from a failing node.  The stimulus program must create repeatable UUT !
! activity and the response file contains the known-good responses for  !
! the outputs in the UUT that are stimulated by the stimulus program.   !
!                                                                       !
! This stimulus program is one of the programs which creates activity   !
! in the kernel area of the UUT.  These programs create activity with   !
! or without the ready circuit working properly.  Because of this, all  !
! the stimulus programs in the kernel area must disable the READY input !
! to the pod, then perform the stimulus, then re-enable the READY input !
! to the pod.  The 80286 microprocessor has a separate bus controller;  !
! for this reason, disabling ready and performing stimulus can get the  !
! bus controller out of synchronization with the pod.  Two fault        !
! handlers trap pod timeout conditions that indicate the bus controller !
! is out of synchronization.  The recover() program is executed to      !
! resynchronize the bus controller and the pod.                         !
!                                                                       !
! TEST PROGRAMS CALLED:                                                 !
!    recover     ()                        The 80286 microprocessor has a!
!                                          bus controller that is totally!
!                                          separate from the pod.  In    !
!                                          some cases the pod can get out!
!                                          of sync with the bus control- !
!                                          ler.  The recover program     !
!                                          resynchronizes the pod and the!
!                                          bus controller.               !
!                                                                       !
!    pia_init()                            Initalization program for the !
!                                          8255.  Sets port A and B to   !
!                                          output with port C to input.  !
!                                                                       !
! GRAPHICS PROGRAMS CALLED:                                             !
!    (none)                                                             !
!                                                                       !
! Local Variables Modified:                                             !
!    devname                               Measurement device           !
!                                                                       !
! Global Variables Modified:                                            !
!    recover_times                         Reset to Zero                !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

(continued on the next page)

Figure 4-55: Stimulus Program *(pia_data)*

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!   FAULT HANDLERS:                                                    !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

handle pod_timeout_enabled_line
   recover()
end handle
handle pod_timeout_recovered
   recover()
end handle

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Main Declarations                                                    !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

declare global numeric recover_times
recover_times = 0

! Let GFI user select which I/O module to use

  if (gfi control) = "yes" then
    devname = gfi device
  else
    devname = "/mod1"
  end if
   print "Stimulus Program PIA_DATA"

! Initialize the PIA and setup the measurement device.

  reset device devname
  pia_init()
  setspace space (getspace space "i/o", size "byte")
  write addr $4002, data  $AA              ! set port B to known value.
  sync device devname, mode "pod"
  sync device "/pod", mode "data"

! Present stimulus to the UUT, read PIA port B register onto data bus.

  arm device devname         ! Start response capture.
      read addr $4002         ! read port B
      write addr $4002, data  $55
      read addr $4002
  readout device devname     ! End response capture.

end pia_data
```

Figure 4-55: Stimulus Program *(pia_data) - continued*

STIMULUS PROGRAM NAME: PIA_DATA
DESCRIPTION: SIZE: 326 BYTES

Node Signal Src	Learned With	SIG	Async LVL	Clk LVL	Counter Mode	Counter Range	Priority Pin
	Response Data						
U31-34	PROBE	0003			TRANS		
U31-34	I/O MODULE	0003			TRANS		U21-5
U31-33	PROBE	0004			TRANS		
U31-33	I/O MODULE	0004			TRANS		U21-5
U31-32	PROBE	0003			TRANS		
U31-32	I/O MODULE	0003			TRANS		U21-5
U31-31	PROBE	0004			TRANS		
U31-31	I/O MODULE	0004			TRANS		U21-5
U31-30	PROBE	0003			TRANS		
U31-30	I/O MODULE	0003			TRANS		U21-5
U31-29	PROBE	0004			TRANS		
U31-29	I/O MODULE	0004			TRANS		U21-5
U31-28	PROBE	0003			TRANS		
U31-28	I/O MODULE	0003			TRANS		U21-5
U31-27	PROBE	0004			TRANS		
U31-27	I/O MODULE	0004			TRANS		U21-5

Figure 4-56: Response File *(pia_data)*

```
program pia_leds

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! STIMULUS PROGRAM to exercise PIA output signals.                      !
!                                                                       !
! Stimulus programs and response files are used by GFI to backtrace     !
! from a failing node.  The stimulus program must create repeatable UUT !
! activity and the response file contains the known-good responses for  !
! the outputs in the UUT that are stimulated by the stimulus program.   !
!                                                                       !
! This Stimulus program uses rampdata at the PIA output port addresses  !
! to toggle port B.                                                     !
!                                                                       !
! TEST PROGRAMS CALLED:                                                 !
!    pia_init()                               Initalization program for the !
!                                             8255.  Sets port A and B to   !
!                                             output with port C to input.  !
!                                                                       !
! GRAPHICS PROGRAMS CALLED:                                             !
!    (none)                                                             !
!                                                                       !
! Local Variables Modified:                                             !
!    devname                                  Measurement device        !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

! Let GFI user select which I/O module to use

   if (gfi control) = "yes" then
     devname = gfi device
   else
     devname = "/mod1"
   end if
   print "Stimulus Program PIA_LEDS"

! Initialize the PIA port and setup measurent device.

   reset device devname
   execute pia_init()
   setspace space (getspace space "i/o", size "word")
   sync device devname, mode "pod"
   sync device "/pod", mode "data"

! Present stimulus to the UUT

   arm device devname              ! Start response capture.
      rampdata addr $4000, data  0, mask $FF
      rampdata addr $4002, data  0, mask $FF
   readout device devname          ! End response capture

end pia_leds
```

Figure 4-57: Stimulus Program *(pia_leds)*

STIMULUS PROGRAM NAME: PIA_LEDS
DESCRIPTION: SIZE: 1,134 BYTES

Node Signal Src	Learned With	SIG	Async LVL	Clk LVL	Counter Mode	Counter Range	Priority Pin
U31-4	I/O MODULE	EFF7	1	0	TRANS		
U31-3	I/O MODULE	7628	1	0	TRANS		
U31-2	I/O MODULE	790E	1	0	TRANS		
U31-1	I/O MODULE	49CB	1	0	TRANS		
U31-40	I/O MODULE	C04E	1	0	TRANS		
U31-39	I/O MODULE	1D3A	1	0	TRANS		
U31-38	I/O MODULE	A1C7	1	0	TRANS		
U31-37	I/O MODULE	63EB	1	0	TRANS		
U31-18	I/O MODULE	D37A	1	0	TRANS		
U31-19	I/O MODULE	A121	1	0	TRANS		
U31-20	I/O MODULE	6AFA	1	0	TRANS		
U31-21	I/O MODULE	B5FC	1	0	TRANS		
U31-22	I/O MODULE	A71E	1	0	TRANS		
U31-23	I/O MODULE	DAF9	1	0	TRANS		
U31-24	I/O MODULE	23EF	1	0	TRANS		
U31-25	I/O MODULE	2F53	1	0	TRANS		
U46-18	PROBE	D37A	1	0	TRANS		
U46-18	I/O MODULE	D37A	1	0	TRANS		
U46-16	PROBE	A121	1	0	TRANS		
U46-16	I/O MODULE	A121	1	0	TRANS		
U46-14	PROBE	6AFA	1	0	TRANS		
U46-14	I/O MODULE	6AFA	1	0	TRANS		
U46-12	PROBE	B5FC	1	0	TRANS		
U46-12	I/O MODULE	B5FC	1	0	TRANS		
U46-9	PROBE	A71E	1	0	TRANS		
U46-9	I/O MODULE	A71E	1	0	TRANS		
U46-7	PROBE	DAF9	1	0	TRANS		
U46-7	I/O MODULE	DAF9	1	0	TRANS		
U46-5	PROBE	23EF	1	0	TRANS		
U46-5	I/O MODULE	23EF	1	0	TRANS		
U46-3	PROBE	2F53	1	0	TRANS		
U46-3	I/O MODULE	2F53	1	0	TRANS		
U32-18	PROBE	EFF7	1	0	TRANS		
U32-18	I/O MODULE	EFF7	1	0	TRANS		
U32-16	PROBE	7628	1	0	TRANS		
U32-16	I/O MODULE	7628	1	0	TRANS		
U32-14	PROBE	790E	1	0	TRANS		
U32-14	I/O MODULE	790E	1	0	TRANS		
U32-12	PROBE	49CB	1	0	TRANS		
U32-12	I/O MODULE	49CB	1	0	TRANS		
U32-9	PROBE	C04E	1	0	TRANS		
U32-9	I/O MODULE	C04E	1	0	TRANS		

(continued on the next page)

Figure 4-58: Response File *(pia_leds)*

U32-7	PROBE	1D3A	1 0	TRANS
U32-7	I/O MODULE	1D3A	1 0	TRANS
U32-5	PROBE	A1C7	1 0	TRANS
U32-5	I/O MODULE	A1C7	1 0	TRANS
U32-3	PROBE	63EB	1 0	TRANS
U32-3	I/O MODULE	63EB	1 0	TRANS
R11-2	PROBE	4596	1	TRANS
R12-2	PROBE	4596	1	TRANS
R13-2	PROBE	4596	1	TRANS
R14-2	PROBE	4596	1	TRANS
R15-2	PROBE	4596	1	TRANS
R16-2	PROBE	4596	1	TRANS
R17-2	PROBE	4596	1	TRANS
R18-2	PROBE	4596	1	TRANS
R19-2	PROBE	4596	1	TRANS
R23-2	PROBE	4596	1	TRANS
R24-2	PROBE	4596	1	TRANS
R25-2	PROBE	4596	1	TRANS
R27-2	PROBE	4596	1	TRANS
R28-2	PROBE	4596	1	TRANS
R29-2	PROBE	4596	1	TRANS
R30-2	PROBE	4596	1	TRANS

Figure 4-58: Response File *(pia_leds) - continued*

```
program pia_init

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! INITIALIZATION PROGRAM to set up the PIA.                              !
!                                                                       !
! TEST PROGRAMS CALLED:                                                 !
!    (none)                                                             !
!                                                                       !
! GRAPHICS PROGRAMS CALLED:                                             !
!    (none)                                                             !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

! Set address space

   setspace space (getspace space "i/o", size "byte")


! Initialize the PIA port

   write data $89, addr $4006            ! SET CONTROL REG
   write data $FF, addr $4000            ! CLEAR THE A REG
   write data $FF, addr $4002            ! CLEAR THE B REG

end pia_init
```

Figure 4-59: Initialization Program *(pia_init)*

Summary of Complete Solution for Parallel I/O 4.5.7.

The entire set of programs and files needed to test and GFI troubleshoot the Parallel I/O functional block is shown below. The format below is similar to a 9100A/9105A UUT directory (you could consider the functional block to be a small UUT), but in addition shows the use of each program and the location in this manual for each file.

UUT DIRECTORY
(Complete File Set for Parallel I/O)

Programs (PROGRAM):

TEST_PIA	Functional Test	Section 4.5.5
PIA_DATA	Stimulus Program	Figure 4-55
PIA_LEDS	Stimulus Program	Figure 4-57
KEY_1	Stimulus Program	Figure 4-47
KEY_2	Stimulus Program	Figure 4-49
KEY_3	Stimulus Program	Figure 4-51
KEY_4	Stimulus Program	Figure 4-53
PIA_INIT	Initialization Program	Figure 4-59

Stimulus Program Responses (RESPONSE):

PIA_DATA	Figure 4-56
PIA_LEDS	Figure 4-58
KEY_1	Figure 4-48
KEY_2	Figure 4-50
KEY_3	Figure 4-52
KEY_4	Figure 4-54

Node List (NODE):

NODELIST	Appendix A

Text Files (TEXT):

Reference Designator List (REF):

REFLIST	Appendix B

Compiled Database (DATABASE):

GFIDATA	Compiled by the 9100A

(This page is intentionally blank.)

SERIAL INPUT/OUTPUT FUNCTIONAL BLOCK 4.6.

Introduction to Serial I/O 4.6.1.

The block diagram in Figure 4-60 shows a typical serial I/O port implemented with a UART (universal asynchronous receiver-transmitter) surrounded by its direct support circuitry. For the UART to function properly, all of the support circuitry in Figure 4-1 must function properly.

SIA (serial interface adaptor) chips typically implement all of the UART block and most of the clock and interrupt blocks. On the Demo/Trainer UUT, address decoding and interrupt generation circuits are grouped as separate functional blocks and are described later in Sections 4.11 and 4.13.

Considerations for Testing and Troubleshooting 4.6.2.

Testing

The external I/O lines can be divided into two types:

- Serial lines.
- Handshake and control lines.

Testing the handshake lines is straightforward. The status of input handshake lines can usually be checked by reading a register and testing the appropriate bit. Similarly, output handshake lines can be toggled by setting and clearing a bit in an output register. Testing can be done using the probe or by connecting output lines back to input lines. Some SIA chips need initialization before they respond properly.

Testing the serial input and serial output lines is usually done by connecting the output back to the input. On the Demo/Trainer UUT, this can be done by setting switches. In general, it is

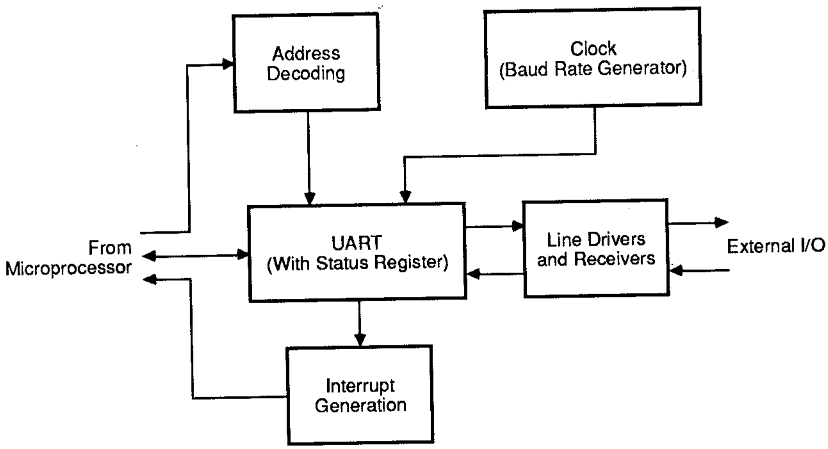


Figure 4-60: Typical Serial I/O Port, With Support Circuitry

preferable to wire a connector to perform the loopback. This allows testing the entire interface, including the connector.

UART chips provide data buffers on their inputs. Therefore, characters can be written to the output side of the UART and the read at the input side. If this technique is used, two limitations should be kept in mind:

- Since the input and output baud rates are usually derived from the same clock, loopback testing will not test for proper baud-rate timing.
- The UART must be initialized with the same transmit and receive baud rate.

One approach to testing the baud rate clock frequency is to set up the transmitter to send seven bits with no parity. Under these conditions, when a null character (00 hex) is sent, the result will be a pulse that is high for eight bit times (start bit and seven data bits). If the probe is connected to a known-frequency clock signal and the start and stop lines are connected to the serial output, the baud rate can be computed. The start line should cause counting to start on the first bit and the stop line should stop the count at the end of the last bit. For example, on the Demo/Trainer UUT, the 8 MHz clock on U1-5 (Figure 4-61) can be probed and the start and stop lines from the clock module can be connected to one of the serial output pins (U13-8 or U12-7). Eight bits at 1200 baud (8/1200 sec) counting 8 MHz the result should be about 53,333 (D055 hex) counts.

The procedures above do not test the interrupt generation block. This circuitry, which is described in detail later in Section 4.13, can be tested by individually enabling the interrupts that are of interest and then stimulating them by exercising the UART. For example, to test the character-received interrupt, perform the following steps:

1. Initialize the interface.
2. Enable the receiver interrupt (usually a bit in a command register).

3. With loopback wired, send a character.

4. Verify that the pod received an interrupt using the *readstatus* TL/1 command. (This assumes that the interrupt stays active until serviced.)

Here are some potential problems in testing serial I/O ports:

- The I/O module may load a crystal oscillator enough to shift the frequency or make it stop oscillating.
- Some SIA chips will not send characters if their handshake lines are in the wrong state.
- If a loopback test cannot be performed on your UUT, you can use the RS-232 port on your 9100A/9105A to test the serial I/O port on the UUT.

Troubleshooting

The central element of a serial I/O port is the UART or SIA chip. If troubleshooting is started by clipping the UART, the problem should be easily isolated. The UART either receives or generates signals from all of the other circuit blocks. If all inputs to the UART are good and all outputs are bad, the UART is bad or its outputs are loaded. If an input is bad, the problem can be traced into the circuitry that generated it. All of this is done automatically in GFI.

The serial input and output can be evaluated by writing a series of characters and counting transitions. The Demo/Trainer UUT stimulus programs for the serial I/O block work this way.

The Demo/Trainer UUT has built-in switches that loop the serial outputs back to the inputs. If GFI troubleshooting is done with the loopback in place, the nodelist must show this connection; if loopback is done at the connector, the appropriate pins of the connector can simply be shown on the same node.

The probe has a special threshold level for testing RS-232 signals, which is set up with the TL/1 command:

```
threshold device "/probe", mode "rs232"
```

or the operator's keypad command:

```
SET PROBE LOGIC INPUT LEVEL TO RS232.
```

If a part has RS-232-level signals, it should be specified as a probe device in the reflist for the UUT.

The *gfi control* TL/1 command determines when a stimulus program is under GFI (or UFI) control. There are many examples of its use in the stimulus programs that follow. When a program is under GFI (or UFI) control, the *gfi reference* function will return a string describing the device being clipped or the pin being probed. The following TL/1 example shows how the *gfi ref* command could be used in a stimulus program to change the threshold levels if the components to be tested require such a change.

