Application Report
SCAA044 - August 2000

# Test-Bus Controller SN74ACT8990

*Peter Forstner* — *Standard Linear & Logic*

**ABSTRACT**

This application report describes IEEE Std 1149.1 test-bus controller (TBC) SN74ACT8990 from Texas Instruments (TI™). The first part explains the architecture and operation of the TBC; the second part uses examples to explain the programming procedure.

**Contents**

**1 Introduction . . . 3**
**2 Hardware . . . 5**
2.1 Processor Interface . . . 5
2.2 Test Bus Interface . . . 7
2.3 Architecture . . . 10
2.3.1 Sequence Block . . . 11
2.3.2 Serial Block . . . 13
2.3.3 Event Block . . . 14
2.3.4 Counter Block . . . 18
2.3.5 Command Block . . . 19
2.3.6 Host Block . . . 20
**3 Programming . . . 21**
3.1 Registers . . . 22
3.2 Typical Programming Procedure . . . 42
**4 Examples . . . 44**
4.1 Example Test Board . . . 44
4.2 Software Examples . . . 45
4.2.1 Load Commands in Test Board . . . 46
4.2.2 Load Test Vector in Test Board . . . 52
4.2.3 Copy SHIFTER-FIFO Into the Read Buffer . . . 56
4.2.4 EXTEST . . . 57
4.2.5 PRPG/PSA . . . 61
4.2.6 Event-Controlled PRPG/PSA . . . 72
4.2.7 Implement PRPG/PSA . . . 73
**5 Summary . . . 77**

## List of Figures


1 Test System With TBC SN74ACT8990 . . . 3
2 System With Built-In Self-Test (BIST) . . . 4
3 Connection of the TBC to a Microprocessor . . . 5
4 Timing Diagram of Processor Interface When Writing . . . 6
5 Timing Diagram of Processor Interface When Reading . . . 6
6 Connection of a IEEE Std 1149.1 Test Bus to the TBC . . . 7
7 Connection of Two IEEE Std 1149.1 Test Buses to the TBC . . . 8
8 Connection of Several IEEE Std 1149.1 Test Buses to the TBC, With Buffering of the Test Bus Signal . . . 9
9 Block Diagram of TBC SN74ACT8990 . . . 10
10 Sequence Block . . . 11
11 Link-Delay Logic . . . 12
12 Serial Block . . . 13
13 Inputs of the Event Block . . . 15
14 Outputs of the Event Block . . . 16
15 COUNTER20 and COUNTER21 Configured as Separate 16-Bit Counters . . . 17
16 COUNTER20 and COUNTER21 Configured as Combined 16-Bit Counters . . . 17
17 COUNTER20 and COUNTER21 Configured as One 32-Bit Counter (COUNTER2) . . . 18
18 Counter Block . . . 18
19 Command Block . . . 19
20 Host Block . . . 20
21 IEEE Std 1149.1 Test Access Port (TAP) Status Diagram . . . 44
22 Hardware Example: Two SCOPE Octals SN74BCT8244N . . . 45

# 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 [surface-mount device (SMD)], and the consequent use of double-sided circuit boards. However, increased component density on circuit boards presents challenges to testability because the number of necessary test vectors increases out of proportion with complexity. By using nail-bed adapters, it is possible to partition the system to be tested, thereby reducing 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. Therefore, the escalating problems of testability can be solved only with a completely new concept.

In 1985, leading electronic manufacturers founded the Joint Test Action Group (JTAG) to develop a new and cost-effective test concept. The result of this was IEEE Std 1149.1. 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 that is controlled via the test bus. In this way, with only four lines, the complete system can be partitioned and tested.

TI application report EB193 describes these test methods in detail, and presents IEEE Std 1149.1-compatible system controllability observability partitioning environment (SCOPE™) bus drivers from TI. Application report EB203 assumes an understanding of test methods compliant to IEEE Std 1149.1.

The control of an IEEE Std 1149.1-compatible test system usually is performed by a computer. TBC SN74ACT8990, which can be connected to a computer like a normal interface circuit, completely controls the IEEE Std 1149.1 test bus (see Figure 1). The computer first configures the TBC, then loads in parallel the test commands and test vectors. The TBC transfers these commands and vectors to the system, thereby generating the signal sequence required by IEEE Std 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.

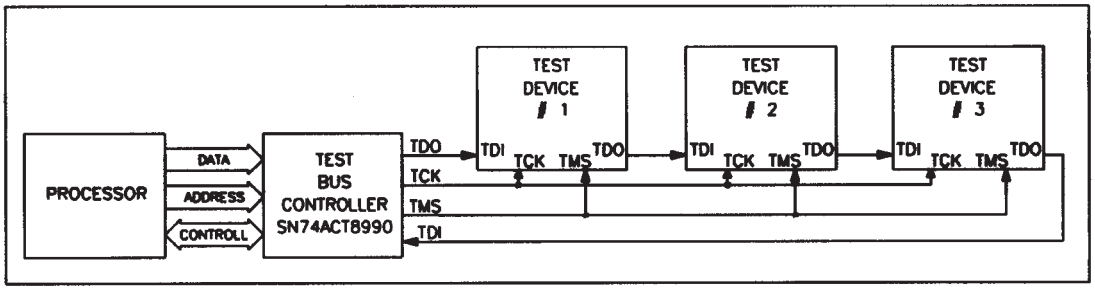



**Figure 1. Test System With TBC SN74ACT8990**

SCOPE is a trademark of Texas Instruments.

To reduce the applications work by the user needed to control IEEE Std 1149.1-compatible test systems, TI offers the computer program Advanced Support System for Emulation and Test (ASSET™) together with a plug-in computer board. TBC SN74ACT8990 is used on this board. ASSET allows easy development of test programs for which the user needs no understanding of the function of the TBC. First, a library with the IEEE Std 1149.1-compatible circuits that 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 also is suited ideally to supporting the circuit designer during the test phase.

If built-in self-test (BIST) is to be incorporated into an electronic system, TBC SN74ACT8990, in combination with the scan-path selector SN74ACT8999 and a microprocessor, provides an ideal basis (see Figure 2). The main computer can give the microprocessor the signal to start BIST via the IEEE Std 1149.1 test bus. The result of the self-test is communicated from the microprocessor to the main computer.

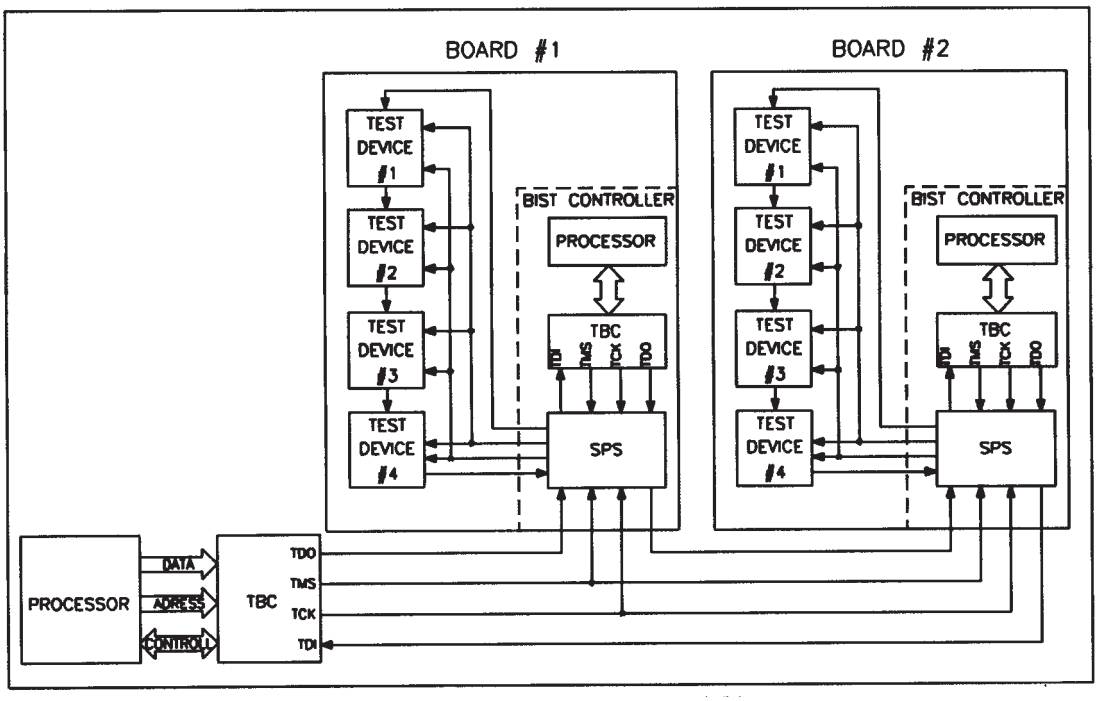



**Figure 2. System With Built-In Self-Test (BIST)**

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

This is one of the cases in which direct programming of the TBC is necessary, and in which this application report is intended to give assistance.

ASSET is a trademark of Texas Instruments.

# 2 Hardware

TBC SN74ACT8990 provides the interface between a processor and the IEEE Std 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 Processor Interface

The processor interface consists of the following:

- 5-bit address bus (A0–A4)
- 16-bit data bus (D0–D15)
- Read line ($\overline{RD}$)
- Write line ($\overline{WR}$)
- Status line ($\overline{RDY}$)
- Interrupt line ($\overline{INT}$)

The TBC is connected to the processor like a normal interface circuit, whereby the address decoder can be implemented as PAL (see Figure 3). Clock signal TCLK is buffered in the TBC, then transferred to the test system as the TCK signal, which can be derived from an independent clock generator or the processor clock, or be provided from the test system.

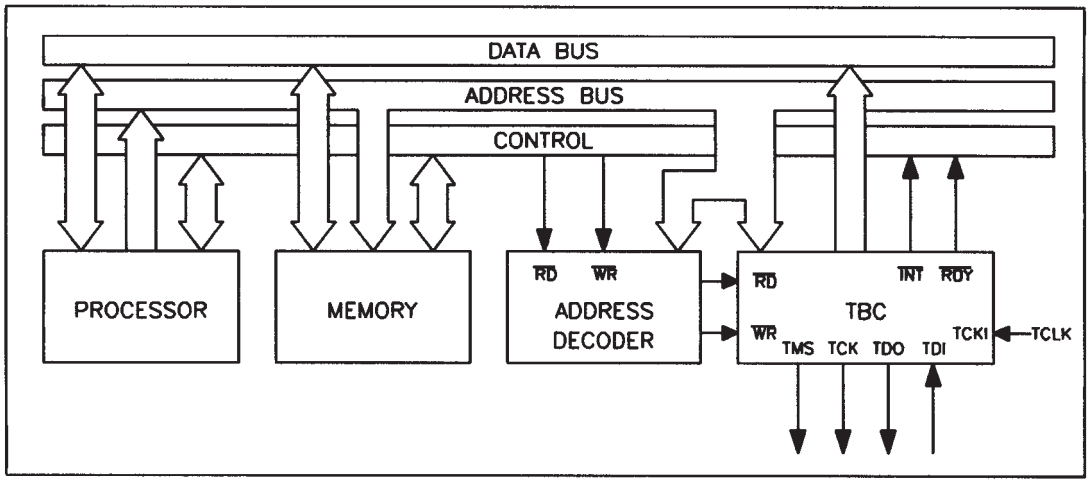



**Figure 3. Connection of the TBC to a Microprocessor**

Five address lines can be used to address 24 internal registers:

- Ten control registers
- Two command registers
- Six registers to control internal counters
- Four status registers
- One register for transmit data
- One register for receive data

Sixteen registers can be read and overwritten, seven are readable only, and one register can only be written to. The remaining eight potential registers are not occupied. Section 3, Programming, gives details of the registers. The timing of a write cycle is shown in Figure 4, and a read cycle is shown in Figure 5.

