EB203

Test Bus Controller

SN74ACT8990

Author Peter Forstner

Date: 01.07.92

Rev.: 1.0

This report describes the IEEE 1149.1 Test Bus Controller SN74ACT8990 from Texas Instruments. The first part explains the architecture and operation of the Test Bus Controller; the second part uses examples to explain the programming procedure.

Contents:

1. Introduction

There has been a dramatic increase in the complexity of electronic systems, as a result of advances in the integration of semiconductors, the introduction of new packaging techniques (SMD), and the consequent use of double-sided circuit boards. However, increased component density on circuit boards brings with it new problems of testability, since the number of the necessary test vectors increases out of proportion with complexity. By making use of nail bed adapters, it is possible to partition the system to be tested, and so to reduce significantly the number of test vectors, although high SMD component density on double-sided boards reduces the number of possible contact areas for nail bed adapters. The escalating problems of testability can therefore only be solved with a completely new concept.

Back in 1985, leading electronic manufacturers founded the Joint Test Action Group (JTAG), in order to develop a new and cost-effective test concept. The result of this was the IEEE 1149.1 standard. This standard requires the use of special test-circuits at the inputs and outputs of selected semiconductor components, together with logic to control such test-circuits. A 4-wire serial test bus combines the test-circuits into a complete test-group, which is controlled via the test bus; in this way, with only 4 lines the complete system can be partitioned and tested.

The Texas Instruments(TI) Application Report EB193 describes these test methods in detail, and presents the IEEE 1149.1-compatible SCOPE™ bus drivers from TI. This Application Report EB203 assumes an understanding of test-methods according to IEEE 1149.1

The control of an IEEE 1149.1-compatible test system is usually performed by a computer. The TEST BUS CONTROLLER (TBC) SN74ACT8990 from Texas Instruments can be connected to a computer like a normal interface circuit, and it then controls completely the IEEE 1149.1 test bus.(Figure 1). The computer first configures the TBC, and then loads in parallel the test commands and test vectors. The TBC transfers these commands and vectors to the system, and thereby generates the signal sequence required by IEEE 1149.1. The processor can read the result in parallel from the TBC, after the test data has addressed the logic to be tested. This Application Report describes the operation of the TBC, and explains the programming procedure with examples.

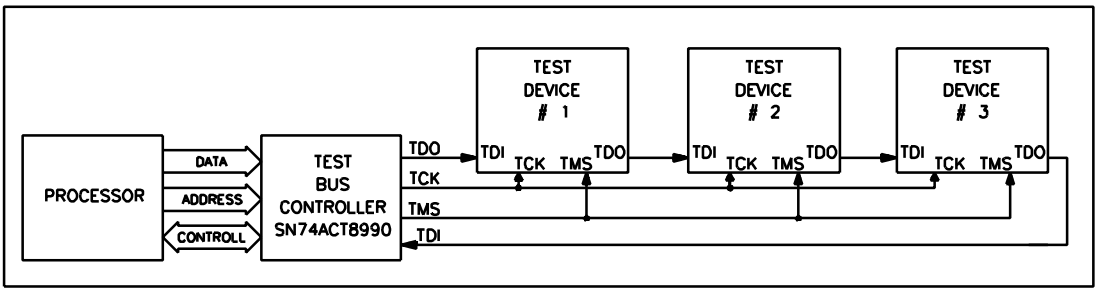


Figure 1: Test System with the SN74ACT8990 TEST BUS CONTROLLER

In order to reduce the applications work needed from the user to control IEEE 1149.1-compatible test systems, TI offers the computer program ASSET (ADVANCED SUPPORT SYSTEM FOR EMULATION AND TEST) together with a plug-in computer board. The TEST BUS CONTROLLER SN74ACT8990 is used on this board. ASSET allows an easy development of test programs; for this, the user needs no understanding of the function of the TEST BUS CONTROLLER. First, a library with the IEEE 1149.1-compatible circuits which are to be used must be assembled; a library with all standard TI components is supplied. Then, the complete system to be tested must be described with components from the library. After the system description, ASSET needs only the test vectors before testing can begin. This system is also ideally suited to supporting the circuit designer during the test phase.

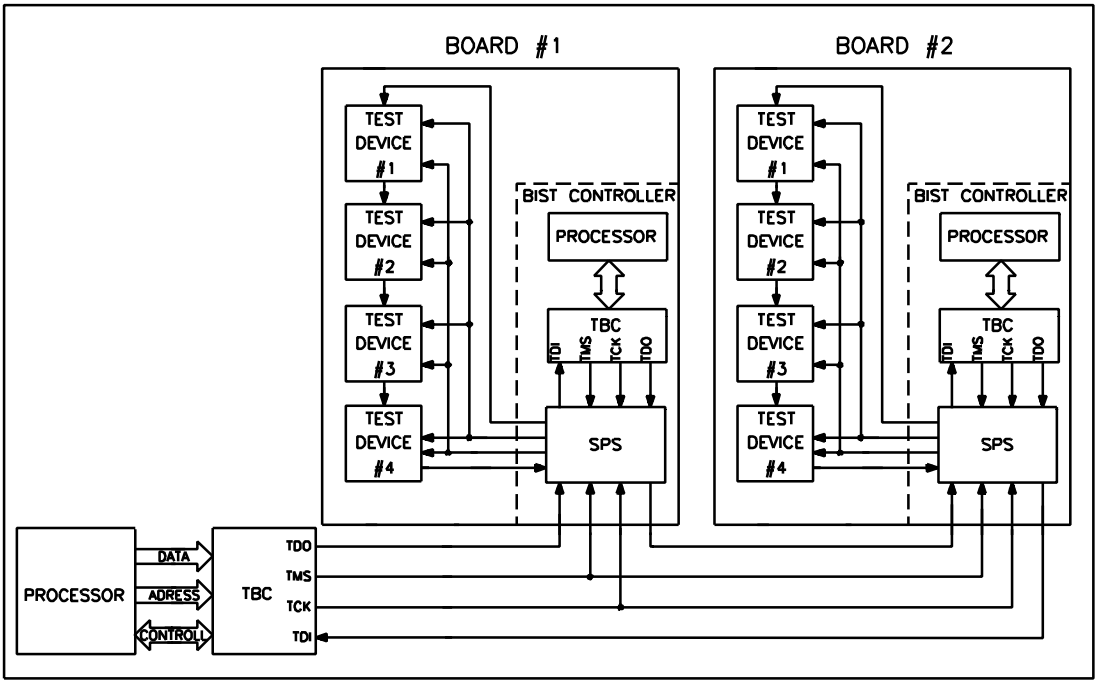


Figure 2: System with BIST (BUILT IN SELF TEST)

If BIST (BUILT IN SELF TEST) is to be incorporated into an electronic system, the TEST BUS CONTROLLER SN74ACT8990 in combination with the SCAN PATH SELECTOR SN74ACT8999 and a microprocessor provide

an ideal basis (Figure 2). The main computer can give the microprocessor the signal to start BIST via the IEEE 1149.1 test bus. The result of the self - test is now communicated from the microprocessor to the main computer.

If a large system is composed of several subsystems, it is advantageous to have BIST in each subsystem. The self-test can then be implemented simultaneously in all subsystems, resulting in an enormous reduction of test time.

This is one of the cases in which direct programming of the TEST BUS CONTROLLER is necessary, and in which this Application Report is intended to give assistance.

2. Hardware

The TEST BUS CONTROLLER SN74ACT8990 provides the interface between a processor and the IEEE 1149.1 test bus. Both interfaces can be operated asynchronously, that is, the clock of the computer does not need to be synchronized with the test clock TCK.

2.1. The Processor Interface

The processor interface consists of the following:

⇒ 5 Bit Address Bus A0 .. A4,

⇒ 16 Bit Data Bus D0 .. D15,

⇒ A Read Line /RD,

⇒ A Write Line /WR,

⇒ A Status Line /RDY, and

⇒ An Interrupt Line /INT

The TBC is connected to the processor like a normal interface circuit (Figure 3) , whereby the ADDRESS DECODER can be implemented as PAL. The clock signal TCLK is buffered in the TEST BUS CONTROLLER, and then transferred to the test system as the TCK signal. It can

⇒ be derived from an independent clock generator,

⇒ from the processor clock, or

⇒ can be provided from the test system.

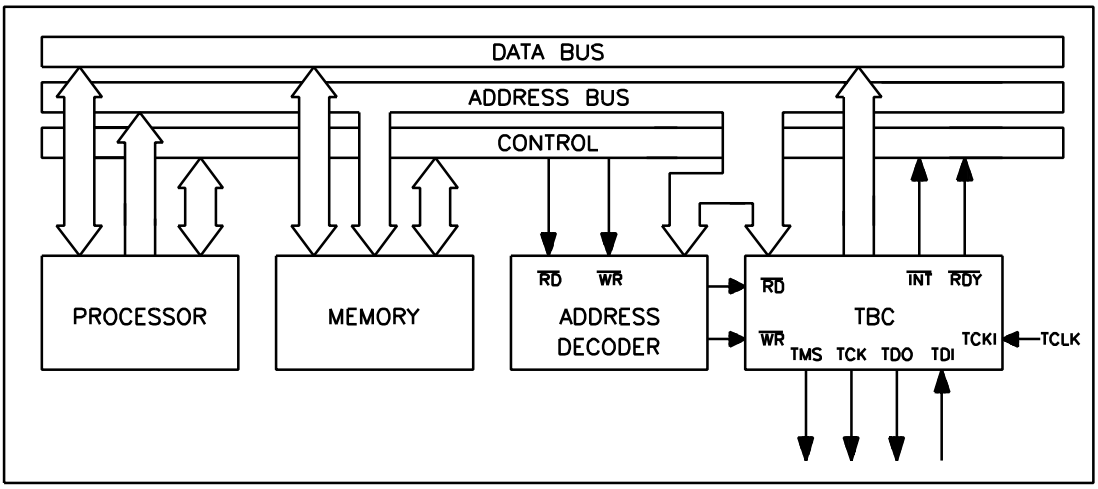


Figure 3: Connection of the TBC to a Microprocessor

5 Address-lines can be used to address 24 internal registers:

⇒ 10 Control Registers,

⇒ 2 Command Registers,

⇒ 6 Registers to control internal counters,

⇒ 4 Status Registers,

⇒ 1 Register for Transmit data, and

⇒ 1 Register for Receive data.

16 Registers can be read and overwritten, 7 are only readable, and 1 register can only be written into. The remaining 8 potential registers are not occupied. Table 1 on page 23 gives details of the Registers. The wave form of a writing cycle is shown in Figure 4, and of a read cycle in Figure 5.

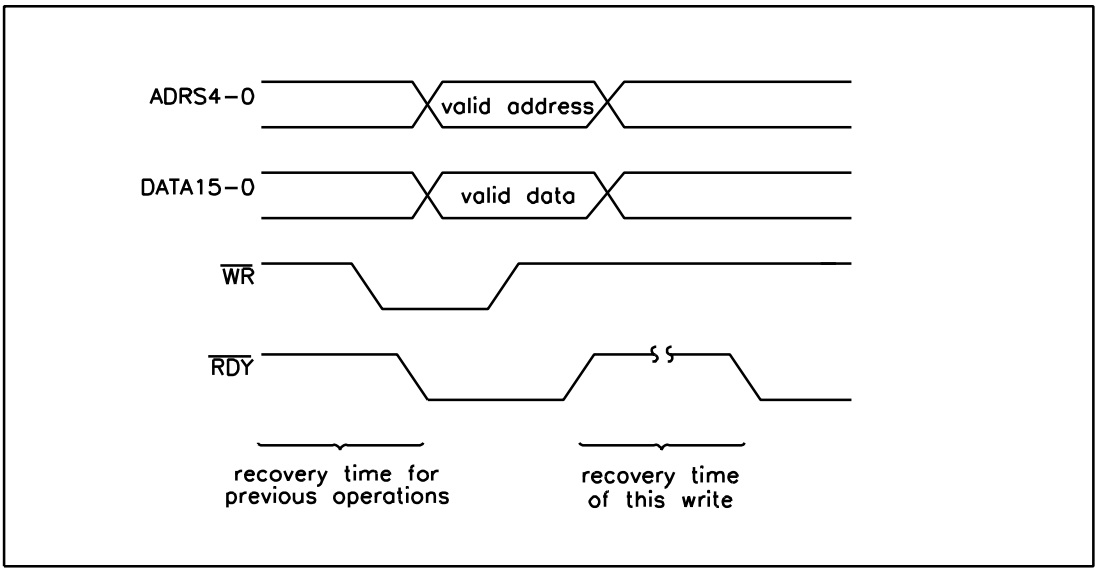


Figure 4: Timing Diagram of Processor Interface when Writing

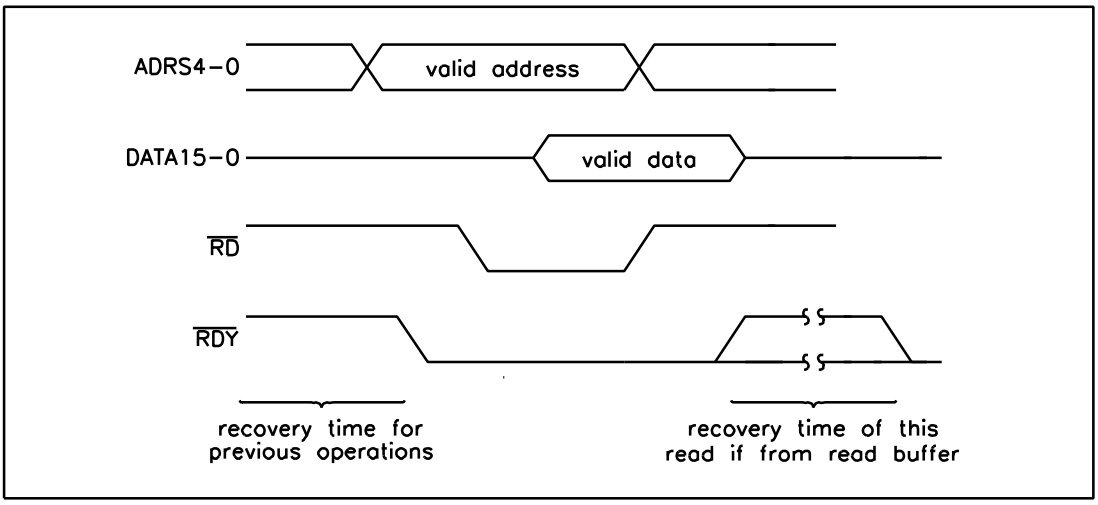


Figure 5: Timing Diagram of Processor when Reading

2.2. The Test Bus Interface

According to IEEE 1149.1, the test bus interface consists of:

⇒ 1 Test pulse input (TCKI),

⇒ 1 Test pulse output (TCKO),

⇒ 1 Test data output (TDO),

⇒ 2 Test data inputs (TDI0, TDI1),

⇒ 2 TEST MODE SELECT outputs (TMS0, TMS1),

⇒ 4 lines which can be used as further TMS outputs (TMS2 .. TMS5) or can be programmed as event inputs/outputs (EVENT0 .. EVENT3),

⇒ 1 line /TOFF (TEST BUS OFF) in order to reset all test outputs into a high resistance state, and

⇒ 1 Input /TRST (TEST BUS RESET) in order to reset the TEST BUS CONTROLLER and all tests circuits. The use of this input is optional, since the TBC and also the test systems are provided with a software reset.

Figure 6 shows the connection of an IEEE 1149.1 test system to the TBC. Here only one TMS signal is needed, so that in this case all four event inputs/outputs (EVENT0 .. EVENT3) are available. Since the signals TMS2 .. TMS5 use the same lines as the event inputs/outputs, the number of available event inputs/outputs is reduced if more than 2 test systems are connected to the TBC (Figure 7).

The test data outputs (TDO) of several test systems can be connected in parallel and to a TDI input of the TEST BUS CONTROLLERs, since they are active only during the shift operation (SHIFT-DR, SHIFT-IR), and are

otherwise in a high impedance state. If the test bus signals must be buffered (Figure 8), the TDO output of the test system can no longer be in a high impedance state. For this eventuality, two test data inputs (TDI0, TDI1) are available.