```
if (gfi control) = "yes" then
    str = gfi ref
    if ((str = "U12-14") or (str = "U12-7")) then
        threshold device "/probe", mode "rs232"
    else
        threshold device "/probe", mode "ttl"
    end if
end if
```

Serial I/O Example 4.6.3.

Figure 4-61 shows the serial I/O port on the Demo/Trainer UUT. The DUART (dual universal asynchronous receiver-transmitter), U11, receives serial data input from the keyboard (RXDA/TXDA) and handles bidirectional signal flow with the RS-232 port (RXDB/TXDB). Keyboard input must be at 1200 baud. U12 acts as a level shifter, coupling TTL signal levels on the Demo/Trainer UUT to RS-232 levels at the serial interface; U12 uses a charge pump to shift levels from a +5V source.

The keystroke functional test that follows is not a complete test of the RS-232 circuit. The keyboard receive, port 1 transmit, and port 2 receive lines are not tested between the loopback switch and the connectors. Also, the test assumes that the interrupt functional block is good when testing the INT pin (U11-24).

Keystroke Functional Test 4.6.4.

1. Initialize the Dual UART using the EXEC key with the following command:

```
EXECUTE UUT DEMO PROGRAM RS232_INIT
```

2. Close switches SW4-4, SW4-5 and SW6-4. Now the Transmit line (Txd) is looped back to the receive line (RxD) and transmitting a character on TxD will cause the UART to receive a character on RxD. Then use the SETUP MENU key with the following command to turn off reporting of interrupts:

```
SETUP POD REPORT INTR ACTIVE OFF
```

3. Use the WRITE and READ keys with the following commands to test Port A of the DUART:

```
WRITE DATA 45 TO ADDR 2006
... (ADDR OPTION: I/O BYTE)
READ ADDR 2006 =
... (ADDR OPTION: I/O BYTE)
    The value read should be 45.
```

4. Use the WRITE and READ keys with the following commands to test the Transmit to Receive loopback of Port B of the DUART:

```
WRITE DATA 55 TO ADDR 2016
... (ADDR OPTION: I/O BYTE)
READ ADDR 2016 =
... (ADDR OPTION: I/O BYTE)
     The value read should be 55.
```

   You may need to do the READ step up to three times to get the expected value, since the read buffer can be stacked three-deep.

5. Use the WRITE and READ keys with the following commands to test the RTS to CTS loopback of Port B of the DUART:

```
WRITE DATA 0 TO ADDR 201A
... (ADDR OPTION: I/O BYTE)
WRITE DATA FF TO ADDR 201C
... (ADDR OPTION: I/O BYTE)
READ ADDR 201A =
... (ADDR OPTION: I/O BYTE)
     Examine the hexadecimal value to make sure
     bit 1 is a 0.  Bit 0 is the LSB.
WRITE DATA FF TO ADDR 201E
... (ADDR OPTION: I/O BYTE)
READ ADDR 201A =
... (ADDR OPTION: I/O BYTE)
     Examine the hexadecimal value to make sure
     bit 1 is a 1.  Bit 0 is the LSB.
```

Keystroke Functional Test

CONNECTION TABLE

STIMULUS	FEEDBACK LOOP	MEASUREMENT
POD	DIP SWITCHES	POD
TEST ACCESS SOCKET	SW6-4 SW4-4 SW4-5	TEST ACCESS SOCKET

STIMULUS AND RESPONSE TABLE FOR DUART PORT A

DATA SENT TO PORT A (ADDRESS 2006)	DATA RECEIVED FROM PORT A (ADDRESS 2006)
45	45

STIMULUS AND RESPONSE TABLE FOR DUART PORT B

DATA SENT TO PORT B (ADDRESS 2016)	DATA RECEIVED FROM PORT B (ADDRESS 2016)
55	55

STIMULUS AND RESPONSE TABLE FOR TIMER INTERRUPT

DATA 55 SENT TO TIMER INTERRUPT	BIT 1 LEVEL AT ADDRESS 201A (BIT 0 IS LSB)
ADDRESS 201C	LOW
ADDRESS 201E	HIGH

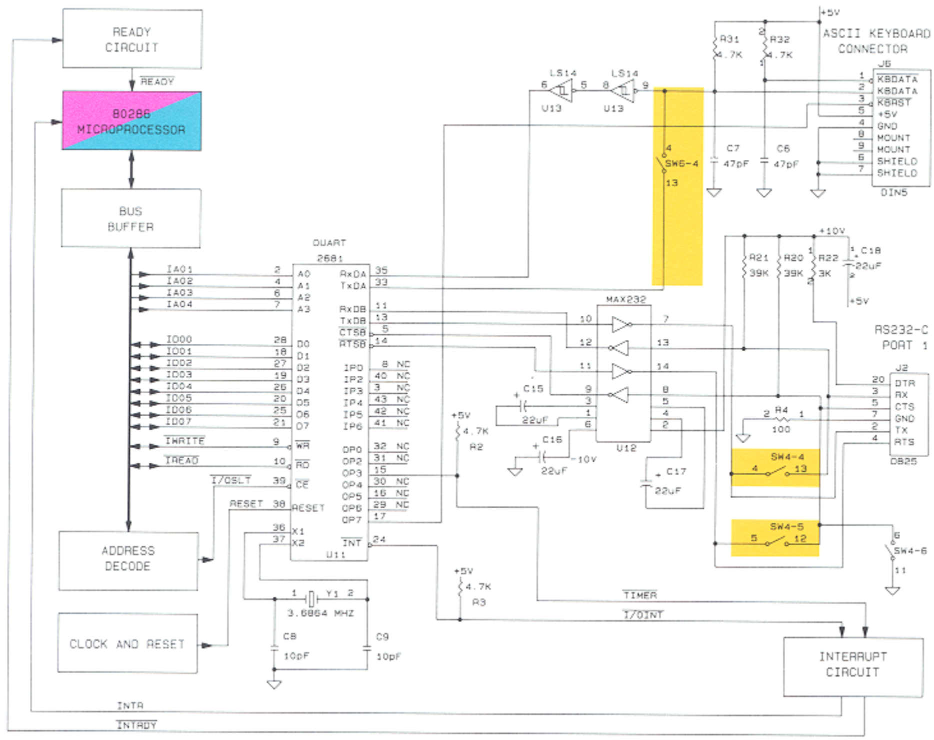


Figure 4-61: Serial I/O Functional Test

Programmed Functional Test 4.6.5.

The *test_rs232* program is the programmed functional test for the Serial I/O functional block. This program also tests for interrupt conditions generated by the Serial I/O circuit.

First, the program initializes the DUART U11 and prompts the test operator to close the loopback switches which connect Port A transmit to Part A receive, connect Port B transmit to Port B receive, and connect Port B Request To Send (RTS) to Port B Clear To Send (CTS).

Next, Port A is checked by transmitting a character and examining the receive buffer for the same character.

And finally, a character is transmitted on Port B which also generates an interrupt condition. Two pod programs called *frc_int* and *rd_cscd* are executed to check proper operation of the interrupt logic. After that, the receive buffer is examined for the same character that was transmitted. This clears the interrupt condition. Then the *frc_int* program is executed again to make sure the interrupt condition has been cleared. A register in the DUART is then checked to see that the RTS/CTS loopback worked properly.

If any of the above operations fail, the *gfi test* command is used to find a failing signal. GFI then takes control and backtraces to the source of the failure.

If a problem is detected in the interrupt circuit, the *tst_intrpt* program (programmed test of the Interrupt Circuit functional block) is executed.

```
program test_rs232

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! FUNCTIONAL TEST of the SERIAL I/O functional block.                    !
!                                                                        !
! This program tests the SERIAL I/O functional block of the              !
! Demo/Trainer.  The two RS-232 ports are tested by setting three Dip    !
! Switches to loop back the two ports (SW4-4, SW4-5 and SW6-4 loop back !
! ports A and B).  The SERIAL I/O functional block also outputs two      !
! interrupt request signals.  This program also checks the interrupt     !
! circuitry.                                                             !
!                                                                        !
!    frc_int     ()                          POD PROGRAM forces repetitive !
!                                            interrupt acknowledge cycles  !
!                                            and returns first interrupt   !
!                                            vector found on data bus.     !
!                                                                        !
!    rd_cscd     ()                          POD PROGRAM returns the 24 bit!
!                                            interrupt cascade address that!
!                                            was found on the address bus  !
!                                            during the last interrupt     !
!                                            acknowledge cycle.            !
!                                                                        !
!    rd_rearm    ()                          POD PROGRAM returns the most  !
!                                            recent interrupt vector and   !
!                                            rearms the pod to respond to  !
!                                            the next interrupt.           !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Main Declarations                                                      !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

declare
   string q                          ! used to get input from keyboard
   global string rev                 ! Reverse Video escape sequence
   global string norm                ! Normal Video escape sequence
end declare

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! FUNCTIONS                                                              !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

function sync_buffer( address, data )
   declare numeric address
   declare numeric data

! Synchronize FIFO buffer in DUART.  Write and then read until correct data
! is returned or count has expired.

   write addr address, data data      ! Transmit Data 31 on port A
   wait time $200
   cnt = 0  \ x = 0
   loop until x = data or cnt > 3
      x = read addr address
      cnt = cnt + 1
   end loop
end function
```

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! FUNCTIONAL TEST of the SERIAL I/O Functional Block.                        !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

! Set interrupt acknowledge cycles on and use the 80286
! pod specific programs rd_rearm(), frc_int() & rd_cscd().

   podsetup 'report intr' "off"
   podsetup 'intr_ack on'          ! Enable Interrupt Ack. cycles
   option = getspace space "i/o", size "byte"
   setspace (option)
   execute check_loop()
   execute rd_rearm()              ! Clear interrupts

! Main part of Test.  Verify DUART port A.

   sync_buffer( $2006, $61 )       ! Synchronize FIFO in DUART for port A
   write addr $2006, data $55      ! Transmit Data 31 on port A
   wait time $200
   if ((read addr $2002) and $F) <> $D then fault 'RS232 Port A failed' \ return
   if (read addr $2006) <> $55 then fault 'RS232 Port A failed' \ return
   write addr $2006, data $55      ! Transmit Data 31 on port A
   wait time $200
   if ((read addr $2002) and $F) <> $D then fault 'RS232 Port A failed' \ return
   if (read addr $2006) <> $55 then fault 'RS232 Port A failed' \ return

! Verify DUART port B and interrupts.

   sync_buffer( $2016, $61 )       ! Synchronize FIFO in DUART for port B
   write addr $201E, data $FF      ! set output port low
   write addr $2016, data $31      ! Transmit Data 31 on port B
   if frc_int() <> $22 then fault 'Interrupt failed' \ return
   if rd_cscd() <> $2016 then fault 'Interrupt failed' \ return
   if (readstatus() and 8) <> 8 then fault 'Interrupt failed' \ return
   if (read addr $2016) <> $31 then fault 'RS232 Port B failed' \ return
   if frc_int() <> $27 then fault 'Interrupt failed' \ return
   write addr $201C, data $FF
   if ((read addr $201A) and 2) <> 0 then fault 'RS232 Port B failed' \ return

end program
```

Stimulus Programs and Responses 4.6.6.

Figure 4-62 is the stimulus program planning diagram for the Serial I/O functional block. The Serial I/O stimulus programs require the test operator to close the loopback switches which loop the transmit lines back to the receive lines and loop the Port B RTS output back to the Port B CTS input.

The *rs232_data* stimulus program outputs data from the DUART onto the data bus. The *rs232_lvl* stimulus program sends a character out the transmit line and then monitors RS232-level signals using the probe with the threshold levels set to "rs232". The *ttl_lvl* stimulus program is the same as *rs232_lvl* except that signals are measured using a level threshold of "ttl".

All the stimulus programs execute *rs232_init* before any measurements are made on the Serial I/O circuitry.

Stimulus Program Planning

PROGRAM: RS232_DATA

EXECUTES RS232_INIT AND READS DATA FROM DUART REGISTERS

MEASUREMENT AT:

U11-28,18,27,19,26,20,25,21

PROGRAM: TTL_LVL

EXECUTES RS232_INIT AND EXERCISES RS-232 CIRCUITRY AT TTL LEVELS

MEASUREMENT AT:

U11-33,14,24,13,15,17
U12-12,9
U13-6,8

PROGRAM: RS232_LVL

EXECUTES RS232_INIT AND EXERCISES RS-232 CIRCUITRY AT RS-232 LEVELS

MEASUREMENT AT:

U12-7,14,1,2,4,6
J2-3,5
R22-2
C15-2
C17-2

PROGRAM: FREQUENCY

MEASURES FREQUENCY

MEASUREMENT AT:

Y1-1

INITIALIZATION PROGRAM: RS232_INIT

INITIALIZES THE DUART

MEASUREMENT AT:

(NONE)

PROGRAM: LEVELS

MEASURES STATIC LEVELS

MEASUREMENT AT:

R4-1
R32-1

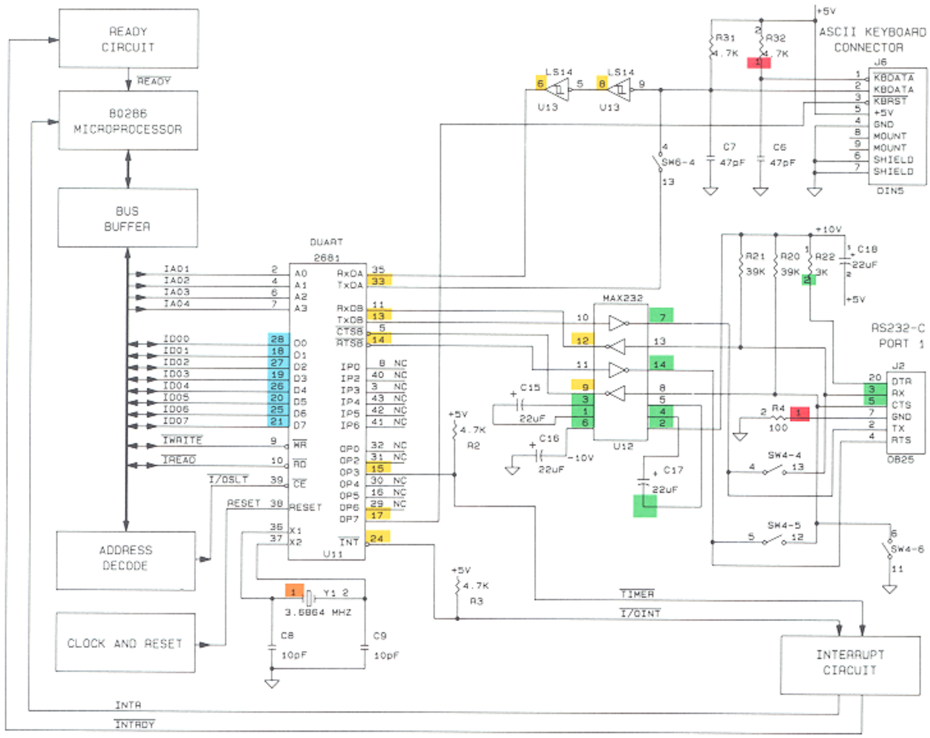


Figure 4-62: Serial I/O Stimulus Program Planning

```
program rs232_data

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! STIMULUS PROGRAM for U11 data lines as outputs.                          !
!                                                                          !
! Stimulus programs and response files are used by GFI to backtrace        !
! from a failing node.  The stimulus program must create repeatable UUT !
! activity and the response file contains the known-good responses for    !
! the outputs in the UUT that are stimulated by the stimulus program.     !
!                                                                          !
! This stimulus program is one of the programs which creates activity     !
! in the kernel area of the UUT.  These programs create activity with     !
! or without the ready circuit working properly.  Because of this, all    !
! the stimulus programs in the kernel area must disable the READY input !
! to the pod, then perform the stimulus, then re-enable the READY input !
! to the pod.  The 80286 microprocessor has a separate bus controller;    !
! for this reason, disabling ready and performing stimulus can get the    !
! bus controller out of synchronization with the pod.  Two fault          !
! handlers trap pod timeout conditions that indicate the bus controller !
! is out of synchronization.  The recover() program is executed to        !
! resynchronize the bus controller and the pod.                           !
!                                                                          !
! TEST PROGRAMS CALLED:                                                    !
!    rs232_init  ()                       Initialize the RS232 circuit. !
!                                                                          !
!    recover     ()                       The 80286 microprocessor has a!
!                                         bus controller that is totally!
!                                         separate from the pod.  In     !
!                                         some cases the pod can get out!
!                                         of sync with the bus control- !
!                                         ler.  The recover program      !
!                                         resynchronizes the pod and the!
!                                         bus controller.                !
!                                                                          !
! GRAPHICS PROGRAMS CALLED:                                                !
!    (none)                                                                !
!                                                                          !
! Global Variables Modified:                                               !
!    recover_times                        Reset to Zero                    !
!    devname                              Measurement device               !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Main Declarations                                                        !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

declare global numeric recover_times
```

(continued on the next page)

Figure 4-63: Stimulus Program *(rs232_data)*

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!   FAULT HANDLERS:                                                    !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

handle pod_timeout_enabled_line
   recover()
end handle
handle pod_timeout_recovered
   recover()
end handle

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!   Main part of STIMULUS PROGRAM                                      !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

recover_times = 0

! Let GFI determine the measurement device.

   if (gfi control) = "yes" then
      devname = gfi device
   else
      devname = "/mod1"
   end if
   print "Stimulus Program RS232_DATA"

! Set addressing mode and setup measurement device.

   reset device devname
   execute rs232_init()
   setspace space (getspace space "i/o", size "byte")
   sync device devname, mode "pod"
   sync device "/pod", mode "data"

! Present stimulus to UUT.

   arm device devname          ! Start response capture.
      read addr $200A
      read addr $201A
      read addr $2012
      read addr $201A
      read addr $2000
   readout device devname      ! End response capture.

end program
```

Figure 4-63: Stimulus Program *(rs232_data) - continued*

STIMULUS PROGRAM NAME: RS232_DATA
DESCRIPTION: SIZE: 318 BYTES

			Response Data				
Node Signal Src	Learned With	SIG	Async LVL	Clk LVL	Counter Mode	Counter Range	Priority Pin
U11-18	PROBE	000B	1	0	TRANS		
U11-18	I/O MODULE	000B	1	0	TRANS		
U11-19	PROBE	000E	1	0	TRANS		
U11-19	I/O MODULE	000E	1	0	TRANS		
U11-20	PROBE	000A	1	0	TRANS		
U11-20	I/O MODULE	000A	1	0	TRANS		
U11-21	PROBE	000A	1	0	TRANS		
U11-21	I/O MODULE	000A	1	0	TRANS		
U11-25	PROBE	000A	1	0	TRANS		
U11-25	I/O MODULE	000A	1	0	TRANS		
U11-26	PROBE	001A	1	0	TRANS		
U11-26	I/O MODULE	001A	1	0	TRANS		
U11-27	PROBE	000F	1	0	TRANS		
U11-27	I/O MODULE	000F	1	0	TRANS		
U11-28	PROBE	001B	1	0	TRANS		
U11-28	I/O MODULE	001B	1	0	TRANS		

Figure 4-64: Response File *(rs232_data)*

```
program rs232_lvl

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! STIMULUS PROGRAM for DUART serial circuits at TTL levels.              !
!                                                                         !
! Stimulus programs and response files are used by GFI to backtrace      !
! from a failing node.  The stimulus program must create repeatable UUT !
! activity and the response file contains the known-good responses for  !
! the outputs in the UUT that are stimulated by the stimulus program.   !
!                                                                         !
! This stimulus program is one of the programs which creates activity   !
! in the kernel area of the UUT.  These programs create activity with   !
! or without the ready circuit working properly.  Because of this, all  !
! the stimulus programs in the kernel area must disable the READY input !
! to the pod, then perform the stimulus, then re-enable the READY input !
! to the pod.  The 80286 microprocessor has a separate bus controller;  !
! for this reason, disabling ready and performing stimulus can get the  !
! bus controller out of synchronization with the pod.  Two fault        !
! handlers trap pod timeout conditions that indicate the bus controller !
! is out of synchronization.  The recover() program is executed to      !
! resynchronize the bus controller and the pod.                          !
!                                                                         !
! TEST PROGRAMS CALLED:                                                   !
!    rs232_init ()                      Initialize the RS232 circuit.  !
!                                                                         !
!    check_loop ()                      Check that loop-back switches !
!                                       are closed.  Prompt if the    !
!                                       switches are not closed.      !
!                                                                         !
! GRAPHICS PROGRAMS CALLED:                                               !
!    (none)                                                               !
!                                                                         !
! Local Variables Modified:                                               !
!    q                                  String to accept keypad input.!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Main Declarations                                                       !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

   declare string q

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!   Main part of STIMULUS PROGRAM                                         !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

! Let GFI determine the measurement device.

   if (gfi control) = "yes" then
      devname = gfi device
   else
      devname = "/mod1"
   end if
   print "Stimulus Program RS232_LVL"
```

(continued on the next page)

Figure 4-65: Stimulus Program *(rs232_lvl)*

```
! Set addressing mode and setup measurement device.

   reset device devname
   execute rs232_init()
   setspace space (getspace space "i/o", size "byte")
   sync device "/probe", mode "freerun"
   threshold device "/probe", level "rs232"

   execute check_loop()          ! check if the loop back switches are set.