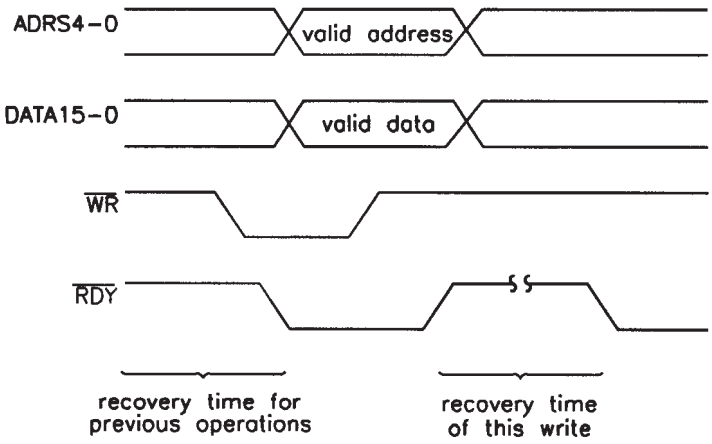



**Figure 4. Timing Diagram of Processor Interface When Writing**

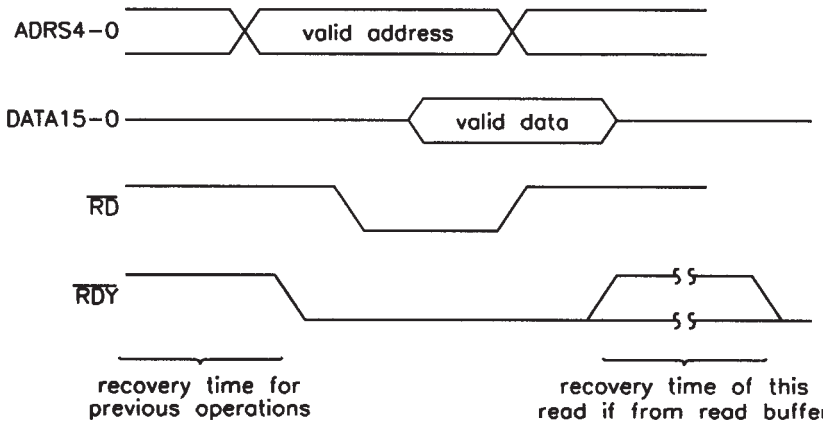



**Figure 5. Timing Diagram of Processor Interface When Reading**

## 2.2 Test Bus Interface

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

- One test pulse input (TCKI)
- One test pulse output (TCKO)
- One test data output (TDO)
- Two test data inputs (TDI0, TDI1)
- Two test mode select outputs (TMS0, TMS1)
- Four lines, which can be used as further TMS outputs (TMS2–TMS5), or can be programmed as event inputs/outputs (EVENT0–EVENT3)
- One line, test bus off ($\overline{\text{TOFF}}$), to reset all test outputs to a high-resistance state
- One input, test bus reset ($\overline{\text{TRST}}$), to reset the TBC and all test circuits. The use of this input is optional because the TBC and the test systems have a software reset.

Figure 6 shows the connection of an IEEE Std 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. Because signals TMS2–TMS5 use the same lines as event inputs/outputs, the number of available event inputs/outputs is reduced if more than two test systems are connected to the TBC (see Figure 7).

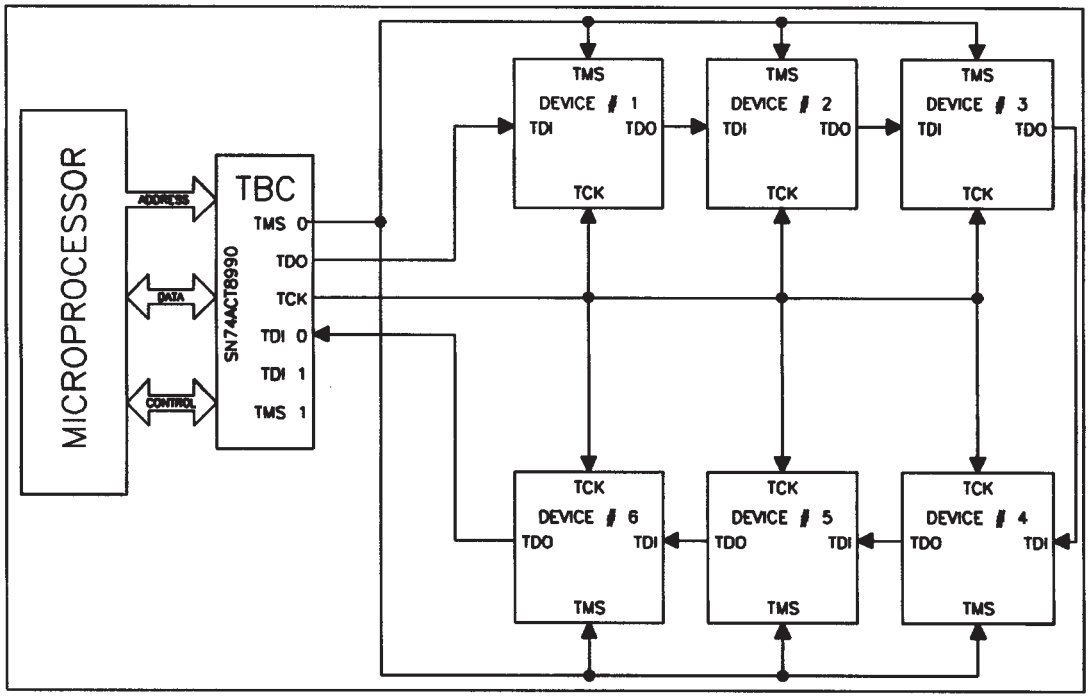



**Figure 6. Connection of a IEEE Std 1149.1 Test Bus to the TBC**

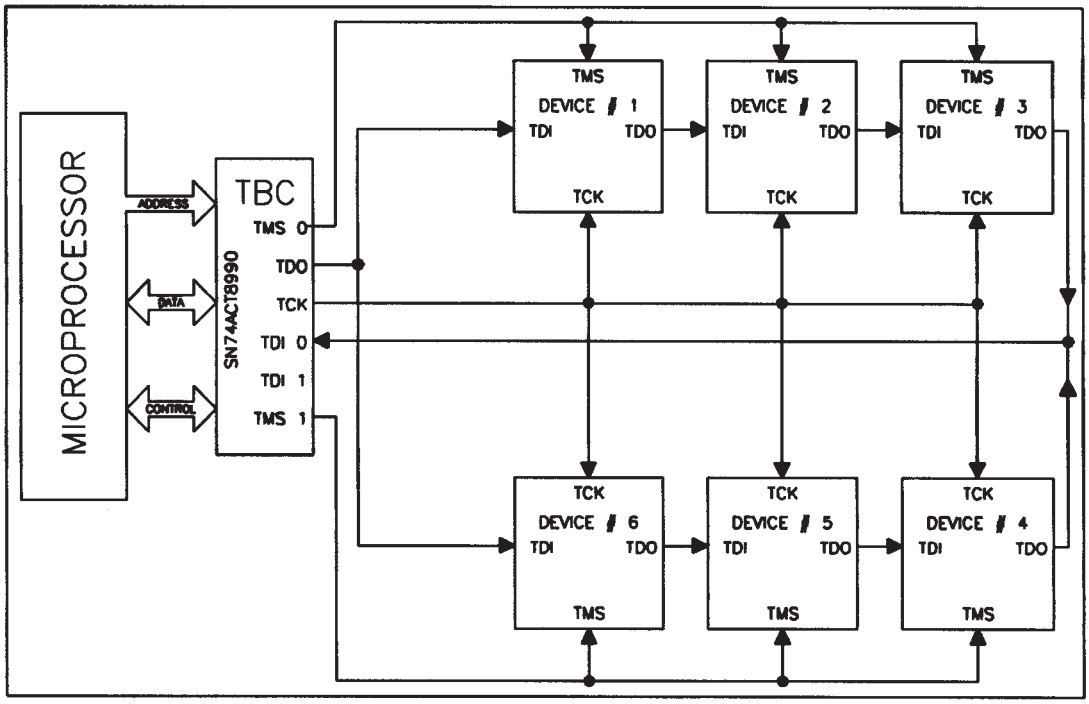



**Figure 7. Connection of Two IEEE Std 1149.1 Test Buses to the TBC**

The TDO of several test systems can be connected in parallel and to a TDI input of the TBCs because 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 (see Figure 8), the TDO output of the test system can no longer be in the high-impedance state. For this eventuality, two test data inputs (TDI0, TDI1) are available.

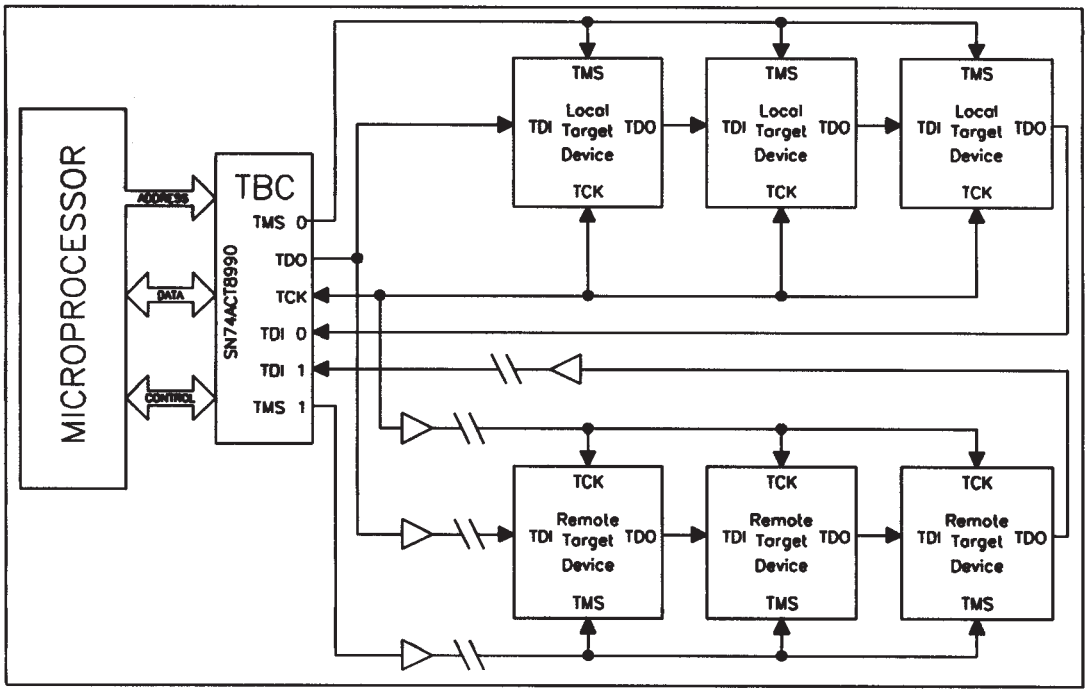



**Figure 8. Connection of Several IEEE Std 1149.1 Test Buses to the TBC, With Buffering of the Test Bus Signal**

## 2.3 Architecture

Figure 9 shows the block diagram of the TBC, consisting of six functional blocks:

- Sequence block
- Serial block
- Event block
- Counter block
- Command block
- Host block

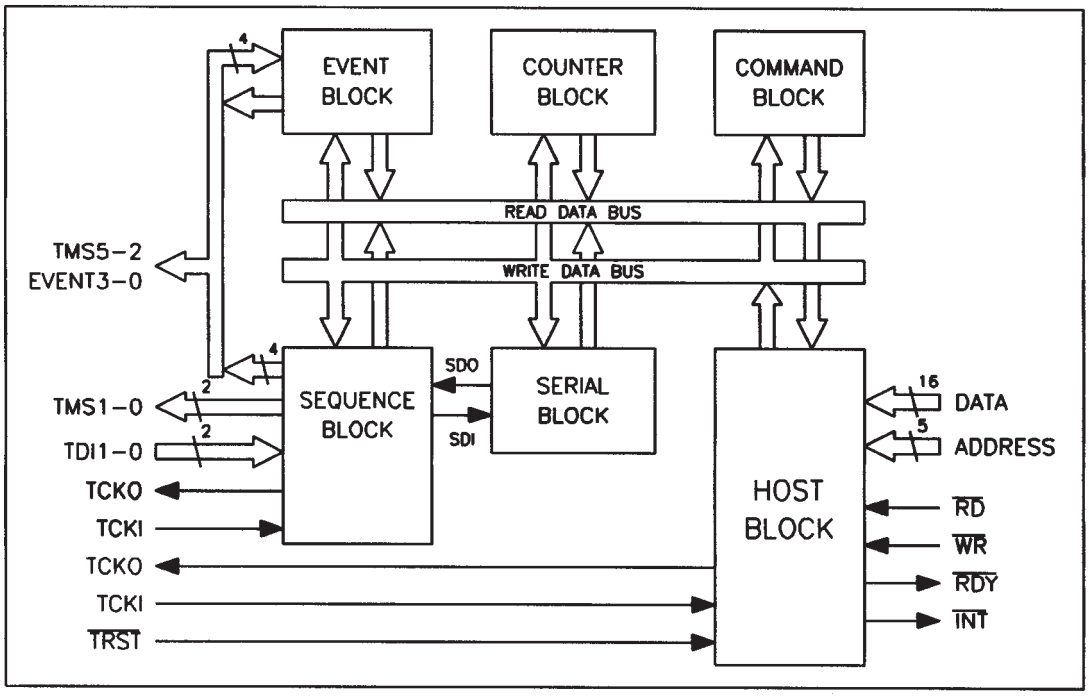



**Figure 9. Block Diagram of TBC SN74ACT8990**

### 2.3.1 Sequence Block

The sequence block (see Figure 10) traces the test system through the status diagram (see Figure 21). 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 Std 1149.1. During the SHIFT-DR and SHIFT-IR states, it uses the serial data out (SDO) data from the serial block to generate the TDO signal. Additionally, the sequence block transfers the TDI received data over the serial data in (SDI) line to the serial block.

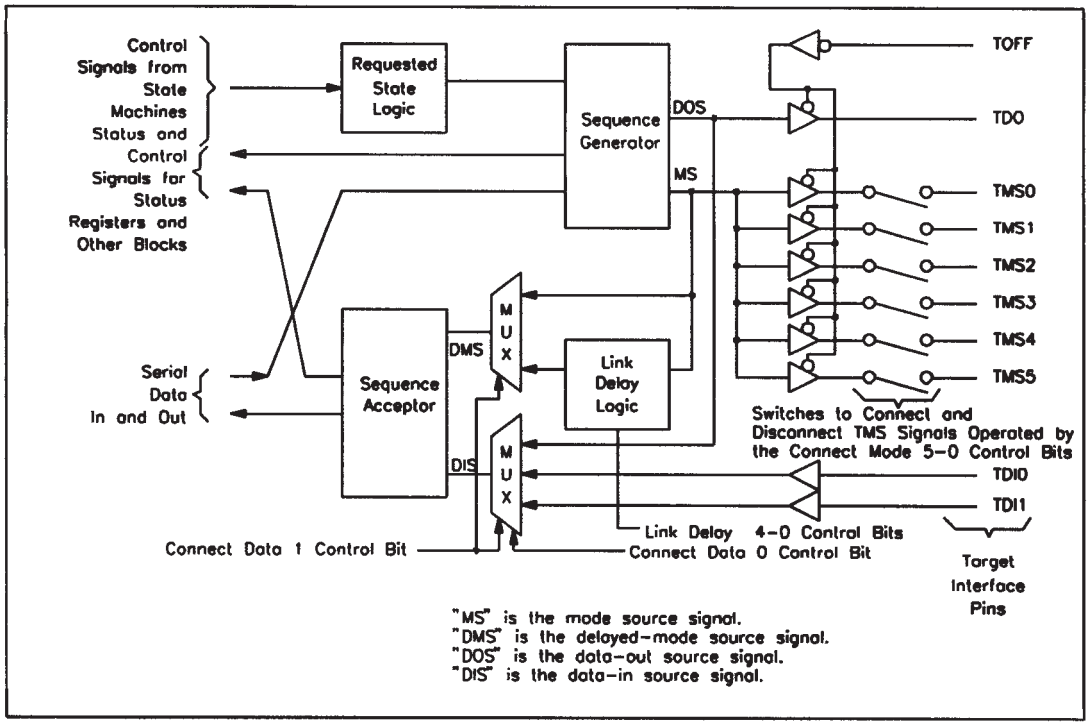



**Figure 10. Sequence Block**

These programming possibilities are important for the user:

- 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)

The link-delay logic allows a simple way to account for flip-flop chains that are switched in the signals TDO, TDI, and TMS between the TBC and the test board (see Figure 11). These flip-flops can be used, for example, to synchronize signals coming from long lines from the TBC SN74ACT8990 to test boards.

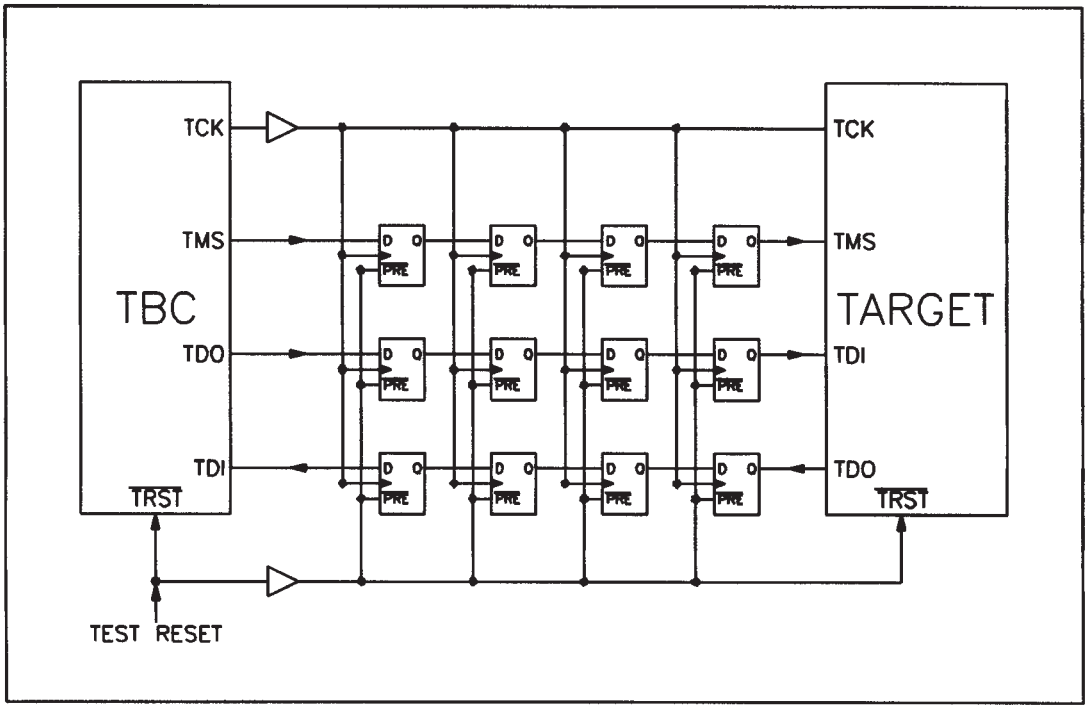



**Figure 11. Link-Delay Logic**

### 2.3.2 Serial Block

An understanding of the operation of the serial block is an important requirement for the application of the TBC (see Figure 12). The serial block supplies the commands and data that are sent to the IEEE Std 1149.1-compatible test boards in the SHIFT-IR and SHIFT-DR states. In the same way, it collects the bits that are read back from the TDO output of the test board.

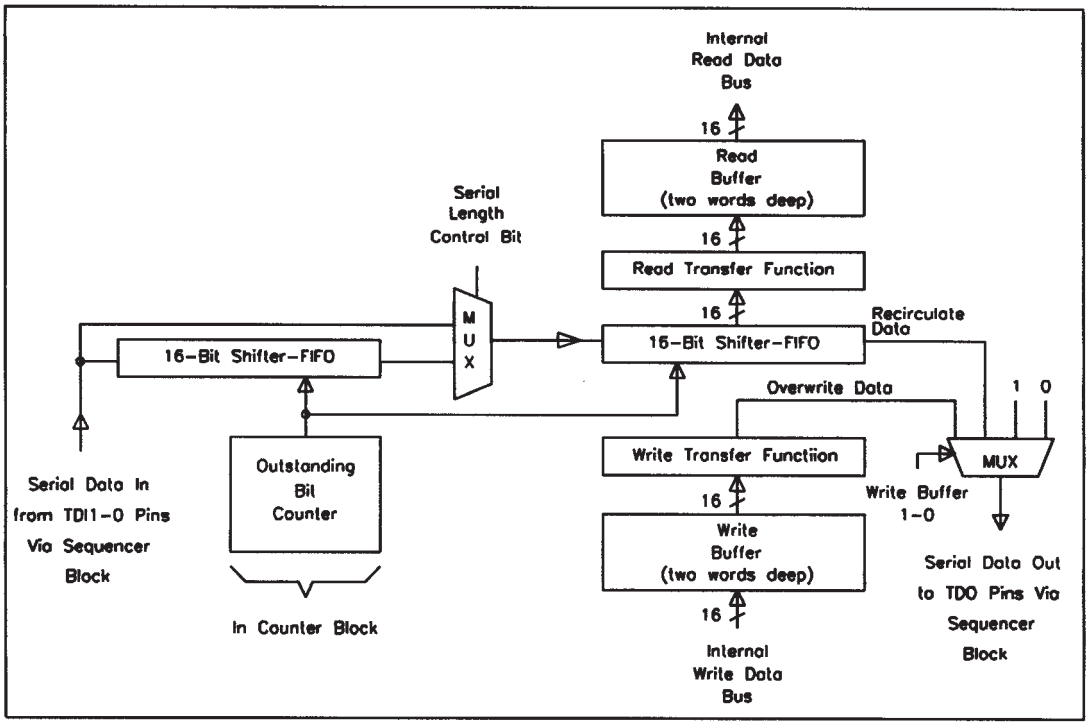



**Figure 12. Serial Block**

The SDO passes through the following sequence block to reach the test board as the TDO signal. The SDO signal can be chosen from the following via the multiplexer MUX2:

- SDO data derives from the WRITE BUFFER.
- Die SDO data derives from the SHIFTER-FIFO.
- SDO data consists only of ones.
- SDO data consists only of zeros.