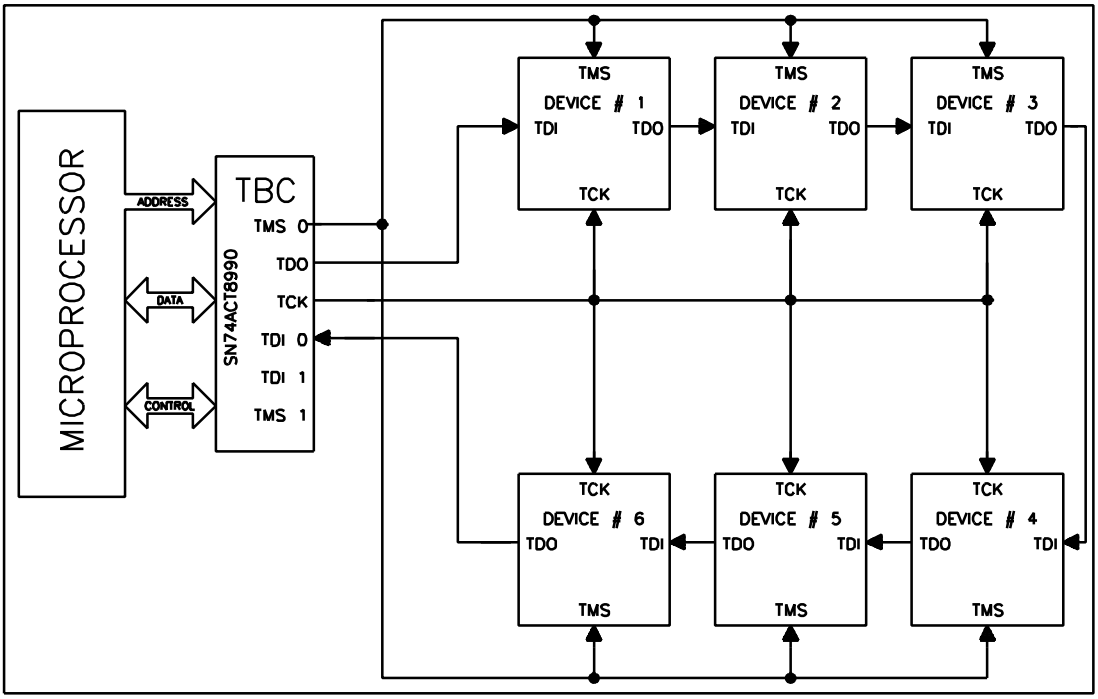


Figure 6: Connection of an IEEE 1149.1 Test Bus to the TBC

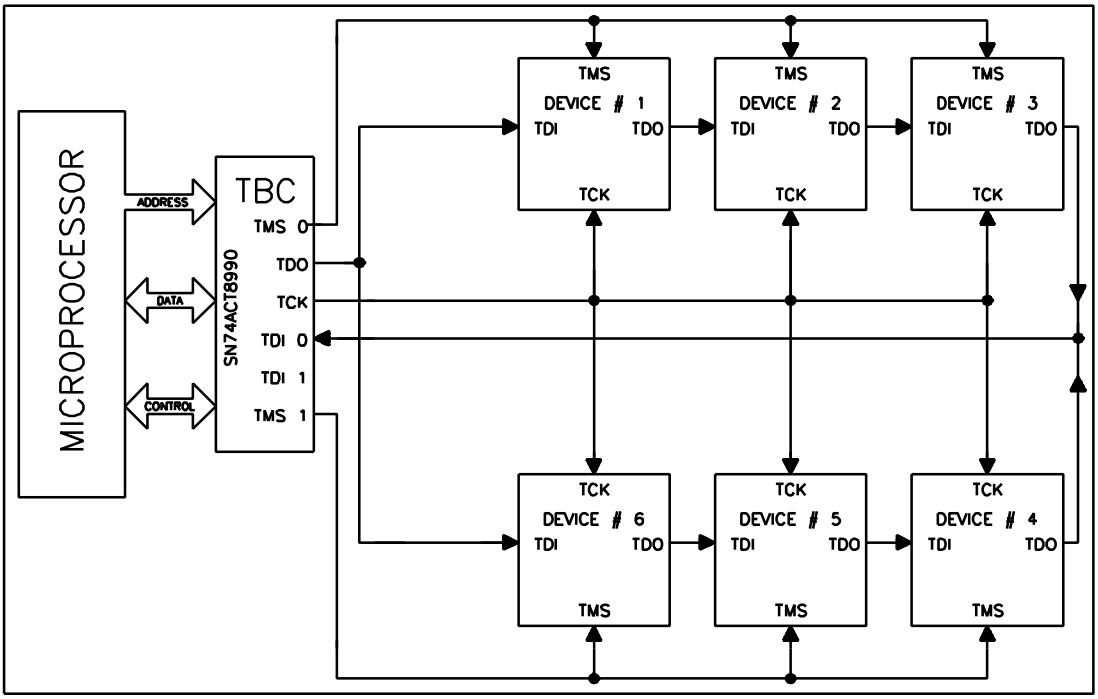


Figure 7: Connection of Two IEEE 1149.1 Test Busses to the TBC

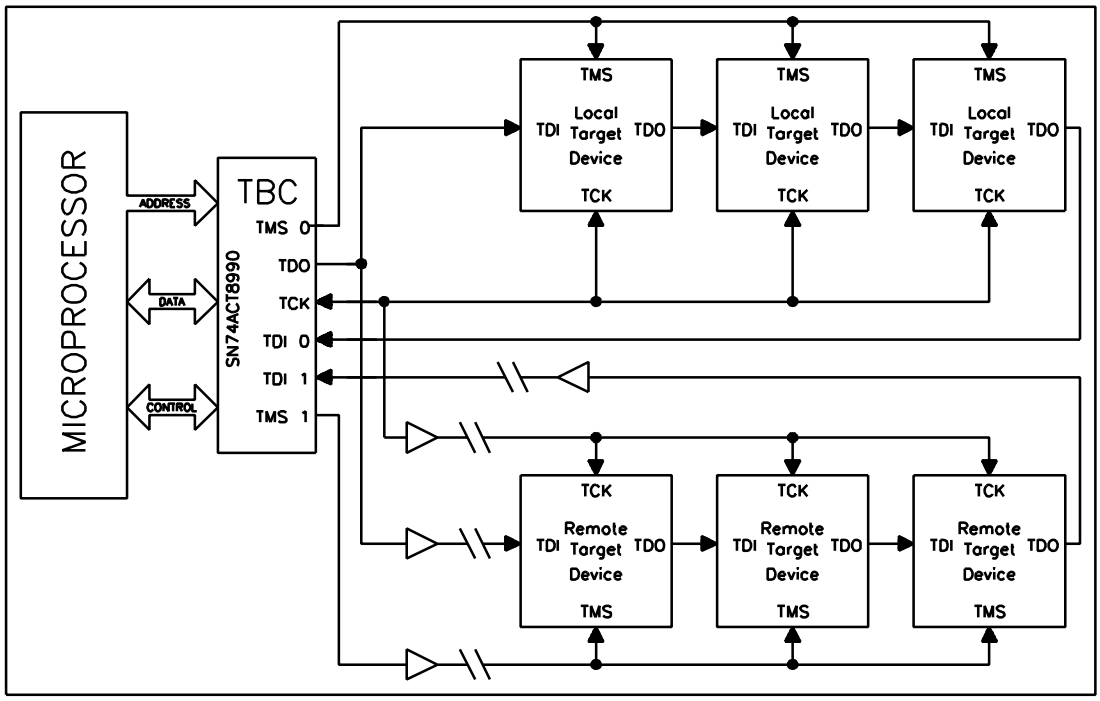


Figure 8: Connection of Several IEEE 1149.1 Test Busses to the TBC, with Buffering of the Test Bus Signal

2.3. Architecture

Figure 9 shows the block diagram of the TEST BUS CONTROLLER consisting of 6 functional blocks:

⇒ SEQUENCE BLOCK,

⇒ SERIAL BLOCK,

⇒ EVENT BLOCK,

⇒ COUNTER BLOCK,

⇒ COMMAND BLOCK and

⇒ HOST BLOCK.

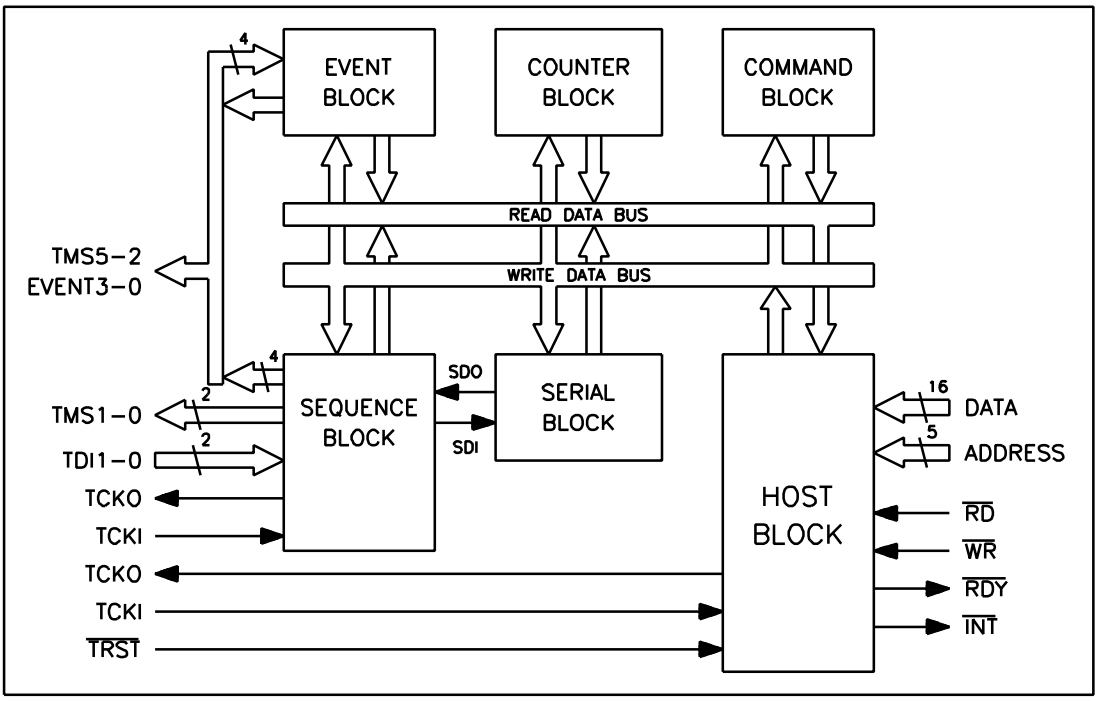


Figure 9: Block Diagram of the TEST BUS CONTROLLER SN74ACT8990

2.3.1. SEQUENCE BLOCK

The SEQUENCE BLOCK (Figure 10) traces the test system through the status diagram (Figure 21, Page 49). During this process, the actual status of the test system is always known. The signals TMS, TDO and TDI are generated by the SEQUENCE BLOCK according to IEEE 1149.1. During the SHIFT-DR and SHIFT-IR states, it uses the SDO (SERIAL DATA OUT) data from the SERIAL BLOCK to generate the TDO signal. Additionally, the SEQUENCE BLOCK transfers the TDI received data over the SDI (SERIAL DATA IN) line to the SERIAL BLOCK.

Important for the user are the programming possibilities:

⇒ Choice of the TMS line to be used (TMS0...TMS5)

⇒ Switching of all test bus lines into a high resistance state

⇒ Choice of the TDI line to be used

⇒ Programmed delay of the test bus signals (LINK DELAY)

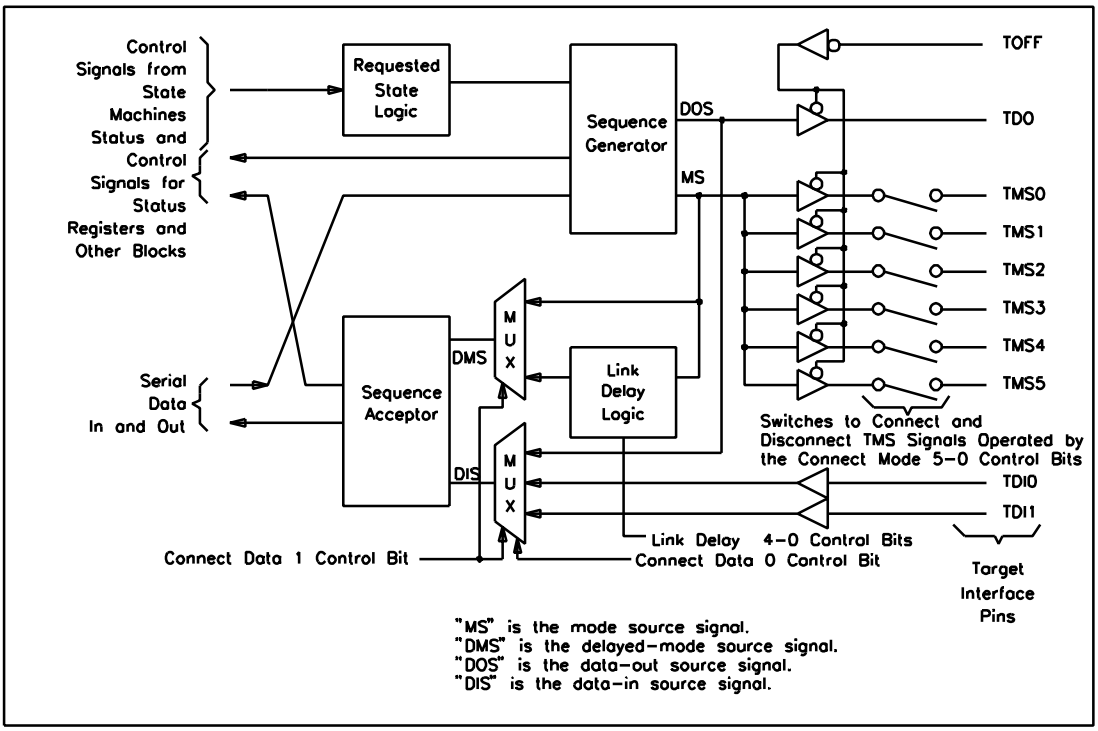


Figure 10: SEQUENCE BLOCK

The LINK DELAY logic (Figure 11) allows a simple way of taking account of flip-flop chains which are switched in the signals TDO, TDI and TMS between the TEST BUS CONTROLLER and the test board. These flip-flops can, for example, be used to synchronize signals coming from long lines, from 'ACT8990 to test boards.

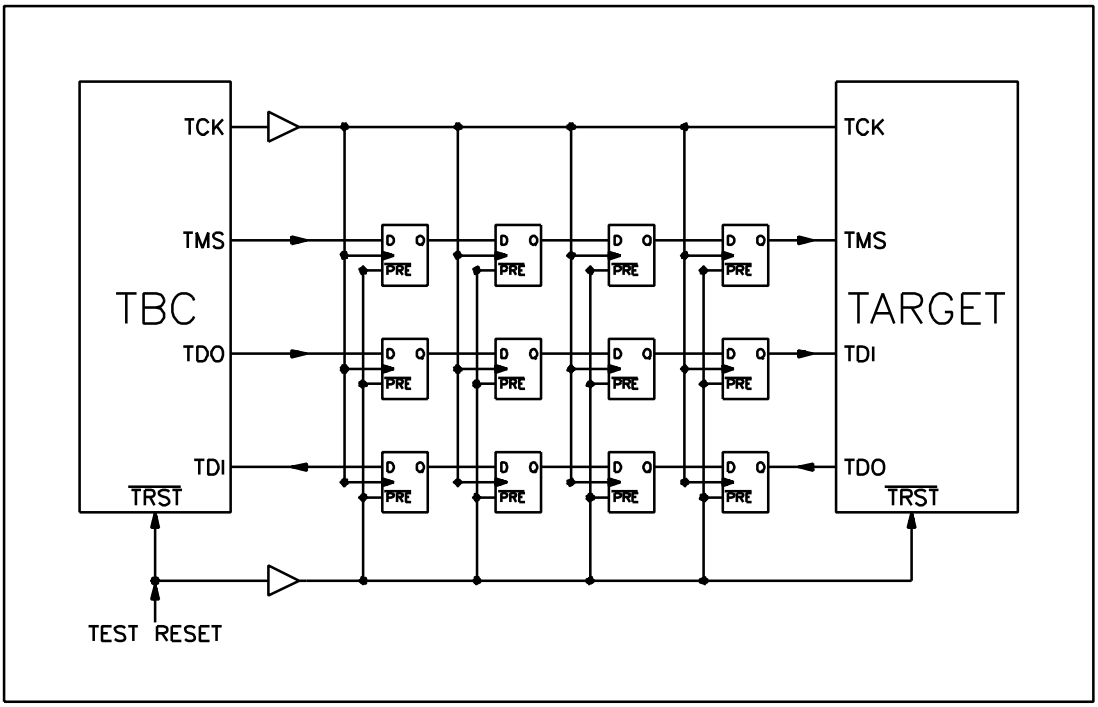


Figure 11: LINK DELAY Logic

2.3.2. SERIAL BLOCK

An understanding of the operation of the SERIAL BLOCK (Figure 12) is an important requirement for the application of the TEST BUS CONTROLLER. The SERIAL BLOCK supplies the commands and data which are sent to the IEEE 1149.1 compatible test boards in the SHIFT-IR and SHIFT-DR states. In the same way it collects the bits which are read back from the TDO output of the test board.

The output SDO (SERIAL DATA OUT) passes through the following SEQUENCE BLOCK in order then to reach the test board as the TDO signal. The SDO signal can be chosen from the following via the multiplexer MUX2:

⇒ The SDO data derives from the WRITE BUFFER.

⇒ Die SDO data derives from the SHIFTER-FIFO.

⇒ Die SDO data consists only of ones

⇒ Die SDO data consists only of zeroes

The microprocessor writes data and commands into the WRITE BUFFER, which are sent to the test board via the SDO output. This WRITE BUFFER consists of two 16-Bit registers. If the computer writes a 16-Bit word into the first register, then this register can be immediately shifted out via SDO. In the meantime it is already possible to write the next 16-Bit word into the second register. The functional control of the register - that is, which register is transferring data and which can be addressed by the computer - is automatically performed by the TBC. The user must only request, via a status register, how many registers of the WRITE BUFFER are empty.

The WRITE TRANSFER FUNCTION allows the user the choice of whether the WRITE BUFFER should be shifted to the right - i.e. starting with the LSB (LEAST SIGNIFICANT BIT), or to the left - i.e. beginning with the MSB (MOST SIGNIFICANT BIT).

The TDI receiver data from the test board pass first through the SEQUENCE BLOCK, and are then shifted over the SDI (SERIAL DATA IN) into the SHIFTER-FIFO. The SHIFTER-FIFO is a shift register, whose length can be programmed to 16 or 32 bits with the help of the multiplexer MUX1. After reset, the SHIFTER-FIFO is loaded with 1's. All bits that leave the SHIFTER-FIFO during a shift process are stored in the READ BUFFER.

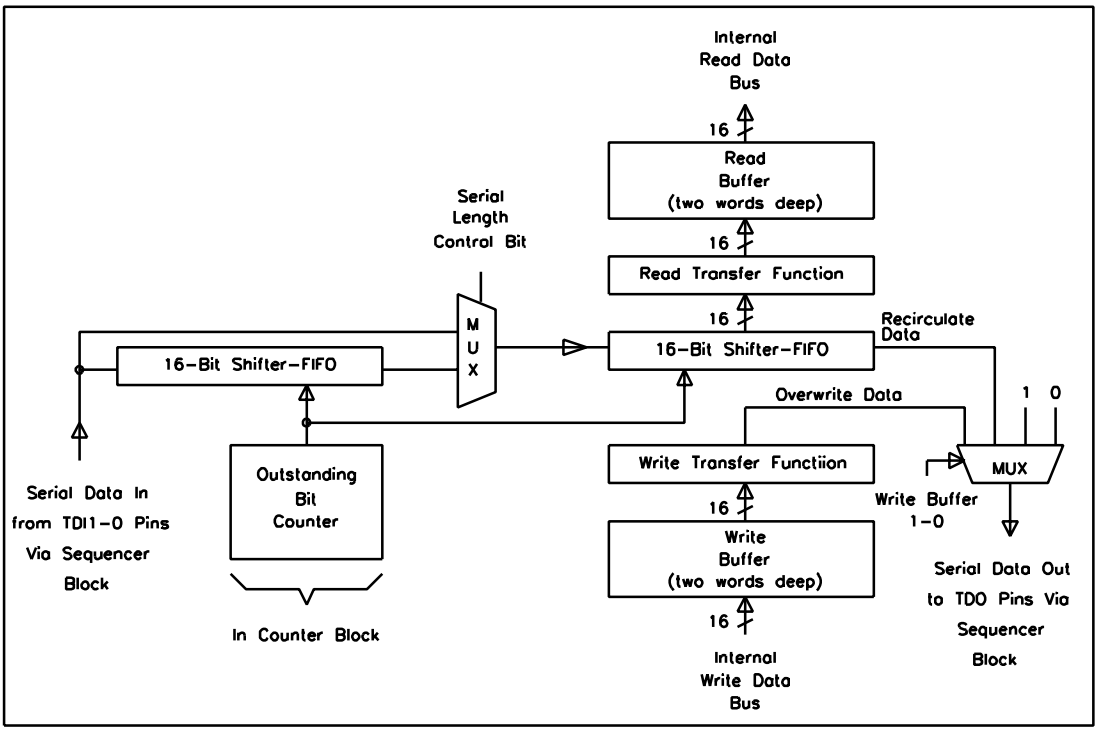


Figure 12: SERIAL BLOCK

The READ BUFFER (like the WRITE BUFFER) consists of two 16 bit registers. The operation is similar to that of the WRITE BUFFER; the only difference is that the data flow is in the reverse direction i.e. the processor reads the data. The processor can find out via a register how many registers of the READ BUFFERs contain data. Since, as explained above, the SHIFTER-FIFO is loaded with 1's on reset, the first bits which the computer reads from the READ BUFFER are 16 or 32 1's, depending on length to which the SHIFTER-FIFOs have been programmed. Only then will the first bits arrive from the TDI input.

The bit-by-bit transfer of data from the SHIFTER-FIFO to the READ BUFFER is performed by the READ TRANSFER FUNCTION. It allows the user the following programs:

⇒ The bits are transferred from the right into the WRITE BUFFER, with the first transmitted bit as LSB in the WRITE BUFFER.

⇒ The bits are transmitted from the right into the WRITE BUFFER, with the last transmitted bit as LSB in the WRITE BUFFER.

2.3.3. EVENT BLOCK

With the help of the EVENT BLOCK, external events can be used to influence the test program, or transferred to the control computer from the EVENT BLOCK as an INTERRUPT. The operation of the EVENT BLOCK is only of interest to those users who want to control the test program in

accordance with events. If no such control is required, the following section can be skipped.