! Present stimulus to UUT.

   arm device devname            ! Start response capture.
      write addr $2006, data $55   !     Txd port A
      write addr $2006, data  $D   !     Txd port A
      write addr $2016, data $55   !     Txd port B
      write addr $2016, data  $D   !     Txd port B
   readout device devname        ! End response capture.

end program
```

Figure 4-65: Stimulus Program *(rs232_lvl) - continued*

STIMULUS PROGRAM NAME: RS232_LVL
DESCRIPTION: SIZE: 249 BYTES

Node Signal Src	Learned With	SIG	Async LVL	Clk LVL	Counter Mode	Counter Range	Priority Pin
			--- Response Data ---				
U12-7	PROBE		1 0		TRANS	8	
U12-14	PROBE		1		TRANS	0	
J2-3	PROBE		1 0		TRANS	8	
J2-5	PROBE		1		TRANS	0	
R22-2	PROBE		1		TRANS	0	
U12-1	PROBE		1		TRANS		
U12-2	PROBE		1		TRANS		
C15-2	PROBE		1X		TRANS		
U12-4	PROBE		1X		TRANS		
C17-2	PROBE		1X0		TRANS		
U12-6	PROBE		X		TRANS		

Figure 4-66: Response File *(rs232_lvl)*

```
program ttl_lvl

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! STIMULUS PROGRAM for DUART serial circuits at TTL levels.              !
!                                                                       !
! Stimulus programs and response files are used by GFI to back-trace    !
! from a failing node.  The stimulus program must create repeatable UUT !
! activity and the response file contains the known-good responses for  !
! the outputs in the UUT that are stimulated by the stimulus program.   !
!                                                                       !
! TEST PROGRAMS CALLED:                                                 !
!    rs232_init ()                      Initialize the RS232 circuit.  !
!                                                                       !
!    check_loop ()                      Check that loop-back switches  !
!                                       are closed.  Prompt if the     !
!                                       switches are not closed.       !
!                                                                       !
! GRAPHICS PROGRAMS CALLED:                                             !
!    (none)                                                             !
!                                                                       !
! Local Variables Modified:                                             !
!    q                                  String to accept keypad input.!
!    devname                            Measurement device             !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Main Declarations                                                     !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

   declare string q

! Let GFI determine the measurement device.

   if (gfi control) = "yes" then
      devname = gfi device
   else
      devname = "/mod1"
   end if
   print "Stimulus Program TTL_LVL"
```

(continued on the next page)

Figure 4-67: Stimulus Program *(ttl_lvl)*

```
! Set addressing mode and setup measurement device.

   reset device devname
   execute rs232_init()
   setspace space (getspace space "i/o", size "byte")
   sync device "/probe", mode "pod"
   sync device "/pod", mode "data"
   threshold device "/probe", level "ttl"

   execute check_loop()            ! Check if loop back switches are closed.

! Present stimulus to UUT.

   arm device devname              ! Start response capture.
      write addr $2006, data $55   !    Txd port A
      write addr $2006, data  $D   !    Txd port A
      write addr $2016, data $55   !    Txd port B
      write addr $2016, data  $D   !    Txd port B
      write addr $201C, data $FF   !
      write addr $201E, data $FF   ! Pulse timer interrupt.
   readout device devname          ! End response capture.

end program
```

Figure 4-67: Stimulus Program *(ttl_lvl) - continued*

STIMULUS PROGRAM NAME: TTL_LVL
DESCRIPTION: SIZE: 368 BYTES

Node Signal Src	Learned With	SIG	Async LVL	Clk LVL	Counter Mode	Counter Range	Priority Pin
			Response Data				
U11-13	PROBE		1 0		TRANS	8	
U11-14	PROBE		1 0		TRANS	1	
U11-33	PROBE		1 0		TRANS	8	
U11-33	I/O MODULE		1 0		TRANS	8	
U11-15	PROBE		1 0		TRANS	1	
U11-15	I/O MODULE		1 0		TRANS	1	
U11-17	PROBE		1 0		TRANS	1	
U11-24	PROBE		1 0		TRANS	0	
U11-24	I/O MODULE		1 0		TRANS	0	
U12-12	PROBE		1 0		TRANS	8	
U12-9	PROBE		1 0		TRANS	1	
U13-6	PROBE		1 0		TRANS	8	
U13-6	I/O MODULE		1 0		TRANS	8	
U13-8	I/O MODULE		1 0		TRANS	8	

Figure 4-68: Response File *(ttl_lvl)*

```
program rs232_init

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! INITIALIZATION PROGRAM for SERIAL I/O functional block.                   !
!                                                                           !
! TEST PROGRAMS CALLED:                                                     !
!    (none)                                                                 !
!                                                                           !
! GRAPHICS PROGRAMS CALLED:                                                 !
!    (none)                                                                 !
!                                                                           !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

    setspace space (getspace space "i/o", size "byte")

    write addr $2004, data $15    ! Cmnd Reg A: reset Rxd
    write addr $2004, data $25    ! Cmnd Reg A: reset Txd
    write addr $2004, data $35    ! Cmnd Reg A: reset Errors
    write addr $2004, data $45    ! Cmnd Reg B: reset Rxd
    write addr $2004, data $55    ! Cmnd Reg B: reset Txd
    write addr $2014, data $15    ! Cmnd Reg A: reset Rxd
    write addr $2014, data $25    ! Cmnd Reg A: reset Txd
    write addr $2014, data $35    ! Cmnd Reg A: reset Errors
    write addr $2014, data $45    ! Cmnd Reg B: reset Rxd
    write addr $2014, data $55    ! Cmnd Reg B: reset Txd
    write addr $2000, data $13    ! Mode register 1A
    write addr $2000, data  7     ! Mode register 2A
    write addr $2010, data $13    ! Mode register 1B
    write addr $2010, data  7     ! Mode register 2B
    write addr $2002, data $66    ! Clock select register A
    write addr $2012, data $BB    ! Clock select register B
    write addr $200A, data $20    ! Interrupts for port B
    read addr $2002               ! Read Status Reg A
    read addr $2000               ! Read Command Reg A

end program
```

Figure 4-69: Initialization Program *(rs232_init)*

Summary of Complete Solution for Serial I/O 4.6.7.

The entire set of programs and files needed to test and GFI troubleshoot the Serial I/O functional block is shown below. The format below is similar to a 9100A/9105A UUT directory (you could consider the functional block to be a small UUT), but in addition shows the use of each program and the location in this manual for each file.

UUT DIRECTORY
(Complete File Set for Serial I/O)

Programs (PROGRAM):

TEST_RS232	Functional Test	Section 4.6.5
RS232_DATA	Stimulus Program	Figure 4-63
RS232_LVL	Stimulus Program	Figure 4-65
TTL_LVL	Stimulus Program	Figure 4-67
FREQUENCY	Stimulus Program	Figure 4-117
LEVELS	Stimulus Program	Figure 4-92
RS232_INIT	Initialization Program	Figure 4-69

Stimulus Program Responses (RESPONSE):

RS232_DATA	Figure 4-64
RS232_LVL	Figure 4-66
TTL_LVL	Figure 4-68
FREQUENCY	Figure 4-118
LEVELS	Figure 4-93

Node List (NODE):

NODELIST	Appendix B

Text Files (TEXT):

Reference Designator List (REF):

REFLIST	Appendix A

Compiled Database (DATABASE):

GFIDATA	Compiled by the 9100A

VIDEO OUTPUT FUNCTIONAL BLOCK 4.7.

Introduction to Video Output Circuits 4.7.1.

Video output circuits are part of larger video display circuits. In general, video display circuits can be divided into two basic classes: video display controllers and intelligent command-oriented display systems, which are a superset of video display controllers. In this manual, we will limit our discussion to video display controllers.

Figure 4-70 is a block diagram of a typical, complete video display controller, of which video output is one functional block. On the Demo/Trainer UUT, address decoding is partitioned as a separate functional block and is described later in Sections 4.11. Often, much of the video control circuitry is performed by a VDC (video display controller) chip. On the Demo/Trainer UUT, most of the video output block is implemented with a single LSI chip.

The video output block typically performs all or some of the following functions:

- Converts video RAM character or dot graphics signals (typically on a bus) to higher-speed (typically serial) pixel outputs that drive the monitor. This is usually done with shift registers.
- Modifies the meaning of video RAM color-data outputs according to a color look-up table or palette RAM.
- Converts the pixel output to analog or digital signals compatible with the monitor.

Considerations for Testing and Troubleshooting 4.7.2.

The Video Output functional block simply processes information presented to it by the Video Control and Video RAM functional blocks. All three video blocks can be considered good if the

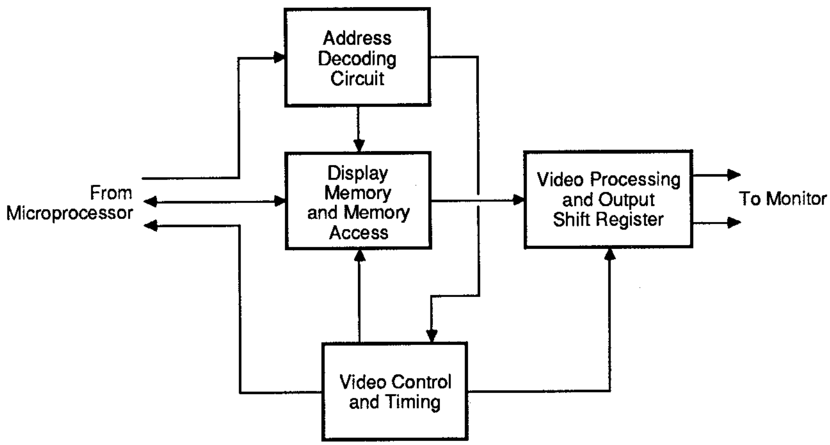


Figure 4-70: Typical Video Controller Circuit

final outputs of the Video Output functional block are good. Because of this, the Video Output functional block is tested first.

While a generalized approach to testing Video Control functional blocks is feasible, testing Video Output and Video RAM functional blocks is strongly dependent on the design of the UUT.

The general approach for testing video circuits is to initialize video RAM and any other RAM sections so that some regular pattern will occur each frame. When this is done for each mode, there should be a way to capture stable signatures on the outputs.

To test video output:

1. Initialize the video control circuit.
2. Initialize the video RAM with blinking disabled.

For horizontal sync and vertical sync:

3. Probe the horizontal sync and vertical sync outputs.
4. Compare all frequencies to those from a known-good UUT.

For video outputs:

3. Connect the clock module's external CLOCK, START, and STOP lines.
4. Compare signatures of TTL-level video outputs to those from a known-good UUT.
5. You can check the level history of any non-TTL-level video outputs to verify that they are toggling.

Connecting the Start and Stop lines to the vertical sync line will usually work. The Clock line should be connected to the high-speed clock that drives the video output shift registers.

Video outputs are sometimes high-speed analog signals. Fortunately, any digital-to-analog conversion is usually done at the last step before the monitor. By measuring the digital signals that drive digital-to-analog converters, most of the circuit can be tested with the 9100A/9105A.

Furthermore, many of the monitors for personal computers accept TTL-level signals. Video cards that put out such TTL-level signals can be checked by the 9100A/9105A at these TTL-level video outputs.

Choose your measurement device to suit the data rate of the signals you are measuring. If the Video Output signals exceed the maximum data rate of the I/O modules (10 MHz), the probe should be used.

Testing should be started in the mode that tests as much of the video display circuitry as possible. In a color graphics circuit, this might be the highest resolution mode with the most colors. Simple tests in other modes can then be used to cover circuitry not tested with the more extensive test.

When selecting the Start and Stop signals for signature analysis, connect to the slowest repetitive signal, relative to the circuitry being tested. This will usually be the vertical sync signal.

To test blinking cursors, it may be easiest simply to probe an internal line to make sure it is blinking rather than run a test program. Other similar modes may also be faster to test with the probe.

Video Output Circuit Example 4.7.3.

The Video Output functional block, shown in Figure 4-71, consists of the 2675 attributes controller chip (U78) and associated circuitry. The 2675 contains a programmable dot clock divider to generate a character clock, a high-speed shift register to convert parallel pixel data into a serial stream, latches and logic to apply visual attributes (e.g. colors) to the resulting display, and logic to display a cursor on the monitor.

Associated circuitry includes latches U87 and U76, which clock in display information provided by the character PROM, and Q1 and Q2, which boost the video signal before it is mixed with the horizontal and vertical sync signals at the monitor to be connected at J3.

The circuitry from the Video Control functional block up to the 2675 attributes controller chip (U78) clocks video data in character format. This means that the code for a character and the attributes for that character are clocked toward the 2675 chip. The attributes controller converts the parallel character information to pixel data.

The circuitry after U78 should be initialized without blinking characters in the video screen, otherwise the pixel stream will change when the characters blink. However, the circuitry between the video control and U78 may contain blinking characters, since the blinking characters are determined by an attribute bit which is stable.

Keystroke Functional Test 4.7.4.

Before testing any part of the video display circuitry, the video controller and video RAM must be initialized. The TL/1 programs *video_init*, *video_fill*, and *video_fil2* are used for initialization of the Demo/Trainer UUT video circuitry. Figure 4-79 shows the *video_init* program, which contains a sequence of *write* commands needed to initialize the Video Control functional block. Figures 4-80 and 4-81 show the *video_fill* and *video_fil2* programs, which write blocks of data to video RAM.

1. Use the EXEC key with the following commands to initialize the video circuit and to fill the video RAM with a test pattern.

```
EXECUTE UUT DEMO PROGRAM VIDEO_INIT
EXECUTE UUT DEMO PROGRAM VIDEO_FIL1
```

2. Connect the external control lines of the clock module as follows:

   Clock to 16MHZ (U25-9)
   Start to VSYNC (U72-18)
   Stop to VSYNC (U72-18)
   Enable to BLANK (U72-17)

3. Use the SYNC and PROBE keys with the following commands to measure the node response for the video output signals (TTV1, TTLV2, and VIDEO). The pins to be probed and the correct responses are shown in the response table of Figure 4-71.

```
SYNC PROBE TO EXT MOD ENABLE LOW CLOCK ↓...
... START ↓ STOP ↑
ARM  PROBE FOR CAPTURE USING SYNC
SHOW PROBE CAPTURED RESPONSES <see ...
... response table>
```

4. Use the PROBE and SOFT KEYS keys with the following command to measure frequency of the video synchronization signals. The results for each sync signal (HSYNC and VSYNC) are shown in the response table of Figure 4-71.

```
FREQ AT PROBE
```

(This page is intentionally blank.)

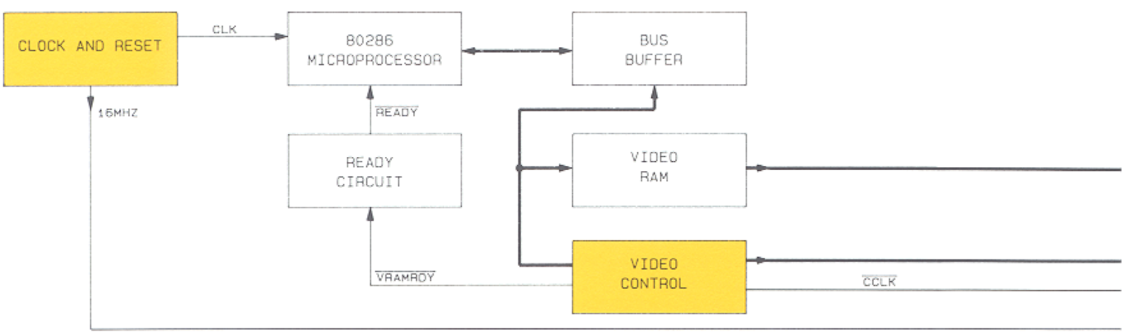
Keystroke Functional Test

CONNECTION TABLE

STIMULUS	MEASUREMENT CONTROL	MEASUREMENT
(NONE)	CLOCK MODULE CLOCK U25-9 START U72-18 STOP U72-18 ENABLE U72-17	PROBE U78 J3

RESPONSE TABLE

SIGNAL	PART PIN	MEASUREMENT
HSYNC	-8	16.7 TO 16.8 KHz
VSYNC	-9	59 TO 61 Hz
VIDEO	J3-7	1X0 (ASYNC LEVEL)
TTLV1	U78-28	B013 (SIG)
TTLV2	-29	E4A7 (SIG)



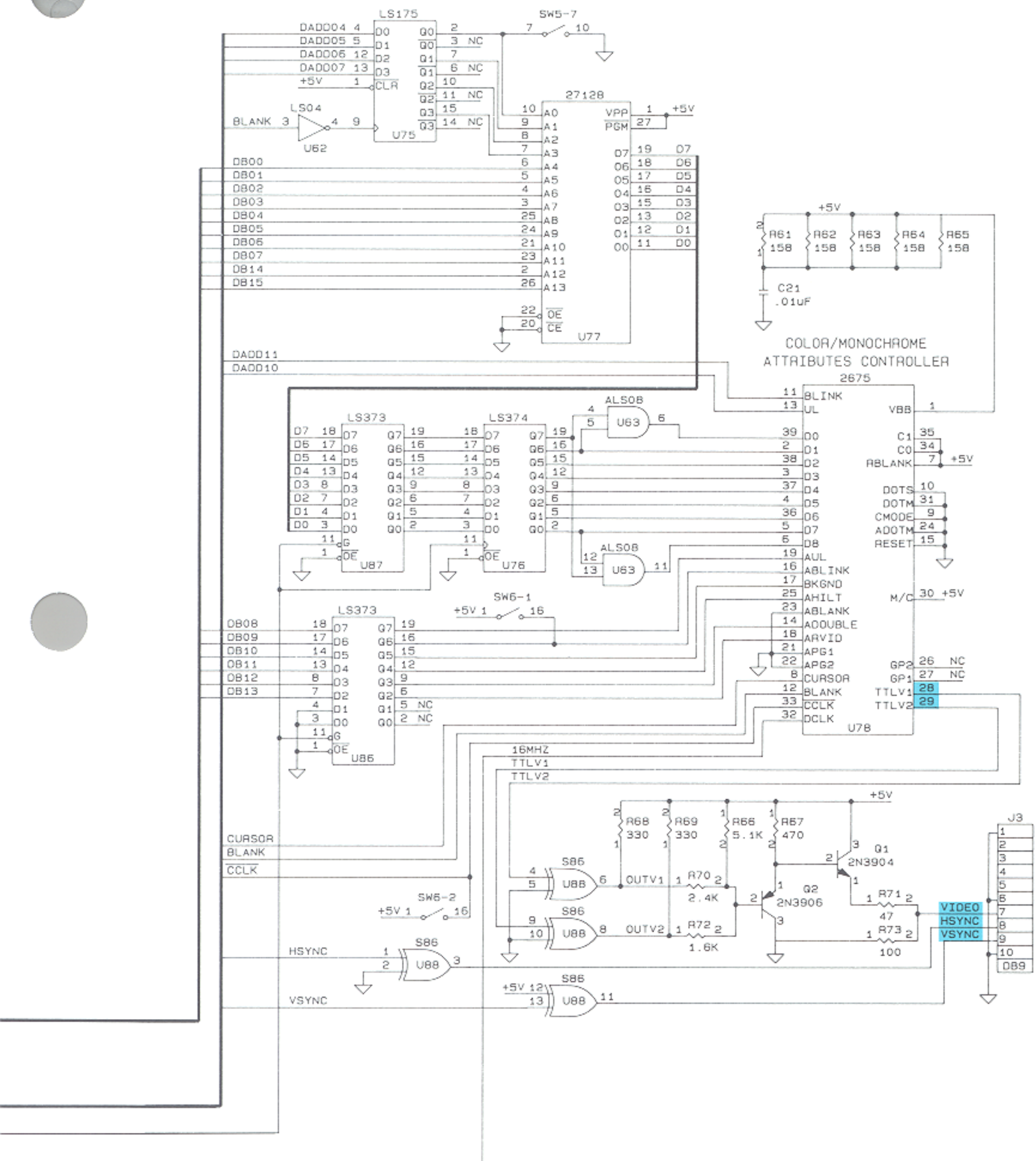


Figure 4-71: Video Output Functional Test

Programmed Functional Test 4.7.5.

The *test_video* program is the programmed functional test for the Video Output functional block. This program uses the *gfi test* command and the probe to measure the output of the video circuit.

If the video outputs fail, the program executes programmed functional tests for the Video Control functional block and the Video RAM functional block. If either of these functional tests fails, GFI will take control and begin backtracing. If neither test fails, the problem is in the Video Output functional block and the *test_video* program passes control to GFI to start backtracing from the video outputs that failed.