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, this register can be shifted out immediately via SDO. In the meantime, it is 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 to choose whether the WRITE BUFFER should be shifted to the right, i.e., starting with the least significant bit (LSB), or to the left, i.e., beginning with the most significant bit (MSB).

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

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 determine via a register how many registers of the READ BUFFERs contain data. Because, as explained previously, the SHIFTER-FIFO is loaded with 1s on reset, the first bits that the computer reads from the READ BUFFER are 16 or 32 1s, 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:

- Bits are transferred from the right into the WRITE BUFFER, with the first transmitted bit as LSB in the WRITE BUFFER.
- 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 (see Figure 15). These counters also can be configured as two combined 16-bit counters (see Figure 16) and as a 32-bit counter (see Figure 17). The multiplexers of the event blocks allow the programming of a variety of different functions.

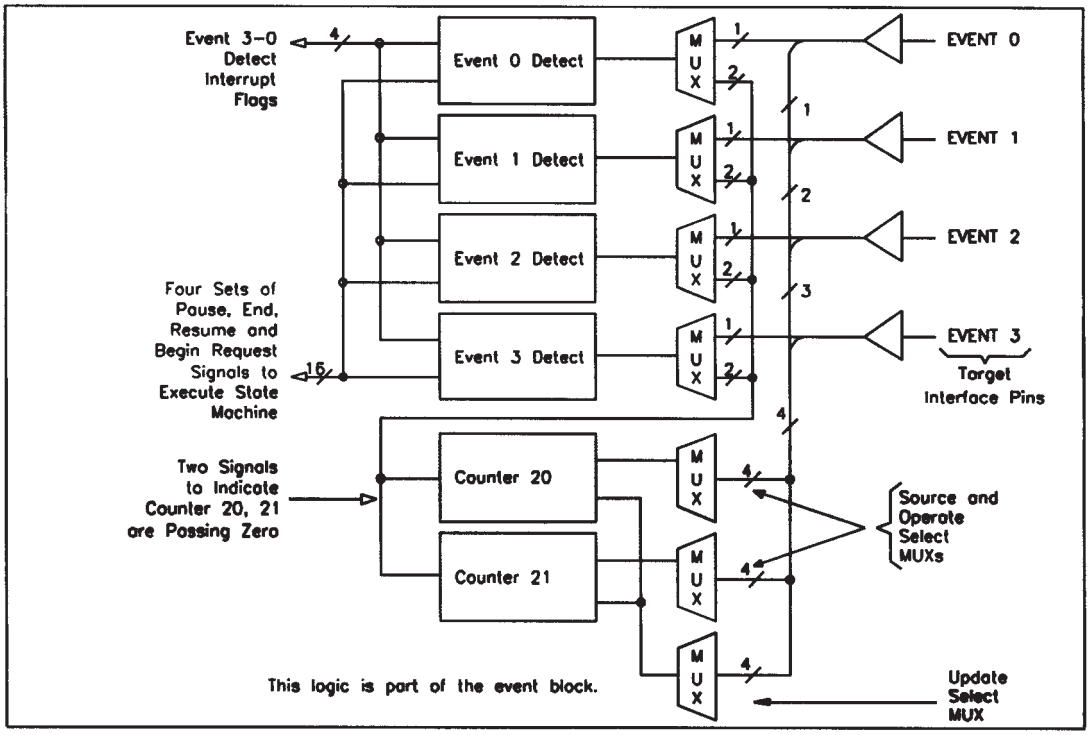



**Figure 13. Inputs of the Event Block**

For example, the EVENT0 input can be switched directly to the event detector by use of the multiplexer MUX 0. 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 that occur at the EVENT0 input only with COUNTER20. After the counter of the COUNTER UPDATE register of the counter blocks (see Figure 18) 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 COUNTER20, the event detector activates its output.

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

The output circuit of the event block is shown 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.

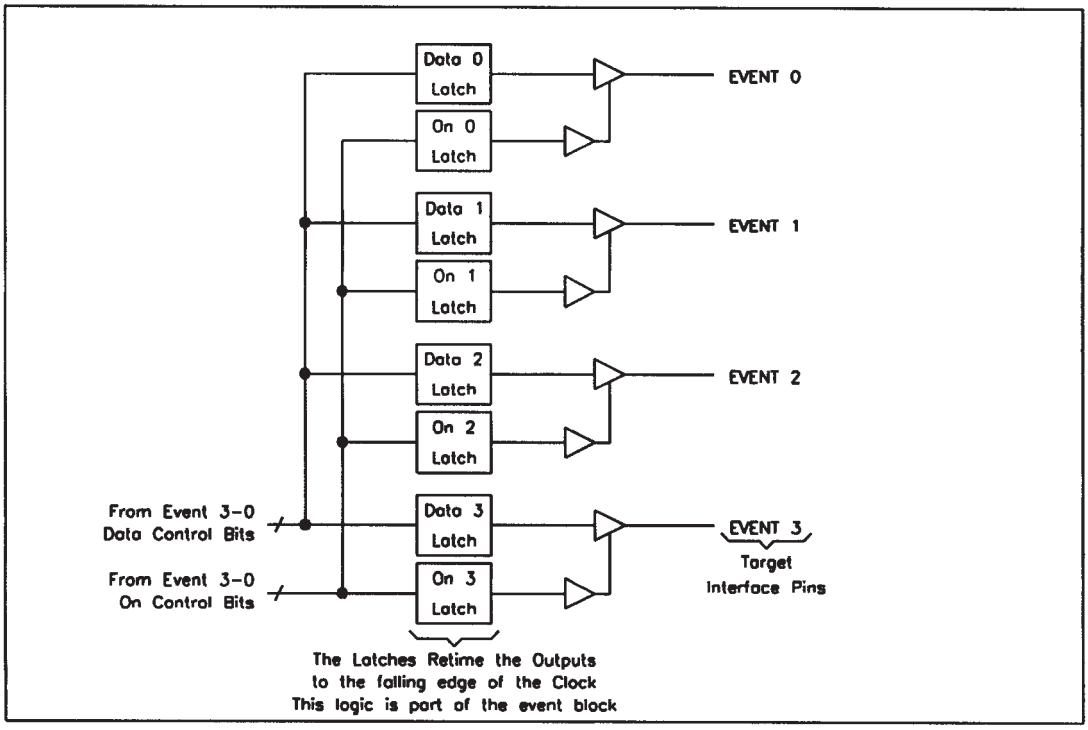



**Figure 14. Outputs of the Event Block**

COUNTER20 and COUNTER21 can be configured as follows:

- COUNTER20 and COUNTER21 work as 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 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 so that the first event must occur 55 times, and then the second event 12,345 times, before the event detector responds.
- Both counters are connected together to make a 32-bit counter (COUNTER2).