Figure 13 shows the block diagram of the EVENT BLOCK inputs. Each input has its own EVENT DETECTOR, which reacts on recognizing predetermined events. In addition the two 16 bit backward counters COUNTER20 and COUNTER21 are available to count events. (Figure 15, Page 19). These counters can also be configured as two combined 16 bit counters (Figure 16, Page 19) and as a 32 bit counter (Figure 17, Page 20). The multiplexers of the EVENT BLOCKs allow the programming of a variety of different functions.

For example, the EVENT0 input can be switched directly to the EVENT DETECTOR with the use of the multiplexer MUX0. The EVENT DETECTOR 0 is programmed to recognize a particular event e.g. the arrival of a rising pulse edge. It is however also possible to count events which occur at the input EVENT0 only with COUNTER20. After the counter of the COUNTER UPDATE register of the COUNTER BLOCKs (Figure 18 on Page 21) has been loaded with the required value, on arrival of an event at the EVENT0 input it will be decremented by one. At the zero crossing of the counter COUNTER20, the EVENT DETECTOR activates its output.

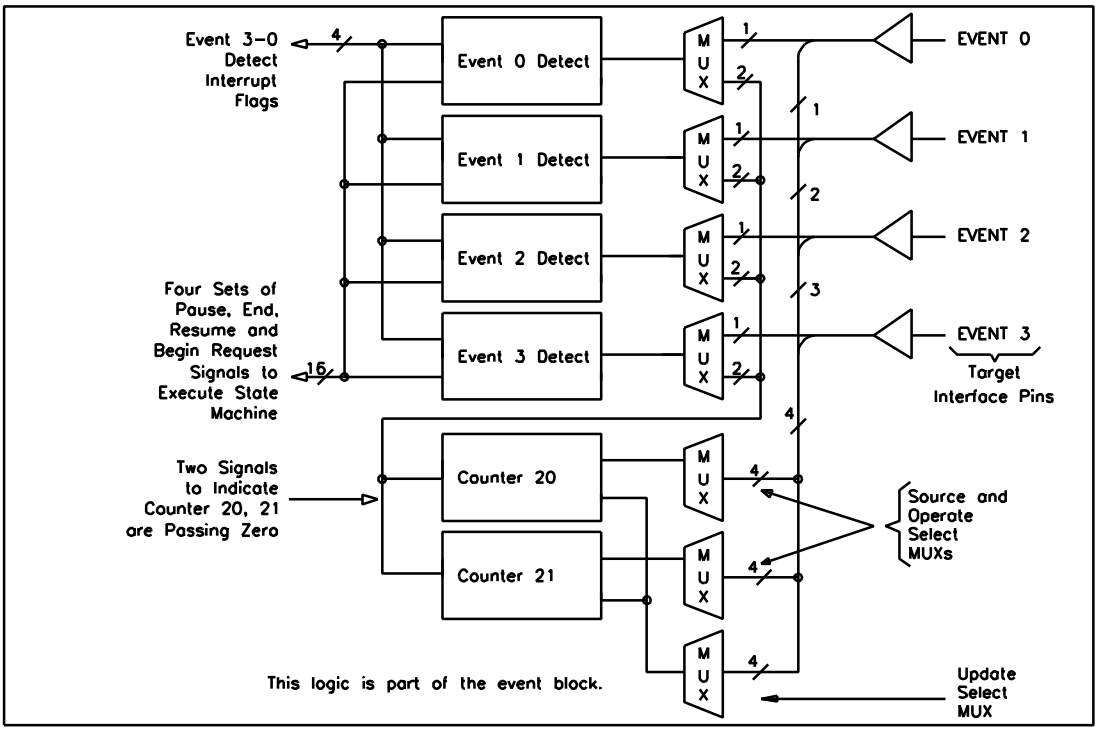


Figure 13: Inputs of the EVENT BLOCK

The control computer has no direct access to the counters COUNTER20 and COUNTER21, however the COUNTER2 UPDATE register can be written into from the control computer. If the counters COUNTER20 and COUNTER21 are loaded in parallel, then this always takes place with the value in the COUNTER2 UPDATE register.

The output circuit of the EVENT BLOCK can be seen in Figure 14. For each output, two latches are available. One latch stores the required output level, the second latch allows the driver to switch into a high resistance state, and therefore to configure the connection as an input.

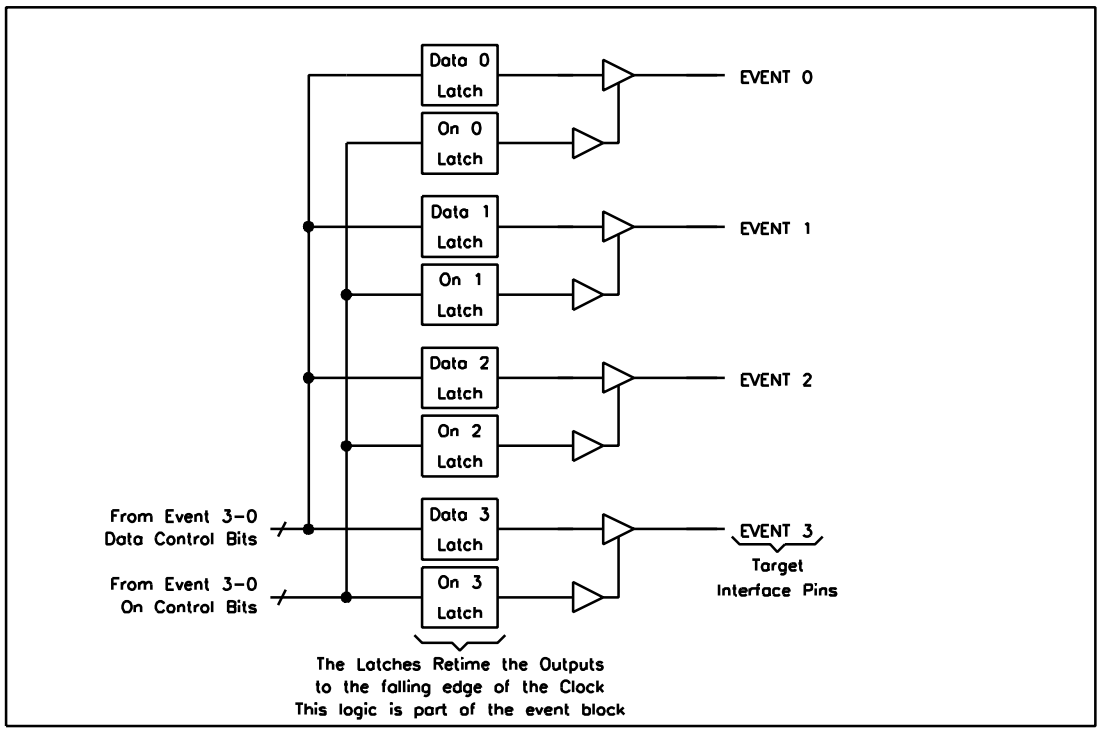


Figure 14: Outputs of the EVENT BLOCK

The two counters COUNTER20 and COUNTER21 can be configured as follows:

⇒ COUNTER20 and COUNTER21 work as two separate 16 bit counters.

⇒ COUNTER20 and COUNTER21 work as combined 16 bit counters. In this case, both counters are implemented as separate 16 bit counters, but counter COUNTER20 is loaded from the COUNTER UPDATE register until COUNTER21 has the first zero crossing. In this case, two events control the event recognition. For example, it can be programmed that the first event must occur 55 times, and then the second event 12345 times, before the EVENT DETECTOR responds.

⇒ Both counters are connected together to make a 32 bit counter COUNTER2.

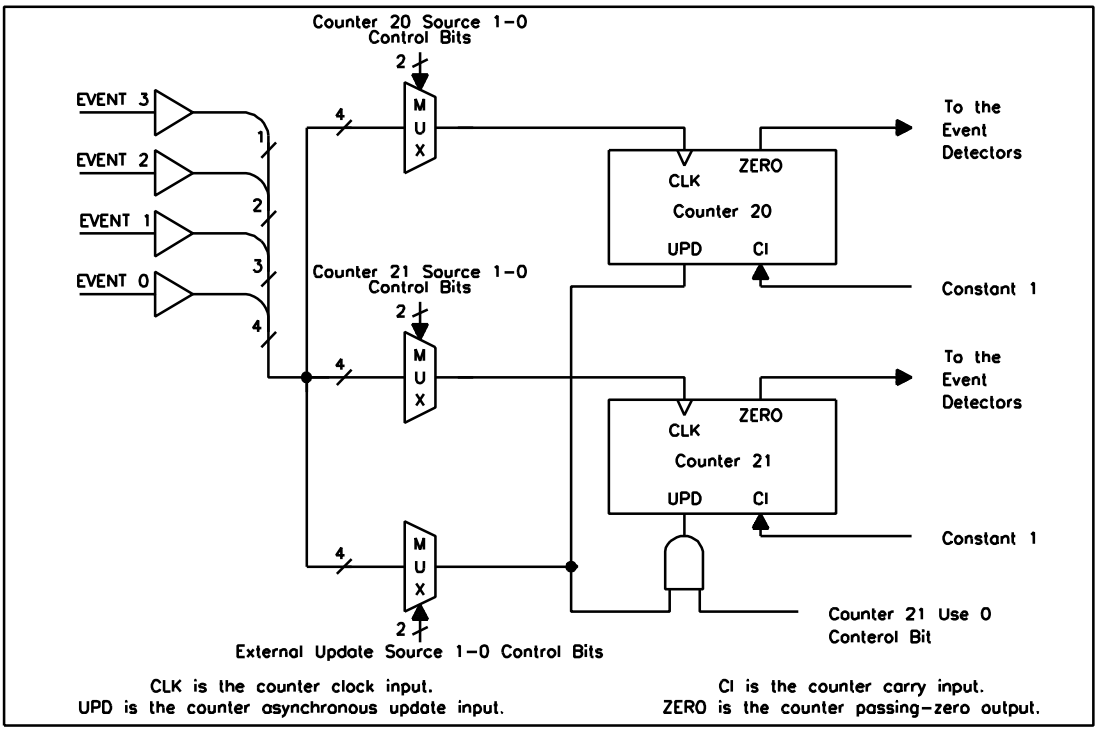


Figure 15: COUNTER20 and COUNTER21 configured as two separate 16 bit counters

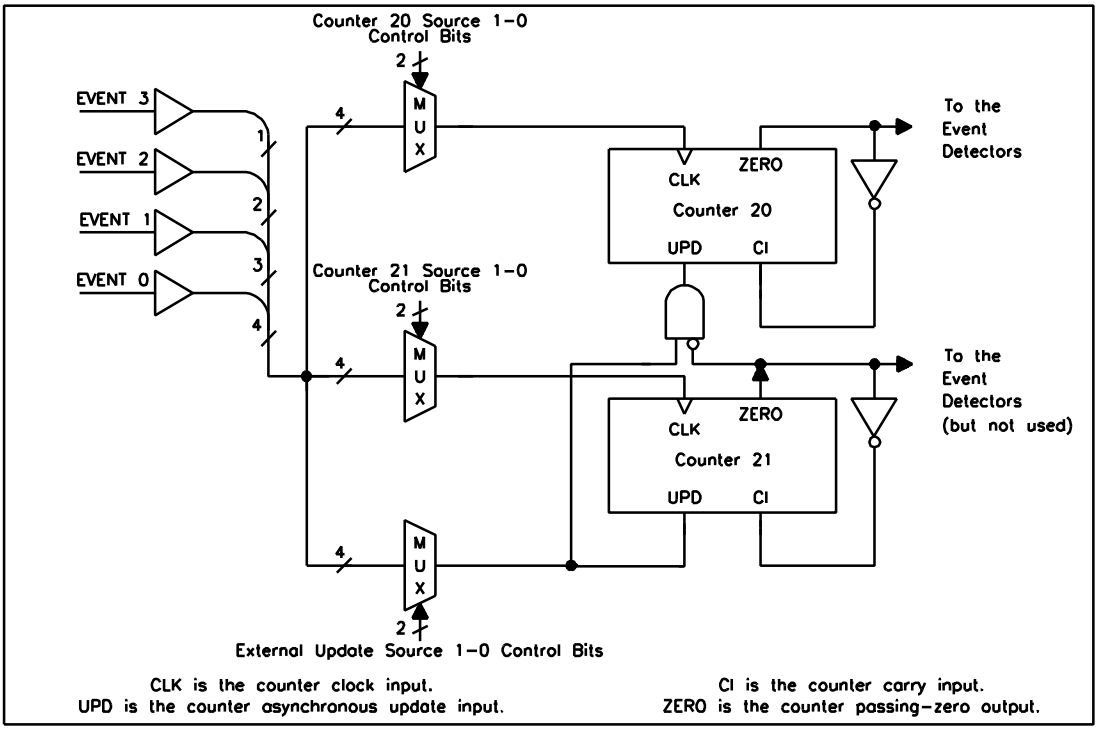


Figure 16: COUNTER20 and COUNTER21 configured as two combined 16 bit counters

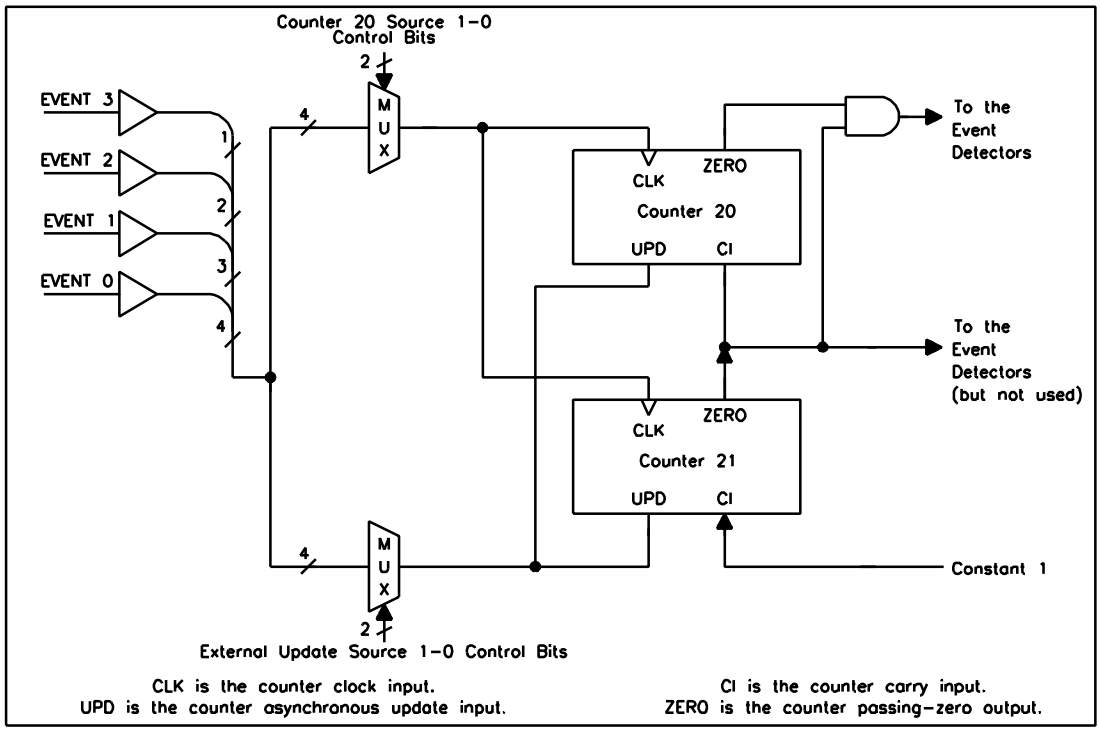


Figure 17: COUNTER20 and COUNTER21 configured as one 32 bit counter COUNTER2

2.3.4. COUNTER BLOCK

The part of the COUNTER BLOCK (Figure 18), which is relevant for the user, consists of:

⇒ the COUNTER1 UPDATE register,

⇒ the 32 bit backwards counter COUNTER1 and

⇒ the COUNTER1 CAPTURE register.

The counter COUNTER1 is used to count the SHIFT-DR, SHIFT-IR and RUN TEST/IDLE states. It is also possible to use it to count events. For this purpose however the counters COUNTER20 and COUNTER21 are primarily available.

The control computer has no direct access to the counter COUNTER1; it can only write into the COUNTER1 UPDATE register and read the COUNTER1 CAPTURE register. If the COUNTER1 is loaded in parallel, then this is always done with the value of the COUNTER1 UPDATE register. The actual count state of the counter COUNTER1 can be copied parallel into the COUNTER1 CAPTURE register. From there, the actual count state can pass to the controlling computer.

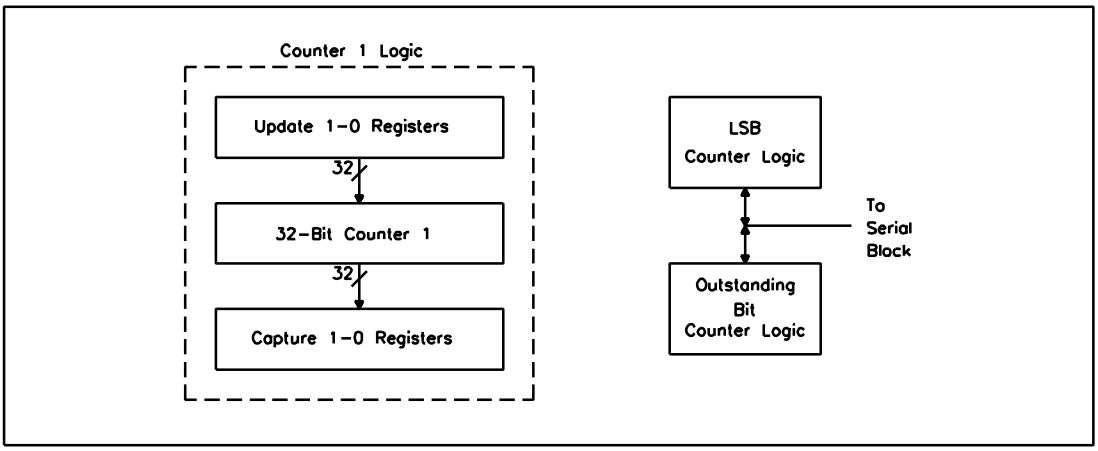


Figure 18: COUNTER BLOCK

2.3.5. COMMAND BLOCK

The COMMAND BLOCK (Figure 19) contains the two command registers MAJOR COMMAND and MINOR COMMAND. In addition, all status registers are part of this block. The COMMAND BLOCK controls with the COMMAND DECODE LOGIC the command process, without the user needing to intervene.