```
program test_video

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! FUNCTIONAL TEST of the VIDEO functional block                          !
!                                                                        !
! This program tests the VIDEO functional block of the Demo/Trainer.     !
! The video test uses the gfi test command to run stimulus programs and !
! to check the outputs of the Video circuit against the stimulus program!
! response files.  The gfi test command returns a passes status if all  !
! the measured results from running the stimulus programs match the      !
! response files.  Otherwise the gfi test command returns a fails        !
! status.                                                                !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

! Setup and initialization.

   connect clear "yes"
   podsetup 'enable ~ready' "on"
   print "\nl\nl"

! Main part of Test.

   if gfi test "J3-8" fails then fault video_scan \ return
   if gfi test "J3-9" fails then fault video_scan \ return

   if gfi test "U78-11" fails then fault video_scan \ return
   if gfi test "U78-28" fails then fault video_output \ return
   if gfi test "U78-29" fails then fault video_output \ return
   if gfi test "J3-7" fails then fault video_output \ return

end program
```

Stimulus Programs and Responses 4.7.6.

Figure 4-72 is the stimulus program planning diagram for the Video Output functional block. The *video_freq* stimulus program initializes the video registers and then measures frequency. The *video_scan* stimulus program initializes video RAM with blinking characters by executing *video_fill*. The *video_out* stimulus program initializes video RAM without any blinking characters by executing *video_fil2*. Not having blinking characters results in stable signatures in the circuitry between U78 and the video output connector.

All the stimulus programs execute *video_init* before any measurements are made on the video circuitry.

Stimulus Program Planning

PROGRAM: VIDEO_FREQ
EXECUTES VIDEO_INIT AND MEASURES FREQUENCY
MEASUREMENT AT: U78-33 U88-3,11 U62-4

PROGRAM: VIDEO_SCAN
EXECUTES VIDEO_INIT, VIDEO_FIL1, AND MEASURES ALL CIRCUITRY WHERE DATA IS CLOCKED THROUGH BY CHARACTERS
MEASUREMENT AT: U75-2,3,7,6,10,11,15,14 U77-11,12,13,15,16,17,18,19 U87-2,5,6,9,12,15,16,19 U76-2,5,6,9,12,15,16,19 U63-6,11 U86-6,9,12,15,16,19

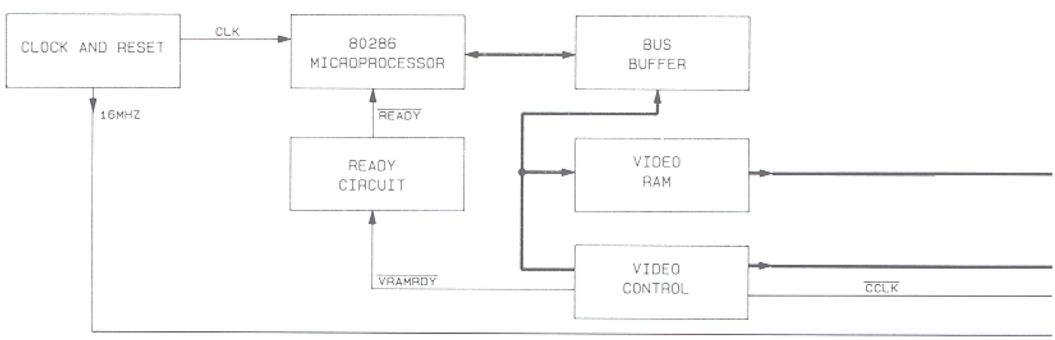
PROGRAM: VIDEO_OUT
EXECUTES VIDEO_INIT, VIDEO_FIL2, AND MEASURES ALL CIRCUITRY WHERE DATA IS CLOCKED THROUGH BY PIXELS
MEASUREMENT AT: U78-28,29 U88-6,8 R72-2, R71-2 Q1-1 Q2-1

INITIALIZATION PROGRAM: VIDEO_FIL1
INITIALIZES VIDEO RAM WITH BLINKING CHARACTERS
MEASUREMENT AT: (NONE)

INITIALIZATION PROGRAM: VIDEO_INIT
INITIALIZES VIDEO REGISTERS TO STANDARD OPERATING MODE
MEASUREMENT AT: (NONE)

INITIALIZATION PROGRAM: VIDEO_FIL2
INITIALIZES VIDEO RAM WITHOUT BLINKING CHARACTERS
MEASUREMENT AT: (NONE)

PROGRAM: LEVELS
MEASURES STATIC LEVELS
MEASUREMENT AT: R61-1



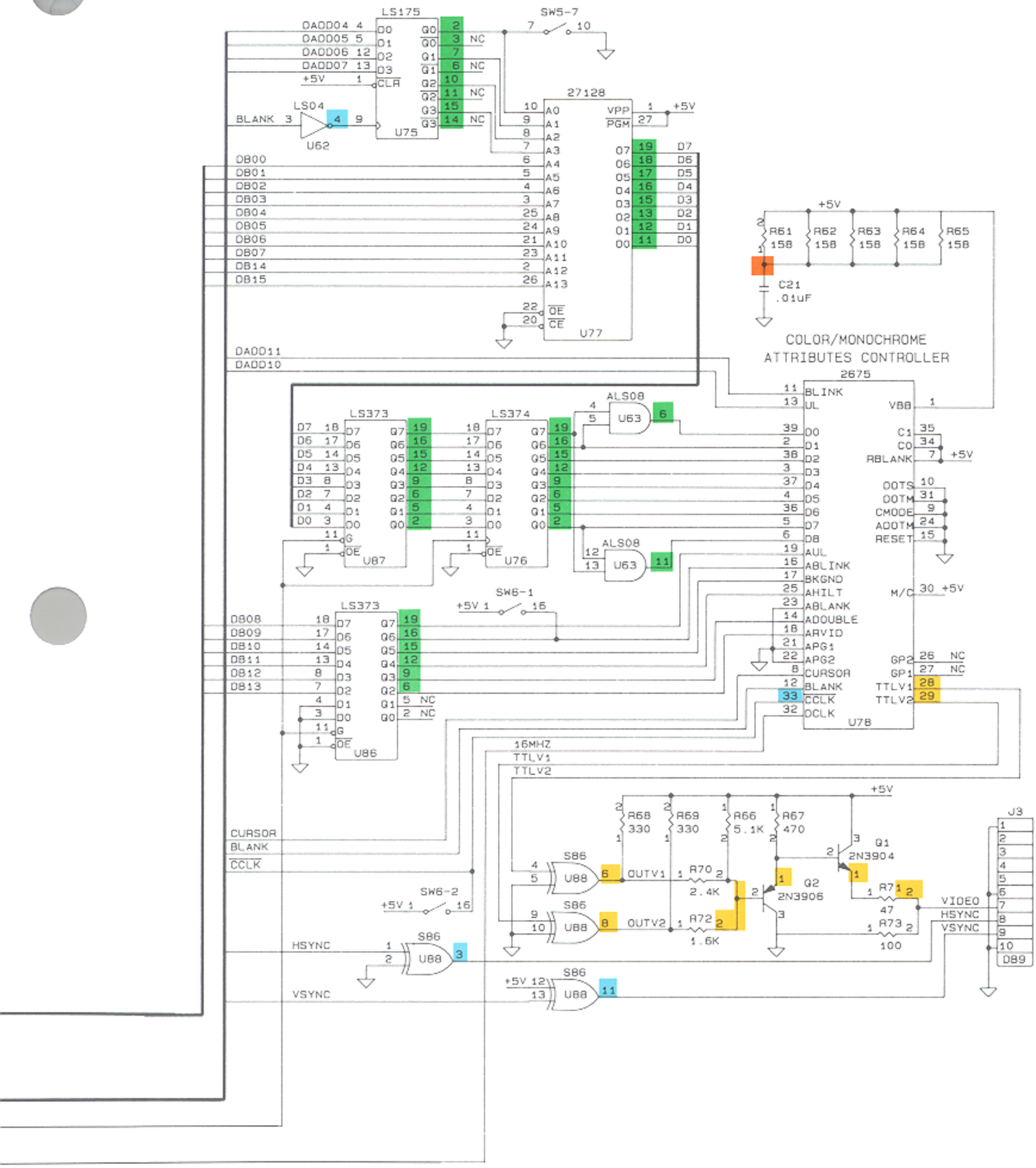


Figure 4-72: Video Output Stimulus Program Planning

```
program video_freq

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! STIMULUS PROGRAM to measure frequency in video circuit.               !
!                                                                       !
! Stimulus programs and response files are used by GFI to backtrace     !
! from a failing node.  The stimulus program must create repeatable UUT !
! activity and the response file contains the known-good responses for  !
! the outputs in the UUT that are stimulated by the stimulus program.   !
!                                                                       !
! TEST PROGRAMS CALLED:                                                 !
!    video_init ()                              Initialize video        !
!                                                                       !
! GRAPHICS PROGRAMS CALLED:                                             !
!    (none)                                                             !
!                                                                       !
! Local Variables Modified:                                             !
!     devname                                   Measurement device      !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!   FAULT HANDLERS:                                                     !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

handle pod_timeout_enabled_line
   recover()
end handle
handle pod_timeout_recovered
   recover()
end handle

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!   Main part of STIMULUS PROGRAM                                       !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

recover_times = 0

! Let GFI determine the measurement device

   if (gfi control) = "yes" then
      devname = gfi device
   else
      devname = "/probe"
   end if
   print "\1B[2J"
   print "Stimulus Program VIDEO_FREQ"

! Initialize and Setup desired measurement mode

   reset device devname
   execute video_init()
   counter device devname, mode "freq"

! No stimulus is applied; response is frequency

   arm device devname           ! Start response capture
   readout device devname       ! End response capture

end program
```

Figure 4-73: Stimulus Program *(video_freq)*

STIMULUS PROGRAM NAME: VIDEO_FREQ
DESCRIPTION: SIZE: 345 BYTES

Node Signal Src	Learned With	SIG	Async LVL	Clk LVL	Counter Mode	Counter Range	Priority Pin
	Response Data						
U72-17	PROBE		1 0		FREQ	14300-14500	
U72-17	I/O MODULE		1 0		FREQ	14300-14500	
U72-18	PROBE		1 0		FREQ	59-61	
U72-18	I/O MODULE		1 0		FREQ	59-61	
U72-19	PROBE		1 0		FREQ	16700-16800	
U72-19	I/O MODULE		1 0		FREQ	16700-16800	
U78-33	PROBE		1 0		FREQ	1770000-1780000	
U78-33	I/O MODULE		1 0		FREQ	1770000-1780000	
U88-3	PROBE		1 0		FREQ	16700-16800	
U88-11	PROBE		1 0		FREQ	59-61	
U70-11	PROBE		1 0		FREQ	1770000-1780000	
U70-11	I/O MODULE		1 0		FREQ	1770000-1780000	
U62-4	I/O MODULE		1 0		FREQ	14300-14500	

Figure 4-74: Response File *(video_freq)*

```
program video_out

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! STIMULUS PROGRAM measures character scan circuitry from U78 to output.!
!                                                                         !
! Stimulus programs and response files are used by GFI to backtrace      !
! from a failing node.  The stimulus program must create repeatable UUT  !
! activity and the response file contains the known-good responses for   !
! the outputs in the UUT that are stimulated by the stimulus program.    !
!                                                                         !
! TEST PROGRAMS CALLED:                                                   !
!    video_init ()                       Initialize video ciruit.         !
!                                                                         !
!    video_fil2 ()                       Initialize data in video RAM     !
!                                        with no blinking characters      !
!                                                                         !
!    check_meas (device, start, stop, clock, enable)                      !
!                                        Checks to see if the measure-    !
!                                        ment is complete using the       !
!                                        TL/1 checkstatus command.  If    !
!                                        the measurement times out then!
!                                        redisplay connect locations.     !
!                                                                         !
! GRAPHICS PROGRAMS CALLED:                                               !
!    (none)                                                               !
!                                                                         !
! Local Variables Modified:                                               !
!    done                                returned from check_meas()       !
!    devname                             Measurement device               !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Main Declarations                                                       !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

   declare numeric done = 0

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!   Main part of STIMULUS PROGRAM                                         !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

! Let GFI determine measurement device.

   if (gfi control) = "yes" then
      devname = gfi device
   else
      devname = "/probe"
   end if
   print "\1B[2J"
   print "Stimulus Program VIDEO_OUT"
```

(continued on the next page)

Figure 4-75: Stimulus Program *(video_out)*

```
! Initialize and Prompt user to connect external lines

   execute video_init()
   execute video_fil2()
   connect device devname, start "U88-13", stop "U88-13", clock "U25-9", common "gnd"

! Setup desired measurement modes.

   reset device devname
   sync device devname, mode "ext"
   enable device devname, mode "always"
   edge device devname, start "-", stop "+", clock "-"
   old_cal = getoffset device devname
   setoffset device devname, offset (1000000 + 40)

! Present stimulus to UUT.

   loop until done = 1
      arm device devname
         done = check_meas(devname, "U88-13", "U88-13", "U25-9", "*")
      readout device devname
   end loop

   setoffset device devname, offset old_cal
end program
```

Figure 4-75: Stimulus Program *(video_out) - continued*

STIMULUS PROGRAM NAME: VIDEO_OUT
DESCRIPTION: SIZE: 200 BYTES

	------------------- Response Data --------------------						
Node Signal Src	Learned With	SIG	Async LVL	Clk LVL	Counter Mode	Counter Range	Priority Pin
U78-28	PROBE	B013	1 0		TRANS	4431	
U78-29	PROBE	E4A7	1 0		TRANS	6359	
U88-6	PROBE		1 0		TRANS	4431	
U88-8	PROBE		1 0		TRANS	6359	
R72-2	PROBE		1X0		TRANS		
Q2-1	PROBE		1X		TRANS		
Q1-1	PROBE		1X0		TRANS		
R71-2	PROBE		1X0		TRANS		

Figure 4-76: Response File *(video_out)*

```
program video_scan

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! STIMULUS PROGRAM to measure character scan circuitry from U72 to U78. !
!                                                                       !
! Stimulus programs and response files are used by GFI to back-trace    !
! from a failing node.  The stimulus program must create repeatable UUT !
! activity and the response file contains the known-good responses for  !
! the outputs in the UUT that are stimulated by the stimulus program.   !
!                                                                       !
! TEST PROGRAMS CALLED:                                                 !
!    video_init ()                       Initialize video ciruit.       !
!                                                                       !
!    video_fill ()                       Initialize data in video RAM   !
!                                                                       !
!    check_meas (device, start, stop, clock, enable)                    !
!                                        Checks to see if the measure-  !
!                                        ment is complete using the     !
!                                        TL/1 checkstatus command.  If  !
!                                        the measurement times out then!
!                                        redisplay connect locations.   !
!                                                                       !
! GRAPHICS PROGRAMS CALLED:                                             !
!    (none)                                                             !
!                                                                       !
! Local Variables Modified:                                             !
!    done                                 returned from check_meas()    !
!    devname                                   Measurement device       !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Main Declarations                                                     !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

   declare numeric done = 0

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! FAULT HANDLERS:                                                       !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

handle pod_timeout_enabled_line
   recover()
end handle
handle pod_timeout_recovered
   recover()
end handle

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!   Main part of STIMULUS PROGRAM                                       !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

recover_times = 0
```

(continued on the next page)

Figure 4-77: Stimulus Program *(video_scan)*

```
! Let GFI determine measurement device.

   if (gfi control) = "yes" then
      devname = gfi device
      measure_ref = gfi ref
   else
      devname = "/mod1"
      measure_ref = "U72"
   end if
   print "Stimulus Program VIDEO_SCAN"

! Initialize and Prompt user to connect external lines

   execute video_init()
   execute video_fill()
   connect device devname, start "U88-13", stop "U88-13", enable "U78-12",
                           clock "U78-33", common "gnd"

! Setup desired mesurement modes.

   reset device devname
   sync device devname, mode "ext"
   enable device devname, mode "low"
   edge device devname, start "-", stop "+", clock "-"

! Present stimulus to the UUT.
! The blink signal node (U72-23 to U78-11) has a signature of 0000 50% of the time
! and the signature in BLINK_SIG the rest of the time.  If U72 or U78-11 is being
! tested, make sure both a zero and the signature in BLINK_SIG are measured
! on the node.  The signature that gfi will evaluate is the signature in the
! variable BLINK_SIG.

   done = 0   \  done2 = 0
   cnt = 0  \  blink = 0
   loop until done = 1 and done2 = 1 or cnt > 12
      arm device devname
         done = check_meas(devname, "U88-13", "U88-13", "U78-33", "U78-12")
         if done = 1 then if checkstatus(devname) <> $F then done2 = 1
      readout device devname
      if measure_ref = "U78-11" then
         if (sig device devname, pin 11)=0 then blink = 1
         if (sig device devname, pin 11)=BLINK_SIG and blink=1 then done2=1
      else if measure_ref = "U72" then
         if (sig device "U72", pin 23)=0 then blink = 1
         if (sig device "U72", pin 23)=BLINK_SIG and blink = 1 then done2 = 1
      else
         done2 = 1              ! Don't loop if not U72 or U78-11
      end if
      cnt = cnt + 1
   end loop

end program
```

Figure 4-77: Stimulus Program *(video_scan) - continued*

```
STIMULUS PROGRAM NAME: VIDEO_SCAN
DESCRIPTION:                                             SIZE:         1,710 BYTES
```

Node Signal Src	Learned With	SIG	Async LVL	Clk LVL	Counter Mode	Counter Range	Priority Pin
U74-9	I/O MODULE	4155	1 0		TRANS		
U74-10	I/O MODULE	3F33	1 0		TRANS		
U74-11	I/O MODULE	A65A	1 0		TRANS		
U74-13	I/O MODULE	9024	1 0		TRANS		
U74-14	I/O MODULE	DE6D	1 0		TRANS		
U74-15	I/O MODULE	D6FA	1 0		TRANS		
U74-16	I/O MODULE	7AC3	1 0		TRANS		
U74-17	I/O MODULE	0477	1 0		TRANS		
U85-9	I/O MODULE	A814	1 0		TRANS		
U85-10	I/O MODULE	C26B	1 0		TRANS		
U85-11	I/O MODULE	D909	1 0		TRANS		
U85-13	I/O MODULE	5FAA	1 0		TRANS		
U85-14	I/O MODULE	5925	1 0		TRANS		
U85-15	I/O MODULE	610D	1 0		TRANS		
U85-16	I/O MODULE	B8AB	1 0		TRANS		
U85-17	I/O MODULE	ADD3	1 0		TRANS		
U84-12	I/O MODULE	4155	1 0		TRANS		
U84-9	I/O MODULE	3F33	1 0		TRANS		
U84-7	I/O MODULE	A65A	1 0		TRANS		
U84-4	I/O MODULE	9024	1 0		TRANS		
U83-12	I/O MODULE	DE6D	1 0		TRANS		
U83-9	I/O MODULE	D6FA	1 0		TRANS		
U83-7	I/O MODULE	7AC3	1 0		TRANS		
U83-4	I/O MODULE	0477	1 0		TRANS		
U73-12	I/O MODULE	E941	1 0		TRANS		
U73-9	I/O MODULE	88B8	1 0		TRANS		
U73-7	I/O MODULE	60B0	1 0		TRANS		
U72-34	I/O MODULE	4155	1 0		TRANS		
U72-33	I/O MODULE	3F33	1 0		TRANS		
U72-32	I/O MODULE	A65A	1 0		TRANS		
U72-31	I/O MODULE	9024	1 0		TRANS		
U72-30	I/O MODULE	DE6D	1 0		TRANS		
U72-29	I/O MODULE	D6FA	1 0		TRANS		
U72-28	I/O MODULE	7AC3	1 0		TRANS		
U72-27	I/O MODULE	0477	1 0		TRANS		
U72-26	I/O MODULE	E941	1 0		TRANS		
U72-25	I/O MODULE	88B8	1 0		TRANS		
U72-24	PROBE	60B0	1 0		TRANS		
U72-24	I/O MODULE	60B0	1 0		TRANS		
U72-23	PROBE	D869	1 0		TRANS		
U72-23	I/O MODULE	D869	1 0		TRANS		
U72-7	PROBE	0000	0		TRANS		
U72-7	I/O MODULE	0000	0		TRANS		
U75-2	I/O MODULE	AC4E	1 0		TRANS		

(continued on the next page)

Figure 4-78: Response File *(video_scan)*

U75-3	I/O MODULE	6FB1	1 0	TRANS
U75-7	I/O MODULE	9B47	1 0	TRANS
U75-6	I/O MODULE	58B8	1 0	TRANS
U75-10	I/O MODULE	762E	1 0	TRANS
U75-11	I/O MODULE	B5D1	1 0	TRANS
U75-15	I/O MODULE	2C30	1 0	TRANS
U75-14	I/O MODULE	EFCF	1 0	TRANS
U77-11	I/O MODULE	7B80	1 0	TRANS
U77-12	I/O MODULE	8FE6	1 0	TRANS
U77-13	I/O MODULE	ADD1	1 0	TRANS
U77-15	I/O MODULE	EB37	1 0	TRANS
U77-16	I/O MODULE	FFE7	1 0	TRANS
U77-17	I/O MODULE	B708	1 0	TRANS
U77-18	I/O MODULE	55C3	1 0	TRANS
U77-19	I/O MODULE	B00D	1 0	TRANS
U87-2	I/O MODULE	7B80	1 0	TRANS
U87-5	I/O MODULE	8FE6	1 0	TRANS
U87-6	I/O MODULE	ADD1	1 0	TRANS
U87-9	I/O MODULE	EB37	1 0	TRANS
U87-12	I/O MODULE	FFE7	1 0	TRANS
U87-15	I/O MODULE	B708	1 0	TRANS
U87-16	I/O MODULE	55C3	1 0	TRANS
U87-19	I/O MODULE	B00D	1 0	TRANS
U76-2	PROBE	1ADB	1 0	TRANS
U76-2	I/O MODULE	1ADB	1 0	TRANS
U76-5	PROBE	444F	1 0	TRANS
U76-5	I/O MODULE	444F	1 0	TRANS
U76-6	PROBE	D65A	1 0	TRANS
U76-6	I/O MODULE	D65A	1 0	TRANS
U76-9	PROBE	4366	1 0	TRANS
U76-9	I/O MODULE	4366	1 0	TRANS
U76-12	PROBE	49EA	1 0	TRANS
U76-12	I/O MODULE	49EA	1 0	TRANS
U76-15	PROBE	4DDC	1 0	TRANS
U76-15	I/O MODULE	4DDC	1 0	TRANS
U76-16	PROBE	5B18	1 0	TRANS
U76-16	I/O MODULE	5B18	1 0	TRANS
U76-19	I/O MODULE	3EF2	1 0	TRANS
U63-11	PROBE	0C5B	1 0	TRANS
U63-11	I/O MODULE	0C5B	1 0	TRANS
U63-6	PROBE	66D3	1 0	TRANS
U63-6	I/O MODULE	66D3	1 0	TRANS
U86-6	PROBE	610D	1 0	TRANS
U86-6	I/O MODULE	610D	1 0	TRANS
U86-9	PROBE	5925	1 0	TRANS
U86-9	I/O MODULE	5925	1 0	TRANS
U86-12	PROBE	5FAA	1 0	TRANS
U86-12	I/O MODULE	5FAA	1 0	TRANS
U86-15	PROBE	D909	1 0	TRANS
U86-15	I/O MODULE	D909	1 0	TRANS
U86-16	PROBE	C26B	1 0	TRANS
U86-16	I/O MODULE	C26B	1 0	TRANS
U86-19	PROBE	A814	1 0	TRANS
U86-19	I/O MODULE	A814	1 0	TRANS

Figure 4-78: Response File *(video_scan) - continued*

