RX Diagnostic Test Modes (version 1.1)

Nomenclatures:

R0: The DAC3xJ8x family uses two SERDES receiver blocks of R0 and R1, and each receiver block contains four receive lanes. R0 is the first SERDES RX block containing RX lanes lane0, lane1, lane2, and lane3.
R1: the second SERDES RX block containing RX lanes lane4, lane5, lane6, and lane7.

RX Basic Diagnostic Test Modes (RX BDTM): Provides basic pattern verification capability for the eight RX lanes. Program the test pattern selection such as 0/1 toggle pattern or PRBS patterns through SIF register. Read the error output through ALARM pin.

RX Advanced Diagnostic Test Modes (RX ADTM): Provides advanced pattern verification capability such as error counter and eye scan for the eight RX lanes. Program the RX ADTM via the JTAG controller. The internal JTAG to IEEE1500 bridge provides instruction translation.

JTAG Controller: IEEE1149.1 based controller. Primary function is to bridge the JTAG instructions to the internal IEEE1500 bridge for RX Advanced Diagnostic Test Modes access.

JTAG Controller

The DAC38J84 JTAG Controller is based on IEEE1149.1 standard. The primary function of the JTAG controller is to interface with the JTAG to IEEE1500 Bridge in order to program the RX ADTM. The test clock (TCLK) and test mode select (TMS) are the inputs to the JTAG controller logic, which is based on the state machine as shown in Figure 1. The sequence of these two inputs can lead to an instruction register scan (IR) or data register scan (DR). The TMS determines the transition of the state machine while the rising edge of TCLK latches the TMS signal to the state machine.

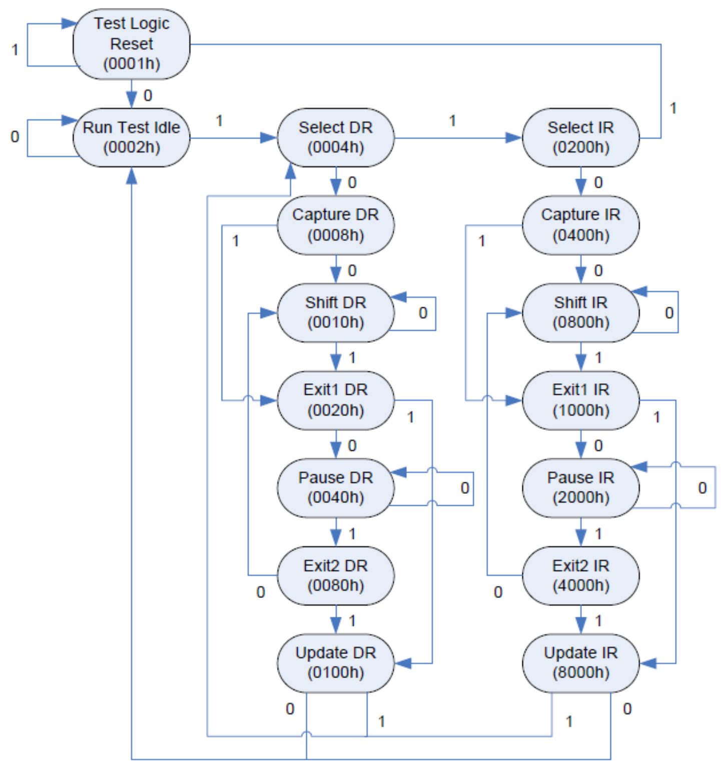


Figure 1. JTAG Controller State Machine

For instance, in order to access IR transaction, the desired state transitions are Run Test Idle (0x0002), Select DR (0x0004), Select IR (0x0200), Capture IR (0x0400), Shift IR (0x0800), performing the necessary input and output instructions for the IR transaction, Exit1 IR (0x1000), Update IR (0x8000), and then return to Run Test Idle (0x002). The corresponding TMS sequence would be 0 -> 1 -> 1 -> 0 -> loop input and output for transaction -> 1 -> 1 -> 0. The corresponding timing diagram and associated states for IR transaction are shown in Figure 2.

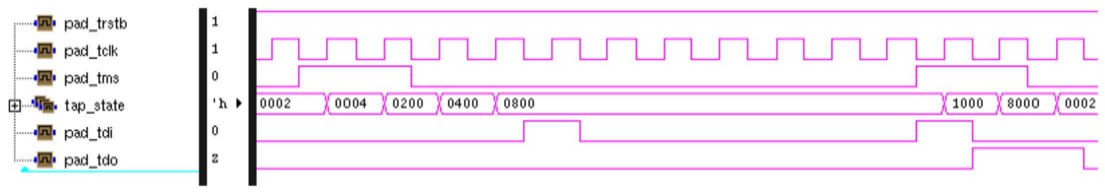


Figure 2. JTAG IR Transaction

When performing the necessary input and output instructions for the transaction, the TDI is the input to the controller and is sampled on the rising edge of TCLK, and the TDO is the output from the controller and transitions on the falling edge of TCLK. Thus TDO can be sampled on the next rising edge of TCLK.