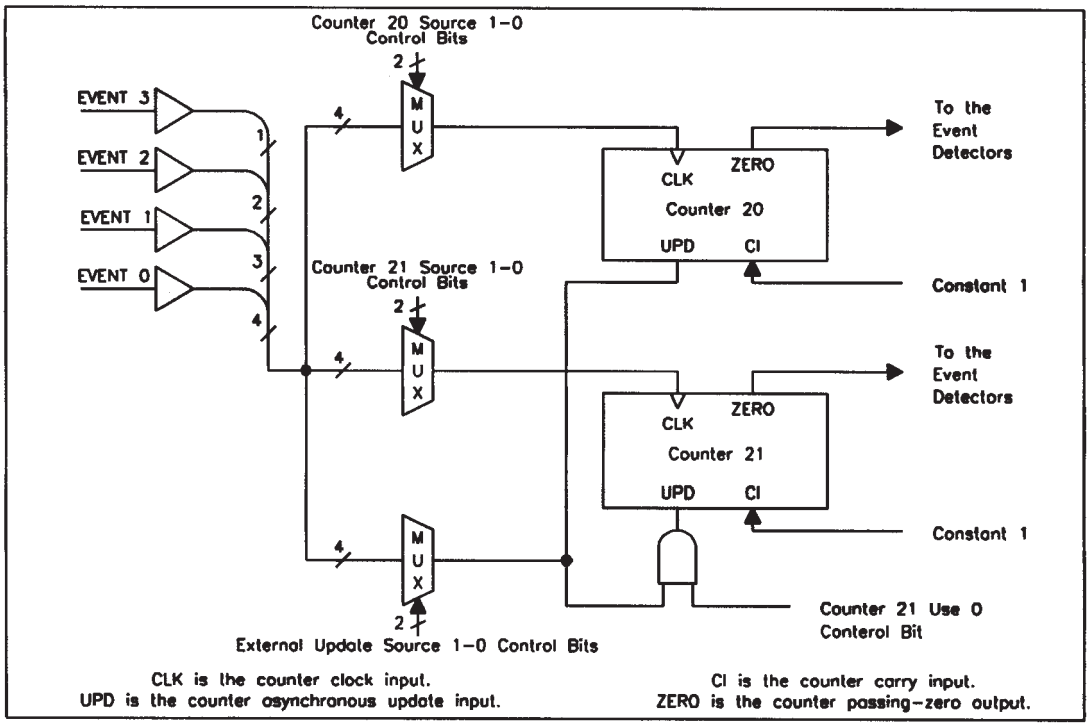



**Figure 15. COUNTER20 and COUNTER21 Configured as Separate 16-Bit Counters**

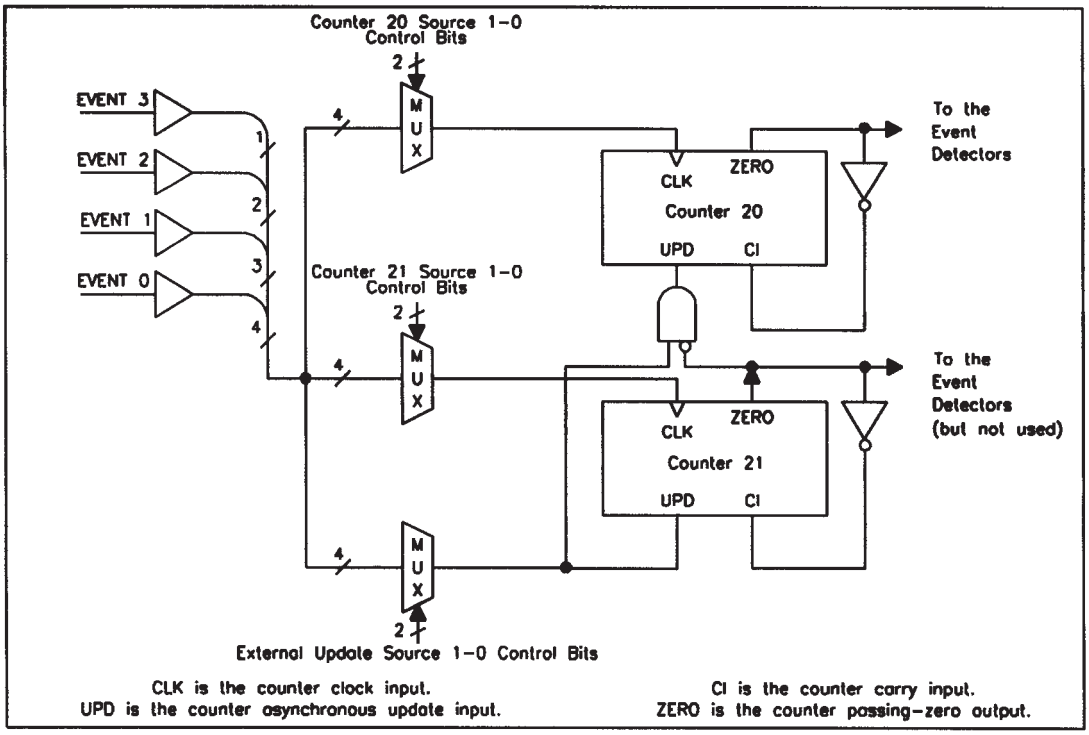



**Figure 16. COUNTER20 and COUNTER21 Configured as Combined 16-Bit Counters**

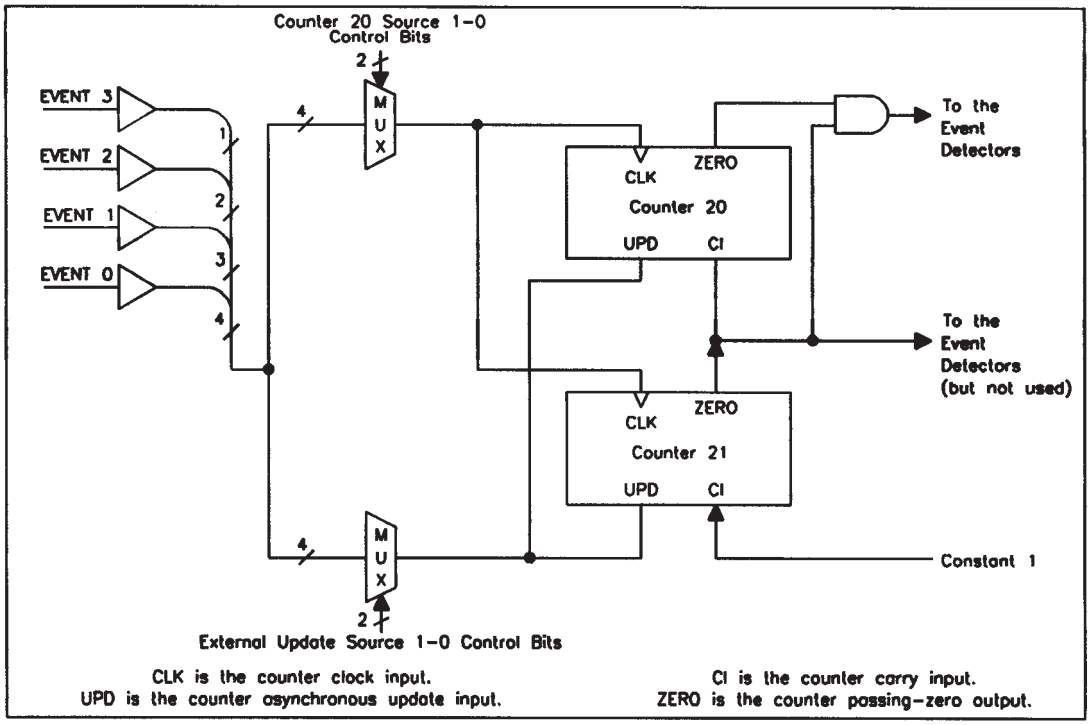



**Figure 17. COUNTER20 and COUNTER21 Configured as One 32-Bit Counter (COUNTER2)**

### *2.3.4 Counter Block*

The part of the counter block (see Figure 18) that is relevant for the user consists of:

- COUNTER1 UPDATE register
- 32-bit backwards counter (COUNTER1)
- COUNTER1 CAPTURE register

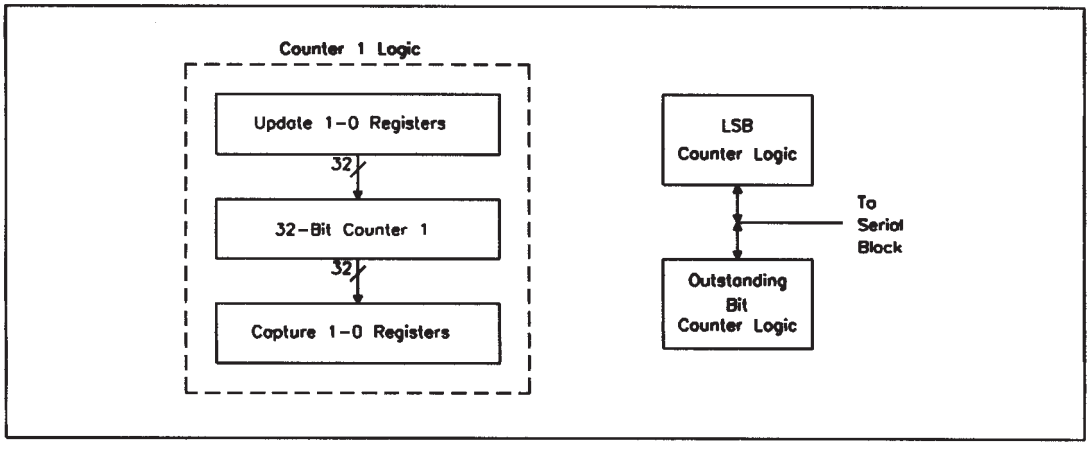



**Figure 18. Counter Block**

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

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

### *2.3.5 Command Block*

The command block contains two command registers (MAJOR COMMAND and MINOR COMMAND) (see Figure 19). In addition, all status registers are part of this block. The command block, with the command decode logic, controls the command process, with no intervention required by the user.

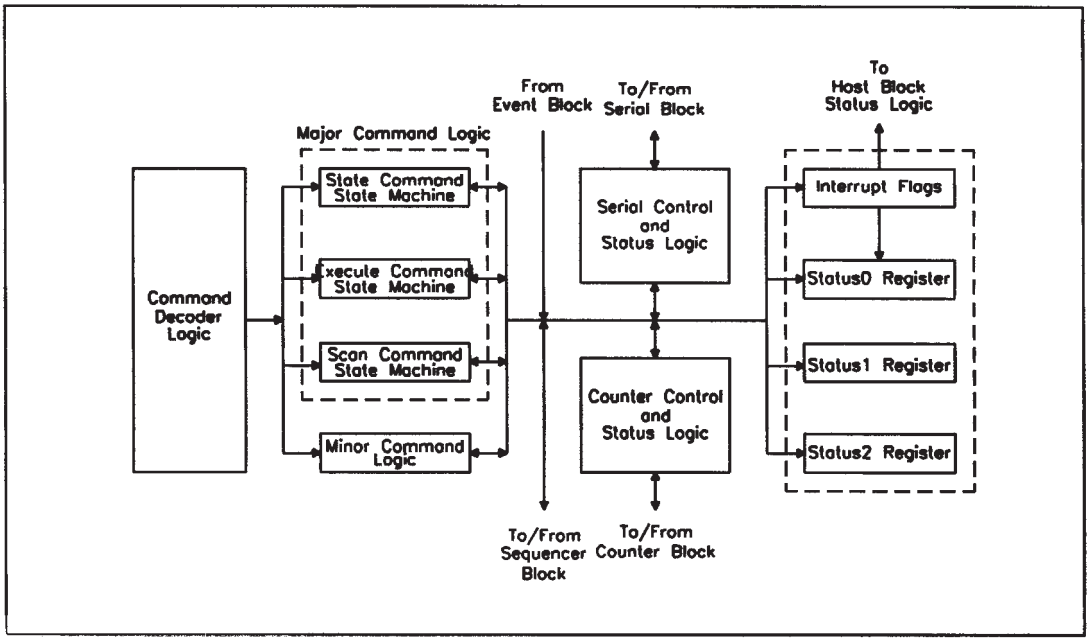



**Figure 19. Command Block**

### 2.3.6 Host Block

The tasks of the host block include data-bus 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 with no intervention required by the user.