```
program video_init

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! INITIALIZATION PROGRAM for the 2674 Advanced Video Display Controller.!
! The program executes two Master Reset commands followed by the init-  !
! ialization of 15 contiguous Initialization Registers.  Next 6 regis-  !
! ers are initialized which determine the screen memory mapping and the !
! cursor location.                                                      !
!                                                                       !
! This program must be executed before any video testing is performed,  !
! and must be re-executed whenever UUT power has been interrupted.      !
!                                                                       !
! TEST PROGRAMS CALLED:                                                 !
!    (none)                                                             !
!                                                                       !
! GRAPHICS PROGRAMS CALLED:                                             !
!    (none)                                                             !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

    setspace space (getspace space "i/o", size "byte")

    write ADDR 2, DATA   0     ! Master Reset Command
    write ADDR 2, DATA   0     ! Master Reset Command
    write ADDR 0, DATA $48     ! Write Initialization Register 0
    write ADDR 0, DATA $20     ! Write Initialization Register 1
    write ADDR 0, DATA $22     ! Write Initialization Register 2
    write ADDR 0, DATA $86     ! Write Initialization Register 3
    write ADDR 0, DATA $17     ! Write Initialization Register 4
    write ADDR 0, DATA $4F     ! Write Initialization Register 5
    write ADDR 0, DATA   9     ! Write Initialization Register 6
    write ADDR 0, DATA $28     ! Write Initialization Register 7
    write ADDR 0, DATA   0     ! Write Initialization Register 8
    write ADDR 0, DATA $10     ! Write Initialization Register 9
    write ADDR 0, DATA   0     ! Write Initialization Register 10
    write ADDR 0, DATA   0     ! Write Initialization Register 11
    write ADDR 0, DATA   0     ! Write Initialization Register 12
    write ADDR 0, DATA   0     ! Write Initialization Register 13
    write ADDR 0, DATA   0     ! Write Initialization Register 14
    write ADDR 4, DATA   1     ! Screen Start 1 Lower Register
    write ADDR 6, DATA   0     ! Screen Start 1 Upper Register
    write ADDR 8, DATA   0     ! Cursor Address Lower Register
    write ADDR $A, DATA  0     ! Cursor Address Upper Register
    write ADDR $C, DATA  0     ! Screen Start 2 Lower Register
    write ADDR $E, DATA  0     ! Screen Start 2 Upper Register
    write ADDR 2, DATA $29     ! Enable Screen On Command

end program
```

Figure 4-79: Initialization Program *(video_init)*

```
program video_fill

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! INITIALIZATION PROGRAM fills video RAM with every attribute & char     !
!                                                                        !
! TEST PROGRAMS CALLED:                                                  !
!    (none)                                                              !
!                                                                        !
! GRAPHICS PROGRAMS CALLED:                                              !
!    (none)                                                              !
!                                                                        !
! Text Files Accessed:                                                   !
!    vid_fill1                                                           !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

   setspace space (getspace space "memory", size "word")
   writeblock file "vid_fill1", format "motorola"

end program
```

Figure 4-80: Initialization Program *(video_fil1)*

```
program video_fil2

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! INITIALIZATION PROGRAM to fill video RAM with Non-Blinking characters !
!                                                                       !
! TEST PROGRAMS CALLED:                                                 !
!                                                                       !
! GRAPHICS PROGRAMS CALLED:                                             !
!    (none)                                                             !
!                                                                       !
! Text Files Accessed:                                                  !
!    vid_fill2                                                          !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

   setspace space (getspace space "memory", size "word")
   writeblock file "vid_fill2", format "motorola"

end program
```

Figure 4-81: Initialization Program *(video_fil2)*

Summary of Complete Solution for Video Output 4.7.7.

The entire set of programs and files needed to test and GFI troubleshoot the Video Output functional block is shown below. The format below is similar to a 9100A/9105A UUT directory (you could consider the functional block to be a small UUT), but in addition shows the use of each program and the location in this manual for each file.

UUT DIRECTORY
(Complete File Set for Video Output)

Programs (PROGRAM):

TEST_VIDEO	Functional Test	Section 4.7.5
VIDEO_FREQ	Stimulus Program	Figure 4-73
VIDEO_OUT	Stimulus Program	Figure 4-75
VIDEO_SCAN	Stimulus Program	Figure 4-77
LEVELS	Stimulus Program	Figure 4-92
VIDEO_INIT	Initialization Program	Figure 4-79
VIDEO_FIL1	Initialization Program	Figure 4-80
VIDEO_FIL2	Initialization Program	Figure 4-81

Stimulus Program Responses (RESPONSE):

VIDEO_FREQ		Figure 4-74
VIDEO_OUT		Figure 4-76
VIDEO_SCAN		Figure 4-78
LEVELS		Figure 4-93

Node List (NODE):

NODELIST		Appendix B

Text Files (TEXT):

VID_FILL1	Initialization Data File	
VID_FILL2	Initialization Data File	

Reference Designator List (REF):

REFLIST		Appendix A

Compiled Database (DATABASE):

GFIDATA		Compiled by the 9100A

VIDEO CONTROL FUNCTIONAL BLOCK 4.8.

Introduction to Video Control Circuits 4.8.1.

After initialization by the microprocessor, the video control block typically generates four major timing functions:

- Character timing for serializing character or dot graphics information to the Video Output functional block.
- Address generation and timing control for accessing the video RAM.
- Cursor timing and control to the Video Output block.
- Vertical and horizontal sync signals.

The frequency of these signals may vary from about 60 Hz for vertical sync to well over 10 MHz for pixel information. Figure 4-82 shows the timing of some of these signals.

Timing Signals

The vertical scan rate is the measure of how often the entire video picture is drawn on the screen (usually 50 or 60 Hz). The screen is scanned horizontally many times during each vertical scan. If the video display is character-oriented, there might be 10 horizontal scans for each row of characters.

When set up properly, the timing outputs and video RAM address outputs will repeat regularly at the vertical scan rate. All the timing signals (such as the character clock, horizontal scan, blanking, vertical sync, blink rate, and cursor signal) are normally derived from the dot clock.

The cursor timing output is a strobe which occurs when the cursor address is sent out.

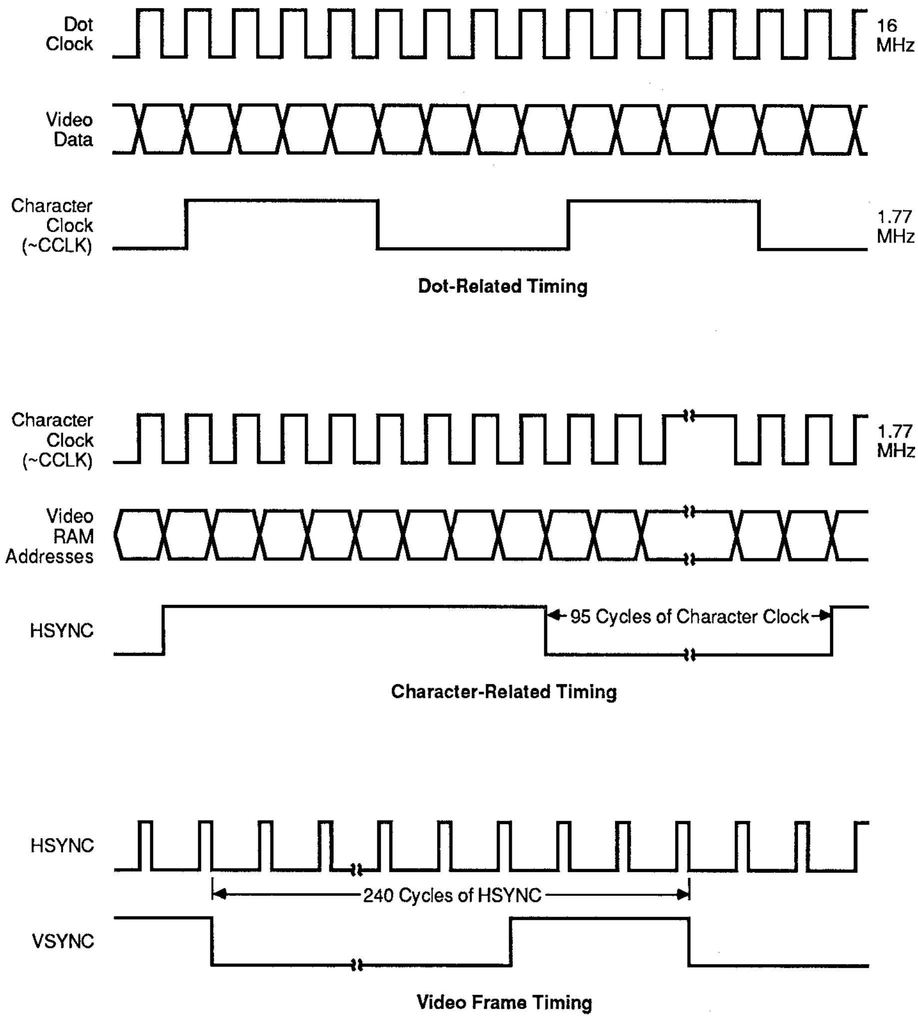


Figure 4-82: Video Display Controller Timing

Considerations for Testing and Troubleshooting 4.8.2.

Video control circuitry can usually be tested in four steps.

1. Initialize the circuitry (set up the video display controller registers if the implementation uses such a chip).

2. Test for proper signature on the scan address lines going to the video RAM to ensure that it cycles through the proper addresses when displaying a frame.

3. Check the vertical and horizontal sync frequency.

   If the timing logic is used in several modes, the three steps described above can be repeated for each mode.

4. Test the cursor strobe generator by clocking from the character clock, starting at the beginning of the frame and stopping at the end of the frame. You may need to test for proper signatures at several cursor positions. For this test approach to work, the cursor cannot be in a blinking mode.

The video RAM access logic, which allows the microprocessor and the video display controller to share video RAM, must arbitrate access to video RAM.

Since the microprocessor and the video display controller are not always synchronous, it may be impossible to find a single clock that gives stable signatures for all of the arbitration logic. One approach to testing the arbitration logic is to count pulses on the outputs of the video control logic while doing a series of writes to video RAM.

The Demo/Trainer UUT contains an example of a memory arbitration circuit which is hard to troubleshoot. It is a state machine with seven inputs and three outputs. In testing this type of circuit, you don't need to worry about how it works. All that is required is to exercise the inputs in a way that causes a stable response on each output. When this type of circuit does not function, it may be necessary to break some of the feedback loops to isolate the problem to one component. This can be done by using an I/O module to overdrive nodes in the feedback loops.

The character clock will probably be the best clock signal for most of the nodes, including scan address lines, video RAM, and circuitry up to the shift register which converts character information to pixel information. The response measurement should start at the end of the vertical retrace and should stop at the beginning of the vertical retrace. This means that the Start and Stop external control lines from the 9100A/9105A Clock Module or an I/O Module should connect to the vertical sync signal.

Video Control Circuit Example 4.8.3.

The Video Control Circuit of the Demo/Trainer UUT, Figure 4-83, uses a Signetics™ 2674 advanced video display controller (AVDC), U72, for video control. The 2674 is a programmable device designed for use with CRT terminals and display systems that employ raster-scan techniques. It is programmed with CRT-terminal setup information, providing cursor, blanking, and clock signals to the 2675 Attributes Controller chip (U78) in the Video Output functional block.

The 2674 outputs to the Video RAM functional block on the scan address lines DADD00-11 in synchronization with the horizontal and vertical sync signals.

The remaining circuitry in this block is a state machine. It is normally inactive, but upon writing to video RAM it produces a variable-length wait state to synchronize the microprocessor bus cycle to the video character clock.

Figure 4-83 shows a timing diagram for the video control circuit of the Demo/Trainer UUT.

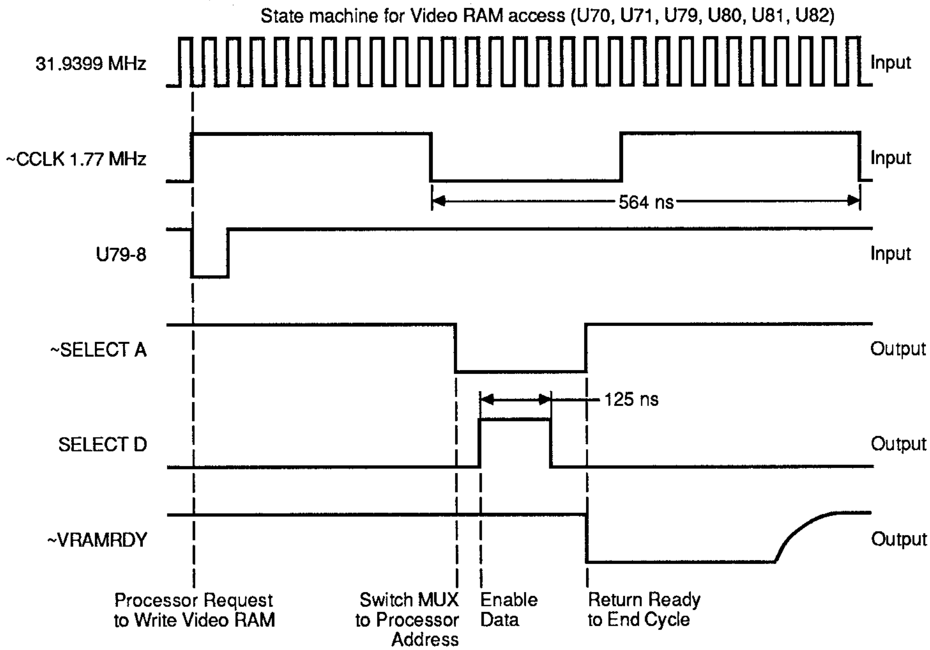


Figure 4-83: Video Control Functional Block Timing

Keystroke Functional Test 4.8.4.

Part A:

1. Clip a 40-pin clip module on I/O module 1 to test U72.

2. Use the the EXEC, I/O MOD, and SOFT KEYS keys with the following commands and check the measured frequency with the correct frequency ranges shown in the response table of Figure 4-84.

```
EXECUTE UUT DEMO PROGRAM VIDEO_INIT
FREQ ON I/O MOD 1 PIN <see response table>
```

Part B:

1. Connect the external control lines of the I/O module 1 as follows:

   Clock to CCLK (U78-33)
   Start to VSYNC (U88-13)
   Stop to VSYNC (U88-13)
   Enable to BLANK (U78-12)

2. Use the EXEC, SYNC, and I/O MOD keys with the following commands, and check the measurements with the response table in Figure 4-85.

```
EXECUTE UUT DEMO PROGRAM VIDEO_INIT
SYNC I/O MOD 1 TO EXT ENABLE LOW ...
... CLOCK ↓ START ↓ STOP ↑
ARM  I/O MOD 1 FOR CAPTURE USING SYNC
SHOW I/O MOD 1 PIN <see response table> ...
... CAPTURED RESPONSES
```

NOTE

The SHOW command requires a clip module pin number rather than a part pin number. This requires you to translate part pin numbers to clip module pin numbers (see Appendix B of the Technical User's Manual). For your convenience, this translation has been done for you in this example, and the results are shown in the "I/O MOD PIN" column of the response table in the Figure 4-85.

Part C:

Use the SYNC, PROBE, and WRITE keys with the probe to test the video ready signals. Compare the results with the response table in Figure 4-86.