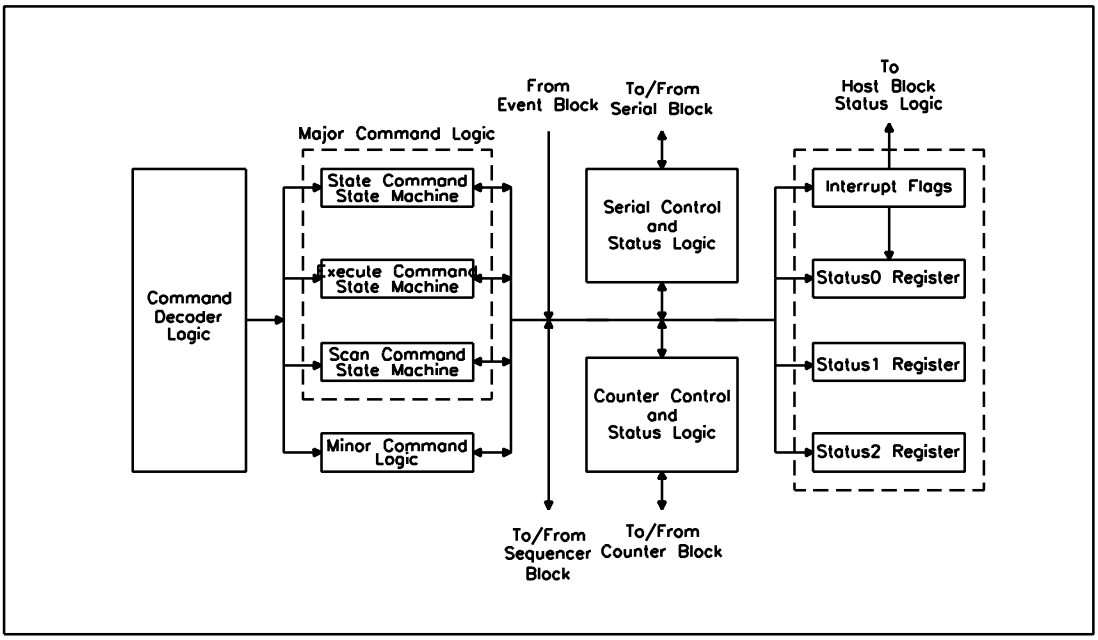


Figure 19: COMMAND BLOCK

2.3.6. HOST BLOCK

The tasks of the HOST BLOCK include databus signal conditioning, decoding addresses and control lines, and conditioning interrupts for the control computer. In addition, the HOST BLOCK includes the RESET LOGIC and the buffering of the test pulse TCK; this block also performs its functions without the user needing to intervene.

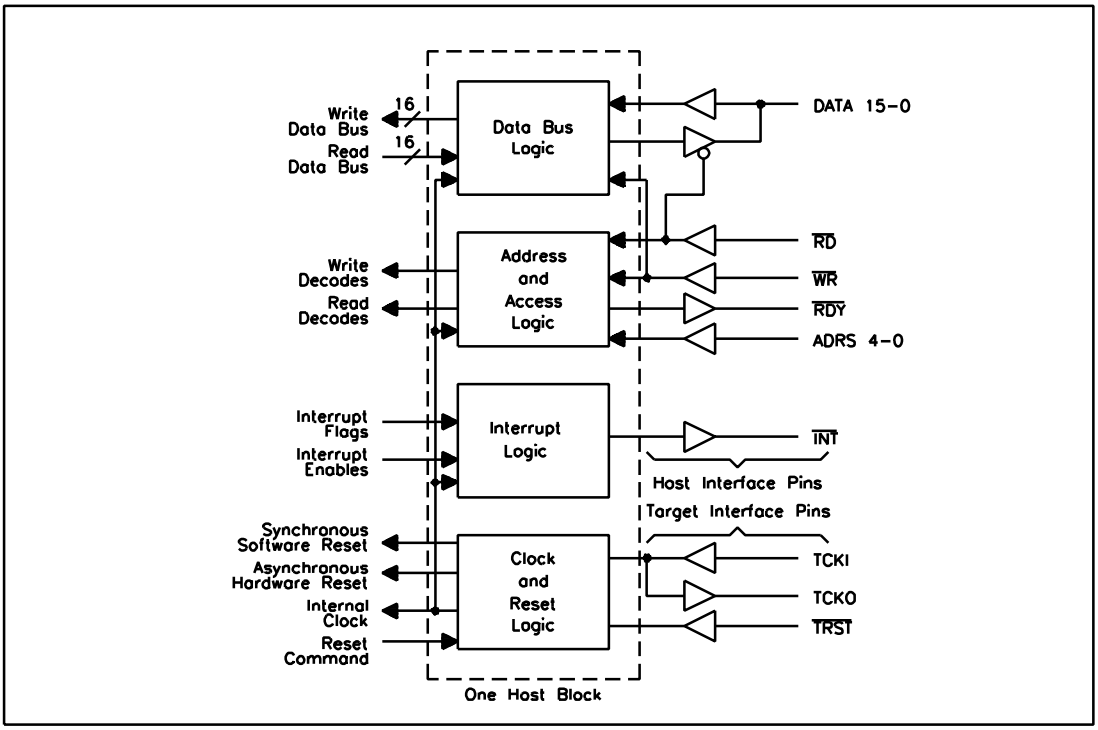


Figure 20: HOST BLOCK

3. Programming

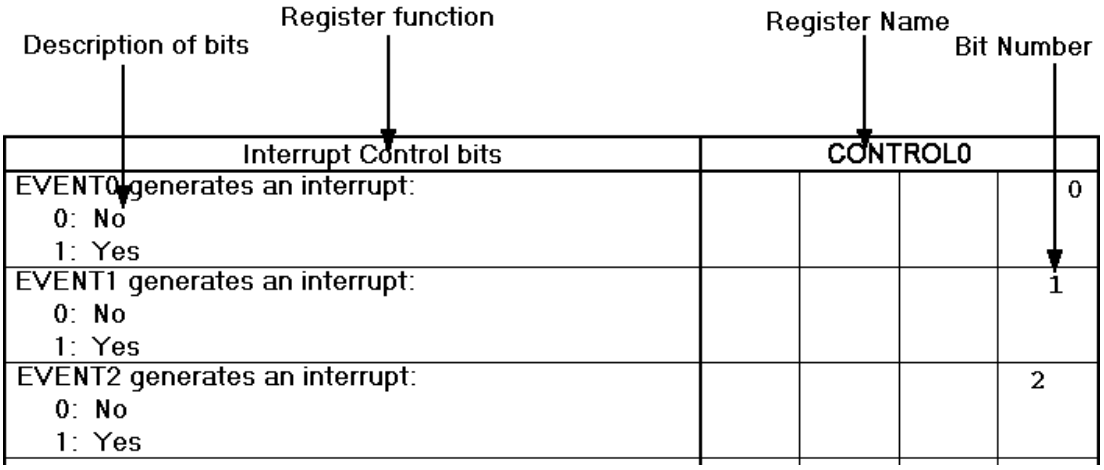
The control computer has access to the functions of the TEST BUS CONTROLLER with 24 registers, which are listed in Table 1. The 10 CONTROL registers configure the TEST BUS CONTROLLER. The 32 bit command register is made up of a combination of the MAJOR COMMAND and MINOR COMMAND registers. With the MAJOR COMMAND, presets of the commands are implemented, while the MINOR COMMAND register controls the execution; the commands are started, interrupted or terminated. With the four COUNTER UPDATE and the two CAPTURE registers, the counters of the EVENT BLOCK and the COUNTER BLOCK are controlled. The internal state of the TEST BUS CONTROLLER can be ascertained over three STATUS registers. The READ BUFFER and WRITE BUFFER are intended for the exchange of data between the test board and the controlling computer.

ADR	Register Name	Application	Access
00	CONTROL0	Interrupt Control Bits	Read/Write
01	CONTROL1	Special Function for Production Test	Read/Write
02	CONTROL2	TMS0 .. TMS5 Connections	Read/Write
03	CONTROL3	TMS/EVENT Function, TDO on/off, TMS/TCK Data Format	Read/Write
04	CONTROL4	Delay, Data Format of TDI and TDO pins	Read/Write
05	CONTROL5	EVENT pins Data Output and Direction	Read/Write
06	CONTROL6	Event Choice of EVENT0/1 Inputs	Read/Write
07	CONTROL7	Event Choice of EVENT2/3 Inputs	Read/Write
08	CONTROL8	COUNTER1 Control	Read/Write
09	CONTROL9	COUNTER2 Control	Read/Write
0A	MINOR COMMAND	Replace Interrupt Flag Bits, Control of Test Run, Read Status, load counters and count, Software RESET	Read/Write
0B	MAJOR COMMAND	STATE Command, EXECUTE Command, SCAN Command	Read/Write
0C	COUNTER1 UPDATE0	COUNTER1 Lower Value Word	Read/Write
0D	COUNTER1 UPDATE1	COUNTER1 Higher Value Word	Read/Write
0E	COUNTER2 UPDATE0	COUNTER2 Lower Value Word or COUNTER20	Read/Write
0F	COUNTER2 UPDATE1	COUNTER2 Higher Value Words or COUNTER21	Read/Write
10	STATUS0	Internal States	Read only
11	STATUS1	External Signals and Requests	Read only
12	STATUS2	State of test board and Buffer	Read only
13	STATUS3	not used	Read only
14	CAPTURE0	COUNTER1 Lower Value Word	Read only
15	CAPTURE1	COUNTER1 Higher Value Word	Read only
16	READ BUFFER	Input Buffer for TDI	Read only
17	WRITE BUFFER	Output Buffer for TDO	Write only

Table 1: Registers of the TEST BUS CONTROLLER SN74ACT8990

3.1. The Registers

In this chapter, the individual bits of all registers of the TEST BUS CONTROLLER will be presented in tables. The table heading includes the register name, and a short description of the register function. Within the tables, all bits are individually described. The bit numbers are assembled in groups each of four bits, in order to simplify the presentation in hexadecimal form.



CONTROL0

Interrupt Control bits	CONTROL0			
EVENT0 generates an Interrupt: 0: No 1: Yes				0
EVENT1 generates an Interrupt: 0: No 1: Yes				1
EVENT2 generates an Interrupt: 0: No 1: Yes				2
EVENT3 generates an Interrupt: 0: No 1: Yes				3
The COUNTER1 EXECUTE Flag generates an Interrupt: 0: No 1: Yes			4	
The COUNTER1 SCAN Flag generates an Interrupt: 0: No 1: Yes			5	
NULL			76	
The SUSPEND Flag generates an Interrupt: 0: No 1: Yes		8		
The END Flag generates an Interrupt: 0: No 1: Yes		9		
The RESUME Flag generates an Interrupt: 0: No 1: Yes		A		
The BEGIN SCAN Flag generates an Interrupt: 0: No 1: Yes		B		
The BUFFER ERROR Flag generates an Interrupt: 0: No 1: Yes	C			
The BUFFER READY Flag generates an Interrupt: 0: No 1: Yes	D			
The ABORT Flag generates an Interrupt: 0: No 1: Yes	E			
The FINISH Flag generates an Interrupt: 0: No 1: Yes	F			

CONTROL1

Special Functions for Production Test	CONTROL1			
Each Bit: 0: Normal operation 1: Test operation for Production Test				3210
Not used			7554	
Each Bit: 0: Normal operation 1: Test operation for Production Test	C	BA98		
Not used	FED			

When using the 'ACT8990 as controller of a test bus according to IEEE 1149.1, all bits of these registers must be set by the user to zero. With the help of these registers, the manufacturer can switch various parts of the component into test operation during production test. This allows a comprehensive but simple functional examination of the internal circuitry.

CONTROL2

TMS0 .. TMS5 Connections	CONTROL2			
TMS0 is connected with internal TMS source: 0: No 1: Yes				0
TMS1 is connected with internal TMS source: 0: No 1: Yes				1
TMS2 is connected with internal TMS source: 0: No 1: Yes				2
TMS3 is connected with internal TMS source: 0: No 1: Yes				3
TMS4 is connected with internal TMS source: 0: No 1: Yes			4	
TMS5 is connected with internal TMS source: 0: No 1: Yes			5	
NULL	FEDC	BA98	76	

The component generates internally a TMS signal. This signal can be extracted at the pins TMS0 to TMS5. All TMS pins which are not connected to the internal TMS source remain in a static state.

CONTROL3

Function of the TMS/EVENT pins, Data Format of the TMS and TCK pins	CONTROL3			
TMS2/EVENT0: 0: EVENT Input or Output 1: TMS2 Output				0
TMS3/EVENT1: 0: EVENT Input or Output 1: TMS3 Output				1
TMS4/EVENT2: 0: EVENT Input or Output 1: TMS4 Output				2
TMS5/EVENT3: 0: EVENT Input or Output 1: TMS5 Output				3
TCKO on/off: 0: TCKO Output is switched on 1: TCKO Output is switched off			4	
TDO on/off: 0: TDO Output is switched on 1: TDO Output is switched off			5	
TMS on/off: 0: all TMS Outputs are switched on 1: all TMS Outputs are switched off			6	
EVENT on/off: 0: all EVENT Outputs are switched on 1: all EVENT Outputs are switched off			7	
Output Bit Format: 0000: for operation to IEEE 1149.1 else: for operation with special TI components		BA98		
TCKO Output signal if TCKO Bit D is set to 1: 0: static 0-signal 1: static 1-signal	C			
TCKO Data Format: 0: TCKO is connected with internal TCK source 1: static signal	D			
NULL	FE			

CONTROL4

Delay, Data Formats of TDI and TDO Pins	CONTROL4			
Delay: 00000 to 11111: with these bits, the LINK DELAY Registers can be set to a delay from 0 to 31 bits length.			4	3210
SHIFTER-FIFO Length: 0: 16 bits 1: 32 bits			5	
TDI Selection: 00, 01: The internal TDO signal is the source for the internal signal TDI 10: The TDI0 Input is the source for the internal signal TDI 11: The TDI1 Input is the source for the internal signal TDI			76	
TDO Sending Data: 00: TDO sends 0's 01: TDO sends 1's 10: TDO sends data of the SHIFTER-FIFO 11: TDO sends data of the WRITE BUFFER		98		
READ BUFFER on/off: 0: The received data is ignored after flowing through the SHIFTER-FIFO 1: The received data is transferred to the READ BUFFER after flowing through the SHIFTER-FIFO		A		
Transfer Format: 0: At data transfer, the LSB is always sent/received first 1: At data transfer, the MSB is always sent/received first		B		
TDI Data Transfer: 0: TDI Data Transfer on leading edge 1: TDI Data Transfer on trailing edge	C			
EVENT Data Transfer: 0: EVENT Data Transfer on leading edge 1: EVENT Data Transfer on trailing edge	D			
NULL	FE			

CONTROL5

EVENT Pins Output Data and Direction of Flow	CONTROL5			
EVENT0 Output Data: 0: 0-Level 1: 1-Level				0
EVENT1 Output Data: 0: 0-Level 1: 1-Level				1
EVENT2 Output Data: 0: 0-Level 1: 1-Level				2
EVENT3 Output Data: 0: 0-Level 1: 1-Level				3
NULL			7654	
EVENT0 Data Flow Direction: 0: EVENT0 is an input 1: EVENT0 is an output		8		
EVENT1 Data Flow Direction: 0: EVENT1 is an input 1: EVENT1 is an output		9		
EVENT2 Data Flow Direction: 0: EVENT2 is an input 1: EVENT2 is an output		A		
EVENT3 Data Flow Direction: 0: EVENT3 is an input 1: EVENT3 is an output		B		
NULL	FEDC			

CONTROL6

Event Choice at the EVENT0/1 Inputs	CONTROL6			
EVENT0 sets the SUSPEND REQUEST Flag: 0: No 1: Yes				0
EVENT0 sets the END REQUEST Flag: 0: No 1: Yes				1
EVENT0 sets the RESUME REQUEST Flag: 0: No 1: Yes				2
EVENT0 sets the BEGIN REQUEST Flag: 0: No 1: Yes				3
EVENT0 is synchronous or asynchronous: 0: The recognition of the EVENT0 is asynchronous 1: The recognition of the EVENT0 is synchronous			4	
EVENT0 Masking: 0: The recognition of the EVENT0 always occurs 1: No EVENT0 recognition during CAPTURE-DR, SHIFT-DR, EXIT1-DR, PAUSE-DR, EXIT2-DR, CAPTURE-IR, SHIFT-IR, EXIT1-IR, PAUSE-IR, EXIT2-IR			5	
Asynchronous EVENT0 recognition: 00: Trailing Edge at Input EVENT0 01: Leading Edge at Input EVENT0 10: Zero passing of COUNTER20 11: Zero passing of COUNTER21 Synchronous EVENT0 recognition of 2 successive TCK edges: 00: 1-Level --> 0-Level 01: 0-Level --> 1-Level 10: 0-Level --> 0-Level 11: 1-Level --> 1-Level			76	
EVENT1 sets the SUSPEND REQUEST Flag: 0: No 1: Yes		8		
EVENT1 sets the END REQUEST Flag: 0: No 1: Yes		9		
EVENT1 sets the RESUME REQUEST Flag: 0: No 1: Yes		A		
EVENT1 sets the BEGIN REQUEST Flag: 0: No 1: Yes		B		
EVENT1 is synchronous or asynchronous: 0: The recognition of the EVENT1 is asynchronous 1: The recognition of the EVENT1 is synchronous	C			
EVENT1 Masking: 0: The recognition of the EVENT1 always occurs 1: No EVENT1 recognition during CAPTURE-DR, SHIFT-DR, EXIT1-DR, PAUSE-DR, EXIT2-DR, CAPTURE-IR, SHIFT-IR, EXIT1-IR, PAUSE-IR, EXIT2-IR	D			

Asynchronous EVENT1 recognition: 00: Trailing Edge at input EVENT1 01: Leading Edge at input EVENT1 10: Zero passing of COUNTER20 11: Zero passing of COUNTER21 Synchronous EVENT1 recognition of 2 successive TCK edges: 00: 1-Level --> 0-Level 01: 0-Level --> 1-Level 10: 0-Level --> 0-Level 11: 1-Level --> 1-Level	FE			

CONTROL7

Event Choice of the EVENT2/3 Inputs	CONTROL7			
EVENT2 sets the SUSPEND REQUEST Flag: 0: No 1: Yes				0
EVENT2 sets the END REQUEST Flag: 0: No 1: Yes				1
EVENT2 sets the RESUME REQUEST Flag: 0: No 1: Yes				2
EVENT2 sets the BEGIN REQUEST Flag: 0: No 1: Yes				3
EVENT2 is synchronous or asynchronous: 0: The recognition of the EVENT2 is asynchronous 1: The recognition of the EVENT2 is synchronous			4	
EVENT2 Masking: 0: The recognition of the EVENT2 always occurs 1: No EVENT2 recognition during CAPTURE-DR, SHIFT-DR, EXIT1-DR, PAUSE-DR, EXIT2-DR, CAPTURE-IR, SHIFT-IR, EXIT1-IR, PAUSE-IR, EXIT2-IR			5	
Asynchronous EVENT2 recognition: 00: Trailing Edge at input EVENT2 01: Leading Edge at input EVENT2 10: Zero passing of COUNTER20 11: Zero passing of COUNTER21 Synchronous EVENT2 recognition of 2 successive TCK edges: 00: 1-Level --> 0-Level 01: 0-Level --> 1-Level 10: 0-Level --> 0-Level 11: 1-Level --> 1-Level			76	
EVENT3 sets the SUSPEND REQUEST Flag: 0: No 1: Yes		8		
EVENT3 sets the END REQUEST Flag: 0: No 1: Yes		9		
EVENT3 sets the RESUME REQUEST Flag: 0: No 1: Yes		A		
EVENT3 sets the BEGIN REQUEST Flag: 0: No 1: Yes		B		
EVENT3 is synchronous or asynchronous: 0: The recognition of the EVENT3 is asynchronous 1: The recognition of the EVENT3 is synchronous	C			
EVENT3 Masking: 0: The recognition of the EVENT3 always occurs 1: No EVENT3 recognition during CAPTURE-DR, SHIFT-DR, EXIT1-DR, PAUSE-DR, EXIT2-DR, CAPTURE-IR, SHIFT-IR, EXIT1-IR, PAUSE-IR, EXIT2-IR	D			

asynchronous EVENT3 recognition: 00: Trailing Edge at input EVENT3 01: Leading Edge at input EVENT3 10: Zero passing of COUNTER20 11: Zero passing of COUNTER21 synchronous EVENT3 recognition of 2 successive TCK edges: 00: 1-Level --> 0-Level 01: 0-Level --> 1-Level 10: 0-Level --> 0-Level 11: 1-Level --> 1-Level	FE			