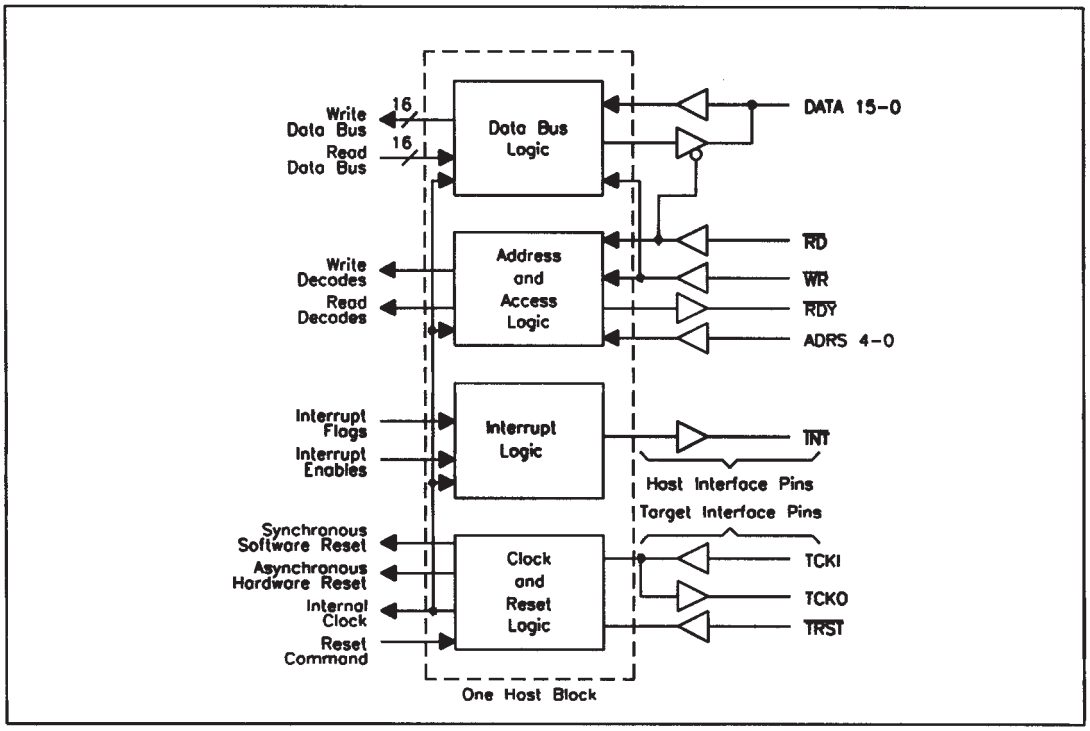



**Figure 20. Host Block**

# 3 Programming

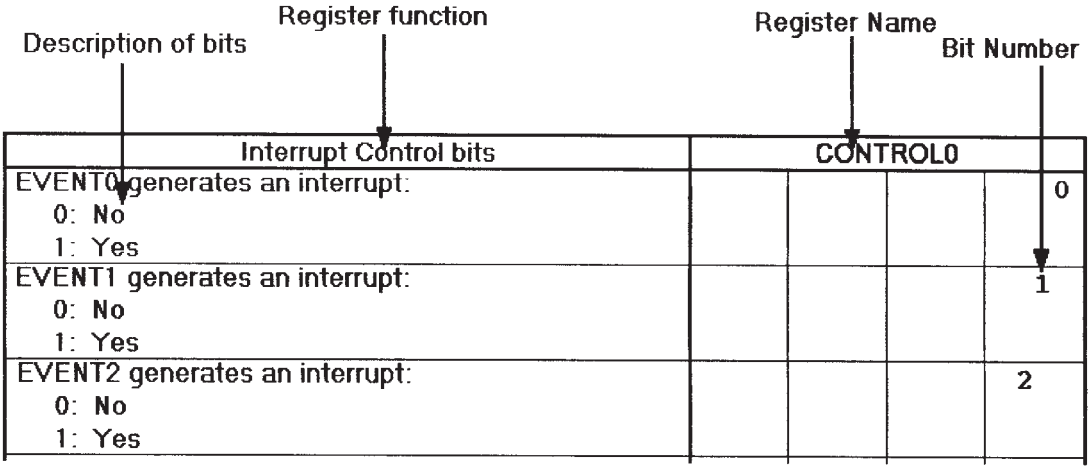
The control computer has access to the functions of the TBC using 24 registers, which are listed in Table 1. The ten control registers configure the TBC. The 32-bit command register is made up of a combination of the MAJOR COMMAND and MINOR COMMAND registers. With the MAJOR COMMAND register, presets 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 two CAPTURE registers, the counters of the event block and the counter block are controlled. The internal state of the TBC can be ascertained over three STATUS registers. The READ BUFFER and WRITE BUFFER are used to exchange data between the test board and the controlling computer.

**Table 1. Registers of TBC SN74ACT8990**

| 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,<br>TDO on/off, TMS/TCK Data Format | Read/Write |
| 04 | CONTROL4 | Delay,<br>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,<br>Control of Test Run,<br>Read Status, load counters and count,<br>Software RESET | Read/Write |
| 0B | MAJOR COMMAND | STATE Command,<br>EXECUTE Command,<br>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<br>or COUNTER20 | Read/Write |
| 0F | COUNTER2 UPDATE1 | COUNTER2 Higher Value Words<br>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 |

## 3.1 Registers

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



## *CONTROL0*

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

### CONTROL1

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

When using the SN74ACT8990 as controller of a test bus according to IEEE Std 1149.1, all bits of these registers must be set to zero by the user. 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:<br>0: No<br>1: Yes | | | | 0 |
| TMS1 is connected with internal TMS source:<br>0: No<br>1: Yes | | | | 1 |
| TMS2 is connected with internal TMS source:<br>0: No<br>1: Yes | | | | 2 |
| TMS3 is connected with internal TMS source:<br>0: No<br>1: Yes | | | | 3 |
| TMS4 is connected with internal TMS source:<br>0: No<br>1: Yes | | | 4 | |
| TMS5 is connected with internal TMS source:<br>0: No<br>1: Yes | | | 5 | |
| NULL | FEDC | BA98 | 76 | |

The component generates a TMS signal internally. This signal can be extracted at the pins TMS0–TMS5. All TMS pins that 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:<br>0: EVENT Input or Output<br>1: TMS2 Output | | | | 0 |
| TMS3/EVENT1:<br>0: EVENT Input or Output<br>1: TMS3 Output | | | | 1 |
| TMS4/EVENT2:<br>0: EVENT Input or Output<br>1: TMS4 Output | | | | 2 |
| TMS5/EVENT3:<br>0: EVENT Input or Output<br>1: TMS5 Output | | | | 3 |
| TCKO on/off:<br>0: TCKO Output is switched on<br>1: TCKO Output is switched off | | | 4 | |
| TDO on/off:<br>0: TDO Output is switched on<br>1: TDO Output is switched off | | | 5 | |
| TMS on/off:<br>0: all TMS Outputs are switched on<br>1: all TMS Outputs are switched off | | | 6 | |
| EVENT on/off:<br>0: all EVENT Outputs are switched on<br>1: all EVENT Outputs are switched off | | | 7 | |
| Output Bit Format:<br>0000: for operation to IEEE 1149.1<br>else: for operation with special TI components | | BA98 | | |
| TCKO Output signal if TCKO Bit D is set to 1:<br>0: static 0-signal<br>1: static 1-signal | C | | | |
| TCKO Data Format:<br>0: TCKO is connected with internal TCK source<br>1: static signal | D | | | |
| NULL | FE | | | |

## CONTROL4

| Delay, Data Formats of TDI and TDO Pins | CONTROL4 | | | |
|---|---|---|---|---|
| Delay:<br>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:<br>0: 16 bits<br>1: 32 bits | | | 5 | |
| TDI Selection:<br>00,<br>01: The internal TDO signal is the source for the internal signal TDI<br>10: The TDI0 Input is the source for the internal signal TDI<br>11: The TDI1 Input is the source for the internal signal TDI | | | 76 | |
| TDO Sending Data:<br>00: TDO sends 0's<br>01: TDO sends 1's<br>10: TDO sends data of the SHIFTER-FIFO<br>11: TDO sends data of the WRITE BUFFER | | 98 | | |
| READ BUFFER on/off:<br>0: The received data is ignored after flowing through the SHIFTER-FIFO<br>1: The received data is transferred to the READ BUFFER after flowing through the SHIFTER-FIFO | | A | | |
| Transfer Format:<br>0: At data transfer, the LSB is always sent/received first<br>1: At data transfer, the MSB is always sent/received first | | B | | |
| TDI Data Transfer:<br>0: TDI Data Transfer on leading edge<br>1: TDI Data Transfer on trailing edge | C | | | |
| EVENT Data Transfer:<br>0: EVENT Data Transfer on leading edge<br>1: EVENT Data Transfer on trailing edge | D | | | |
| NULL | FE | | | |

## CONTROL5

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

## CONTROL6

| Event Choice at the EVENT0/1 Inputs | CONTROL6 | | | |
|---|---|---|---|---|
| EVENT0 sets the SUSPEND REQUEST Flag:<br>0: No<br>1: Yes | | | | 0 |
| EVENT0 sets the END REQUEST Flag:<br>0: No<br>1: Yes | | | | 1 |
| EVENT0 sets the RESUME REQUEST Flag:<br>0: No<br>1: Yes | | | | 2 |
| EVENT0 sets the BEGIN REQUEST Flag:<br>0: No<br>1: Yes | | | | 3 |
| EVENT0 is synchronous or asynchronous:<br>0: The recognition of the EVENT0 is asynchronous<br>1: The recognition of the EVENT0 is synchronous | | | 4 | |
| EVENT0 Masking:<br>0: The recognition of the EVENT0 always occurs<br>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:<br>00: Trailing Edge at Input EVENT0<br>01: Leading Edge at Input EVENT0<br>10: Zero passing of COUNTER20<br>11: Zero passing of COUNTER21<br>Synchronous EVENT0 recognition of 2 successive TCK edges:<br>00: 1-Level --> 0-Level<br>01: 0-Level --> 1-Level<br>10: 0-Level --> 0-Level<br>11: 1-Level --> 1-Level | | | 76 | |
| EVENT1 sets the SUSPEND REQUEST Flag:<br>0: No<br>1: Yes | | 8 | | |
| EVENT1 sets the END REQUEST Flag:<br>0: No<br>1: Yes | | 9 | | |
| EVENT1 sets the RESUME REQUEST Flag:<br>0: No<br>1: Yes | | A | | |
| EVENT1 sets the BEGIN REQUEST Flag:<br>0: No<br>1: Yes | | B | | |
| EVENT1 is synchronous or asynchronous:<br>0: The recognition of the EVENT1 is asynchronous<br>1: The recognition of the EVENT1 is synchronous | C | | | |
| EVENT1 Masking:<br>0: The recognition of the EVENT1 always occurs<br>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 | | | |

## *CONTROL6 (Continued)*

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

## CONTROL7

| Event Choice of the EVENT2/3 Inputs | CONTROL7 | | | |
|---|---|---|---|---|
| EVENT2 sets the SUSPEND REQUEST Flag:<br>0: No<br>1: Yes | | | | 0 |
| EVENT2 sets the END REQUEST Flag:<br>0: No<br>1: Yes | | | | 1 |
| EVENT2 sets the RESUME REQUEST Flag:<br>0: No<br>1: Yes | | | | 2 |
| EVENT2 sets the BEGIN REQUEST Flag:<br>0: No<br>1: Yes | | | | 3 |
| EVENT2 is synchronous or asynchronous:<br>0: The recognition of the EVENT2 is asynchronous<br>1: The recognition of the EVENT2 is synchronous | | | 4 | |
| EVENT2 Masking:<br>0: The recognition of the EVENT2 always occurs<br>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:<br>00: Trailing Edge at input EVENT2<br>01: Leading Edge at input EVENT2<br>10: Zero passing of COUNTER20<br>11: Zero passing of COUNTER21<br>Synchronous EVENT2 recognition of 2 successive TCK edges:<br>00: 1-Level --> 0-Level<br>01: 0-Level --> 1-Level<br>10: 0-Level --> 0-Level<br>11: 1-Level --> 1-Level | | | 76 | |
| EVENT3 sets the SUSPEND REQUEST Flag:<br>0: No<br>1: Yes | | 8 | | |
| EVENT3 sets the END REQUEST Flag:<br>0: No<br>1: Yes | | 9 | | |
| EVENT3 sets the RESUME REQUEST Flag:<br>0: No<br>1: Yes | | A | | |
| EVENT3 sets the BEGIN REQUEST Flag:<br>0: No<br>1: Yes | | B | | |
| EVENT3 is synchronous or asynchronous:<br>0: The recognition of the EVENT3 is asynchronous<br>1: The recognition of the EVENT3 is synchronous | C | | | |
| EVENT3 Masking:<br>0: The recognition of the EVENT3 always occurs<br>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 | | | |