```
SYNC PROBE TO POD DATA
ARM PROBE FOR CAPTURE USING SYNC
WRITE BLOCK INTO MEMORY FROM UUT DEMO ...
... FILE VID_FILL1 USING MOTOROLA
... (ADDR OPTION: MEMORY WORD)
SHOW PROBE CAPTURED RESPONSES
```

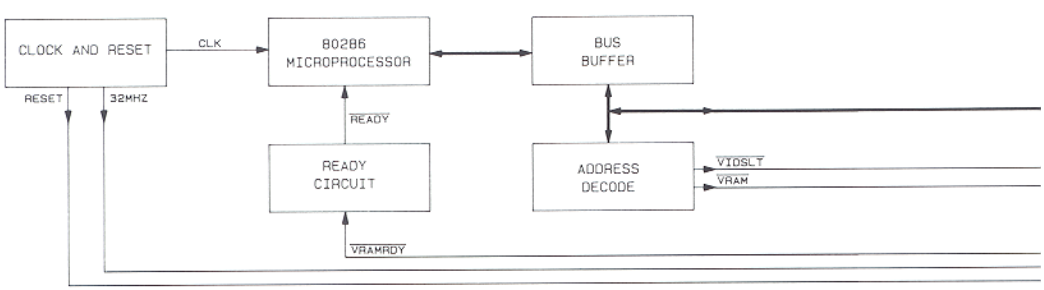
Keystroke Functional Test (Part A)

CONNECTION TABLE

STIMULUS	MEASUREMENT
(NONE)	I/O MOD U72

RESPONSE TABLE

SIGNAL	PIN	I/O MOD PIN	FREQUENCY MINIMUM	FREQUENCY MAXIMUM
CCLK	U72-16	16	1.775 MHZ	1.780 MHZ
BLANK	-17	17	14426 HZ	14435 HZ
VSYNC	-18	18	58 HZ	62 HZ
HSYNC	-19	19	16766 HZ	16774 HZ



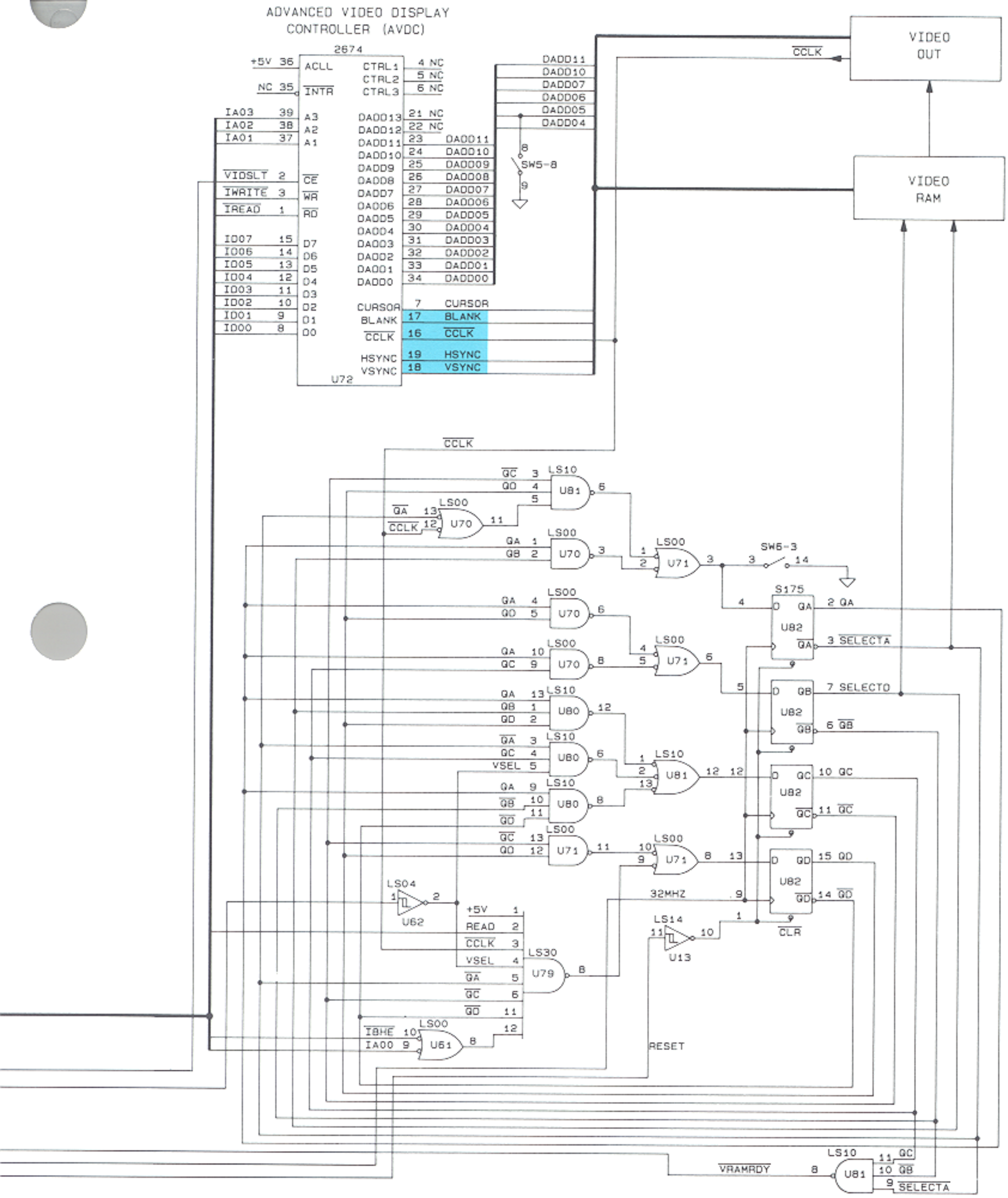


Figure 4-84: Video Control Functional Test (Part A)

Keystroke Functional Test (Part B)

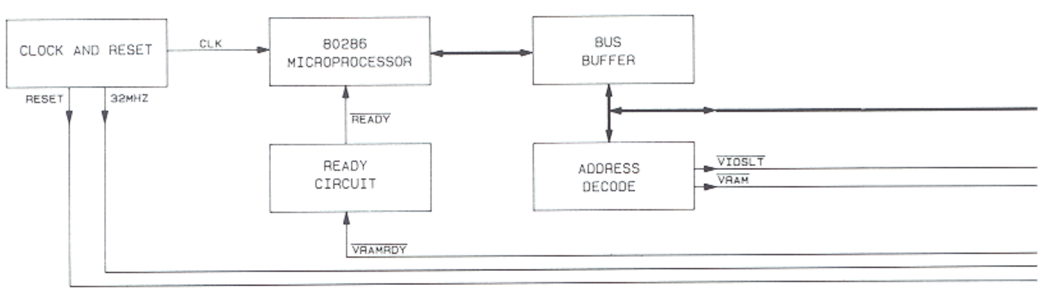
CONNECTION TABLE

STIMULUS	MEASUREMENT CONTROL		MEASUREMENT
(NONE)	I/O MOD		I/O MOD
	CLOCK	U78-33	U72
	START	U88-13	
	STOP	U88-13	
	ENABLE	U78-12	

RESPONSE TABLE

SIGNAL	PART PIN	I/O MOD PIN	SIGNATURE
DAD00	U72-34	34	4155
DAD01	-33	33	3F33
DAD02	-32	32	A65A
DAD03	-31	31	9024
DAD04	-30	30	DE6D
DAD05	-29	29	D6FA
DAD06	-28	28	7AC3
DAD07	-27	27	0477
DAD08	-26	26	E941
DAD09	-25	25	88B8
DAD10	-24	24	60B0
DAD11	-23	23	D869 or 0000*

*DAD11 has a signature of D869 one half of the time and 0000 the other half of the time.



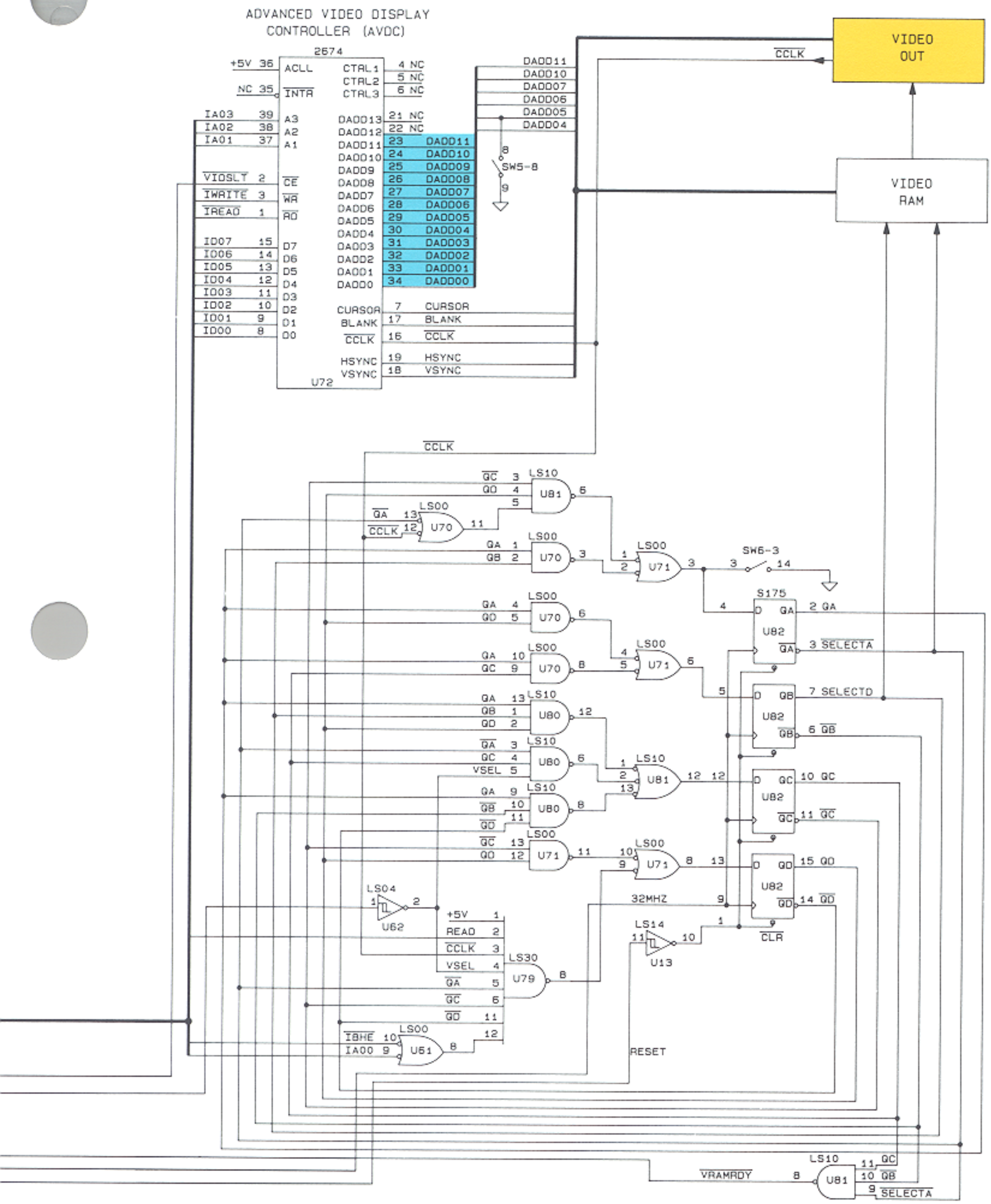


Figure 4-85: Video Control Functional Test (Part B)

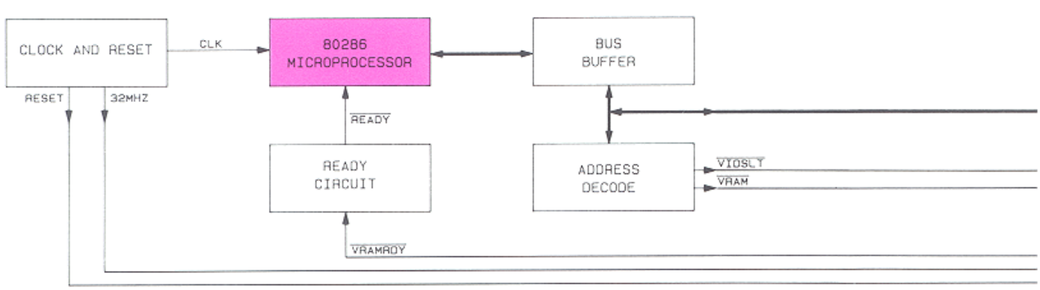
Keystroke Functional Test (Part C)

CONNECTION TABLE

STIMULUS	MEASUREMENT
POD	PROBE U82-3 U81-8

RESPONSE TABLE

SIGNAL	PART PIN	SIGNATURE	TRANS COUNT
SELECTA	U82-3	7A70	2048
VRAMRDY	U81-8	0000	2048



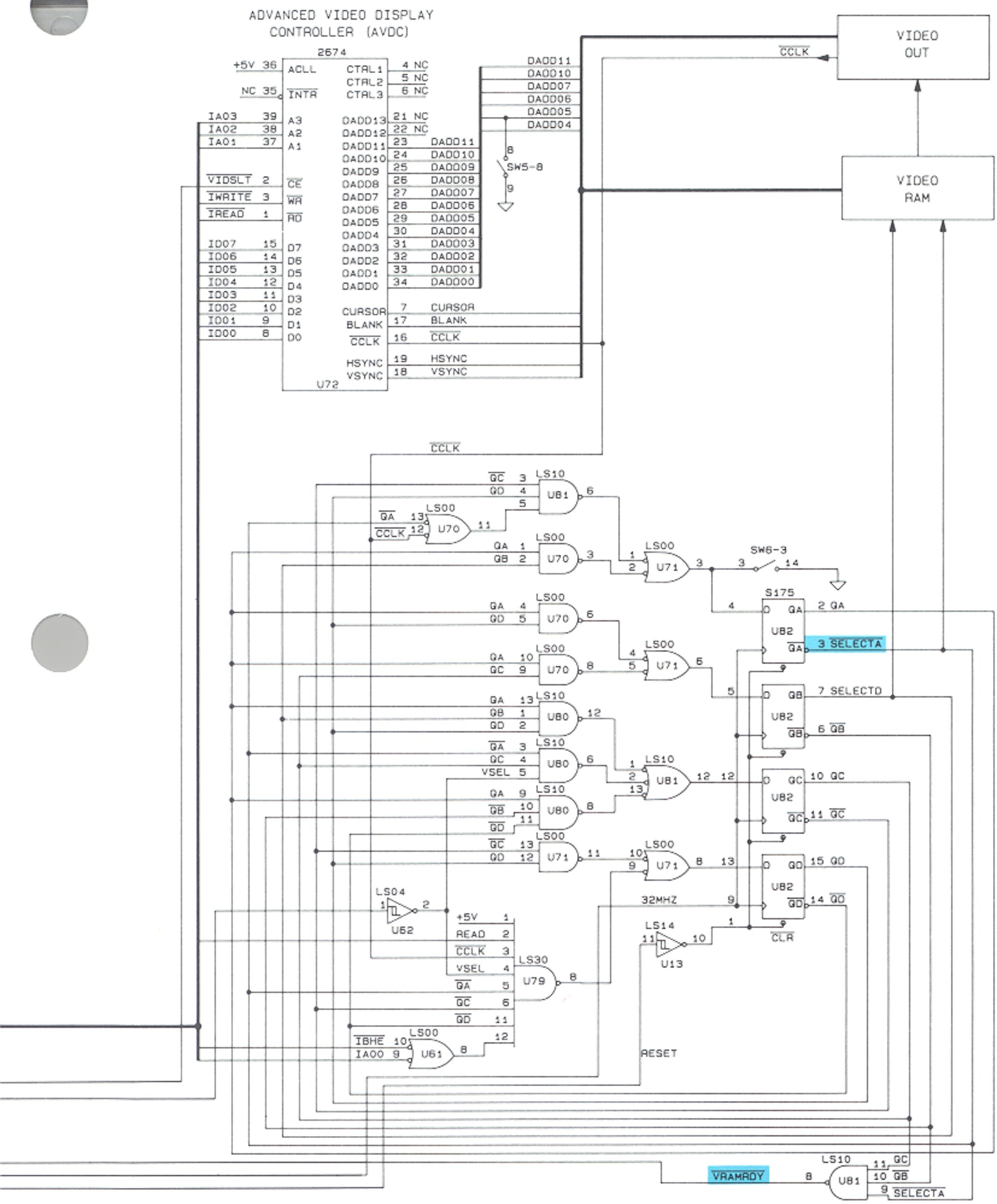


Figure 4-86: Video Control Functional Test (Part C)

Programmed Functional Test 4.8.5.

The *tst_vidctl* program is the programmed functional test for the Video Control functional block. This program checks the video controller IC (U72) and the video RAM ready generator outputs U81-8 and U82-3 using the *gfi test* command. If the *gfi test* command fails, the *abort_test* program is executed and GFI troubleshooting begins. (See the Bus Buffer functional block for a discussion of the *abort_test* program).

```
program tst_vidctl

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! FUNCTIONAL TEST of the VIDEO CONTROL functional block.               !
!                                                                      !
! This program tests the VIDEO CONTROL functional block of the         !
! Demo/Trainer.  The gfi test command and I/O module are used to       !
! perform the test.                                                    !
!                                                                      !
! TEST PROGRAMS CALLED:                                                !
!    abort_test (ref-pin)               If gfi has an accusation       !
!                                       display the accusation else    !
!                                       create a gfi hint for the      !
!                                       ref-pin and terminate the test!
!                                       program (GFI begins trouble-   !
!                                       shooting).                     !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

! Setup

   print "\nl\nlTESTING VIDEO CONTROL Circuit"

! Main part of test

   podsetup 'enable ~ready' "on"

   if gfi test "U72-34" fails then abort_test("U72-34")
   if gfi test "U81-8" fails then abort_test("U81-8")
   if gfi test "U82-3" fails then abort_test("U82-3")

   print "VIDEO CONTROL TEST PASSES"
end program
```

Stimulus Programs and Responses 4.8.6.

Figure 4-87 is the stimulus program planning diagram for the Video Control functional block. The *video_data* stimulus program outputs data onto the data bus. The *video_freq* stimulus program initializes the video registers and then measures frequency. The *video_scan* stimulus program initializes video RAM by executing *video_fill*, which fills video

RAM with characters including blinking characters. The *reset_low* stimulus program prompts the test operator to push the Demo/Trainer UUT RESET pushbutton and measures the level of the reset signal. The *levels* stimulus program stimulates activity appropriate for measuring static levels on a number of nodes in the Video RAM Ready (VRAMRDY) generation circuit. The *video_rdy* stimulus program stimulates the Video RAM Ready (VRAMRDY) generation circuit by writes made to the write-only video RAM.

All the stimulus programs execute *video_init* before any measurements are made on the video circuitry.

Stimulus Program Planning

PROGRAM: RESET_LOW

PROMPTS THE OPERATOR TO PRESS THE RESET KEY AND THEN CHECKS FOR A LOW LEVEL

MEASUREMENT AT:

U13-10

PROGRAM: VIDEO_FREQ

EXECUTES VIDEO_INIT AND MEASURES FREQUENCY

MEASUREMENT AT:

U72-17,19,18
U70-11

PROGRAM: VIDEO_SCAN

EXECUTES VIDEO_INIT, VIDEO_FIL1, AND MEASURES ALL CIRCUITRY WHERE DATA IS CLOCKED THROUGH BY CHARACTERS

MEASUREMENT AT:

U72-34,33,32,31,30,29,28,27,26,25,24,23,7

INITIALIZATION PROGRAM: VIDEO_INIT

INITIALIZES VIDEO REGISTERS TO STANDARD OPERATING MODE

MEASUREMENT AT:

(NONE)

PROGRAM: VIDEO_DATA

EXECUTES VIDEO_INIT AND READS DATA FROM VIDEO CONTROLLER REGISTERS

MEASUREMENT AT:

U72-8,9,10,11,12,13,14,15

PROGRAM: VIDEO_RDY

EXERCISES VIDEO RAM READY CIRCUITRY

MEASUREMENT AT:

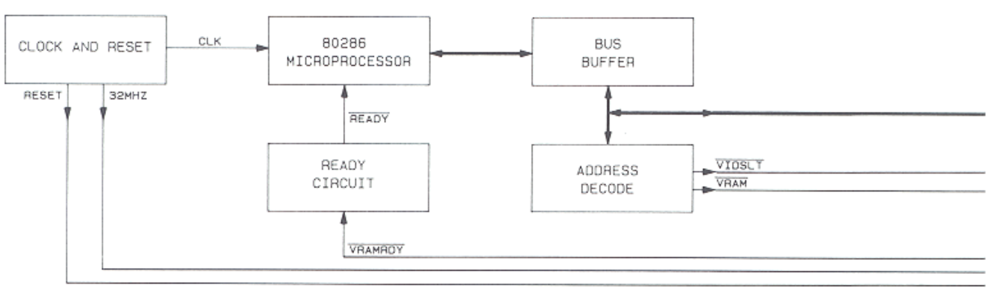
U70-11,3,6,8	U62-2	U82-2,3,7,6
U81-6,12,8	U61-8	U82-10,11,15,14
U71-3,6,11,8	U79-8	
U80-12,6,8		

PROGRAM: LEVELS

MEASURES STATIC LEVELS

MEASUREMENT AT:

U70-11,3,6,8	U62-2	U82-2,3,7,6
U81-6,12,8	U61-8	U82-10,11,15,14
U71-3,6,11,8	U79-8	
U80-12,6,8		



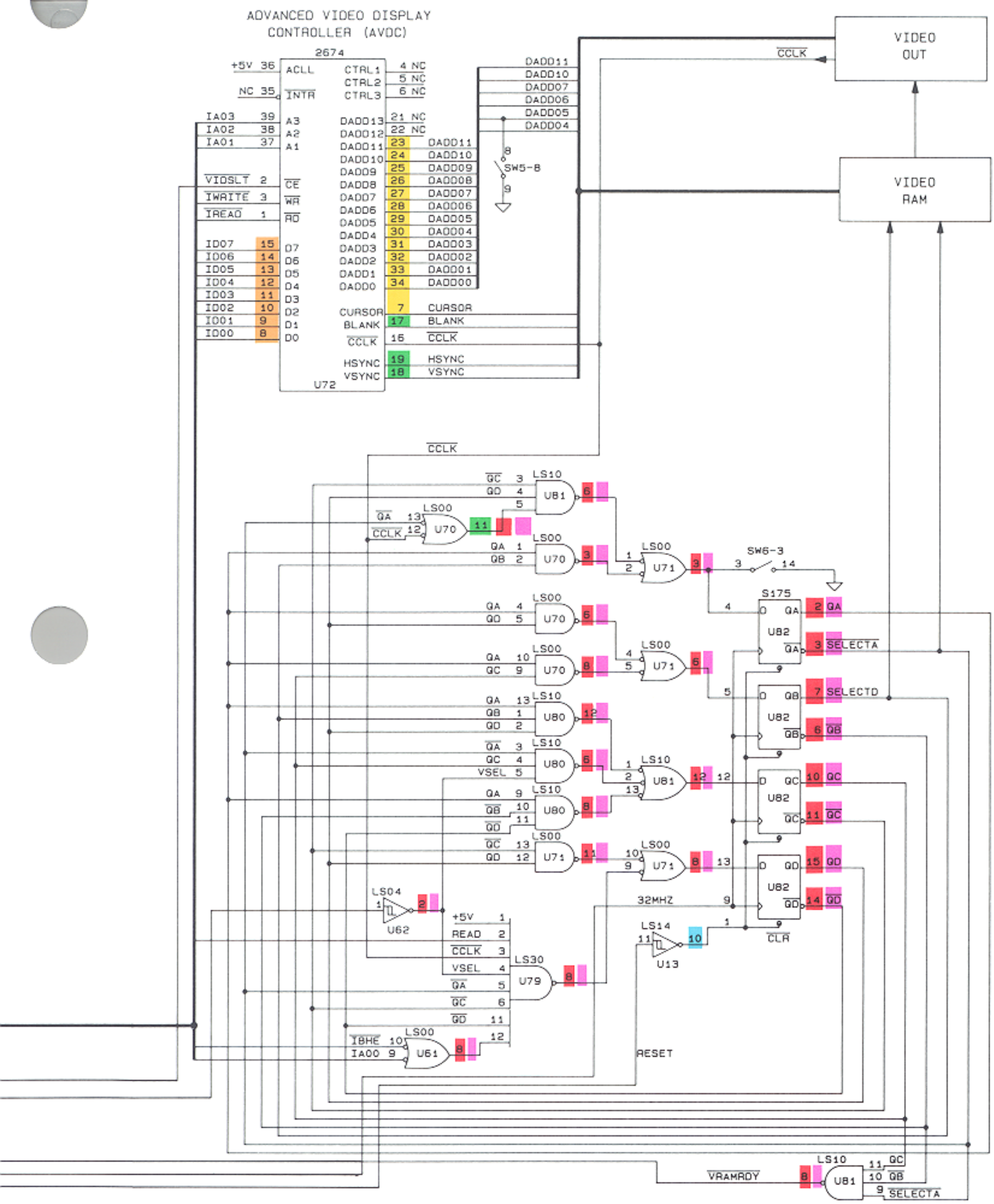


Figure 4-87: Video Control Stimulus Program Planning