CONTROL8

COUNTER1 Control	CONTROL8			
COUNTER1 sets the SUSPEND REQUEST Flag during zero passing: 0: No 1: Yes				0
COUNTER1 sets the END REQUEST Flag during zero passing: 0: No 1: Yes				1
COUNTER1 is loaded by the COUNTER1 UPDATE Register during zero passing: 0: No 1: Yes				2
COUNTER 1 counts 000, 001: never. 010: the clock edge of TCK in RUN-TEST/IDLE state during an EXECUTE command. 011: always when an EXECUTE command is executed. 100, 101: with the EVENT DETECTOR, which is selected with Bits 7 and 6. 110: if an 0-Level exists at the EVENT PIN selected by Bits 7 and 6. 111: if an 1-Level exits at the EVENT PIN selected by Bits 7 and 6.			54	3
COUNTER1 reacts on 00: EVENT DETECTOR0 or the input EVENT0 01: EVENT DETECTOR1 or the input EVENT1 10: EVENT DETECTOR2 or the input EVENT2 11: EVENT DETECTOR3 or the input EVENT3 if the Bits 5 and 4 have the value 100,101,110 or 111. In all other cases the value of these bits is without significance.			76	
not used	FEDC	BA98		

CONTROL9

COUNTER2 Control	CONTROL9			
COUNTER2 is loaded with an 000: 0-Level at the EVENT0 Input. 001: 1-Level at the EVENT0 Input. 010: 0-Level at the EVENT1 Input. 011: 1-Level at the EVENT1 Input. 100: 0-Level at the EVENT2 Input. 101: 1-Level at the EVENT2 Input. 110: 0-Level at the EVENT3 Input. 111: 1-Level at the EVENT3 Input.				210
COUNTER20 is loaded by the COUNTER2 UPDATE0 Register: 0: Yes 1: No				3
COUNTER20 is loaded during zero passing: 0: No 1: Yes			4	
COUNTER20 counts 000: if there is a trailing edge at the input EVENT0. 001: if there is a leading edge at the input EVENT0. 010: if there is a trailing edge at the input EVENT1. 011: if there is a leading edge at the input EVENT1. 100: if there is a trailing edge at the input EVENT2. 101: if there is a leading edge at the input EVENT2. 110: if there is a trailing edge at the input EVENT3. 111: if there is a leading edge at the input EVENT3.			765	
COUNTER2 can be loaded by an 0: asynchronous signal at the EVENT input. 1: synchronous signal at the EVENT input.		8		
COUNTER20 and COUNTER21 work as 00: separate 16-Bit counters; only COUNTER20 can be loaded by an asynchronous input signal. 01: separate 16-Bit counters; both counters can be loaded by an asynchronous input signal. 10: two tied 16-Bit counter 11: one 32-Bit counter. This counter can be loaded by an asynchronous input signal.		A9		
COUNTER21 is loaded by the COUNTER2 UPDATE1 Register: 0: Yes 1: No		B		
COUNTER21 is loaded at zero passing: 0: No 1: Yes	C			
COUNTER21 counts 000: if there is a trailing edge at the input EVENT0. 001: if there is a leading edge at the input EVENT0. 010: if there is a trailing edge at the input EVENT1. 011: if there is a leading edge at the input EVENT1. 100: if there is a trailing edge at the input EVENT2. 101: if there is a leading edge at the input EVENT2. 110: if there is a trailing edge at the input EVENT3. 111: if there is a leading edge at the input EVENT3.	FED			

MINOR COMMAND

Replace Interrupt Flag Bits 0 to 5	MINOR COMMAND			
Set/Reset the Interrupt Flag Bit 0: 0: No 1: Yes				0
Set/Reset the Interrupt Flag Bit 1: 0: No 1: Yes				1
Set/Reset the Interrupt Flag Bit 2: 0: No 1: Yes				2
Set/Reset the Interrupt Flag Bit 3: 0: No 1: Yes				3
Set/Reset the Interrupt Flag Bit 4: 0: No 1: Yes			4	
Set/Reset the Interrupt Flag Bit 5: 0: No 1: Yes			5	
NULL		BA98	76	
0000: Reset the Interrupt Flag Bits 0 to 5 0010: Set the Interrupt Flag Bits 0 to 5	FEDC			

Replace Interrupt Flag Bit 8 to 15	MINOR COMMAND			
Set/Reset the Interrupt Flag Bit 8: 0: No 1: Yes				0
Set/Reset the Interrupt Flag Bit 9: 0: No 1: Yes				1
Set/Reset the Interrupt Flag Bit 10: 0: No 1: Yes				2
Set/Reset the Interrupt Flag Bit 11: 0: No 1: Yes				3
Set/Reset the Interrupt Flag Bit 12: 0: No 1: Yes			4	
Set/Reset the Interrupt Flag Bit 13: 0: No 1: Yes			5	
Set/Reset the Interrupt Flag Bit 14: 0: No 1: Yes			6	
Set/Reset the Interrupt Flag Bit 15: 0: No 1: Yes			7	
NULL		BA98		
0001: Reset the Interrupt Flag Bits 8 to 15 0011: Set the Interrupt Flag Bits 8 to 15	FEDC			

Controlling the Test Operation	MINOR COMMAND			
Set/Reset the SUSPEND REQUEST Flag: 0: No 1: Yes				0
Set/Reset the END REQUEST Flag: 0: No 1: Yes				1
Set/Reset the RESUME REQUEST Flag: 0: No 1: Yes				2
Set/Reset the BEGIN REQUEST Flag: 0: No 1: Yes				3
NULL			4	
Enable start of an EXECUTE Command: 0: No 1: Yes			5	
Canceling a SCAN or EXECUTE Command: 0: No 1: Yes			6	
Enable start of a STATE, SCAN or EXECUTE Command: 0: No 1: Yes			7	
NULL		BA98		
0100: Bits 0 to 3 reset the Flags 0101: Bits 0 to 3 set the Flags	FEDC			

Read status, load counter, and count	MINOR COMMAND			
Load COUNTER1 UPDATE into COUNTER1: 0: No 1: Yes				0
Count COUNTER1: 0: No 1: Yes				1
Load COUNTER1 into the CAPTURE status register: 0: No 1: Yes				2
NULL				3
Load COUNTER2 UPDATE0 into COUNTER20: 0: No 1: Yes			4	
Load COUNTER2 UPDATE1 into COUNTER21: 0: No 1: Yes			5	
Refresh STATUS0, STATUS1 and STATUS2 registers: 0: No 1: Yes			6	
NULL		BA98	7	
0110: Read status, load counter, and count	FEDC			

Software RESET	MINOR COMMAND			
Reset complete 'ACT8990: 0: No 1: Yes				0
Reset interrupt flags: 0: No 1: Yes				1
Reset READ BUFFER and WRITE BUFFER: 0: No 1: Yes				2
Reset REQUEST flags: 0: No 1: Yes				3
NULL		BA98	7654	
0111: Reset the whole component	FEDC			

MAJOR COMMAND

STATE Command	MAJOR COMMAND			
New state after executing the command: 000, 001: TEST-LOGIC-RESET 010, 011: RUN-TEST/IDLE 100: SHIFT-DR 101: SHIFT-IR 110: PAUSE-DR 111: PAUSE-IR				210
NULL		BA98	7654	3
0001: STATE Command	FEDC			

The STATE Command is used to set the test board into a new state. There is no other activity during that command:

⇒ The event recognition, and all counters, are ignored.

⇒ COUNTER1 is ignored and does not count, even if the RUN-TEST/IDLE state occurs.

⇒ The Data Buffer is ignored and remains unchanged, even if a SHIFT-DR state or a SHIFT-IR state occurs.

⇒ The SHIFTER-FIFO is ignored and remains unchanged, even if a SHIFT-DR state or a SHIFT-IR state occurs.

⇒ The Execution of the command can not be suspended.

⇒ To reach the new state, only the temporary states SELECT-DR-SCAN, CAPTURE-DR, EXIT1-DR, UPDATE-DR, SELECT-IR-SCAN, CAPTURE-IR, EXIT1-IR, UPDATE-IR are used. The sole exception is the start of the TEST-LOGIC-RESET state. In this case, first change to RUN-TEST/IDLE state. Only then will this procedure lead exclusively via temporary states to the new state.

EXECUTE Command	MAJOR COMMAND			
New state after execution of command: 000, 001: TEST-LOGIC-RESET 010, 011: RUN-TEST/IDLE 100: SHIFT-DR 101: SHIFT-IR 110: PAUSE-DR 111: PAUSE-IR				210
States for execution and suspension of commands: 0: Execution of command in RUN-TEST/IDLE state and suspension in PAUSE-DR state. 1: Execution of command in RUN-TEST/IDLE and suspension of the execution of command in PAUSE-IR state.				3
Suspension or Cancellation of Commands: 00, 01: During zero passing of COUNTER1 the execution of command is suspended or cancelled. 10: The execution of the command is finished after the EXECUTE state has been passed through once. COUNTER1 is not used. 11: The execution of the Command is cancelled after the EXECUTE state has been passed once. COUNTER1 is not used.			54	
Starting and executing the commands: 00, 01: The selected new state is controlled directly. This case corresponds to the STATE COMMAND, but can not be controlled by events. 10: The execution of the command is suspended immediately after the start (SLEEP STATE) 11: The execution of command begins with it's start			76	
SUSPEND and RESUME Flag: 0: If the execution of command is suspended, then the SUSPEND flag is resetted, if the execution of the command is resumed, then the RESUME flag is Reset. 1: If the execution of the command is suspended or resumed, then the SUSPEND flag as well as the RESUME flag are reset.		8		
NULL		BA9		
0010: EXECUTE command	FEDC			

Using the EXECUTE Command a defined number of RUN-TEST/IDLE cycles can be executed. The execution of this command can be suspended and resumed.

SCAN Command	MAJOR COMMAND			
New state after executing the command: 000, 001: TEST-LOGIC-RESET 010, 011: RUN-TEST/IDLE 100: SHIFT-DR 101: SHIFT-IR 110: PAUSE-DR 111: PAUSE-IR				210
States for execution and suspension of commands: 0: Execution of command in SHIFT-DR state and suspension of execution of command in PAUSE-DR state. 1: Execution of command in SHIFT-IR state and suspension of execution of command in PAUSE-IR state.				3
Suspension and cancellation of commands: 00, 01: During zero passing of COUNTER1 the execution of command is suspended or cancelled. 10: The execution of the command is finished, after the EXECUTE state has been passed through once. COUNTER1 is not used. 11: The execution of the command is cancelled, after the EXECUTE state has been passed through once. COUNTER1 is not used.			54	
NULL		BA98	76	
0011: SCAN Command	FEDC			

Using the SCAN command, a defined number of SHIFT-DR cycles or SHIFT-IR cycles can be executed. The execution of this command can be suspended and resumed.

COUNTER1 UPDATE0

COUNTER1 Lower Value Word	COUNTER1 UPDATE0			
The lower value 16-Bit word of COUNTER1 can be loaded with the contents of this register.	FEDC	BA98	7654	3210

COUNTER1 UPDATE1

COUNTER1 Higher Value Word	COUNTER1 UPDATE1			
The higher value 16-Bit word of COUNTER1 can be loaded with the contents of this register.	FEDC	BA98	7654	3210

The 32-Bit COUNTER1 is used for counting the numbers of shift operations with the SCAN Command in the SHIFT-DR state and the SHIFT-IR state. Additionally, it is used for counting the RUN TEST/IDLE Cycles during an EXECUTE Command. COUNTER1 is also able to count events.

COUNTER2 UPDATE0

COUNTER2 Lower Value Word or COUNTER20	COUNTER2 UPDATE0			
The lower value 16-Bit Word of COUNTER2 or of COUNTER20 can be loaded with the contents of this register.	FEDC	BA98	7654	3210

COUNTER2 UPDATE1

COUNTER2 Higher Value Word or COUNTER20	COUNTER2 UPDATE1			
The higher value 16-Bit word of COUNTER2 or of COUNTER21 can be loaded with the contents of this register.	FEDC	BA98	7654	3210

Both Counters COUNTER20 and COUNTER21 can be used as two independent 16-Bit Counters (Figure 15, Page 19), two combined 16-Bit Counters (Figure 16, Page 19) or as one 32-Bit Counter COUNTER2 (Figure 17, Page 20). In both cases they can be used for counting events which occur at the EVENT0 to EVENT3 Inputs.

STATUS0

Internal States	STATUS0			
An event occurred at the EVENT DETECTOR0: 0: No 1: Yes				0
An event occurred at the EVENT DETECTOR1: 0: No 1: Yes				1
An event occurred at the EVENT DETECTOR2: 0: No 1: Yes				2
An event occurred at the EVENT DETECTOR3: 0: No 1: Yes				3
Zero passing of COUNTER1: 0: No 1: Yes			4	
Zero passing of COUNTER1 during a SCAN command: 0: No 1: Yes			5	
NULL			76	
The execution of an EXECUTE command is suspended: 0: No 1: Yes		8		
The execution of an EXECUTE Command is finished: 0: No 1: Yes		9		
The execution of an EXECUTE Command is resumed: 0: No 1: Yes		A		
The execution of an EXECUTE Command is started: 0: No 1: Yes		B		
A failure occurred at the READ BUFFER or at the WRITE BUFFER: 0: No 1: Yes	C			
The WRITE BUFFER is empty and the READ BUFFER is full: 0: No 1: Yes Only buffers in use are addressed.	D			
The execution of a command is cancelled: 0: No 1: Yes	E			
The execution of a command is finished: 0: No 1: Yes	F			

STATUS1

External Signals and Requests	STATUS1			
The Input signal EVENT0 is an 0: 0-Level 1: 1-Level				0
The Input signal EVENT1 is an 0: 0-Level 1: 1-Level				1
The Input signal EVENT2 is an 0: 0-Level 1: 1-Level				2
The Input signal EVENT3 is an 0: 0-Level 1: 1-Level				3
NULL			7654	
A request for suspending an EXECUTE Command exists: 0: No 1: Yes		8		
A request for finishing an EXECUTE Command exists: 0: No 1: Yes		9		
A request for resuming an EXECUTE Command exists: 0: No 1: Yes		A		
A request for starting an EXECUTE Command exists: 0: No 1: Yes		B		
NULL	DC			
State of execution of a command: 0: The execution of an active command is suspended or no command is in process. 1: An active command is in process or the process is finishing.	E			
A command is active: 0: No 1: Yes	F			

STATUS2

State of the Test Board and of the buffers	STATUS2			
Actual state of Test Board: 0000, 0001: TEST-LOGIC-RESET 0010, 0011: RUN-TEST/IDLE 0100: SHIFT-DR 0101: SHIFT-IR 0110: PAUSE-DR 0111: PAUSE-IR 1000: Temporary state				3210
NULL			7654	
WRITE BUFFER State: 00: both WRITE BUFFERs are empty 01: not used 10: one WRITE BUFFER is empty, one is full 11: both WRITE BUFFERs are full		98		
NULL		BA		
READ BUFFER State: 00: both READ BUFFERs are empty 01: one READ BUFFER is empty, one is full 10: not used 11: both READ BUFFERs are full	DC			
NULL	FE			

STATUS3

Not Used	STATUS3			
NULL	FEDC	BA98	7654	3210

CAPTURE0

COUNTER1 Low Value Word	CAPTURE0			
This register can be loaded with the lower value 16-Bit Word of COUNTER1 and afterwards it can be read.	FEDC	BA98	7654	3210

CAPTURE1

COUNTER1 High Value Word	CAPTURE1			
This register can be loaded with the higher value 16-Bit Word of COUNTER1 and afterwards it can be read.	FEDC	BA98	7654	3210

The actual value of COUNTER1 can be loaded into the registers CAPTURE0 and CAPTURE1.

READ BUFFER

Input Buffer for TDI	READ BUFFER			
16-Bit Receive Data	FEDC	BA98	7654	3210

WRITE BUFFER

Output Buffer for TDO	WRITE BUFFER			
16-Bit Transmission Data	FEDC	BA98	7654	3210

3.2. Typical Programming Procedure

Using a test board according to IEEE 1149.1, the commands are typically first sent to the test components with IR-SCAN, and the necessary test vectors communicated with subsequent DR-SCANs. Simultaneously, a DR-SCAN reads the system answer from the previous test vector. For a test of the connecting lines the following procedure is typical:

1) IR-SCAN: command EXTEST for all test components

2) DR-SCAN: shift first test vector in all test components

3) DR-SCAN: read system reply to the first test vector and shift second test vector to all test components

4) DR-SCAN: read system reply to the second test vector and shift third test vector to all test components.

5) etc.

An IR-SCAN approximates in programming to a DR-SCAN. Only bit 3 in the MAJOR COMMAND register needs to be programmed to one for an IR-SCAN, or to zero for a DR-SCAN. In order to implement an IR-SCAN or DR-SCAN under the control of the TEST BUS CONTROLLER, the following programming steps are necessary:

A) First, the TEST BUS CONTROLLER must be configured using the ten CONTROL registers. If this has already been done, then only possible necessary changes to the configuration need to be made.

B) The COUNTER1 UPDATE register needs to be programmed according to the length of the shift register.

C) Using the MAJOR COMMAND, the IR-SCAN or DR-SCAN are selected, and subsequently the MINOR COMMAND starts the execution of the commands.

D.1) The STATUS register now allow the control of the status of the WRITE BUFFER. If one of the two buffer is empty, then 16 command bits or 16 bits of the test vector must be written into the WRITE BUFFER. These bits are shifted from the TEST BUS CONTROLLER over the TDO line into the test board. Simultaneously, the TEST BUS CONTROLLER collects the bits from the TDI input into the SHIFTER-FIFO. During this shifting operation, bits inevitably fall out at the other end of the shift register; these are read into the READ BUFFER via the READ TRANSFER FUNCTION.

D.2) If the STATUS register shows that data is in the READ BUFFER is available, then this data must be read from the READ BUFFER.

D.3) Steps D.1 and D.2 should be continued until all bits in the test board have been written. The last time round usually less than 16 bits need to be shifted. Unneeded bits of the WRITE BUFFER will be ignored by the TEST BUS CONTROLLER; excess bits from the READ BUFFER are filled by the TEST BUS CONTROLLER with zeroes.

E) The last data bits read from the TDI input are now in the SHIFTER-FIFO, and must be transferred into the READ BUFFER in order to proceed from there to the control computer.