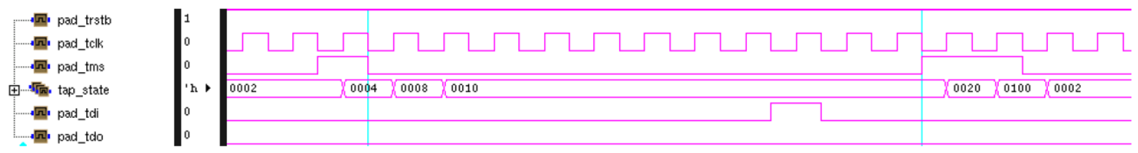


Figure 3. JTAG DR Transaction

The access to DR transaction is similar to IR transaction as shown in Figure 3. For instance, in order to access DR transaction, the desired state transitions are Run Test Idle (0x0002), Select DR (0x0004), Capture DR (0x0008), Shift DR (0x0010), performing the necessary input and output instructions for the IR transaction, Exit1 DR (0x0020), Update DR (0x0100), and then return to Run Test Idle (0x002). The corresponding TMS sequence would be 0 -> 1 -> 0 -> loop input and output for transaction -> 1 -> 1 -> 0. TDI and TDO are sampled in the same fashion as mentioned above.

JTAG Interface to IEEE1500 Instruction Registers

The programming command for the RX ADTM is based on the IEEE 1500-2005 standard. The users can use the JTAG ports, which are based on IEEE 1149.1 standard, to program the RX ADTM with IEEE 1500 instructions through the build-in JTAG to IEEE 15000 bridge. As shown in Figure 4, the logic used to translate signals from JTAG controller output to IEEE 1500 is based on Annex C of IEEE 1500 standard.

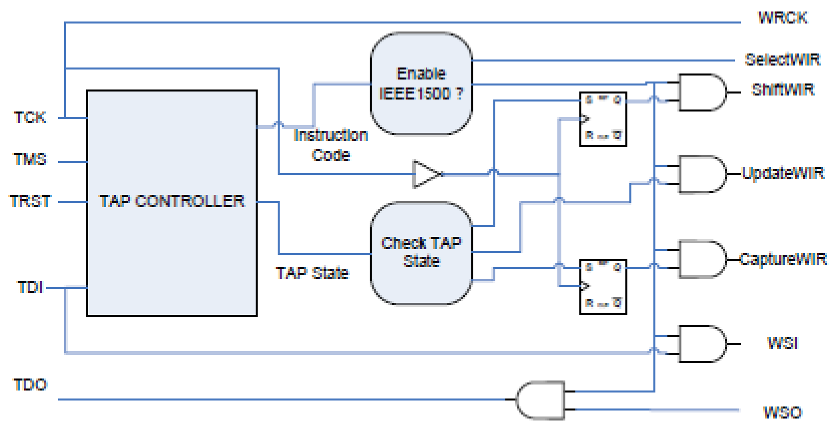


Figure 52 JTAG to IEEE1500 Bridge

Figure 4. JTAG to IEEE 15000 Bridge

The IEEE 1500 standard signals WRCK, WRSTN, WSI, and WSO correspond to their counterparts in JTAG: TCK, TRSTN, TDI, and TDO, respectively. With the build-in JTAG to IEEE1500 bridge, the IEEE 1500 IR or DR transactions are transparent, and the user only need to interface with the JTAG controller IR or DR transactions instead of the IEEE 1500 IR or DR transactions directly. Refer to the JTAG Controller section for details of JTAG IR and DR transaction.

IEEE1500 Instruction Registers

The two SERDES receiver R0 and R1 blocks support the IEEE1500 as shown in Table 1. As described in JTAG Interface to IEEE1500 Instruction Registers, the JTAG interface provides transparent access to these registers.

Instruction	Opcode	Description
ws_bypass	0x00	Bypass. Selects a 1-bit bypass data register. Use when accessing other macros on the same IEEE1500 scan chain.
ws_cfg	0x35	Configuration. Write protection options for other instructions.
ws_core	0x30	Core. Fields also accessible via dedicated core-side ports.
ws_tuning	0x31	Tuning. Fields for fine tuning macro performance.
ws_debug	0x32	Debug. Fields for advanced control, manufacturing test, silicon characterization and debug
ws_unshadowed	0x34	Unshadowed. Fields for silicon characterization.
ws_char	0x33	Char. Fields used for eye scan.

Table 1. IEEE1500 Instruction Codes

The following steps describe the process of accessing the IEEE1500 instructions. Steps one and two describe the JTAG IR and DR transactions needed for setting up the appropriate R0 or R1 block and also the associated instruction code. Steps three and four describe the JTAG IR and DR transactions needed for writing or reading the data from the R0 or R1 instruction register.

1. Set up an IEEE 1500 IR transaction by writing binary 2b'01101101 to the JTAG Controller (IEEE1149.1) IR transaction.
2. Provide instruction code to the IEEE1500 instruction registers of either R0 or R1 block. The IEEE1500 instruction registers are chained in series from the fuse block, through the R0 block, to the R1 block. The instruction register code should go through the JTAG DR transaction with the format of 0x00*R0R1*.