```
program video_data

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! STIMULUS PROGRAM to extract data from U72 registers.                    !
!                                                                         !
! Stimulus programs and response files are used by GFI to backtrace      !
! from a failing node.  The stimulus program must create repeatable UUT  !
! activity and the response file contains the known-good responses for   !
! the outputs in the UUT that are stimulated by the stimulus program.    !
!                                                                         !
! This stimulus program is one of the programs which creates activity    !
! in the kernel area of the UUT.  These programs create activity with    !
! or without the ready circuit working properly.  Because of this, all  !
! the stimulus programs in the kernel area must disable the READY input  !
! to the pod, then perform the stimulus, then re-enable the READY input  !
! to the pod.  The 80286 microprocessor has a separate bus controller;   !
! for this reason, disabling ready and performing stimulus can get the   !
! bus controller out of synchronization with the pod.  Two fault         !
! handlers trap pod timeout conditions that indicate the bus controller  !
! is out of synchronization.  The recover() program is executed to       !
! resynchronize the bus controller and the pod.                          !
!                                                                         !
! TEST PROGRAMS CALLED:                                                   !
!    recover    ()                           The 80286 microprocessor has a!
!                                            bus controller that is totaly !
!                                            separate from the pod.  In    !
!                                            some cases the pod can get out!
!                                            of sync with the bus control- !
!                                            ler.  The recover program     !
!                                            resynchronizes the pod and the!
!                                            bus controller.               !
!                                                                         !
! GRAPHICS PROGRAMS CALLED:                                               !
!    (none)                                                               !
!                                                                         !
! Local Variables Modified:                                               !
!    recover_times                           Reset to Zero                 !
!    devname                                 Measurement device            !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Main Declarations                                                       !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

declare global numeric recover_times

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!   FAULT HANDLERS:                                                       !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

handle pod_timeout_enabled_line
   recover()
end handle
handle pod_timeout_recovered
   recover()
end handle
```

(continued on the next page)

Figure 4-88: Stimulus Program *(video_data)*

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!   Main part of STIMULUS PROGRAM                                      !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

recover_times = 0

! Let GFI determine measurement device.

    if (gfi control) = "yes" then
        devname = gfi device
    else
        devname = "/mod1"
    end if
   print "Stimulus Program VIDEO_DATA"

! Set addressing mode and initialize.

    option = getspace type "i/o", size "byte"
    setspace( option )
    write ADDR 8, DATA $FF      ! Cursor Address Lower
    write ADDR $A, DATA  0      ! Cursor Address Upper
    write ADDR $C, DATA $AA     ! Screen Start 2 Lower
    write ADDR $E, DATA $35     ! Screen Start 2 Upper

! Setup measurement device.

    reset device devname
    sync device devname, mode "pod"
    sync device "/pod", mode "data"

! Present stimulus to UUT.

    arm device devname          ! Start response capture
       read addr  8             !    Lower Cursor Addr Reg
       read addr  $A            !    Upper Cursor Addr Reg
       read addr  $C            !    Lower Screen Start 2
       read addr  $E            !    Upper Screen Start 2
    readout device devname      ! End response capture

end program
```

Figure 4-88: Stimulus Program *(video_data) - continued*

STIMULUS PROGRAM NAME: VIDEO_DATA
DESCRIPTION: SIZE: 318 BYTES

Node Signal Src	Learned With	SIG	Async LVL	Clk LVL	Counter Mode	Counter Range	Priority Pin
		------------------- Response Data	--------------------				
U72-8	PROBE	0009	1	0	TRANS		
U72-8	I/O MODULE	0009	1	0	TRANS		
U72-9	PROBE	000A	1	0	TRANS		
U72-9	I/O MODULE	000A	1	0	TRANS		
U72-10	PROBE	0009	1	0	TRANS		
U72-10	I/O MODULE	0009	1	0	TRANS		
U72-11	PROBE	000A	1	0	TRANS		
U72-11	I/O MODULE	000A	1	0	TRANS		
U72-12	PROBE	0009	1	0	TRANS		
U72-12	I/O MODULE	0009	1	0	TRANS		
U72-13	PROBE	000B	1	0	TRANS		
U72-13	I/O MODULE	000B	1	0	TRANS		
U72-14	PROBE	0008	1	0	TRANS		
U72-14	I/O MODULE	0008	1	0	TRANS		
U72-15	PROBE	000A	1	0	TRANS		
U72-15	I/O MODULE	000A	1	0	TRANS		

Figure 4-89: Response File *(video_data)*

```
program video_rdy

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! STIMULUS PROGRAM activates video ready circuitry.                    !
!                                                                      !
! Stimulus programs and response files are used by GFI to backtrace    !
! from a failing node.  The stimulus program must create repeatable UUT !
! activity and the response file contains the known-good responses for  !
! the outputs in the UUT that are stimulated by the stimulus program.   !
!                                                                      !
! TEST PROGRAMS CALLED:                                                !
!    (none)                                                            !
!                                                                      !
! GRAPHICS PROGRAMS CALLED:                                            !
!    (none)                                                            !
!                                                                      !
! Local Variables Modified:                                            !
!    devname                             Measurement device            !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!   FAULT HANDLERS:                                                    !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

handle pod_timeout_enabled_line
   recover()
end handle
handle pod_timeout_recovered
   recover()
end handle

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!   Main part of STIMULUS PROGRAM                                      !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

recover_times = 0

! Let GFI determine measurement device.

    if (gfi control) = "yes" then
        devname = gfi device
    else
        devname = "/probe"
    end if
    print "Stimulus Program VIDEO_RDY"

! Set addressing mode and Set up measurement device.

    reset device devname
    setspace space (getspace space "memory", size "word")
    sync device devname, mode "pod"
    sync device "/pod", mode "data"

! Present stimulus to UUT

    arm device devname                          ! Start response capture.
      toggledata addr $20000, data 0, mask $FFFF ! Create a burst of writes.
    readout device devname                      ! End response capture.

end program
```

Figure 4-90: Stimulus Program *(video_rdy)*

STIMULUS PROGRAM NAME: VIDEO_RDY
DESCRIPTION: SIZE: 1,411 BYTES

Node	Learned		Response Data				Priority
			Async	Clk	Counter		
Signal Src	With	SIG	LVL	LVL	Mode	Counter Range	Pin
U82-2	PROBE	0000	1 0		TRANS		
U82-2	I/O MODULE	0000	1 0		TRANS		
U82-3	PROBE	3951	1 0		TRANS		
U82-3	I/O MODULE	3951	1 0		TRANS		
U82-7	PROBE	0000	1 0		TRANS		
U82-7	I/O MODULE	0000	1 0		TRANS		
U82-6	PROBE	3951	1 0		TRANS		
U82-10	PROBE	3951	1 0		TRANS		
U82-10	I/O MODULE	3951	1 0		TRANS		
U82-11	PROBE	0000	1 0		TRANS		
U82-11	I/O MODULE	0000	1 0		TRANS		
U82-15	PROBE	0000	1 0		TRANS		
U82-15	I/O MODULE	0000	1 0		TRANS		
U82-14	PROBE	3951	1 0		TRANS		
U82-14	I/O MODULE	3951	1 0		TRANS		
U81-6	PROBE	3951	1 0		TRANS		
U81-6	I/O MODULE	3951	1 0		TRANS		
U81-8	PROBE	0000	1 0		TRANS		
U81-8	I/O MODULE	0000	1 0		TRANS		
U81-12	PROBE	3951	1 0		TRANS		
U80-6	PROBE	0000	1 0		TRANS		
U80-8	PROBE	3951	1 0		TRANS		
U80-12	PROBE	3951	1 0		TRANS		
U79-8	I/O MODULE	3951	1 0	1	TRANS		
U71-3	PROBE	0000	1 0		TRANS		
U71-3	I/O MODULE	0000	1 0		TRANS		
U71-6	PROBE	0000	1 0		TRANS		
U71-6	I/O MODULE	0000	1 0		TRANS		
U71-8	PROBE	0000	1 0		TRANS		
U71-8	I/O MODULE	0000	1 0		TRANS		
U71-11	I/O MODULE	3951	1 0		TRANS		
U70-3	I/O MODULE	3951	1 0		TRANS		
U70-6	I/O MODULE	3951	1 0		TRANS		
U70-8	I/O MODULE	3951	1 0		TRANS		
U70-11	PROBE		1 0		TRANS		
U70-11	I/O MODULE		1 0		TRANS		
U62-2	PROBE	3951	1 0		TRANS		
U62-2	I/O MODULE	3951	1 0		TRANS		
U62-6	I/O MODULE	0000	1 0		TRANS		
U62-10	I/O MODULE	3951	1		TRANS		
U62-12	I/O MODULE	3951	1		TRANS		
U61-6	I/O MODULE	3951	1 0		TRANS		
U61-3	I/O MODULE	3951	1 0		TRANS		
U61-8	I/O MODULE	3951	1		TRANS		

(continued on the next page)

Figure 4-91: Response File *(video_rdy)*

U84-4	I/O MODULE	1 0	TRANS	8300-9500
U84-7	I/O MODULE	1 0	TRANS	14000-17500
U84-9	I/O MODULE	1 0	TRANS	30000-36000
U84-12	I/O MODULE	1 0	TRANS	61000-71000
U83-4	I/O MODULE	1 0	TRANS	950-1300
U83-7	I/O MODULE	1 0	TRANS	1400-1800
U83-9	I/O MODULE	1 0	TRANS	2300-2700
U83-12	I/O MODULE	1 0	TRANS	4100-4700
U73-7	I/O MODULE	1 0	TRANS	475-800
U73-9	I/O MODULE	1 0	TRANS	500-900
U73-12	I/O MODULE	1 0	TRANS	700-1000
U69-18	I/O MODULE	1 0	TRANS	
U69-16	I/O MODULE	1 0	TRANS	
U69-14	I/O MODULE	1 0	TRANS	
U69-12	I/O MODULE	1 0	TRANS	
U69-9	I/O MODULE	1 0	TRANS	
U69-7	I/O MODULE	1 0	TRANS	
U69-5	I/O MODULE	1 0	TRANS	
U69-3	I/O MODULE	1 0	TRANS	
U68-18	I/O MODULE	1 0	TRANS	
U68-16	I/O MODULE	1 0	TRANS	
U68-14	I/O MODULE	1 0	TRANS	
U68-12	I/O MODULE	1 0	TRANS	
U68-9	I/O MODULE	1 0	TRANS	
U68-7	I/O MODULE	1 0	TRANS	
U68-5	I/O MODULE	1 0	TRANS	
U68-3	I/O MODULE	1 0	TRANS	

Figure 4-91: Response File *(video_rdy) - continued*

```
program levels

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! STIMULUS PROGRAM to measure level history.                            !
!                                                                       !
! Stimulus programs and response files are used by GFI to backtrace     !
! from a failing node.  The stimulus program must create repeatable UUT !
! activity and the response file contains the known-good responses for  !
! the outputs in the UUT that are stimulated by the stimulus program.   !
!                                                                       !
! This is a general purpose routine that is used to characterize the    !
! level history both sync and async of a node that may not lend itself  !
! to signatures or frequency.                                           !
!                                                                       !
! TEST PROGRAMS CALLED:                                                 !
!    (none)                                                             !
!                                                                       !
! GRAPHICS PROGRAMS CALLED:                                             !
!    (none)                                                             !
!                                                                       !
! Local Variables Modified:                                             !
!    devname                             Measurement device             !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!   FAULT HANDLERS:                                                     !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

handle pod_timeout_no_clk
end handle

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!   Main part of STIMULUS PROGRAM                                       !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

! Let GFI user select which I/O module to use.

   if (gfi control) = "yes" then
     devname = gfi device
   else
     devname = "/mod1"
   end if
   print "Stimulus Program LEVELS"

! Set desired measurement modes.

   reset device devname

! No stimulus is applied; response is async levels.

   arm device devname        ! Start response capture.
   readout device devname    ! End response capture

end levels
```

Figure 4-92: Stimulus Program *(levels)*

STIMULUS PROGRAM NAME: LEVELS
DESCRIPTION: SIZE: 1,435 BYTES

Node Signal Src	Learned With	SIG	Async LVL	Clk LVL	Counter Mode	Counter Range	Priority Pin
			---- Response Data ----				
U82-2	PROBE		0		TRANS		
U82-2	I/O MODULE		0		TRANS		
U82-3	PROBE		1		TRANS		
U82-3	I/O MODULE		1		TRANS		
U82-7	PROBE		0		TRANS		
U82-7	I/O MODULE		0		TRANS		
U82-6	PROBE		1		TRANS		
U82-10	PROBE		0		TRANS		
U82-10	I/O MODULE		0		TRANS		
U82-11	PROBE		1		TRANS		
U82-11	I/O MODULE		1		TRANS		
U82-15	PROBE		0		TRANS		
U82-15	I/O MODULE		0		TRANS		
U82-14	PROBE		1		TRANS		
U82-14	I/O MODULE		1		TRANS		
U81-6	PROBE		1		TRANS		
U81-6	I/O MODULE		1		TRANS		
U81-8	PROBE		1		TRANS		
U81-8	I/O MODULE		1		TRANS		
U81-12	PROBE		0		TRANS		
U80-6	PROBE		1		TRANS		
U80-8	PROBE		1		TRANS		
U80-12	PROBE		1		TRANS		
U79-8	I/O MODULE		1		TRANS		
U71-3	PROBE		0		TRANS		
U71-3	I/O MODULE		0		TRANS		
U71-6	PROBE		0		TRANS		
U71-6	I/O MODULE		0		TRANS		
U71-8	PROBE		0		TRANS		
U71-8	I/O MODULE		0		TRANS		
U71-11	I/O MODULE		1		TRANS		
U70-3	I/O MODULE		1		TRANS		
U70-6	I/O MODULE		1		TRANS		
U70-8	I/O MODULE		1		TRANS		
U70-11	PROBE		1 0		TRANS		
U70-11	I/O MODULE		1 0		TRANS		
U62-2	PROBE		0		TRANS		
U62-2	I/O MODULE		0		TRANS		
U61-8	I/O MODULE		1		TRANS		
U62-6	I/O MODULE		0		TRANS		
U61-3	I/O MODULE		1		TRANS		
U61-6	I/O MODULE		1		TRANS		
U84-4	I/O MODULE		1 0		TRANS		
U84-7	I/O MODULE		1 0		TRANS		

(continued on the next page)

Figure 4-93: Response File *(levels)*

U84-9	I/O MODULE	1 0	TRANS
U84-12	I/O MODULE	1 0	TRANS
U83-4	I/O MODULE	1 0	TRANS
U83-7	I/O MODULE	1 0	TRANS
U83-9	I/O MODULE	1 0	TRANS
U83-12	I/O MODULE	1 0	TRANS
U73-7	I/O MODULE	1 0	TRANS
U73-9	I/O MODULE	1 0	TRANS
U73-12	I/O MODULE	1 0	TRANS
U69-18	I/O MODULE	1 0	TRANS
U69-16	I/O MODULE	1 0	TRANS
U69-14	I/O MODULE	1 0	TRANS
U69-12	I/O MODULE	1 0	TRANS
U69-9	I/O MODULE	1 0	TRANS
U69-7	I/O MODULE	1 0	TRANS
U69-5	I/O MODULE	1 0	TRANS
U69-3	I/O MODULE	1 0	TRANS
U68-18	I/O MODULE	1 0	TRANS
U68-16	I/O MODULE	1 0	TRANS
U68-14	I/O MODULE	1 0	TRANS
U68-12	I/O MODULE	1 0	TRANS
U68-9	I/O MODULE	1 0	TRANS
U68-7	I/O MODULE	1 0	TRANS
U68-5	I/O MODULE	1 0	TRANS
U68-3	I/O MODULE	1 0	TRANS
J4-6	PROBE	0	TRANS
J4-6	I/O MODULE	0	TRANS
J4-10	PROBE	1	TRANS
J4-10	I/O MODULE	1	TRANS
R34-1	PROBE	1	TRANS
DS1-2	PROBE	1	TRANS
R26-1	PROBE	0	TRANS
R26-1	I/O MODULE	0	TRANS
R32-1	PROBE	1	TRANS
R4-1	PROBE	0	TRANS
R61-1	PROBE	X	TRANS
R77-1	PROBE	1	TRANS
R78-2	PROBE	1	TRANS
R79-2	PROBE	1	TRANS
R80-1	PROBE	1	TRANS
U26-3	I/O MODULE	X	TRANS
U13-4	PROBE	1	TRANS
U13-4	I/O MODULE	1	TRANS
U13-12	PROBE	0	TRANS
U13-12	I/O MODULE	0	TRANS
C13-1	PROBE	1X0	TRANS
C4-1	PROBE	0	TRANS
U14-65	PROBE	1	TRANS
U14-65	I/O MODULE	1	TRANS

Figure 4-93: Response File *(levels) - continued*

Summary of Complete Solution for Video Control 4.8.7.