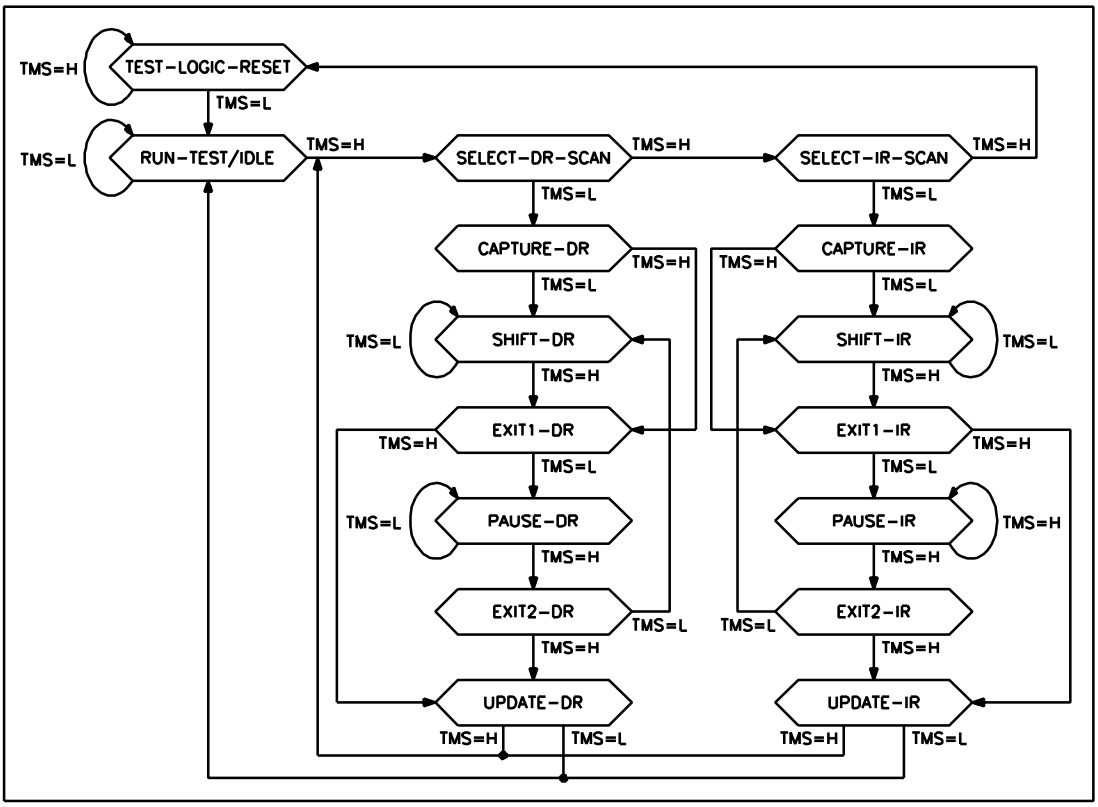


Figure 21: IEEE 1149.1 TAP (TEST ACCESS PORT) Status Diagram

Since the TEST BUS CONTROLLER continually sends the clock signal TCK to the test board, the walk through the status diagram (Figure 21) can only be halted in one of the six static states:

TEST-LOGIC-RESET

RUN-TEST/IDLE

SHIFT-DR

PAUSE-DR

SHIFT-IR

PAUSE-IR

If an interrupt in the shift cycle is necessary (e.g. because the control computer has not yet written the next data in the WRITE BUFFER, or because both READ BUFFERs are full), then the TEST BUS CONTROLLER

changes to the PAUSE-DR or PAUSE-IR state. It waits there until the shift cycle can again proceed. The state in which the test board should wait after executing a command is chosen with the MAJOR COMMAND. In most cases the RUN-TEST/IDLE state will be used. If the PAUSE-DR or PAUSE-IR states are chosen, account should then be taken of the fact that, after ending a command, the UPDATE-DR or UPDATE-IR phases would not proceed.

4. Examples

The programming of the TEST BUS CONTROLLER SN74BCT8244N will now be demonstrated with worked examples. All examples assume the use of the same test board with two SN74BCT8244N SCOPE™ (SYSTEM CONTROLLABILITY OBSERVABILITY PARTITIONING ENVIRONMENT) OCTALs.

4.1. Example Test Board

All examples in this chapter are based on the same test board. In order to simplify an understanding of the examples, this test board (Figure 22) consists of two SN74BCT8244N SCOPE™ OCTALs from Texas Instruments. The programming methods with more complex test boards are, however, basically the same.

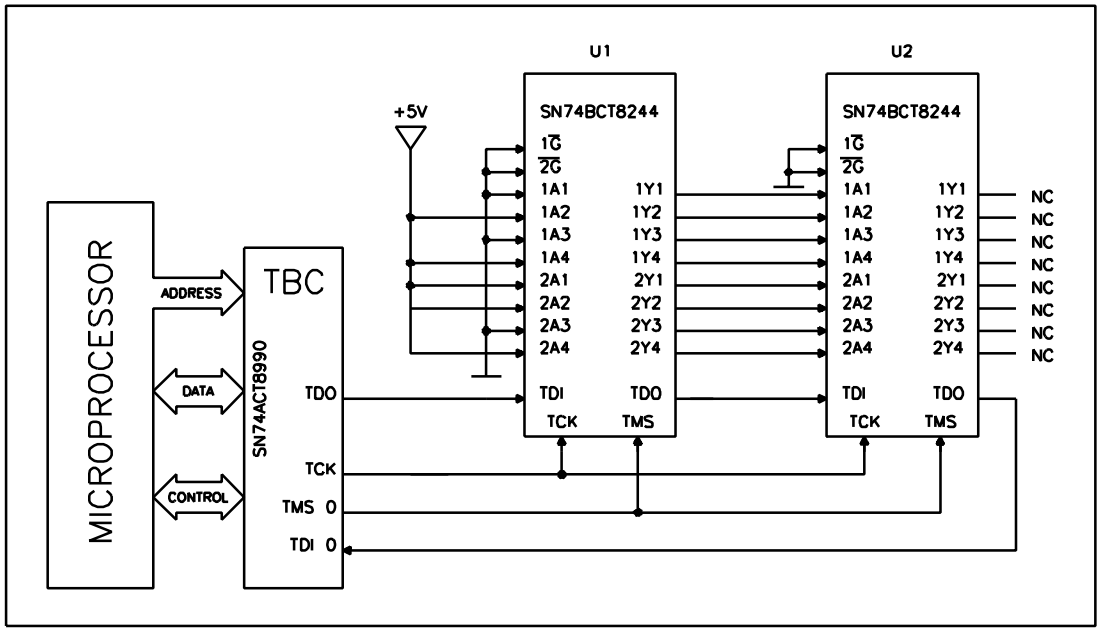


Figure 22: Hardware Example: Two SCOPE™ OCTALs SN74BCT8244N

The shift registers of the test boards are in the following states:

SHIFT-IR State ⇒ 16 Bits IR (INSTRUCTION REGISTER),

SHIFT-DR State ⇒ 36 BSCs (BOUNDARY SCAN CELL),

⇒ 4 Bit BCR (BOUNDARY CONTROL REGISTER),

⇒ 2 Bit BYPASS Register.

4.2. Software Examples

The first examples show the programming of three basic functions:

⇒ Load commands in the Test Board,

⇒ Load test vectors in the Test Board,

⇒ Copy SHIFTER-FIFO in the READ BUFFER .

This next example is essentially combinations of three basic functions:

⇒ EXTEST

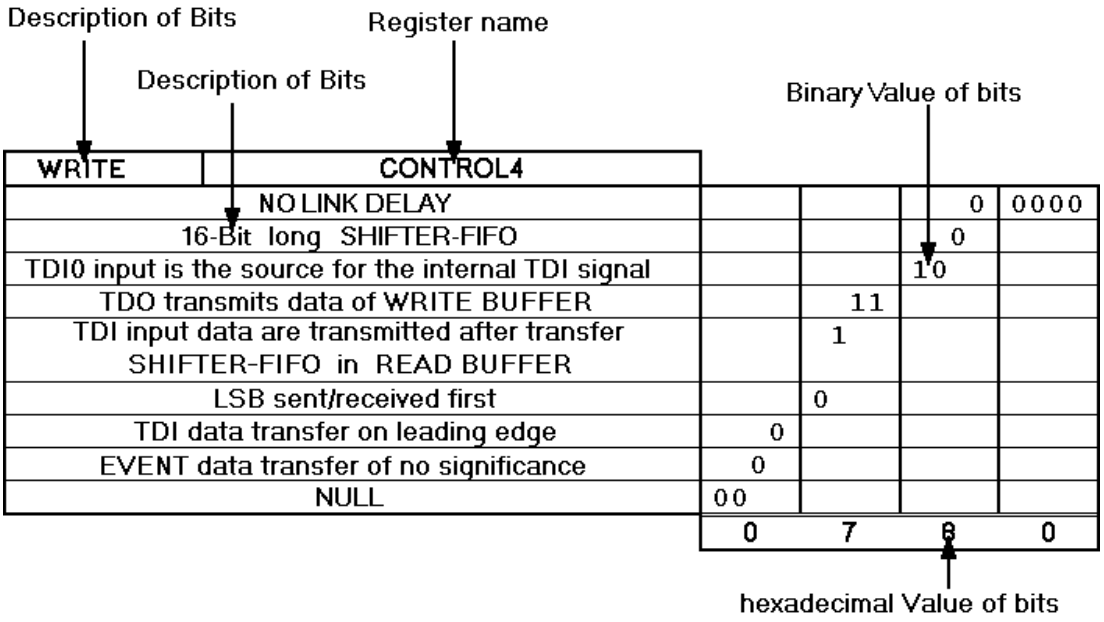
Lastly, the PRPG/PSA (PSEUDO RANDOM PATTERN GENERATION / PARALLEL SIGNATURE ANALYSIS) will be demonstrated with examples, with and without event control:

⇒ PRPG/PSA without event control

⇒ Event controlled PRPG/PSA

Each software example commences with a table, giving an overview of the basic functions which are used.

The programming of the registers is explained in detail in tables. The tables show the register name and the direction of data flow i.e. whether the register is being written into or read. In addition descriptions of the individual bits are included and the resulting bit combinations, whether binary or hexadecimal.



4.2.1. Load commands in test board

Nr.	Commands for both SCOPE™ OCTALs	MAJOR COMMAND		Description
		Command execution	Final state	
1	each 8-Bits READBT	IR-SCAN	RUN-TEST/IDLE	Load READBT command in test board

At the beginning of a test, the TBC should always be reset.

Write	MINOR COMMAND				
Complete reset					0001
NULL			0000	0000	
MINOR COMMAND Type: Software RESET		0111			
		7	0	0	1

After reset of the whole TBC, it should be configured with the 10 control registers, as follows:

⇒ TDI0 as test data input,

⇒ TMS0 as TEST MODE SELECT output,

⇒ No interrupt transferred to the control computer,

⇒ The TMS2..5/EVENT0..3 inputs/outputs are made TMS outputs but not used, so that the event logic remains unchanged.

Write	CONTROL0				
No interrupt at event					0000
No interrupt with COUNTER1 Flags				00	
NULL				00	
No interrupt from command output flags			0000		
No interrupt from change of a buffer state		00			
No interrupt on ending command		00			
		0	**0**	**0**	**0**

Write	CONTROL1				
No production test operation					0000
NULL				0000	
No production test operation		0	0000		
NULL		000			
		0	**0**	**0**	**0**

Write	CONTROL2				
Switch on TMS0, switch off TMS1..5				00	0001
NULL		0000	0000	00	
		0	**0**	**0**	**1**

Write	CONTROL3				
TMS2..5/EVENT0..3 are TMS outputs (here without significance)					1111
Test pulse output TCK0 is switched on				0	
Test data output TDO is switched on				0	
all TMS outputs are switched on				0	
all EVENT inputs/outputs are switched off				1	
Operation to IEEE 1149.1			0000		
TCKO is connected with the internal TCK source		00			
NULL		00			
		0	**0**	**8**	**F**

Write	CONTROL4				
No LINK DELAY				0	0000
16-Bit long SHIFTER-FIFO				0	
TDI0 input is the source for the internal TDI signal				10	
TDO transmits data of WRITE BUFFER			11		
TDI input data are transmitted into the READ BUFFER after passing through the SHIFTER-FIFO			1		
The LSB is transmitted or received first			0		
TDI Data transfer at the leading edge		0			
EVENT Data transfer without significance		0			
NULL		00			
		0	**7**	**8**	**0**

Write	CONTROL5				
The EVENT output levels are here without significance					0000
NULL				0000	
The EVENT Data direction is here without significance			0000		
NULL		0000			
		0	**0**	**0**	**0**

Write	CONTROL6				
EVENT0 sets no Flags					0000
No recognition of EVENT0				0000	
EVENT1 sets no flags			0000		
No recognition of EVENT1		0000			
		0	**0**	**0**	**0**

Write	CONTROL7				
EVENT2 sets no Flags					0000
No recognition of EVENT2				0000	
EVENT3 sets no Flags			0000		
No recognition of EVENT3		0000			
		0	**0**	**0**	**0**

Write	CONTROL8				
COUNTER1 does not set the SUSPEND REQUEST Flag					0
COUNTER1 does not set the END REQUEST Flag					0
COUNTER1 is loaded at zero passing by the COUNTER1 UPDATE register					1
COUNTER1 does not count at events or EXECUTE commands				00	0
COUNTER1 does not respond to an event				00	
NULL		0000	0000		
		0	**0**	**0**	**4**

Write	CONTROL9				
COUNTER2 is not used		0000	0000	0000	0000
		0	**0**	**0**	**0**

After configuration of the TEST BUS CONTROLLER with the ten control registers, the full length of the shift register should be loaded into COUNTER1.

Write	COUNTER1 UPDATE0				
Command registers of two SCOPE OCTALs are together 16 bits long ⇒ COUNTER1 lower value bits: 16-1=15		0000	0000	0000	1111
		0	**0**	**0**	**F**

Write	COUNTER1 UPDATE1			
COUNTER1 higher value bits are 0	0000	0000	0000	0000
	0	**0**	**0**	**0**

Write	MINOR COMMAND			
Load COUNTER1 from COUNTER1 UPDATE register				001
NULL				0
No operation with COUNTER2 or Status			000	
NULL		0000	0	
MINOR COMMAND Type: Read status, load counter and count	0110			
	6	**0**	**0**	**1**

With the MAJOR COMMAND, presets are chosen and the MINOR COMMAND then starts the shift operation.

Write	MAJOR COMMAND			
State after completing the command: RUN-TEST/IDLE				011
Execute command in SHIFT-IR and put command in PAUSE-IR state				1
Terminate command at zero passing of COUNTER1			00	
NULL		0000	00	
MAJOR COMMAND Type: SCAN command	0011			
	3	**0**	**0**	**B**

Write	MINOR COMMAND			
All Flags are reset				0000
NULL			0	
Start of a SCAN command			100	
NULL		0000		
MINOR COMMAND Type: Control of the test procedure	0101			
	5	**0**	**8**	**0**

The TEST BUS CONTROLLER is now ready to send commands to the test board via TDO, and to receive the status of the test board via TDI.

Before the WRITE BUFFER is written, the status of the WRITE BUFFER must be interrogated.

Write	MINOR COMMAND				
Load COUNTER1 in the CAPTURE register					100
NULL					0
refresh all 3 STATUS registers				100	
NULL			0000	0	
MINOR COMMAND Type: Read status, load counter and count		0110			
		6	**0**	**4**	**4**

The STATUS2 register now contains the following bit combination.

Read	STATUS2				
test board is in PAUSE-IR state					0111
NULL				0000	
both WRITE BUFFER are empty			00		
NULL			00		
both READ BUFFER are empty		00			
NULL		00			
		0	**0**	**0**	**7**

Write	WRITE BUFFER				
READBT command for SCOPE™ OCTAL U2				1000	1011
READBT command for SCOPE™ OCTAL U1		1000	1011		
		8	**B**	**8**	**B**

As soon as the WRITE BUFFER has been loaded, the TEST BUS CONTROLLER begins to shift the data. Since the 16 bits of the WRITE BUFFER exactly match the length of the shift register, the WRITE BUFFER needs no further data. The next step is the reading of the READ BUFFER. For this, the status must first be interrogated, and only when the READ BUFFER is filled with data can it be read out.

Write	MINOR COMMAND				
Load COUNTER1 in the CAPTURE register					100
NULL					0
Refresh all 3 STATUS registers				100	
NULL			0000	0	
MINOR COMMAND type: Read status, load counter and count		0110			
		6	**0**	**4**	**4**

The READ BUFFER can be read out, if the STATUS3 register contains the following bit combination:

Read	STATUS2				
Test board is in RUN-TEST/IDLE state					0011
NULL				0000	
both WRITE BUFFER are empty			00		
NULL			00		
one READ BUFFER contains Data		01			
NULL		00			
		1	**0**	**0**	**3**

The following value can now be read out from the READ BUFFER:

Read	READ BUFFER				
1's of SHIFTER-FIFOs after reset		1111	1111	1111	1111
		F	**F**	**F**	**F**

The status information of the 2 SCOPE™ OCTALs is now in the SHIFTER-FIFO. If now a test vector is shifted into the test board, the next 16 bits of the READ BUFFER will contain the status information of the 2 SCOPE™ OCTALs. If it is required to receive the status information immediately, the contents of the SHIFTER-FIFO must be copied into the READ BUFFER. An example of this is in chapter 4.2.3.

4.2.2. Load test vector in test board

Nr.	Data for both SCOPE™ OCTALs	MAJOR COMMAND		Description
		Command execution	Final state	
1	Each 18 bits test vector	DR-SCAN	RUN-TEST/IDLE	Load test vector

Loading a test vector into the test board differs from loading commands in the following respects:

⇒ the length of the shift register is 36 bits,

⇒ the DR-SCAN must be chosen to shift the test vector.

The control registers only need to be programmed, if they differ from a previous program. Since the DR-SCAN usually follows an IR-SCAN, the control registers do not need to be programmed again. A detailed description of the control register programming is to be found in chapter 4.2.1.

Write	**CONTROL0**	**0**	**0**	**0**	**0**
Write	**CONTROL1**	**0**	**0**	**0**	**0**
Write	**CONTROL2**	**0**	**0**	**0**	**1**
Write	**CONTROL3**	**0**	**0**	**8**	**F**
Write	**CONTROL4**	**0**	**7**	**8**	**0**
Write	**CONTROL5**	**0**	**0**	**0**	**0**
Write	**CONTROL6**	**0**	**0**	**0**	**0**
Write	**CONTROL7**	**0**	**0**	**0**	**0**
Write	**CONTROL8**	**0**	**0**	**0**	**4**
Write	**CONTROL9**	**0**	**0**	**0**	**0**

Now COUNTER1 needs to be programmed to the length of the shift register: 36 - 1 = 35 = 0x23

Write	**COUNTER1 UPDATE0**	**0**	**0**	**2**	**3**
Write	**COUNTER1 UPDATE1**	**0**	**0**	**0**	**0**
Write	**MINOR COMMAND**	**6**	**0**	**0**	**1**

With the MAJOR COMMAND, the DR-SCAN is chosen here. The MINOR COMMAND then starts the shifting operation again.

Write	**MAJOR COMMAND**				
State after executing command RUN-TEST/IDLE					011
Execution of command in SHIFT-DR state and suspension of command in PAUSE-DR state					0
Terminate command at zero passing of COUNTER1				00	
NULL			0000	00	
MAJOR COMMAND type: SCAN command		0011			
		3	**0**	**0**	**3**

Write	**MINOR COMMAND**				
All Flags are reset.					0000
NULL				0	
Start of a SCAN command				100	
NULL			0000		
MINOR COMMAND type: Control of test process		0101			
		5	**0**	**8**	**0**

Now the WRITE BUFFER can again be fed with data, and the READ BUFFER read out. Reading out STATUS2 register again ensures that the buffer is ready.

Write	MINOR COMMAND	6	0	4	4

Read	STATUS2				
Test board is in PAUSE-DR state					0110
NULL				0000	
both WRITE BUFFER are empty			00		
NULL			00		
both READ BUFFER are empty		00			
NULL		00			
		0	0	0	6

If both WRITE BUFFERs are empty, two times 16 bits can be immediately entered.