   NOTE:
   0x*R0* is the two nibble instruction opcode for the R0 block, while 0x*R1* is the two nibble instruction opcode for the R1 block. Refer to Table 1 for the instruction codes for each block. The instruction register code always starts with 0x00. Most importantly, the instruction register should address to one block at a time. This means either R0 or R1 should have valid instruction code at a time while another block's instruction code set to 0x00. (i.e. ws_bypass)

   A. To provide instruction code to R0 block, write 0x00*R0*00 to the JTAG Controller DR transaction.
   For example, to access ws_cfg instruction, write 0x003500 to the JTAG Controller DR transaction.

   B. To provide instruction code to R1 block, write 0x0000*R1* to the JTAG Controller DR transaction.
   For example, to access ws_cfg instruction, write 0x000035 to the JTAG Controller DR transaction.

3. Set up an IEEE 1500 DR transaction by writing binary b10011011 to the JTAG Controller IR transaction.
4. Provide data code to the IEEE1500 data registers of the desired block. Step 2 sets up and addresses the desired block while this step provides the data to the block. The data register code should go through the JTAG DR transaction with the format described for each instruction.

   NOTE:
   The data for each instruction has the format of chained components of head, body, and tail. The body is consisted of the four receiver lanes for either the R0 or R1 block. Therefore, the overall data format for the R0 or R1 block is head, receiver lane0, receiver lane1, receiver lane2, receiver lane3, and tail.

   All multi-bit registers in each chain are packed with bits reversed. For example, the head section of the ws_core chain is packed as {retime, enpll, mpy[0:7], vrange, endivclk, and lb[0:1]}, and the tail section of the ws_core chain is packed as {clkbyp[0:1], sleeppll, …, unlock, cfg_ovr, and retime}.

   The instruction chains are shifted in and out of the IEEE1500 data registers with the LSB leading. For example, shift the ws_core chain into the data register as {retime, cfg_ovr, unlock, …, sleeppll, clkbyp[1], clkbyp[0], …, lb[1], lb[0], endivclk, vrange, mpy[7], mpy[6], …, mpy[0], enpll, and retime}. Please note that the example above skipped register fields of the four receivers within a block.

   All data register reads from SerDes Block R0 should read 1 bit more than the desired number of bits and discard the first bit received on TDO. For instance, to read 40-bit data from R0 block, 41 bits should be read off from TDO and the first bit received should be discarded. Similarly, any data written to SerDes Block R0 Data Registers should be prefixed with an extra 0.

   These bit packing instructions are summarized in Table TBD to Table TBD.

Table 21. *ws_cfg* Chain

FIELD	DESCRIPTION
HEAD (STARTING FROM THE MSB OF CHAIN)	
RETIME	No function.
CORE_WE	Core chain write enable.
RECEIVER (FOR EACH LANE 0,1,2,3)	
CORE_WE	Core chain write enable.
TUNING_WE	Tuning chain write enable.
DEBUG_WE	Reserved.
CHAR_WE	Char chain write enable.
UNSHADOWED_WE	Reserved.
TAIL (ENDING WITH THE LSB OF CHAIN)	
CORE_WE	Core chain write enable.
TUNING_WE	Tuning chain write enable.
DEBUG_WE	Reserved.
RETIME	No function.
CHAIN LENGTH = 26 BITS	

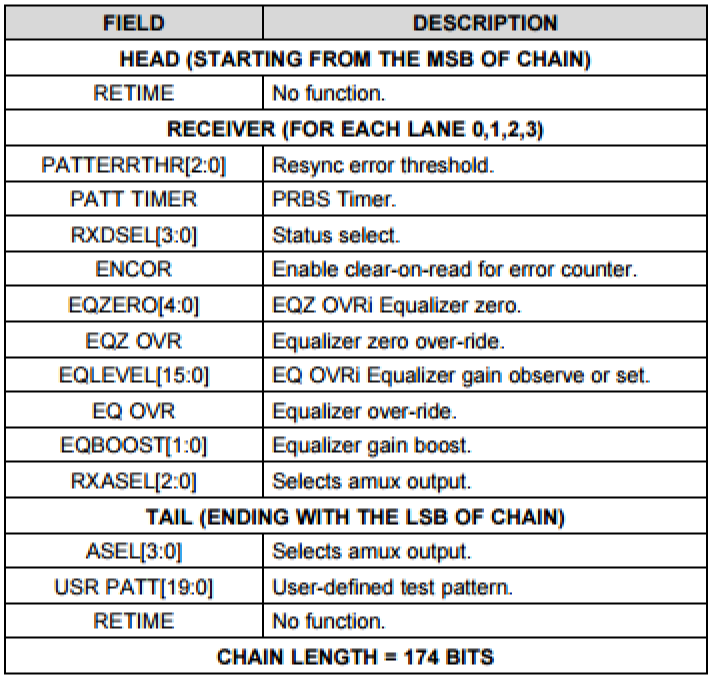
Table 22. *ws_core* Chain

FIELD	DESCRIPTION
HEAD (STARTING FROM THE MSB OF CHAIN)	
RETIME	No function.
ENPLL	PLL enable.
MPY[7:0]	PLL multiply.
VRANGE	VCO range.
ENDIVCLK	Enable DIVCLK output

Table 22. *ws_core* Chain (continued)