The entire set of programs and files needed to test and GFI troubleshoot the Video Control functional block is shown below. The format below is similar to a 9100A/9105A UUT directory (you could consider the functional block to be a small UUT), but in addition shows the use of each program and the location in this manual for each file.

UUT DIRECTORY
(Complete File Set for Video Control)

Programs (PROGRAM):

TST_VCTRL	Functional Test	Section 4.8.5
RESET_LOW	Stimulus Program	Figure 4-115
VIDEO_DATA	Stimulus Program	Figure 4-88
VIDEO_FREQ	Stimulus Program	Figure 4-73
VIDEO_RDY	Stimulus Program	Figure 4-90
VIDEO_SCAN	Stimulus Program	Figure 4-77
LEVELS	Stimulus Program	Figure 4-92
VIDEO_INIT	Initialization Program	Figure 4-79

Stimulus Program Responses (RESPONSE):

RESET_LOW		Figure 4-116
VIDEO_DATA		Figure 4-89
VIDEO_FREQ		Figure 4-74
VIDEO_RDY		Figure 4-91
VIDEO_SCAN		Figure 4-78
LEVELS		Figure 4-93

Node List (NODE):

NODELIST		Appendix B

Text Files (TEXT):

Reference Designator List (REF):

REFLIST		Appendix A

Compiled Database (DATABASE):

GFIDATA		Compiled by the 9100A

(This page is intentionally blank.)

VIDEO RAM FUNCTIONAL BLOCK 4.9.

Introduction to Video RAM 4.9.1.

Video RAM blocks come in several forms. Here are some of the common configurations:

- Character-oriented video RAM, with secondary character-generation ROM or RAM.
- Pixel-oriented video RAM.
- Combinations of the above.

Access to video RAM can be provided in several ways, including:

- The video display controller may directly access microprocessor memory by stealing memory cycles.
- Video RAM may be separate but still mapped into microprocessor memory space. In this case, access to this memory may be write-only or read/write.
- Access to video RAM may be through I/O-mapped registers.
- If character-generation RAM is used, access to character RAM may be different than access to video RAM.

Considerations for Testing and Troubleshooting 4.9.2.

Testing of video display circuits is complicated by the fact that there may be as many as three separate hierarchical memory spaces, each of which may be sectioned for use only in a particular mode of operation:

- Video RAM
- Character ROM or RAM
- Color palette RAM

Video RAM

If video RAM has read/write access and is mapped into the microprocessor memory space, it can be tested with the 9100A/9105A's built-in RAM test (Section 4.4 discusses this built-in test). If video RAM does not have read access, the video RAM output must be tested with the I/O module or the probe. The 9100A/9105A external Start and Stop control lines should be connected (probably to vertical sync) so that one frame is captured. The 9100A/9105A external Clock control lines should be connected to the appropriate clock signal so that valid RAM output will be captured for each read cycle.

With the above connections, the following procedure will usually test video RAM:

1. Initialize the video circuitry, if not already initialized.
2. Initialize the video RAM with blinking enabled. The TL/1 *writeblock* and *writefill* commands can be used to do this.
3. Set the video control mode so that it accesses as much video RAM as possible.
4. Measure signatures at the video RAM output and compare them to good signatures.
5. Steps 2, 3, and 4 can be repeated, varying the test pattern loaded into video RAM. For example, with 16-bit-wide memory try test patterns like FFFF, 0000, 7777, and AAAA, or ramping data over the entire video RAM.

Character ROM or RAM

If the video RAM is character oriented, with secondary character ROM or RAM, a pattern can be written into the video RAM that cycles through the character-memory addresses. In the case of character ROM, signatures collected at the ROM outputs serve to test the ROM. In the case of character RAM, a pattern must be loaded into the RAM before testing.

Video RAM Circuit Example 4.9.3.

Figure 4-94 shows the Video RAM functional block for the Demo/Trainer UUT. Components U74 and U85 provide 2K bytes of static video RAM. When addressed over the main address bus (IA01-11), video RAM is used to store ASCII character codes supplied by the microprocessor over the main data bus (DB00-15). The system is character-mapped: a specific video RAM address maps into a physical location on the monitor screen.

The video control logic sequentially samples these addresses over lines DADD00-11 to generate display characters using the ASCII codes at these addresses and the corresponding display-character information in the character PROM (see U77 in the Video Output functional block).

The multiplexers U73, U83, and U84 select between the video control address lines (DAD00-11) and the buffered microprocessor lines (IA01-11). The selection control for this multiplexing comes from the Video Control functional block.

Keystroke Functional Test 4.9.4.

1. Connect the external control lines of I/O module 1 as follows:

   Clock to CCLK (U78-33)
   Start to VSYNC (U88-13)
   Stop to VSYNC (U88-13)
   Enable to BLANK (U78-12)

2. Use a 24-pin clip module on side A of I/O module 1 to test the video scan signal. Use the EXEC, SYNC, and I/O MOD keys to enter the following commands. Then, compare the measurements with the response tables in Figure 4-94.

```
EXECUTE UUT DEMO PROGRAM VIDEO_INIT
EXECUTE UUT DEMO PROGRAM VIDEO_FIL1
SYNC I/O MOD 1 TO EXT ENABLE LOW ...
... CLOCK ↓ START ↓ STOP ↑
ARM  I/O MOD 1 FOR CAPTURE USING SYNC
SHOW I/O MOD 1 PIN <see response table> ...
... CAPTURED RESPONSES
```

NOTE

The SHOW command requires a clip module pin number rather than a part pin number. This requires you to translate part pin numbers to clip module pin numbers (see Appendix B of the Technical User's Manual). For your convenience, this translation has been done for you in this example, and the results are shown in the "I/O MOD PIN" column of the response table in Figure 4-94.

(This page is intentionally blank.)

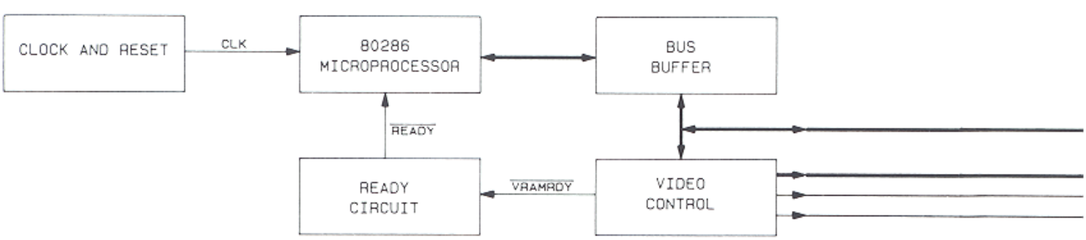
Keystroke Functional Test

CONNECTION TABLE

STIMULUS	MEASUREMENT CONTROL		MEASUREMENT
(NONE)	I/O MOD		I/O MOD
	CLOCK	U78-33	U74
	START	U88-13	U85
	STOP	U88-13	
	ENABLE	U78-12	

RESPONSE TABLE

SIGNAL	PART PIN	I/O MOD PIN	SIGNATURE
DB00	U74-9	9	4155
DB01	-10	10	3F33
DB02	-11	29	A65A
DB03	-13	31	9024
DB04	-14	32	DE6D
DB05	-15	11	D6FA
DB06	-16	12	7AC3
DB07	-17	13	0477
DB08	U85-9	9	A814
DB09	-10	10	C26B
DB10	-11	29	D909
DB11	-13	31	5FAA
DB12	-14	32	5925
DB13	-15	11	610D
DB14	-16	12	B8AB
DB15	-17	13	ADD3



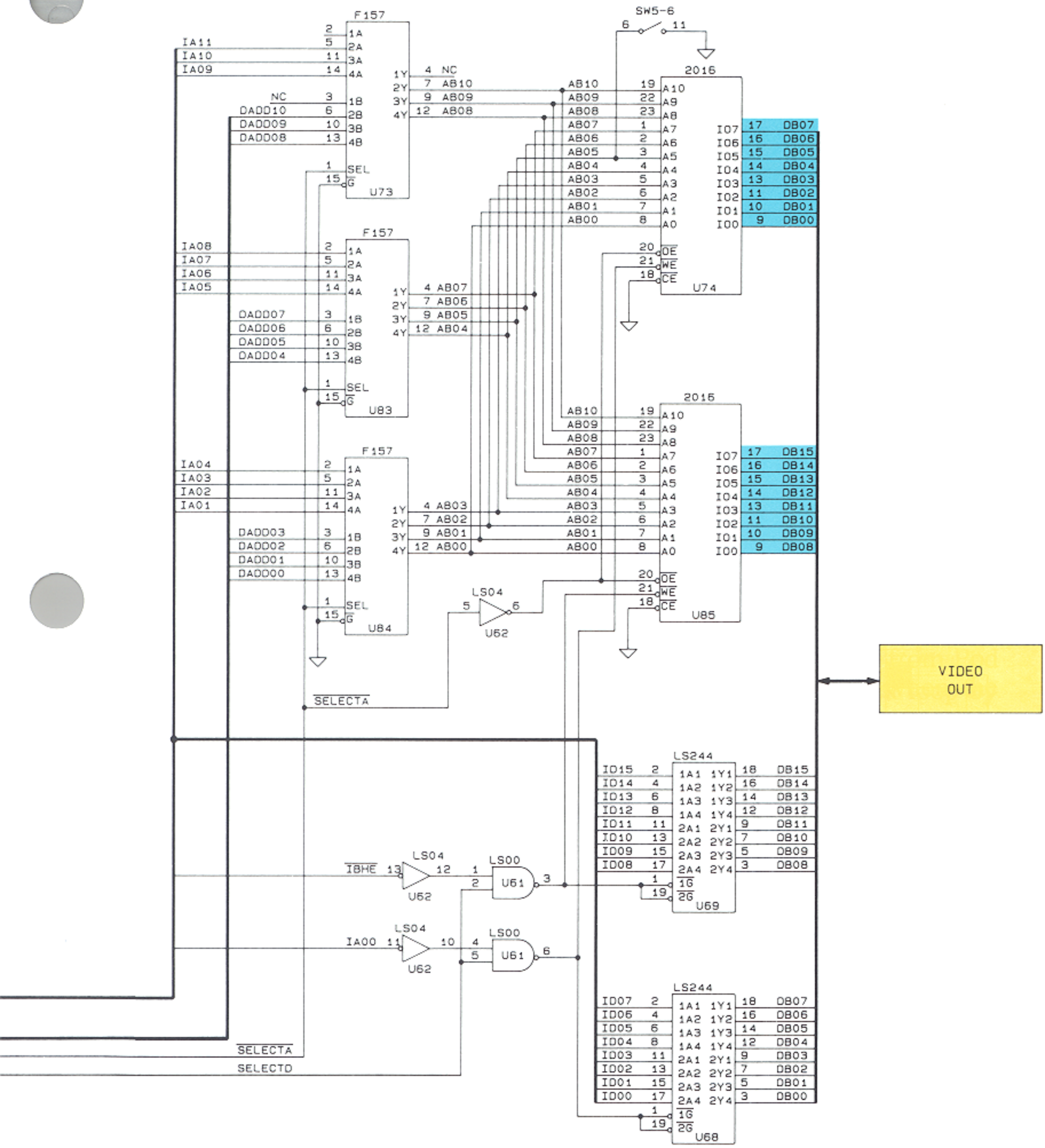


Figure 4-94: Video RAM Functional Test

Programmed Functional Test 4.9.5.

The *tst_vidram* program is the programmed functional test for the Video RAM functional block. This program checks the two RAM ICs U74 and U85 using the *gfi test* command. If the *gfi test* command fails, the *abort_test* program is executed and GFI troubleshooting begins. (See the Bus Buffer functional block for a discussion of the *abort_test* program).

```
program tst_vidram

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! FUNCTIONAL TEST of the VIDEO RAM functional block.                      !
!                                                                        !
! This program tests the VIDEO RAM functional block of the Demo/Trainer.!
! The gfi test command and I/O module are used to perform the test.      !
!                                                                        !
! TEST PROGRAMS CALLED:                                                  !
!    abort_test (ref-pin)                   If gfi has an accusation     !
!                                           display the accusation else  !
!                                           create a gfi hint for the    !
!                                           ref-pin and terminate the test!
!                                           program (GFI begins trouble- !
!                                           shooting).                   !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

! Setup

   print "\nl\nlTESTING VIDEO RAM Circuit"

! Main part of Test

   podsetup 'enable ~ready' "on"

   if gfi test "U74-9" fails then abort_test("U74-9")
   if gfi test "U85-9" fails then abort_test("U85-9")

   print "VIDEO RAM TEST PASSES"
end program
```

Stimulus Programs and Responses 4.9.6.

Figure 4-95 is the stimulus program planning diagram for the Video RAM functional block. The *video_scan* stimulus program initializes video RAM by executing *video_fill*, which fills video RAM with characters including blinking characters. The *levels* stimulus program provides the appropriate stimuli to measure the asynchronous level of various outputs. The *video_rdy* stimulus program stimulates the Video RAM Ready (VRAMRDY) generation circuit by writes made to the write-only video RAM.

All these stimulus programs (except *levels*) execute *video_init* before any measurements are made on the video circuitry.

Stimulus Program Planning

PROGRAM: LEVELS

MEASURES STATIC LEVELS

MEASUREMENT AT:

U62-6
U61-3,6
U84-12,9,7,4
U83-12,9,7,4
U73-12,9,7
U69-3,5,7,9,12,14,16,18
U68-3,5,7,9,12,14,16,18

PROGRAM: VIDEO_SCAN

EXECUTES VIDEO_INIT, VIDEO_FIL1, AND MEASURES ALL CIRCUITRY WHERE DATA IS CLOCKED THROUGH BY CHARACTERS

MEASUREMENT AT:

U84-12,9,7,4
U83-12,9,7,4
U73-12,9,7
U85-9,10,11,13,14,15,16,17
U74-9,10,11,13,14,15,16,17

INITIALIZATION PROGRAM: VIDEO_FIL1

INITIALIZES VIDEO RAM WITH BLINKING CHARACTERS

MEASUREMENT AT:

(NONE)

PROGRAM: VIDEO_RDY

EXERCISES THE VIDEO RAM DATA BUFFERS AND VIDEO RAM ADDRESS MULTIPLEXERS

MEASUREMENT AT:

U84-12,9,7,4
U83-12,9,7,4
U73-12,9,7
U62-6,12,10
U61-3,6
U68-3,5,7,9,12,14,16,18
U69-3,5,7,9,12,14,16,18

INITIALIZATION PROGRAM: VIDEO_INIT

INITIALIZES VIDEO REGISTERS TO STANDARD OPERATING MODE

MEASUREMENT AT:

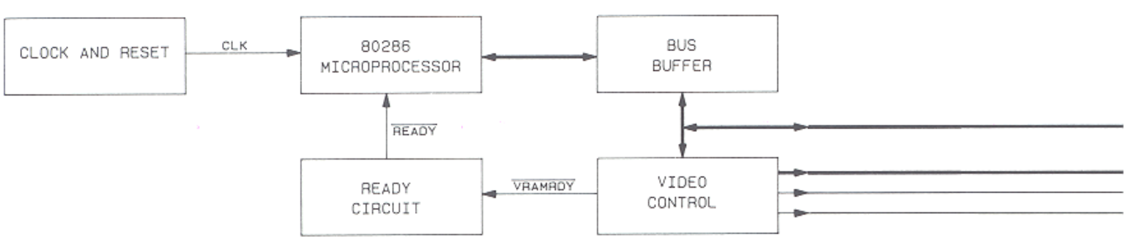
(NONE)

INITIALIZATION PROGRAM: VIDEO_FIL2

INITIALIZES VIDEO RAM WITHOUT BLINKING CHARACTERS

MEASUREMENT AT:

(NONE)



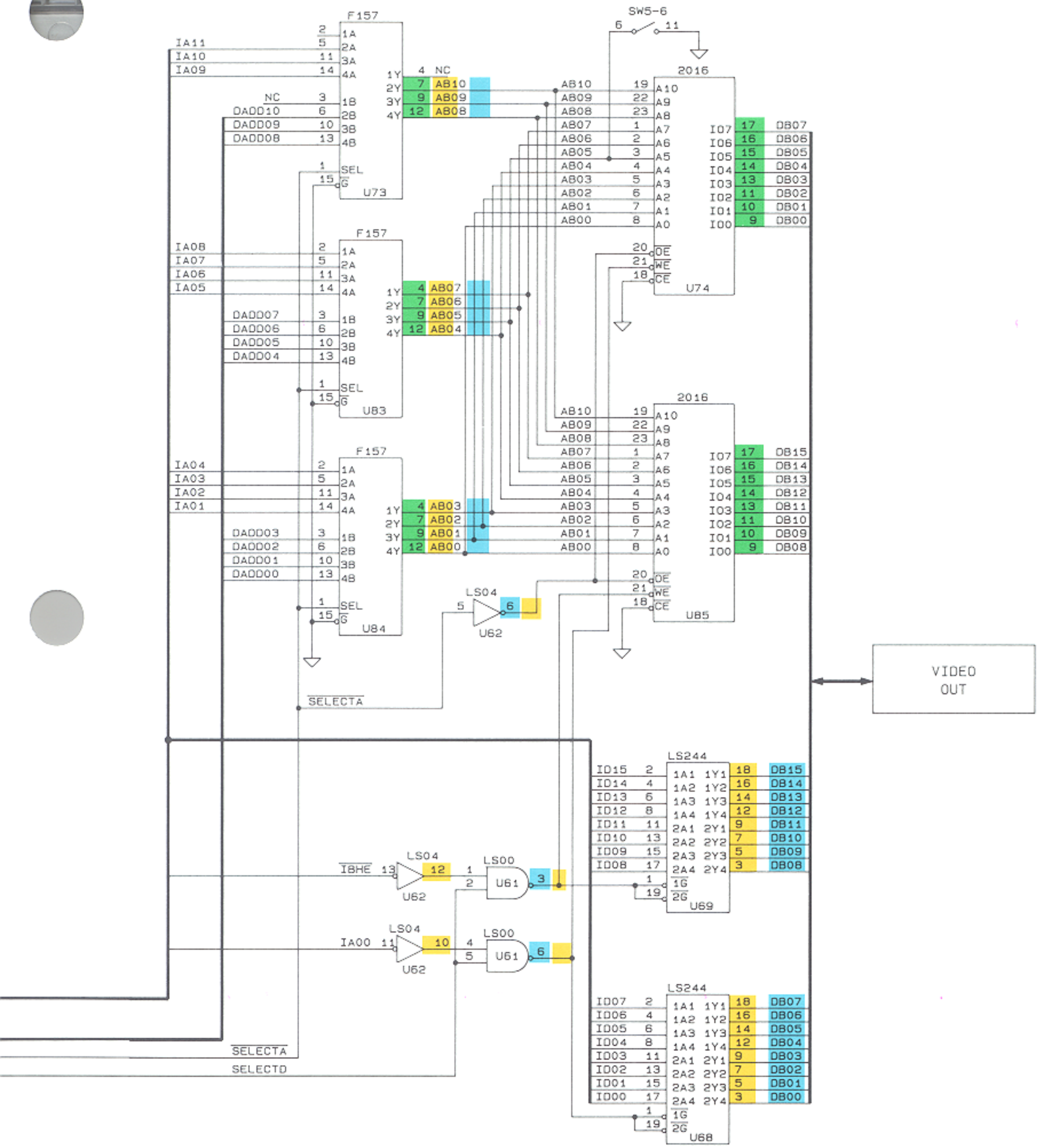


Figure 4-95: Video RAM Stimulus Program Planning