Write	WRITE BUFFER				
1Y1..1Y4,2Y1..2Y4 Test sample of the SCOPE™ OCTAL U2				0010	0001
1A1..1A4,2A1..2A4 Test sample of the SCOPE™ OCTAL U2		0100	0011		
		4	3	2	1

Write	WRITE BUFFER				
1G, 2G Test sample of the SCOPE™ OCTAL U2					00
1Y1..1Y4,2Y1..2Y4 Test sample of the SCOPE™ OCTAL U1			10	0111	10
1A1..1A4,2A1..2A2 Test sample of the SCOPE™ OCTAL U1		0101	01		
		5	6	7	8

If the TEST BUS CONTROLLER has shifted out the 32 bits of the WRITE BUFFER via TDO, it reads also 32 bits at the TDI input into the SHIFTER-FIFO, and from there into the READ BUFFER. In order to be able to shift further data into the test board, at least one 16 bit word from the READ BUFFER must next be read. Via the STATUS2 register must also be checked, whether data is already available.

Write	MINOR COMMAND	6	0	4	4

Read	STATUS2				
Momentary state of the test board					????
NULL				0000	
Momentary state of WRITE BUFFER			??		
NULL			00		
At least one register of the READ BUFFER is full		?1			
NULL		00			
		3/1	?	0	?

If this DR-SCAN example occurred before the example from chapter 4.2.1., the following values can be read from the READ BUFFER:

Read	READ BUFFER				
Status information of SCOPE™ OCTAL U2				1000	0001
Status information of SCOPE™ OCTAL U1		1000	0001		
		8	1	8	1

If the READ BUFFER has been read, then at least 16 bits were shifted through the test board, and as a result at least one register of the WRITE BUFFER must be empty. To be safe, the status can also be ascertained before addressing the WRITE BUFFER. Since the shift register is 36 bits long and 32 bits have already been written into the WRITE BUFFER, only 4 bits from the test vector are missing. The remaining 12 bits of the WRITE BUFFER can be occupied with any desired words, since they will anyway not be shifted out into the test board.

Write	WRITE BUFFER				
2A3..2A4 Test sample of the SCOPE™ OCTAL U1					11
1G, 2G Test sample of the SCOPE™ OCTAL U1					00
bits not used anymore		0000	0000	0000	
		0	0	0	3

Finally, the last data from the READ BUFFER must be read out. Status interrogation again ensures, that data is available.

Write	MINOR COMMAND	6	0	4	4
Read	STATUS2	3/1	?	0	?
Read	READ BUFFER				
1Y1..1Y4,2Y1..2Y4 Test sample of the SCOPE™ OCTAL U2				????	????
1A1..1A4,2A1..2A4 Test sample of the SCOPE™ OCTAL U2		????	????		
		?	?	?	?

Because only 4 bits have been shifted at the last shift operation, only four bits arrived from the SHIFTER-FIFO to the READ BUFFER.

Write	MINOR COMMAND	6	0	4	4
Read	STATUS2	1	0	0	?
Read	READ BUFFER				
1G, 2G Test sample of the SCOPE™ OCTAL U2					??
1Y1..1Y2 Test sample of the SCOPE™ OCTAL U1					??
NULL		0000	0000	0000	
		0	0	0	?

The following 16 Bit are still in the SHIFTER-FIFO:

⇒ 6 Bit: 1Y3..1Y4,2Y1..2Y4 Test sample of the SCOPE™ OCTAL U1

⇒ 8 Bit: 1A1..1Y4,2Y1..2Y4 Test sample of the SCOPE™ OCTAL U1

⇒ 2 Bit: 1G, 2G Test sample of the SCOPE™ OCTAL U1

The example of chapter 4.2.3. shows how these 16 bits can be copied into the READ BUFFER.

4.2.3. Copy SHIFTER-FIFO into the READ BUFFER

Nr.	Commands and Data for both SCOPE™ OCTALs	MAJOR COMMAND		Description
		Command execution	Final state	
1	-	-	-	Read SHIFTER-FIFO

In order to copy the contents of the SHIFTER-FIFO into the READ BUFFER, the component must be configured in such a way, that the output data of the SHIFTER-FIFO serves as the source for the input data of the SHIFTER-FIFO. Therefore the FIFO must be connected as a circular memory. Then the SHIFTER-FIFO will be shifted - according to the program - 16 or 32

times. Because each bit, which leaves the SHIFTER-FIFO is copied into the READ BUFFER, the READ BUFFER will contain a copy of the SHIFTER-FIFO after this operation.

Write	CONTROL4				
No LINK DELAY				0	0000
16-bit long SHIFTER-FIFO				0	
Internal TDO is source for the internal TDI signal				00	
TDO transmits data of the SHIFTER-FIFOs			10		
TDI input data are transmitted after copying the SHIFTER-FIFO into the READ BUFFER			1		
The LSB is transmitted or received first			0		
TDI data taking over at leading edge		0			
EVENT data taking over here without significance		0			
NULL		00			
		0	**6**	**0**	**0**

Write	COUNTER1 UPDATE0	0	0	0	F
Write	COUNTER1 UPDATE1	0	0	0	0
Write	MINOR COMMAND	6	0	0	1

Write	MINOR COMMAND				
All Flags are reset.					0000
NULL				0	
Start of a SCAN command				100	
NULL			0000		
MINOR COMMAND type: Control of test process		0101			
		5	**0**	**8**	**0**

Now the contents of SHIFTER-FIFO can be removed from the READ BUFFER. If this is done as the example in chapter 4.2.2, the READ BUFFER will contain the following bit sample:

Read	READ BUFFER				
1Y3..1Y4,2Y1..2Y4 Test sample of SCOPE™ OCTAL U1				??	????
1A1..1A4,2A1..2A4 Test sample of SCOPE™ OCTAL U1		??	????	??	
1G, 2G Test sample of SCOPE™ OCTAL U1		??			
		?	**?**	**?**	**?**

With this operation the contents of the SHIFTER-FIFO remains unchanged, and also the state of the test board remains completely uninfluenced.

4.2.4. EXTEST

With the 3 basic functions for the preceding chapter, an EXTEST for the test board used can very easily be made. Following shifting actions are necessary for this:

Nr.	Commands and data for both SCOPE™ OCTALs	MAJOR COMMAND		Description
		Command execution	Final state	
1	each 8 bits EXTEST	IR-SCAN	RUN-TEST/IDLE	Load command in test board
2	each 18 bits test vector	DR-SCAN	RUN-TEST/IDLE	Load 1st test vector
3	each 18 bits test vector	DR-SCAN	RUN-TEST/IDLE	Read system reply to 1st test vector and load 2nd test vector
4	-	-	-	Read SHIFTER-FIFO

The following tables show the test vectors used, and the system reply to this vector which can be expected.

SCOPE™ OCTAL U1			**SCOPE™ OCTAL U2**		
1G, 2G	1A1..1A4, 2A1..2A4	1Y1..1Y4, 2Y1..2Y4	1G, 2G	1A1..1A4, 2A1..2A4	1Y1..1Y4, 2Y1..2Y4
00	11010101	10011110	00	01000011	00100001
00	01011101	11010101	00	10011110	01000011

The necessary program steps for this have already been explained in detail in previous examples, and are summarized here:

First the TEST BUS CONTROLLER is reset and subsequently configured with the help of the control register.

Write	**MINOR COMMAND**	**7**	**0**	**0**	**1**

1.) Load command in test board

Write	**CONTROL0**	**0**	**0**	**0**	**0**
Write	**CONTROL1**	**0**	**0**	**0**	**0**
Write	**CONTROL2**	**0**	**0**	**0**	**1**
Write	**CONTROL3**	**0**	**0**	**8**	**F**
Write	**CONTROL4**	**0**	**7**	**8**	**0**
Write	**CONTROL5**	**0**	**0**	**0**	**0**
Write	**CONTROL6**	**0**	**0**	**0**	**0**
Write	**CONTROL7**	**0**	**0**	**0**	**0**
Write	**CONTROL8**	**0**	**0**	**0**	**4**
Write	**CONTROL9**	**0**	**0**	**0**	**0**

Now both SCOPE™ OCTALs are loaded with the command EXTEST. The value which will be read from the READ BUFFER, will be the 1's from the reset SHIFTER-FIFO.

Write	**COUNTER1 UPDATE0**	**0**	**0**	**0**	**F**
Write	**COUNTER1 UPDATE1**	**0**	**0**	**0**	**0**
Write	**MINOR COMMAND**	**6**	**0**	**0**	**1**

Write	**MAJOR COMMAND**	**3**	**0**	**0**	**B**
Write	**MINOR COMMAND**	**5**	**0**	**8**	**0**

Write	**MINOR COMMAND**	**6**	**0**	**4**	**4**
Read	**STATUS2**	**0**	**0**	**0**	**7**
Write	**WRITE BUFFER**	**0**	**0**	**0**	**0**

Write	**MINOR COMMAND**	**6**	**0**	**4**	**4**
Read	**STATUS2**	**1**	**0**	**0**	**3**
Read	**READ BUFFER**	**F**	**F**	**F**	**F**

2.) Load first test vector

Now the first test vector is shifted into the 2 SCOPE™ OCTALs. The first 16 bit word of the READ BUFFER contains the status information, which the SCOPE™ OCTALs produce on writing in a command. Subsequently, bitmaps are read, which were already present in the SCOPE™ OCTALs before the beginning of the test.

Write	**COUNTER1 UPDATE0**	**0**	**0**	**2**	**3**
Write	**COUNTER1 UPDATE1**	**0**	**0**	**0**	**0**
Write	**MINOR COMMAND**	**6**	**0**	**0**	**1**

Write	**MAJOR COMMAND**	**3**	**0**	**0**	**3**
Write	**MINOR COMMAND**	**5**	**0**	**8**	**0**

Write	**MINOR COMMAND**	**6**	**0**	**4**	**4**
Read	**STATUS2**	**0**	**0**	**0**	**6**
Write	**WRITE BUFFER**	**4**	**3**	**2**	**1**
Write	**WRITE BUFFER**	**5**	**6**	**7**	**8**

Write	**MINOR COMMAND**	**6**	**0**	**4**	**4**
Read	**STATUS2**	**3/1**	**?**	**0**	**?**
Read	**READ BUFFER**				
Status information of the SCOPE™ OCTALs U2				1000	0001
Status information of the SCOPE™ OCTALs U1		1000	0001		
		8	**1**	**8**	**1**

Write	**WRITE BUFFER**	**0**	**0**	**0**	**3**

Write	**MINOR COMMAND**	**6**	**0**	**4**	**4**
Read	**STATUS2**	**3/1**	**?**	**0**	**?**
Read	**READ BUFFER**				
1Y1..1Y4,2Y1..2Y4 Test sample of the SCOPE™ OCTAL U2				????	????
1A1..1A4,2A1..2A4 Test sample of the SCOPE™ OCTAL U2		????	????		
		?	**?**	**?**	**?**

Write	**MINOR COMMAND**	**6**	**0**	**4**	**4**
Read	**STATUS2**	**1**	**0**	**0**	**3**
Read	**READ BUFFER**				
1G, 2G Test sample of the SCOPE™ OCTAL U2					??
1Y1..1Y2 Test sample of the SCOPE™ OCTAL U1					??
NULL		0000	0000	0000	
		0	**0**	**0**	**?**

3.) Read system reply to first test vector and load second test vector

The next test vector is now shifted into the SCOPE™ OCTALs. From the READ BUFFER, first the rest of the previous bitmap of the SCOPE™ OCTALs can be read and then the system reply of the SCOPE™ OCTALs to the first test vector.

Write	**MINOR COMMAND**	**6**	**0**	**0**	**1**
Write	**MAJOR COMMAND**	**3**	**0**	**0**	**3**
Write	**MINOR COMMAND**	**5**	**0**	**8**	**0**

Write	MINOR COMMAND	6	0	4	4
Read	STATUS2	0	0	0	6
Write	WRITE BUFFER	4	3	2	1
Write	WRITE BUFFER	5	6	7	8

Write	MINOR COMMAND	6	0	4	4
Read	STATUS2	3/1	?	0	?
Read	READ BUFFER				
1Y3..1Y4,2Y1..2Y4 Test sample of the SCOPE™ OCTAL U1				??	????
1A1..1Y4,2Y1..2Y4 Test sample of the SCOPE™ OCTAL U1		??	????	??	
2 Bit: 1G, 2G Test sample of the SCOPE™ OCTAL U1		??			
		?	?	?	?

Write	WRITE BUFFER	0	0	0	3

Write	MINOR COMMAND	6	0	4	4
Read	STATUS2	3/1	?	0	?
Read	READ BUFFER				
1Y1..1Y4,2Y1..2Y4 Test sample of the SCOPE™ OCTAL U2				0100	0011
1A1..1A4,2A1..2A4 Test sample of the SCOPE™ OCTAL U2		1001	1110		
		9	E	4	3

Write	MINOR COMMAND	6	0	4	4
Read	STATUS2	1	0	0	3
Read	READ BUFFER				
1G, 2G Test sample of the SCOPE™ OCTAL U2					00
1Y1..1Y2 Test sample of the SCOPE™ OCTAL U1					01
NULL		0000	0000	0000	
		0	0	0	4

4.) Read SHIFTER FIFO

The missing 16 bits of the system reply are still in the SHIFTER-FIFO. To receive these bits, the contents of the SHIFTER-FIFO must now be copied into the READ BUFFER.

Write	**CONTROL4**	**0**	**6**	**0**	**0**
Write	**COUNTER1 UPDATE0**	**0**	**0**	**0**	**F**
Write	**COUNTER1 UPDATE1**	**0**	**0**	**0**	**0**
Write	**MINOR COMMAND**	**6**	**0**	**0**	**1**
Write	**MINOR COMMAND**	**5**	**0**	**8**	**0**
Write	**CONTROL4**	**0**	**7**	**8**	**0**
Read	**READ BUFFER**				
1Y3..1Y4, 2Y1..2Y4 Test sample of the SCOPE™ OCTAL U1				11	0101
1A1..1A4, 2A1..2A4 Test sample of the SCOPE™ OCTAL U1		01	0111	01	
1G, 2G Test sample of the SCOPE™ OCTAL U1		00			
		1	**7**	**7**	**5**

The EXTEST will now have been completed. The test vector was written with 3 shift operations into the test board. In the same way the reading of the system reply to the test vector of the READ BUFFER is made in three steps:

3.) 16Bit	**2.) 4 Bit**	**1.) 16 Bit**
0001011101110101	0100	1001111001000011

The result is coded into the two SCOPE™ OCTALs as follows:

SCOPEô OCTAL U1			**SCOPEô OCTAL U2**		
1G, 2G	1A1..1A4, 2A1..2A4	1Y1..1Y4, 2Y1..2Y4	1G, 2G	1A1..1A4, 2A1..2A4	1Y1..1Y4, 2Y1..2Y4
00	01011101	11010101	00	10011110	01000011

4.2.5. PRPG/PSA

This example shows how, with the 3 basic functions IR-SCAN, DR-SCAN and read SHIFTER-FIFO and an additional EXECUTE RUN-TEST/IDLE block a PRPG/PSA (PSEUDO RANDOM PATTERN GENERATION / PARALLEL SIGNATURE ANALYSIS) test can be built up. New in this example is the command EXECUTE from the TBC, which can generate a defined number of RUN-TEST/IDLE cycles.

Nr.	Command and Data for both SCOPE™ OCTALs	MAJOR COMMAND		Description
		Command execution	Final state	
1	each 8 bits SAMPLE	IR-SCAN	RUN-TEST/IDLE	Load initial value
2	each 18 bits test vector	DR-SCAN	RUN-TEST/IDLE	
3	each 8 bits SCANCT	IR-SCAN	RUN-TEST/IDLE	Load BOUNDARY CONTROL register with PRPG/PSA
4	each 2 bits PRPG/PSA	DR-SCAN	RUN-TEST/IDLE	
5	each 8 bits RUNT	IR-SCAN	PAUSE-IR	load RUNT command into common register
6		EXECUTE	PAUSE-IR	execute PRPG/PSA
7	each 8 bits READBT	IR-SCAN	RUN-TEST/IDLE	read result vector
8	each 18 bits next test vector	DR-SCAN	RUN-TEST/IDLE	
9	-	-	-	Load SHIFTER FIFO

The example hardware uses the bit combination 01011101 as the input signal for the component U1. The signature at the inputs of U1 is determined from the bit combination and the initial value used.

The signature at the inputs of U2 is based on the output signal from U1 and the initial value. Only the initial value is decisive for the bit sequence of the PRPG. Now the PRPG and the PSA of both SN74BCT8244N generate following bit sequences:

PRPG PSA Nr.	**SCOPE™ OCTAL U1**		**SCOPE™ OCTAL U2**	
	1A1..1A4, 2A1..2A4	1Y1..1Y4, 2Y1..2Y4	1A1..1A4, 2A1..2A4	1Y1..1Y4, 2Y1..2Y4
	11010101	10011110	01000011	00100001
1	10110111	11011111	10111111	00010000
2	10000110	01100111	00010000	10001000
3	00011110	10110011	11101111	01000100
4	10110010	11011001	01000100	10100010
5	00110100	11101100	01111011	11010001
6	01000111	01110110	11010001	11101000
7	01111110	10111011	10011110	01110100
8	11100010	11011101	01110100	10111010
9	00101100	11101110	01100111	01011101
10	11001011	01110111	01011101	10101110

The following Tables show the test vector used as an initial value and the expected system reply to this vector after 10 PRPG/PSA cycles:

SCOPE™ OCTAL U1			SCOPE™ OCTAL U2		
1G, 2G	1A1..1A4, 2A1..2A4	1Y1..1Y4, 2Y1..2Y4	1G, 2G	1A1..1A4, 2A1..2A4	1Y1..1Y4, 2Y1..2Y4
00	11010101	10011110	00	01000011	00100001
00	11001011	01110111	00	01011101	10101110

First the TEST BUS CONTROLLER must be reset.