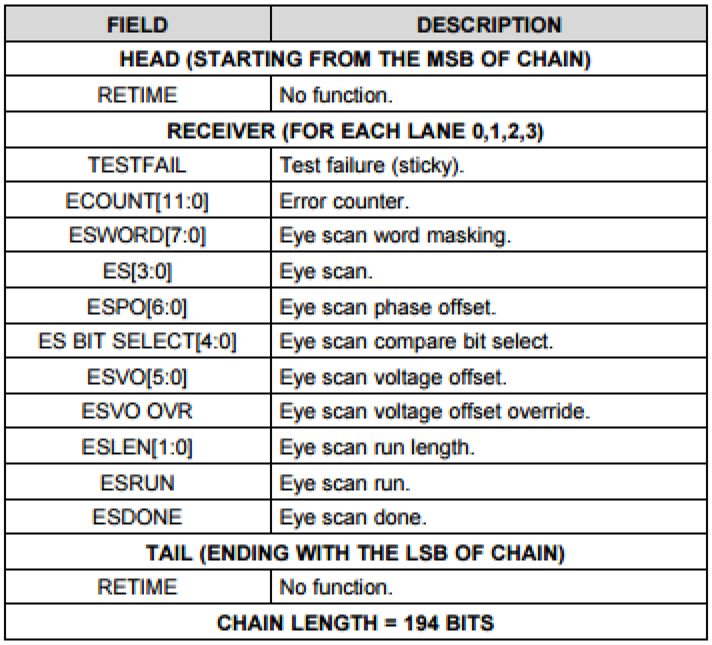
FIELD	DESCRIPTION
LB[1:0]	Loop bandwidth
RECEIVER (FOR EACH LANE 0,1,2,3)	
ENRX	Receiver enable.
SLEEPRX	Receiver sleep mode.
BUSWIDTH[2:0]	Bus width.
RATE[1:0]	Operating rate.
INVPAIR	Invert polarity.
TERM[2:0]	Termination.
ALIGN[1:0]	Symbol alignment.
LOS[2:0]	Loss of signal enable.
CDR[2:0]	Clock/data recovery.
EQ[2:0]	Equalizer.
EQHLD	Equalizer hold.
ENOC	Offset compensation.
LOOPBACK[1:0]	Loopback.
BSINRXP	Boundary scan initialization.
BSINRXN	Boundary scan initialization.
RESERVED	Reserved.
testpatt[2:0]	Testpattern selection.
TESTFAIL	Test failure (real time).
LOSDTCT	Loss of signal detected (real time).
BSRXP	Boundary scan data.
BSRXN	Boundary scan data.
OCIP	Offset compensation in progress.
EQOVER	Received signal over equalized.
EQUNDER	Received signal under equalized.
LOSDTCT	Loss of signal detected (sticky).
SYNC	Re-alignment done, or aligned comma output (sticky)
RETIME	No function.
TAIL (ENDING WITH THE LSB OF CHAIN)	
CLKBYP[1:0]	Clock bypass.
SLEEPPLL	PLL sleep mode.
RESERVED	Reserved.
LOCK	PLL lock (real time).
BSINITCLK	Boundary scan initialization clock.
ENBSTX	Enable Tx boundary scan.
ENBSRX	Enable Rx boundary scan.
ENBSPT	Rx pulse boundary scan.
RESERVED	Reserved.
NEARLOCK	PLL near to lock.
UNLOCK	PLL lock (sticky).
CFG OVR	Configuration over-ride.
RETIME	No function.
CHAIN LENGTH = 196 BITS	

Table 23. ***ws_tuning*** **Chain**



FIELD	DESCRIPTION
HEAD (STARTING FROM THE MSB OF CHAIN)	
RETIME	No function.
RECEIVER (FOR EACH LANE 0,1,2,3)	
PATTERRTHR[2:0]	Resync error threshold.
PATT TIMER	PRBS Timer.
RXDSEL[3:0]	Status select.
ENCOR	Enable clear-on-read for error counter.
EQZERO[4:0]	EQZ OVRi Equalizer zero.
EQZ OVR	Equalizer zero over-ride.
EQLEVEL[15:0]	EQ OVRi Equalizer gain observe or set.
EQ OVR	Equalizer over-ride.
EQBOOST[1:0]	Equalizer gain boost.
RXASEL[2:0]	Selects amux output.
TAIL (ENDING WITH THE LSB OF CHAIN)	
ASEL[3:0]	Selects amux output.
USR PATT[19:0]	User-defined test pattern.
RETIME	No function.
CHAIN LENGTH = 174 BITS	

Table 24. ***ws_char*** **Chain**



FIELD	DESCRIPTION
HEAD (STARTING FROM THE MSB OF CHAIN)	
RETIME	No function.
RECEIVER (FOR EACH LANE 0,1,2,3)	
TESTFAIL	Test failure (sticky).
ECOUNT[11:0]	Error counter.
ESWORD[7:0]	Eye scan word masking.
ES[3:0]	Eye scan.
ESPO[6:0]	Eye scan phase offset.
ES BIT SELECT[4:0]	Eye scan compare bit select.
ESVO[5:0]	Eye scan voltage offset.
ESVO OVR	Eye scan voltage offset override.
ESLEN[1:0]	Eye scan run length.
ESRUN	Eye scan run.
ESDONE	Eye scan done.
TAIL (ENDING WITH THE LSB OF CHAIN)	
RETIME	No function.
CHAIN LENGTH = 194 BITS	

Receiver Basic Diagnostic Test Modes (RX BDTM - SIF Register Access Required)

The SERDES receiver blocks support a number of basic pattern verification via SIF registers. The default test pattern selection includes an alternating 0/1 pattern, pseudo random bit stream (PRBS) sequences with three different pattern repeating lengths, and along with programmable custom pattern that can be accessed via the DAC3xJ84x JTAG/IEEE1500 port. The verification pattern selection is programmed via the TESTPATT fields of rw_cfgrx0[14:12], as shown in Table 18.