## CONTROL7 (Continued)

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

## CONTROL8

| COUNTER1 Control | CONTROL8 | | | |
|---|---|---|---|---|
| COUNTER1 sets the SUSPEND REQUEST Flag during zero passing:<br>0: No<br>1: Yes | | | | 0 |
| COUNTER1 sets the END REQUEST Flag during zero passing:<br>0: No<br>1: Yes | | | | 1 |
| COUNTER1 is loaded by the COUNTER1 UPDATE Register during zero passing:<br>0: No<br>1: Yes | | | | 2 |
| COUNTER 1 counts<br>000,<br>001: never.<br>010: the clock edge of TCK in RUN-TEST/IDLE state during an EXECUTE command.<br>011: always when an EXECUTE command is executed.<br>100,<br>101: with the EVENT DETECTOR, which is selected with Bits 7 and 6.<br>110: if an 0-Level exists at the EVENT PIN selected by Bits 7 and 6.<br>111: if an 1-Level exits at the EVENT PIN selected by Bits 7 and 6. | | | 54 | 3 |
| COUNTER1 reacts on<br>00: EVENT DETECTOR0 or the input EVENT0<br>01: EVENT DETECTOR1 or the input EVENT1<br>10: EVENT DETECTOR2 or the input EVENT2<br>11: EVENT DETECTOR3 or the input EVENT3<br>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<br>000: 0-Level at the EVENT0 Input.<br>001: 1-Level at the EVENT0 Input.<br>010: 0-Level at the EVENT1 Input.<br>011: 1-Level at the EVENT1 Input.<br>100: 0-Level at the EVENT2 Input.<br>101: 1-Level at the EVENT2 Input.<br>110: 0-Level at the EVENT3 Input.<br>111: 1-Level at the EVENT3 Input. | | | | 210 |
| COUNTER20 is loaded by the COUNTER2 UPDATE0 Register:<br>0: Yes<br>1: No | | | | 3 |
| COUNTER20 is loaded during zero passing:<br>0: No<br>1: Yes | | | 4 | |
| COUNTER20 counts<br>000: if there is a trailing edge at the input EVENT0.<br>001: if there is a leading edge at the input EVENT0.<br>010: if there is a trailing edge at the input EVENT1.<br>011: if there is a leading edge at the input EVENT1.<br>100: if there is a trailing edge at the input EVENT2.<br>101: if there is a leading edge at the input EVENT2.<br>110: if there is a trailing edge at the input EVENT3.<br>111: if there is a leading edge at the input EVENT3. | | | 765 | |
| COUNTER2 can be loaded by an<br>0: asynchronous signal at the EVENT input.<br>1: synchronous signal at the EVENT input. | | 8 | | |
| COUNTER20 and COUNTER21 work as<br>00: separate 16-Bit counters; only COUNTER20 can be loaded by an asynchronous input signal.<br>01: separate 16-Bit counters; both counters can be loaded by an asynchronous input signal.<br>10: two tied 16-Bit counter<br>11: one 32-Bit counter. This counter can be loaded by an asynchronous input signal. | | A9 | | |
| COUNTER21 is loaded by the COUNTER2 UPDATE1 Register:<br>0: Yes<br>1: No | | B | | |
| COUNTER21 is loaded at zero passing:<br>0: No<br>1: Yes | C | | | |
| COUNTER21 counts<br>000: if there is a trailing edge at the input EVENT0.<br>001: if there is a leading edge at the input EVENT0.<br>010: if there is a trailing edge at the input EVENT1.<br>011: if there is a leading edge at the input EVENT1.<br>100: if there is a trailing edge at the input EVENT2.<br>101: if there is a leading edge at the input EVENT2.<br>110: if there is a trailing edge at the input EVENT3.<br>111: if there is a leading edge at the input EVENT3. | FED | | | |

## *MINOR COMMAND*

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

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

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

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

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

### *MAJOR COMMAND*

| STATE Command | MAJOR COMMAND | | | |
|---|---|---|---|---|
| New state after executing the command:<br>000,<br>001: TEST-LOGIC-RESET<br>010,<br>011: RUN-TEST/IDLE<br>100: SHIFT-DR<br>101: SHIFT-IR<br>110: PAUSE-DR<br>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:

- Event recognition, and all counters, are ignored.
- COUNTER1 is ignored and does not count, even if the RUN-TEST/IDLE state occurs.
- Data buffer is ignored and remains unchanged, even if a SHIFT-DR state or a SHIFT-IR state occurs.
- SHIFTER-FIFO is ignored and remains unchanged, even if a SHIFT-DR state or a SHIFT-IR state occurs.
- Execution of the command cannot 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, and 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:<br>000,<br>001: TEST-LOGIC-RESET<br>010,<br>011: RUN-TEST/IDLE<br>100: SHIFT-DR<br>101: SHIFT-IR<br>110: PAUSE-DR<br>111: PAUSE-IR | | | | 210 |
| States for execution and suspension of commands:<br>0: Execution of command in RUN-TEST/IDLE state and suspension in PAUSE-DR state.<br>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:<br>00,<br>01: During zero passing of COUNTER1 the execution of command is suspended or cancelled.<br>10: The execution of the command is finished after the EXECUTE state has been passed through once. COUNTER1 is not used.<br>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:<br>00,<br>01: The selected new state is controlled directly. This case corresponds to the STATE COMMAND, but can not be controlled by events.<br>10: The execution of the command is suspended immediately after the start (SLEEP STATE)<br>11: The execution of command begins with it's start | | | 76 | |
| SUSPEND and RESUME Flag:<br>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.<br>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:<br>000,<br>001: TEST-LOGIC-RESET<br>010,<br>011: RUN-TEST/IDLE<br>100: SHIFT-DR<br>101: SHIFT-IR<br>110: PAUSE-DR<br>111: PAUSE-IR | | | | 210 |
| States for execution and suspension of commands:<br>0: Execution of command in SHIFT-DR state and suspension of execution of command in PAUSE-DR state.<br>1: Execution of command in SHIFT-IR state and suspension of execution of command in PAUSE-IR state. | | | | 3 |
| Suspension and cancellation of commands:<br>00,<br>01: During zero passing of COUNTER1 the execution of command is suspended or cancelled.<br>10: The execution of the command is finished, after the EXECUTE state has been passed through once. COUNTER1 is not used.<br>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 also is 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 COUNTER20 and COUNTER21 can be used as two independent 16-bit counters (see Figure 15), two combined 16-bit counters (see Figure 16), or as one 32-bit counter (COUNTER2) (see Figure 17). In both cases, they can be used for counting events that occur at the EVENT0–EVENT3 inputs.

### *STATUS0*

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

### *STATUS1*

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

### STATUS2

| State of the Test Board and of the buffers | STATUS2 | | | |
|---|---|---|---|---|
| Actual state of Test Board:<br>0000,<br>0001: TEST-LOGIC-RESET<br>0010,<br>0011: RUN-TEST/IDLE<br>0100: SHIFT-DR<br>0101: SHIFT-IR<br>0110: PAUSE-DR<br>0111: PAUSE-IR<br>1000: Temporary state | | | | 3210 |
| NULL | | | 7654 | |
| WRITE BUFFER State:<br>00: both WRITE BUFFERs are empty<br>01: not used<br>10: one WRITE BUFFER is empty, one is full<br>11: both WRITE BUFFERs are full | | 98 | | |
| NULL | | BA | | |
| READ BUFFER State:<br>00: both READ BUFFERs are empty<br>01: one READ BUFFER is empty, one is full<br>10: not used<br>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 CAPTURE0 and CAPTURE1 registers.

### *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 Std 1149.1, the commands typically are sent first 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 must be programmed to 1 for an IR-SCAN, or to 0 for a DR-SCAN. To implement an IR-SCAN or DR-SCAN under the control of the TBC, the following programming steps are necessary:

1. First, the TBC must be configured using the ten control registers. If this already has been done, only some necessary changes to the configuration might need to be made.
2. The COUNTER1 UPDATE register must be programmed according to the length of the shift register.
3. Using the MAJOR COMMAND, IR-SCAN or DR-SCAN is selected and, subsequently, the MINOR COMMAND starts the execution of the commands.

4. a) The STATUS register allows control of the status of the WRITE BUFFER. If one of the two buffers is empty, 16 command bits or 16 bits of the test vector must be written into the WRITE BUFFER. These bits are shifted from the TBC, over the TDO line, into the test board. Simultaneously, the TBC 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.

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

   c) Steps 4a and 4b should be continued until all bits in the test board have been written. For the final iteration, usually less than 16 bits need to be shifted. Unneeded bits of the WRITE BUFFER will be ignored by the TBC; excess bits from the READ BUFFER are filled by the TBC with zeros.

5. 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.

Since the TBC continually sends the clock signal TCK to the test board, movement through the status diagram (see Figure 21) can be halted in only 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., the control computer has not yet written the next data in the WRITE BUFFER, or because both READ BUFFERs are full), the TBC changes to the PAUSE-DR or PAUSE-IR state. It remains 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 is used. If the PAUSE-DR or PAUSE-IR states are chosen, the fact should be noted that, after ending a command, the UPDATE-DR or UPDATE-IR phases would not proceed.

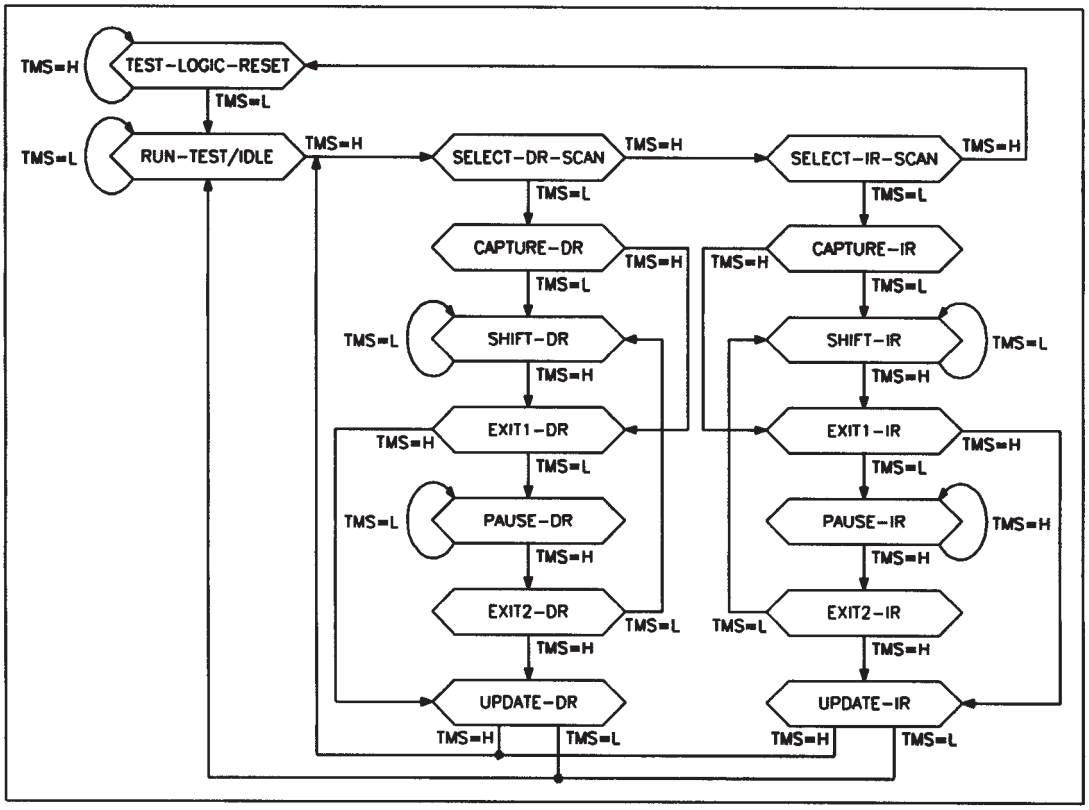



**Figure 21. IEEE Std 1149.1 Test Access Port (TAP) Status Diagram**

# 4 Examples

The programming of TBC SN74BCT8244N is demonstrated in the following examples. All examples assume the use of the same test board with two SN74BCT8244N SCOPE octals.