Write	**MINOR COMMAND**	**7**	**0**	**0**	**1**

1.) Load initial values: IR-SCAN SAMPLE

Write	**CONTROL0**	**0**	**0**	**0**	**0**
Write	**CONTROL1**	**0**	**0**	**0**	**0**
Write	**CONTROL2**	**0**	**0**	**0**	**1**
Write	**CONTROL3**	**0**	**0**	**8**	**F**
Write	**CONTROL4**	**0**	**7**	**8**	**0**
Write	**CONTROL5**	**0**	**0**	**0**	**0**
Write	**CONTROL6**	**0**	**0**	**0**	**0**
Write	**CONTROL7**	**0**	**0**	**0**	**0**
Write	**CONTROL8**	**0**	**0**	**0**	**4**
Write	**CONTROL9**	**0**	**0**	**0**	**0**

Write	**COUNTER1 UPDATE0**	**0**	**0**	**0**	**F**
Write	**COUNTER1 UPDATE1**	**0**	**0**	**0**	**0**
Write	**MINOR COMMAND**	**6**	**0**	**0**	**1**

Write	**MAJOR COMMAND**	**3**	**0**	**0**	**B**
Write	**MINOR COMMAND**	**5**	**0**	**8**	**0**

Write	**MINOR COMMAND**	**6**	**0**	**4**	**4**
Read	**STATUS2**	**0**	**0**	**0**	**7**
Write	**WRITE BUFFER**				
SAMPLE command for SCOPE™ OCTAL U2				1000	1011
SAMPLE command for SCOPE™ OCTAL U1		1000	1011		
		8	**B**	**8**	**B**

Write	**MINOR COMMAND**	**6**	**0**	**4**	**4**
Read	**STATUS2**	**1**	**0**	**0**	**3**
Read	**READ BUFFER**	**F**	**F**	**F**	**F**

2.) Load initial values: DR-SCAN test vector

Write	**COUNTER1 UPDATE0**	**0**	**0**	**2**	**3**
Write	**COUNTER1 UPDATE1**	**0**	**0**	**0**	**0**
Write	**MINOR COMMAND**	**6**	**0**	**0**	**1**

Write	**MAJOR COMMAND**	**3**	**0**	**0**	**3**
Write	**MINOR COMMAND**	**5**	**0**	**8**	**0**

Write	**MINOR COMMAND**	**6**	**0**	**4**	**4**
Read	**STATUS2**	**0**	**0**	**0**	**6**
Write	**WRITE BUFFER**	**4**	**3**	**2**	**1**
Write	**WRITE BUFFER**	**5**	**6**	**7**	**8**

Write	**MINOR COMMAND**	**6**	**0**	**4**	**4**
Read	**STATUS2**	**3/1**	**?**	**0**	**?**
Read	**READ BUFFER**				
Status information of the SCOPE™ OCTALs U2				1000	0001
Status information of the SCOPE™ OCTALs U1		1000	0001		
		8	**1**	**8**	**1**

Write	**WRITE BUFFER**	**0**	**0**	**0**	**3**

Write	**MINOR COMMAND**	**6**	**0**	**4**	**4**
Read	**STATUS2**	**3/1**	**?**	**0**	**?**
Read	**READ BUFFER**				
1Y1..1Y4,2Y1..2Y4 Test result of the SCOPE™ OCTAL U2				????	????
1A1..1A4,2A1..2A4 Test result of the SCOPE™ OCTAL U2		????	????		
		?	**?**	**?**	**?**

Write	**MINOR COMMAND**	**6**	**0**	**4**	**4**
Read	**STATUS2**	**1**	**0**	**0**	**3**
Read	**READ BUFFER**				
1G, 2G Test result of the SCOPE™ OCTAL U2					??
1Y1..1Y2 Test result of the SCOPE™ OCTAL U1					??
NULL		0000	0000	0000	
		0	**0**	**0**	**?**

3.) Load BOUNDARY CONTROL register with PRPG/PSA: IR-SCAN SCANCT

Write	**COUNTER1 UPDATE0**	**0**	**0**	**0**	**F**
Write	**COUNTER1 UPDATE1**	**0**	**0**	**0**	**0**
Write	**MINOR COMMAND**	**6**	**0**	**0**	**1**

Write	MAJOR COMMAND	3	0	0	B
Write	MINOR COMMAND	5	0	8	0

Write	MINOR COMMAND	6	0	4	4
Read	STATUS2	0	0	0	7
Write	WRITE BUFFER				
SCANCT Command for SCOPE™ OCTAL U2				0000	1111
SCANCT Command for SCOPE™ OCTAL U1		0000	1111		
		0	F	0	F

Write	MINOR COMMAND	6	0	4	4
Read	STATUS2	1	0	0	3
Read	READ BUFFER				
1Y3..1Y4,2Y1..2Y4 Test result of the SCOPE™ OCTAL U1				??	????
1A1..1Y4,2Y1..2Y4 Test result of the SCOPE™ OCTAL U1		??	????	??	
2 Bit: 1G, 2G Test result of the SCOPE™ OCTAL U1		??			
		?	?	?	?

4.) Load BOUNDARY CONTROL register: DR-SCAN 8-Bit PRPG/PSA

Write	COUNTER1 UPDATE0	0	0	0	3
Write	COUNTER1 UPDATE1	0	0	0	0
Write	MINOR COMMAND	6	0	0	1

Write	MAJOR COMMAND	3	0	0	3
Write	MINOR COMMAND	5	0	8	0

Write	MINOR COMMAND	6	0	4	4
Read	STATUS2	0	0	0	6
Write	WRITE BUFFER				
8 Bit PRPG/PSA Command for SCOPE™ OCTAL U2					11
8 Bit PRPG/PSA Command for SCOPE™ OCTAL U1					11
NULL		0000	0000	0000	
		0	0	0	F

Write	MINOR COMMAND	6	0	4	4
Read	STATUS2	3/1	?	0	?
Read	READ BUFFER				
1st Byte, Status information of SCOPE™ OCTALs U2					0001
NULL		0000	0000	0000	
		0	0	0	1

5.) Load RUNT command in command register

Write	**COUNTER1 UPDATE0**	**0**	**0**	**0**	**F**
Write	**COUNTER1 UPDATE1**	**0**	**0**	**0**	**0**
Write	**MINOR COMMAND**	**6**	**0**	**0**	**1**

Write	**MAJOR COMMAND**				
State after command execution: PAUSE-IR					111
Execute command in SHIFT-IR and suspension of command execution in PAUSE-IR state					1
Terminate command at zero passing of COUNTER1				00	
NULL			0000	00	
MAJOR COMMAND type: SCAN command		0011			
		3	**0**	**0**	**F**
Write	**MINOR COMMAND**	**5**	**0**	**8**	**0**

Write	**MINOR COMMAND**	**6**	**0**	**4**	**4**
Read	**STATUS2**	**0**	**0**	**0**	**7**
Write	**WRITE BUFFER**				
RUNT command for SCOPE™ OCTAL U2				0000	1001
RUNT command for SCOPE™ OCTAL U1		0000	1001		
		0	**9**	**0**	**9**

Write	**MINOR COMMAND**	**6**	**0**	**4**	**4**
Read	**STATUS2**	**1**	**0**	**0**	**7**
Read	**READ BUFFER**				
2nd Byte, Status information of SCOPE™ OCTAL U2					1000
Status information of SCOPE™ OCTAL U1			1000	0001	
BOUNDARY CONTROL register of SCOPE™ OCTAL U2		??			
BOUNDARY CONTROL register of SCOPE™ OCTAL U1		??			
		?	**8**	**1**	**8**

6.) Implement PRPG/PSA

Before the PRPG/PSA implementation can begin, the configuration of the TEST BUS CONTROLLER must be changed. The COUNTER1 is now used in order to count the TCK edges during the RUN-TEST/IDLE states.

Write	CONTROL8				
COUNTER1 does not set the SUSPEND REQUEST Flag					0
COUNTER1 sets the END REQUEST Flag					1
COUNTER1 is loaded by the COUNTER1 UPDATE register at zero passing					1
COUNTER1 does not count at EXECUTE commands in RUN-TEST/IDLE state				01	0
COUNTER1 does not react on any event				00	
NULL		0000	0000		
		0	**0**	**1**	**6**

COUNTER1 is now loaded with the number of RUN-TEST/IDLE cycles. In this case 10 cycles should be made. The register is then to be programmed with the value (10 - 1) = 9.

Write	COUNTER1 UPDATE0	0	0	0	9
Write	COUNTER1 UPDATE1	0	0	0	0
Write	MINOR COMMAND	6	0	0	1

MINOR COMMAND and MAJOR COMMAND start the PRPG/PSA implementation.

Write	MAJOR COMMAND				
State after executing command: PAUSE-IR					111
Command execution in RUN-TEST/IDLE state and suspension of command in PAUSE-IR state					1
Terminate command at zero passing of COUNTER1				00	
The command implementation starts immediately.				11	
no significance here, since execution is not interrupted			0		
NULL			000		
MAJOR COMMAND type: EXECUTE command		0010			
		2	**0**	**C**	**F**

Write	MINOR COMMAND				
All Flags are reset.					1111
NULL				0	
immediate start of EXECUTE command				100	
NULL			0000		
MINOR COMMAND type: Control of test run		0100			
		4	**0**	**8**	**F**

Write	CONTROL8				
COUNTER1 does not set SUSPEND REQUEST Flag					0
COUNTER1 does not set END REQUEST Flag					0
COUNTER1 is loaded by COUNTER1 UPDATE register at zero passing					1
COUNTER1 does not count events or EXECUTE commands				00	0
COUNTER1 does not react to any event				00	
NULL		0000	0000		
		0	**0**	**0**	**4**

7.) Read result vector: IR-SCAN READBT

In order to read the result vectors, one of the two commands READBT or READBN must be used. Other commands, such as EXTEST or SAMPLE, were written over during the UPDATE-IR phase with the result of the PRPG/PSA.

Write	COUNTER1 UPDATE0	0	0	0	F
Write	COUNTER1 UPDATE1	0	0	0	0
Write	MINOR COMMAND	6	0	0	1

Write	MAJOR COMMAND	3	0	0	B
Write	MINOR COMMAND	5	0	8	0

Write	MINOR COMMAND	6	0	4	4
Read	STATUS2	0	0	0	7
Write	WRITE BUFFER				
READBT command for SCOPE™ OCTAL U2				1000	1011
READBT command for SCOPE™ OCTAL U1		1000	1011		
		8	**B**	**8**	**B**

Write	MINOR COMMAND	6	0	4	4
Read	STATUS2	1	0	0	3
Read	READ BUFFER				
Status information of SCOPE™ OCTAL U2				1000	0001
Status information of SCOPE™ OCTAL U1		1000	0001		
		8	**1**	**8**	**1**

8.) Read result vector: DR-SCAN

Write	COUNTER1 UPDATE0	0	0	2	3
Write	COUNTER1 UPDATE1	0	0	0	0
Write	MINOR COMMAND	6	0	0	1

Write	**MAJOR COMMAND**	**3**	**0**	**0**	**3**
Write	**MINOR COMMAND**	**5**	**0**	**8**	**0**

Write	**MINOR COMMAND**	**6**	**0**	**4**	**4**
Read	**STATUS2**	**0**	**0**	**0**	**6**
Write	**WRITE BUFFER**	**4**	**3**	**2**	**1**
Write	**WRITE BUFFER**	**5**	**6**	**7**	**8**

Write	**MINOR COMMAND**	**6**	**0**	**4**	**4**
Read	**STATUS2**	**3/1**	**?**	**0**	**?**
Read	**READ BUFFER**				
Status information of SCOPE™ OCTAL U2				1000	0001
Status information of SCOPE™ OCTAL U1		1000	0001		
		8	**1**	**8**	**1**

Write	**WRITE BUFFER**	**0**	**0**	**0**	**3**

Write	**MINOR COMMAND**	**6**	**0**	**4**	**4**
Read	**STATUS2**	**3/1**	**?**	**0**	**?**
Read	**READ BUFFER**				
1Y1..1Y4,2Y1..2Y4 Test result of the SCOPE™ OCTAL U2				1010	1110
1A1..1A4,2A1..2A4 Test result of the SCOPE™ OCTAL U2		0101	1101		
		5	**D**	**A**	**E**

Write	**MINOR COMMAND**	**6**	**0**	**4**	**4**
Read	**STATUS2**	**1**	**0**	**0**	**3**
Read	**READ BUFFER**				
1G, 2G Test result of the SCOPE™ OCTAL U2					00
1Y1..1Y2 Test result of the SCOPE™ OCTAL U1					11
NULL		0000	0000	0000	
		0	**0**	**0**	**C**

9.) Read SHIFTER FIFO

Write	**CONTROL4**	**0**	**6**	**0**	**0**
Write	**COUNTER1 UPDATE0**	**0**	**0**	**0**	**F**
Write	**COUNTER1 UPDATE1**	**0**	**0**	**0**	**0**
Write	**MINOR COMMAND**	**6**	**0**	**0**	**1**
Write	**MINOR COMMAND**	**5**	**0**	**8**	**0**
Write	**CONTROL4**	**0**	**7**	**8**	**0**
Read	**READ BUFFER**				
1Y3..1Y4, 2Y1..2Y4 Test result of the SCOPE™ OCTAL U1				01	1101
1A1..1A4, 2A1..2A4 Test result of the SCOPE™ OCTAL U1		11	0010	11	
1G, 2G Test result of the SCOPE™ OCTAL U1		00			
		3	**2**	**D**	**D**

The system reply to the test vector was read by the READ BUFFER in three steps:

3.) 16 Bit	2.) 4 Bit	1.) 16 Bit
0011001011011101	1100	0101110110101110

The result coded in the two SCOPE™ OCTALs appears as follows:

SCOPE™ OCTAL U1			SCOPE™ OCTAL U2		
1G, 2G	1A1..1A4, 2A1..2A4	1Y1..1Y4, 2Y1..2Y4	1G, 2G	1A1..1A4, 2A1..2A4	1Y1..1Y4, 2Y1..2Y4
00	11001011	01110111	00	01011101	10101110

4.2.6. Event controlled PRPG/PSA

This example is a further development of the PRPG/PSA example. In this case, however, the PRPG/PSA should only start when a rising edge has arrived five times at the event input EVENT2.

The programming of the TEST BUS CONTROLLER is as a result mostly identical with that of the PRPG/PSA example. Only the sixth step, the execution of the PRPG/PSA, is somewhat changed. Only this sixth step will therefore be explained in detail; all other steps should be taken from the PRPG/PSA example.

Nr.	Commands and data for both SCOPE™ OCTALs	MAJOR COMMAND		Description
		Command execution	Final state	
1	each 8 bits SAMPLE	IR-SCAN	RUN-TEST/IDLE	Load initial values
2	each 18 bits test vector	DR-SCAN	RUN-TEST/IDLE	
3	each 8 bits SCANCT	IR-SCAN	RUN-TEST/IDLE	Load BOUNDARY CONTROL register with PRPG/PSA
4	each 2 bits PRPG/PSA	DR-SCAN	RUN-TEST/IDLE	
5	each 8 bits RUNT	IR-SCAN	PAUSE-IR	Load RUNT command in command register
6		EXECUTE	PAUSE-IR	execute PRPG/PSA
7	each 8 its READBT	IR-SCAN	RUN-TEST/IDLE	Read result vector
8	each 8 bits next test vector	DR-SCAN	RUN-TEST/IDLE	
9	-	-	-	Read SHIFTER FIFO

6.) Implement PRPG/PSA

Before execution of the PRPG/PSA can begin, the configuration of the TEST BUS CONTROLLER must be changed. COUNTER1 is now used to count the TCK edges during the RUN-TEST/IDLE state. COUNTER20 should count the events occurring at the EVENT2 input.

Write	**CONTROL3**				
TMS2/EVENT0 is TMS output (here without significance)					0
TMS3/EVENT1 is EVENT input					1
TMS4 .. TMS5/EVENT2 .. EVENT3 are TMS outputs (here without significance)					00
Test pulse output TCK0 is switched on				0	
Test data output TDO is switched on				0	
All TMS outputs are switched on				0	
All EVENT outputs are switched off				1	
Operation to IEEE 1149.1			0000		
TCKO is connected with internal TCK source		00			
NULL		00			
		0	**0**	**8**	**3**

Write	CONTROL4				
No LINK DELAY				0	0000
16-Bit long SHIFTER-FIFO				0	
TDI0 input is source for the internal TDI signal				10	
TDO transmits data of the WRITE BUFFER			11		
TDI input data are transmitted into READ BUFFER after passing the SHIFTER-FIFO			1		
The LSB is transmitted/received first			0		
TDI data transfer at leading edge		0			
EVENT data transfer at trailing edge		0			
NULL		00			
		0	**7**	**8**	**0**

Write	CONTROL7				
EVENT2 sets the RESUME REQUEST Flag					0100
Asynchronous recognition of EVENT2				0	
EVENT2 recognition always occurs				0	
Event is detected at zero passing of COUNTER20				10	
EVENT3 sets no Flags			0000		
No recognition of EVENT3		0000			
		0	**0**	**8**	**4**

Write	CONTROL8				
COUNTER1 does not set the SUSPEND REQUEST Flag					0
COUNTER1 sets the END REQUEST Flag					1
COUNTER1 is loaded by the COUNTER1 UPDATE register at zero crossing					1
COUNTER1 counts the test edges during EXECUTE commands in RUN-TEST/IDLE state				01	0
Here without significance, because COUNTER1 does not respond to any EVENT input				00	
NULL		0000	0000		
		0	**0**	**1**	**6**

Write	CONTROL9				
COUNTER2 is loaded with an 0-Level at input EVENT0 (here without significance, since EVENT0 is switched as TMS2 output)					000
COUNTER20 is loaded by COUNTER2 UPDATE0 register					1
COUNTER20 is not loaded at zero passing				0	
COUNTER20 counts leading edges at the input EVENT2				101	
Loading of COUNTER2 takes place with an asynchronous signal at the event input			0		
COUNTER20 and COUNTER21 work as separate 16-bit counters			00		
COUNTER21 is not used		0000	0		
		0	**0**	**A**	**8**

COUNTER1 is now loaded with the number of RUN-TEST/IDLE cycles.

Write	**COUNTER1 UPDATE0**	**0**	**0**	**0**	**9**
Write	**COUNTER1 UPDATE1**	**0**	**0**	**0**	**0**
Write	**MINOR COMMAND**	**6**	**0**	**0**	**1**

COUNTER20 is now loaded with the number of events. For five events the value (5 - 1) = 4 is to be programmed.

Write	**COUNTER1 UPDATE0**	**0**	**0**	**0**	**4**
Write	**COUNTER1 UPDATE1**	**0**	**0**	**0**	**0**
Write	**MINOR COMMAND**	**6**	**0**	**3**	**0**

MINOR COMMAND and MAJOR COMMAND start the execution of the PRPG/PSA.

Write	**MAJOR COMMAND**				
State after executing the command: PAUSE-IR					111
Execution of command in RUN-TEST/IDLE state and suspension of the execution of command in PAUSE-IR state					1
Terminate command at zero passing of COUNTER1				00	
Execution of command is suspended right after the start (SLEEP STATE)				10	
here of no significance, since the process was not interrupted			0		
NULL			000		
MAJOR COMMAND type: EXECUTE command		0010			
		2	**0**	**8**	**F**

Write	**MINOR COMMAND**				
All Flags are reset					1111
NULL				0	
Immediate start of EXECUTE Command				100	
NULL			0000		
MINOR COMMAND type: Control of test run		0100			
		4	**0**	**8**	**F**

Now the TEST BUS CONTROLLER waits for five leading edges at the event input of EVENT2. Only then does PRPG/PSA execution begin. After PRPG/PSA has been completed, the CONTROL registers must be reset into the standard state.

Write	**CONTROL0**	**0**	**0**	**0**	**0**
Write	**CONTROL3**	**0**	**0**	**8**	**F**
Write	**CONTROL4**	**0**	**7**	**8**	**0**
Write	**CONTROL7**	**0**	**0**	**0**	**0**
Write	**CONTROL8**	**0**	**0**	**0**	**4**
Write	**CONTROL9**	**0**	**0**	**0**	**0**

The result can now be read out, as in example PRPG/PSA.

5. Summary

The JOINT TEST ACTION GROUP (JTAG) has developed a coordinated test concept based on BOUNDARY SCAN techniques, whose specification is embodied in the IEEE 1149.1 standard. This allows cost-effective testing of complex electronic systems.

When using BOUNDARY SCAN test methods according to IEEE 1149.1, it is however necessary to control this test bus with a computer. For this purpose Texas Instruments has made available a suitable peripheral component, the TEST BUS CONTROLLER SN74ACT8990. If the ASSET™ development system from Texas Instruments is used to develop a test program, then the computer program supplied undertakes the programming of the TEST BUS CONTROLLER.

BIST systems require direct programming of the TEST BUS CONTROLLER. Also in this case, the ASSET™ system and additional translation programs offer easy and flexible programming solutions.

If the register of the TEST BUS CONTROLLER needs, however, to be directly programmed, then a detailed understanding of this component and methods of programming it are necessary. The examples which have been given in this Application Report should have made it easier for the user to undertake the direct programming of the TEST BUS CONTROLLER.