Table 18. SerDes Test Pattern Selection

TESTPATT	EFFECT
000	Test mode disabled.
001	Alternating 0/1 Pattern. An alternating 0/1 pattern with a period of 2UI.
010	Generate or Verify 2^7-1 PRBS. Uses a 7-bit LFSR with feedback polynomial $x^7 + x^6 + 1$.
011	Generate or Verify 2^{23} -1 PRBS. Uses an ITU O.150 conformant 23-bit LFSR with feedback polynomial $x^{23} + x^{18} + 1$.
100	Generate or Verify 2^{31}-1 PRBS. Uses an ITU O.150 conformant 31-bit LFSR with feedback polynomial $x^{31} + x^{28} + 1$.
101	User-defined 20-bit pattern. Uses the USR PATT IEEE1500 Tuning instruction field to specify the pattern. The default value is 0x66666.
11x	Reserved

All the test pattern sequences can often be found programmed into standard test equipment, such as a Bit Error Rate Tester (BERT). With the selection of PRBS sequence, the pattern verifier in the receiver compares the received word with an expected value calculated using an LFSR (shift register). Three PRBS sequence lengths are available: 2^7-1, 2^{23}-1, and 2^{31}-1, and the user can select the desired PRBS length to ensure the sequences are not repeated within certain amount time.

The received pattern on the RX port is serialized, and the DAC3xJ8x internal serial to parallel converter converts the serial data stream to parallel data stream. Afterwards, the pattern verification circuit compares the output of the internal serial to parallel converter with an expected pattern from the test pattern selection. When there is a mismatch, the TESTFAIL bit in ws_core chain is driven high, which can be programmed to come out the ALARM pin by setting dtest[3:0] to 2b'0011.

To perform SERDES test pattern, refer to the following SIF register read and write sequences:

1. config74, set bits 4:0 to 0x1E to disable JESD block
2. config61, set bits 14:12 to 0x01 to enable the alternating 0/1 pattern, 0x02 to enable the 7-bit PRBS test pattern, 0x03 to enable the 23-bit PRBS test pattern, 0x04 enable the 31-bit PRBS test pattern. Set bits 14:12 to 0x05 to enable user-defined 20-bit pattern, and the value of the pattern needs to be programmed through JTAG/IEEE1500 port.
3. config27, set bits 11:8 to 0x3 to output pattern verification TESTFAIL bit on ALARM pin.
4. config27, set bits 14:12 to the lane to be tested (0 through 7).
5. config62, make sure bits 12:11 are set to 0x0 to disable character alignment.

Users should monitor the ALARM pin to see the results of the test. If the test is failing, ALARM will be high (or toggling if marginal). If the test is passing, the ALARM will be low.

Receiver Advanced Test Modes (RX ADTM - JTAG/IEEE1500 Access Required)

Each SERDES receiver block has advanced diagnostic capabilities that provides accumulation of pattern verification errors and the ability to map out the width and height of the receive eye, also known as eye scan. The following sections provide details of performing pattern verification and eye scan.

Error Counter

All receive channels include a 12-bit counter for accumulating pattern verification errors. This counter is accessible via the ***ECOUNT****i* IEEE1500 Char field, as shown in table TBD. It is an essential part of the eye scan capability (see next section), though can be used independently of this.

The counter operates synchronously to **rxbclk[*i*]**, and increments once for every cycle that the ***TESTFAIL*** bit of **stsrx*i*[9:0]** (**stsrx*i*[0]**) is high. The counter will not increment when at its maximum value (i.e. all 1s).

When an IEEE1500 capture is performed, the count value is loaded into the ***ECOUNT****i* scan elements (so that it can be scanned out), and the counter is then reset, provided ***ENCOR****i* is set high.

ECOUNT*i* can be used to get a measure of the bit error rate. However, as the error rate increases, it will become less accurate due to limitations of the pattern verification capabilities. Specifically, the pattern verifier checks multiple bits in parallel (as determined by the Rx bus width, which should be set as 20bit for DAC3xJ8x family), and it is not possible to distinguish between 1 or more errors in this.

Note that the ***RXDSEL****i* IEEE1500 field can be used to mux other other functions onto the ***TESTFAIL****i* output, in which case ***ECOUNT****i* can be used to count these as well as pattern verification errors. However, because it shares a clock domain with the pattern verifier, the pattern verifier must still be enabled (***TESTPATT****i* ≠ 000).