## 4.1 Example Test Board

All examples in this section are based on the same test board. To simplify an understanding of the examples, this test board consists of two SN74BCT8244N SCOPE octals from TI (see Figure 22). The programming methods used with more complex test boards are basically the same.

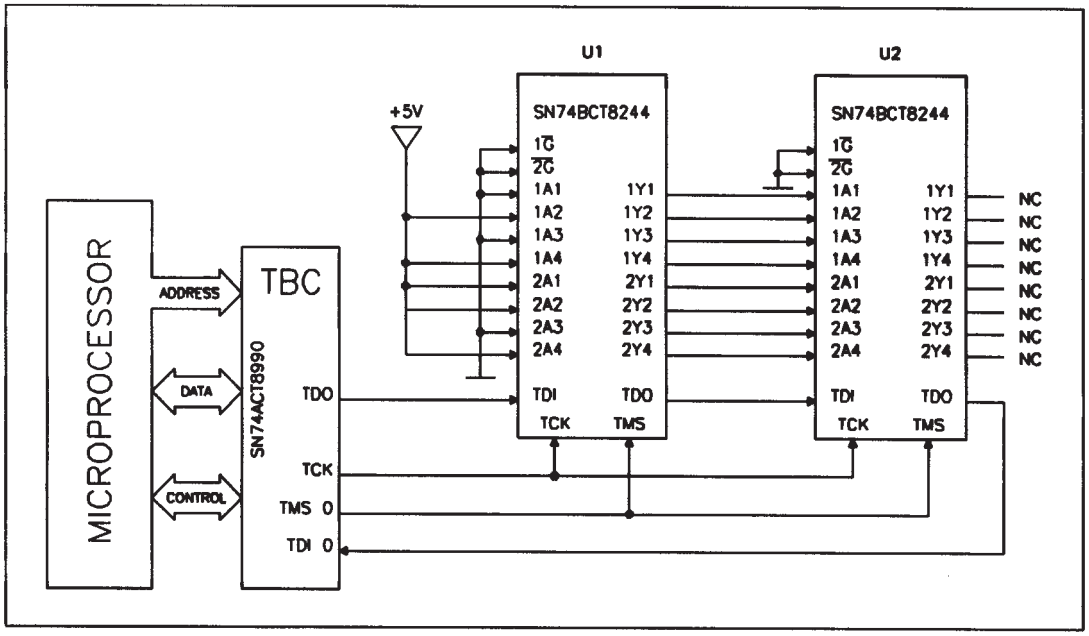



**Figure 22. Hardware Example: Two SCOPE Octals SN74BCT8244N**

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

- SHIFT-IR state
  - 16-bit INSTRUCTION register (IR)
- SHIFT-DR state
  - 36 boundary scan cells (BSCs)
  - 4-bit BOUNDARY CONTROL register (BCR)
  - 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

The next example essentially is a combination of the three basic functions:

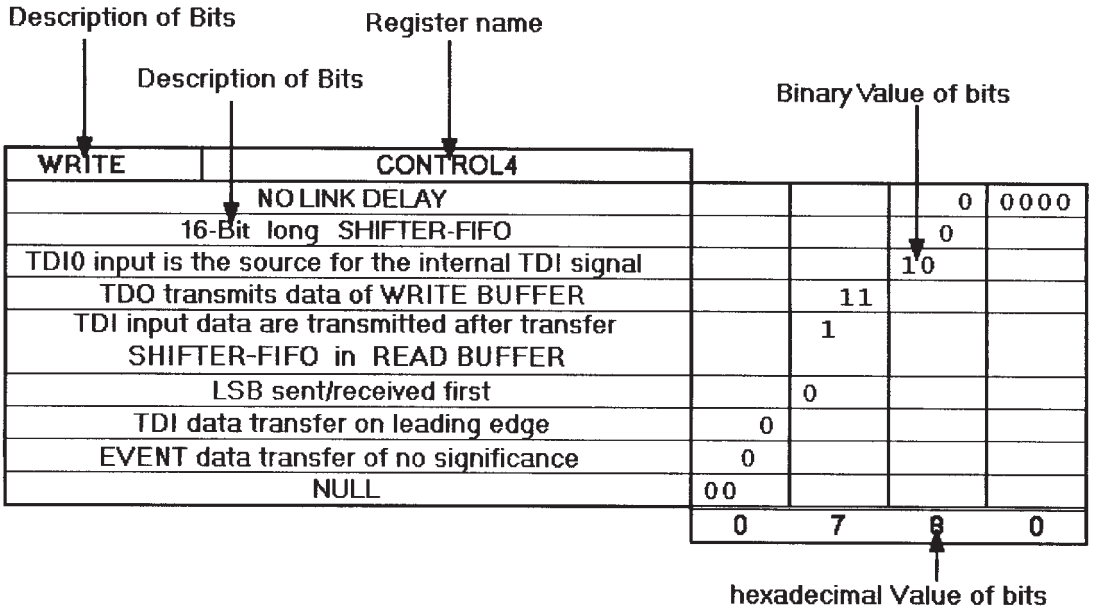
- EXTEST

Finally, the pseudo-random pattern generation/parallel signature analysis (PRPG/PSA) is 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 that are used.

The programming of the registers is explained in detail in the 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 and the resulting bit combinations are included, whether binary or hexadecimal.




| 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 after transfer SHIFTER-FIFO in READ BUFFER | | | 1 | | |
| LSB sent/received first | | | 0 | | |
| TDI data transfer on leading edge | | 0 | | | |
| EVENT data transfer of no significance | | 0 | | | |
| NULL | | 00 | | | |
| | | 0 | 7 | 8 | 0 |

### *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 always should 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 ten control registers:

- TDI0 as test data input
- TMS0 as TEST MODE SELECT output
- No interrupt transferred to the control computer
- The TMS2–TMS5/EVENT0–EVENT3 inputs/outputs are made TMS outputs, but are 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 TBC 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 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 TBC now is 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 TBC 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 be interrogated first, 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:<br>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 now can 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 two SCOPE octals is now in the SHIFTER-FIFO. If a test vector is shifted into the test board, the next 16 bits of the READ BUFFER will contain the status information of the two 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 given in section 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:

- Length of the shift register is 36 bits.
- DR-SCAN must be chosen to shift the test vector.

The control registers should be programmed only 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 given in section 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** |

COUNTER1 now must 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, now the DR-SCAN is chosen. The MINOR COMMAND then starts the shifting operation again.

| **Write** | **MAJOR COMMAND** | | | | |
|---|---|---|---|---|---|
| State after executing command<br>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:<br>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:<br>Control of test process | | 0101 | | | |
| | | **5** | **0** | **8** | **0** |

The WRITE BUFFER now can be fed with data again, 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, 2 × 16 bits can be entered immediately.

| 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 TBC 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. To be able to shift further data into the test board, at least one 16-bit word from the READ BUFFER must be read next. Via the STATUS2 register, it also must be determined if 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 section 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 also can 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, because they will 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 four 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 bits are still in the SHIFTER-FIFO:

- Six bits: 1Y3–1Y4, 2Y1–2Y4 (test sample of SCOPE octal U1)
- Eight bits: 1A1–1Y4, 2Y1–2Y4 (test sample of SCOPE octal U1)
- Two bits: 1G, 2G (test sample of SCOPE octal U1)

The example in section 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 |

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 that leaves the SHIFTER-FIFO is copied into the READ BUFFER, the READ BUFFER contains 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** |

The contents of SHIFTER-FIFO can be removed from the READ BUFFER. If this is done as shown in the example in section 4.2.2, the READ BUFFER contains 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 remain unchanged, and the state of the test board also remains unchanged.

### *4.2.4 EXTEST*

With the three basic functions for the preceding section, an EXTEST for the test board used can be made very easily. The 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 that results.

| 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 already have been explained in detail in previous examples, and are summarized here:

First, the TBC 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 |

Both SCOPE octals are loaded with the command EXTEST. The value to be read from the READ BUFFER is the 1s 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

   The first test vector is shifted into the two 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 that were 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 now is shifted into the SCOPE octals. From the READ BUFFER, the rest of the previous bitmap of the SCOPE octals can be read first, 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 still are in the SHIFTER-FIFO. To receive these bits, the contents of the SHIFTER-FIFO must 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 has been completed. The test vector was written with three shift operations into the test board. In the same way, reading the system reply to the test vector of the READ BUFFER is done 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:

| 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 | 01011101 | 11010101 | 00 | 10011110 | 01000011 |

### *4.2.5 PRPG/PSA*

This example shows how, with the three basic functions, IR-SCAN, DR-SCAN, and read SHIFTER-FIFO, and an additional EXECUTE RUN-TEST/IDLE block, a PRPG/PSA 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 the 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 ten 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 TBC 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 BCR 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 BCR: 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 TBC must be changed. The COUNTER1 is used 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 loaded now with the number of RUN-TEST/IDLE cycles. In this case, ten cycles should be made. The register must 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

To read the result vectors, one of the two commands, READBT or READBN, must be used. Other commands, such as EXTEST or SAMPLE, are written over during the UPIDATE-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 is 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, is given below:

| 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 previous PRPG/PSA example. In this case, however, the PRPG/PSA will start only when a rising edge has arrived five times at the event input EVENT2.

The programming of the TBC is nearly identical with that of the PRPG/PSA example. Only the sixth step, the execution of the PRPG/PSA, is somewhat changed. Therefore, only this sixth step will 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 |

### *4.2.7 Implement PRPG/PSA*

Before execution of the PRPG/PSA can begin, the configuration of the TBC must be changed. COUNTER1 is used to count the TCK edges during the RUN-TEST/IDLE state. COUNTER20 counts 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 now is 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 now is 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** |

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

# 5 Summary

JTAG has developed a coordinated test concept based on boundary-scan techniques, whose specification is embodied in IEEE Std 1149.1. This allows cost-effective testing of complex electronic systems.

When using boundary-scan test methods according to IEEE Std 1149.1, it is necessary to control this test bus with a computer. For this purpose, TI has made available a suitable peripheral component, the TBC SN74ACT8990. If the ASSET development system from TI is used to develop a test program, the computer program supplied programs the TBC.

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

However, if the register of the TBC must be programmed directly, a detailed understanding of this component and methods of programming it are necessary. The examples in this application report make it easier for the user to undertake direct programming of the TBC.

## IMPORTANT NOTICE

Texas Instruments and its subsidiaries (TI) reserve the right to make changes to their products or to discontinue any product or service without notice, and advise customers to obtain the latest version of relevant information to verify, before placing orders, that information being relied on is current and complete. All products are sold subject to the terms and conditions of sale supplied at the time of order acknowledgment, including those pertaining to warranty, patent infringement, and limitation of liability.

TI warrants performance of its semiconductor products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Customers are responsible for their applications using TI components.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards must be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance or customer product design. TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used. TI's publication of information regarding any third party's products or services does not constitute TI's approval, warranty or endorsement thereof.