Using the error counter to measure bit error rate requires the time interval over which errors are accumulated to be known. If an external time reference is not available, the counter can be configured as a timer, so that the time interval can be measured. To do this, select a suitable receiver PRBS pattern via ***TESTPATT****i*, set the ***PATT TIMER****i* IEEE1500 Tuning bit to 1, and configure the error counter to count ***PATTVSYNC****i* pulses via ***RXDSEL****i*. By programming ***RXDSEL****i* in the IEEE1500 Tuning chain filed to value of 4b’0010, the default TESTFAIL signal will now become PATTVSYNC signal. Program register config27, dtest field (bit11:8) to 4b’0011 to enable TESTFAIL signal routed to the CMOS ALARM terminal. The time interval can now be measured on the CMOS ALARM terminal.

Eye Scan and Symbol Response Extraction

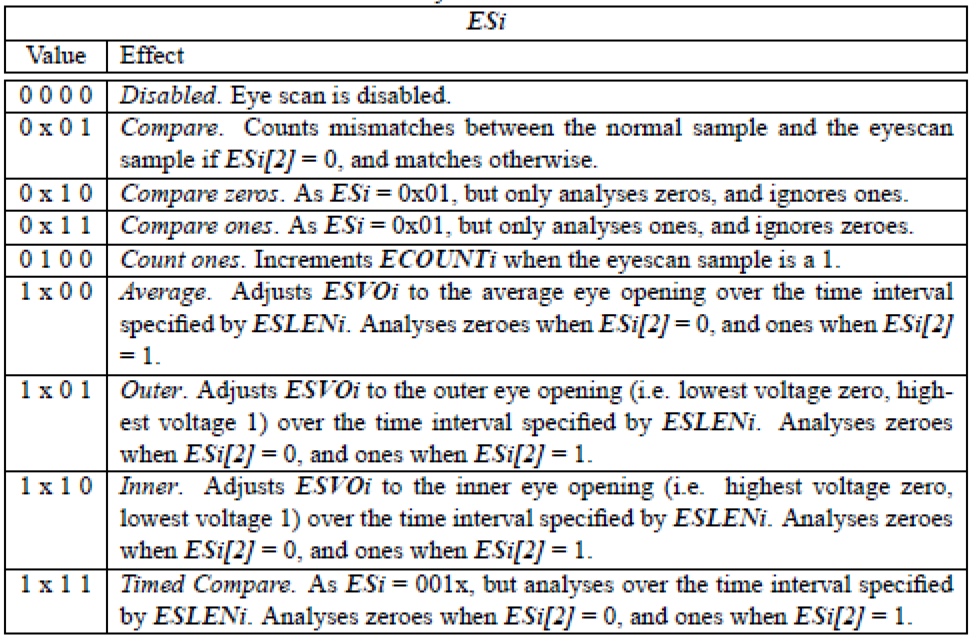
All receive channels provide features which facilitate mapping the received data eye or extracting a symbol response. A number of fields accessible via the IEEE1500 Char scan chain allow the required low level data to be gathered. The process of transforming this data into a map of the eye or a symbol response must then be performed externally, typically in software.

The basic principle used is as follows:

- Enable dedicated eye scan input samplers, and generate an error when the value sampled differs from the normal data sample;
- Apply a voltage offset to the dedicated eye scan input samplers, to effectively reduce their sensitivity;
- Apply a phase offset to adjust the point in the eye that the dedicated eye scan data samples are taken;
- Reset the error counter to remove any false errors accumulated as a result of the voltage or phase offset adjustments;
- Run in this state for a period of time, periodically checking to see if any errors have occured;
- Change voltage and/or phase offset, and repeat.

Alternatively, the algorithm can be configured to optimize the voltage offset at a specified phase offset, over a specified time interval.

Table 8.10: Eyescan Mode Selection



	ESi
Value	Effect
0 0 0 0	*Disabled*. Eye scan is disabled.
0 x 0 1	*Compare*. Counts mismatches between the normal sample and the eyescan sample if *ESi[2]* = 0, and matches otherwise.
0 x 1 0	*Compare zeros*. As *ESi* = 0x01, but only analyses zeros, and ignores ones.
0 x 1 1	*Compare ones*. As *ESi* = 0x01, but only analyses ones, and ignores zeroes.
0 1 0 0	*Count ones*. Increments *ECOUNTi* when the eyescan sample is a 1.
1 x 0 0	*Average*. Adjusts *ESVOi* to the average eye opening over the time interval specified by *ESLENi*. Analyses zeroes when *ESi[2]* = 0, and ones when *ESi[2]* = 1.
1 x 0 1	*Outer*. Adjusts *ESVOi* to the outer eye opening (i.e. lowest voltage zero, highest voltage 1) over the time interval specified by *ESLENi*. Analyses zeroes when *ESi[2]* = 0, and ones when *ESi[2]* = 1.
1 x 1 0	*Inner*. Adjusts *ESVOi* to the inner eye opening (i.e. highest voltage zero, lowest voltage 1) over the time interval specified by *ESLENi*. Analyses zeroes when *ESi[2]* = 0, and ones when *ESi[2]* = 1.
1 x 1 1	*Timed Compare*. As *ESi* = 001x, but analyses over the time interval specified by *ESLENi*. Analyses zeroes when *ESi[2]* = 0, and ones when *ESi[2]* = 1.

Eye scan can be used in both synchronous and asynchronous systems, whilst receiving normal data traffic.

The IEEE1500 Char fields used to directly control eyescan and symbol response extraction are ***ESi***, ***ESWORDi***, ***ES BIT SELECTi***, ***ESLIMi***, ***ESPOi***, ***ESVOi***, ***ESVO OVRi***, ***ESRUNi***, and ***ESDONEi***. ***ESPOi*** and ***ESVOi***. Eye scan errors are accumulated in ***ECOUNTi*** (see section 8.2.1).

The required eyescan mode is selected via the ***ESi*** field, as shown in table 8.10.

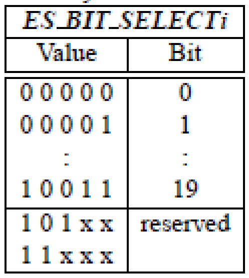
When enabled, only data from the bit position within **rd*i*[19:0]** specified via ***ES BIT SELECTi*** is analysed (see table 8.11). In other words, only eye scan errors associated with data output at this bit position will accumulate in ***ECOUNTi***. The maximum legal ***ES BIT SELECTi*** value is 10011 (bit19) for 20-bit bus width is selected via ***BUSWIDTH***.

Data from all words is analyzed, provided ***ESWORDi*** is set to all 1's. Other values are used for extracting a symbol response.

All the eyescan options apart from setting 0x01 analyze only ones or zeroes. Note that the value of ***INVPAIR*** is applied before deciding whether a bit is a one or a zero.

To build a complete eye, data from all positions within the word and of both polarities should be superimposed. Alternately, assembling eyes based on every second or fourth bit can establish whether there is any duty cycle or quadrature distortion present in the data stream.

Table 8.11: Eye Scan Bit Selection



ES_BIT_SELECTi	
Value	Bit
0 0 0 0 0	0
0 0 0 0 1	1
:	:
1 0 0 1 1	19
1 0 1 x x	reserved
1 1 x x x	

When ***ESi[3]*** = 0, the selected analysis runs continuously. However, when ***ESi[3]*** = 1, only the number of qualified samples specified by ***ESLENi***, as shown in table TBD. In this case, analysis is started by writing a 1 to ***ESRUNi*** (it is not necessary to set it back to 0). When analysis completes, ***ESDONEi*** will be set to 1.

Applications of Voltage Offsets to Determine Eye Height

When ***ESVO OVRi*** = 1, the ***ESVOi*** field determines the amount of offset voltage that is applied to the eye scan data samplers associated with **rxp*i*** and **rxn*i***. The amount of offset is variable between 0 and ±300mV in increments of ˜10mV, as shown in table TBD.

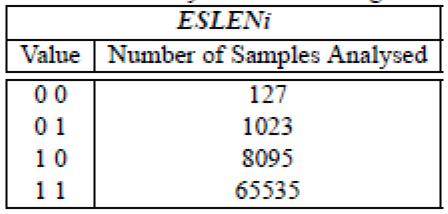


Table 8.12: Eye Scan Run Length

ESLENi	
Value	Number of Samples Analysed
0 0	127
0 1	1023
1 0	8095
1 1	65535

When ***ESi[3]*** = 1, ***ESVO OVRi*** must be 0 to allow the optimized voltage offset to be read back via ***ESVOi***.

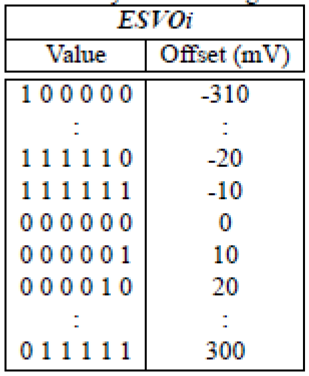


Table 8.13: Eyescan Voltage Offset

ESVOi	
Value	Offset (mV)
1 0 0 0 0 0	-310
:	:
1 1 1 1 1 0	-20
1 1 1 1 1 1	-10
0 0 0 0 0 0	0
0 0 0 0 0 1	10
0 0 0 0 1 0	20
:	:
0 1 1 1 1 1	300

Increasing positive offsets make it progressively more difficult to correctly sample a logical 1, whereas negative offsets make it progressively more difficult to sample a logical 0. This means the number of errors counted for a given polarity will effectively be a factor of two too low because the offset is only affecting half the bits received. It is recommended that measurements be taken using both polarities and combined.

The relationship between voltage offset and ***ESVOi*** is not entirely linear. For applications where high accuracy is required, the offset amount can be calibrated. This is achieved by selecting the **rxp***i* and **rxn***i* offsets for voltage monitoring. The offset for each ***ESVOi*** value can be measured by taking half of the difference between the **rxp***i* and **rxn***i* offsets. These voltage offsets should only be selected for monitoring during calibration. Whilst ***ESi*** 6= 0000, ***RXASELi*** should not be set to 1xx.

Applications of Phase Offsets to Determine Eye Width

The phase position of the samplers associated with **rxp***i* and **rxn***i*, is controlled to a precision of 1/32UI. When ***ESi*** is not 00, the phase position can be adjusted forwards or backwards by more than one UI using the ***ESPOi*** field, as shown in table 8.14.

Table 8.14: Eyescan Phase Offset

ESPOi	
Value	Offset (1/32UI)
0 1 1 1 1 1 1	+63
:	:
0 0 0 0 0 0 1	+1
0 0 0 0 0 0 0	0
1 1 1 1 1 1 1	-1
:	:
1 0 0 0 0 0 0	-64

In normal use, the range should be limited to +/-0.5UI (+15 to -16 phase steps).

Symbol Response Extraction

To extract a symbol response, data must be gathered for each bit in a known repetitive pattern, and then the response deconvolved from the known data pattern in software.

The pattern verifier is used to generate synchronisation pulses which are used as a timing reference. The interval between pulses will be the length of the PRBS pattern selected via ***TESTPATT*** multiplied by the bus width selected via ***BUSWIDTH***. For example, a 2^7-1 pattern with a bus width of 16 will result in a 2032-bit interval.

The transmitting link partner must transmit a known data pattern with a length that corresponds to this interval, or an integer fraction thereof (in which case it must also be divisible by the bus width). Note that the pattern itself does not have to match the PRBS pattern used by the pattern verifier; it just has to be the same length.

Table 8.15: Symbol Response Word Selection

ESWORDi	
Value	Word
0 0 0 0 0 0 0 0	0
0 0 0 0 0 0 0 1	1
:	:
1 1 1 1 1 1 0 1	253
1 1 1 1 1 1 1 0	reserved
1 1 1 1 1 1 1 1	All

The ***ESWORD****i* field is used to specify a word offset with respect to the synchronization pulse, as shown in table TBD. In conjunction with ***ES BIT SELECT****i*, this defines a specific bit offset. Only eye scan errors for the selected bit in the selected word will be accumulated into ***ECOUNT****i*. Note that when using a 2^7-1 PRBS as the timing reference, the maximum useful value of ***ESWORD****i* is 01111110 (126), and when using 2^{23}-1 or 2^{31}-1, only the first 254 words of the pattern can be analysed.

By also using ***ESPO****i* the pattern can be analysed in steps of as little as 1/32UI.

Example Instructions for R0 Block RX ADTM

To perform pattern verification on received data for R0 block, follow the steps below:

Select read-only behavior:
*Instruction register access to select the **ws_cfg** instruction;*
1. Write **B"01101101"** to the JTAG Instruction Register to setup an IEEE1500 IR transaction
2. Write **X"00_35_00"** to the JTAG Data Register. The data is passed as Instruction Codes to IEEE1500 Instruction Registers of R0 block and R1 block respectively. R1 blocks receive no instruction code whereas R0 block receives instruction code **0x35**.

Data register access to set all write enable bits to zero;
3. Write **B" 10011011"** to the JTAG Instruction Register to setup an IEEE1500 DR transaction
4. Write **{0, ws_cfg_head, ws_cfg_rx, ws_cfg_rx, ws_cfg_rx, ws_cfg_rx, ws_cfg_tail}** to the JTAG Data Register.

Read out the value of the core-side inputs:
*Instruction register access to select the **ws_core** instruction;*
5. Write **B"01101101"** to the JTAG Instruction Register to setup an IEEE1500 IR transaction
6. Write **X"00_30_00"** to the JTAG Data Register.

Data register access to read out the value of all the core-side ports;
7. Write **B" 10011011"** to the JTAG Instruction Register to setup an IEEE1500 DR transaction
8. Read JTAG Data Register which contains **ws_core** data.

Select read-write behaviour:
*Instruction register access to select the **ws_cfg** instruction;*
9. Write **B"01101101"** to the JTAG Instruction Register to setup an IEEE1500 IR transaction
10. Write **X"00_35_00"** to the JTAG Data Register.

Data register access to set all CORE WE bits and CHAR WEi to 1;
11. Write **B" 10011011"** to the JTAG Instruction Register to setup an IEEE1500 DR transaction
12. Write **{0, ws_cfg_head, ws_cfg_rx, ws_cfg_rx, ws_cfg_rx, ws_cfg_rx, ws_cfg_tail}** to the JTAG Data Register.

Select PRBS patterns for all receive channels:
*Instruction register access to select the **ws_core** instruction;*
13. Write **B"01101101"** to the JTAG Instruction Register to setup an IEEE1500 IR transaction
14. Write **X"00_30_00"** to the JTAG Data Register.

Data register access to set CFG OVR to 1, TESTPATTi to 100, and all other bits to the values obtained in step 8;
15. Write **B" 10011011"** to the JTAG Instruction Register to setup an IEEE1500 DR transaction
16. Write **{0, ws_core_head, ws_core_rx, ws_core_rx, ws_core_rx, ws_core_rx, ws_core_tail}** to the JTAG Data Register. The data should contain *CFG OVR* and *TESTPATT* bits set to appropriate values.

Check data is PRBS:
Instruction register access to select the ws_char instruction;
17. Write **B"01101101"** to the JTAG Instruction Register to setup an IEEE1500 IR transaction
18. Write **X"00_33_00"** to the JTAG Data Register.

Data register access to clear false errors from ECOUNTi resulting from pattern verification initialization. Discard the value read out;
19. Write **B" 10011011"** to the JTAG Instruction Register to setup an IEEE1500 DR transaction
20. Read JTAG Data Register which contains **ws_char** data.

Wait the desired gating time;
Data register access to read ECOUNTi;
21. Read JTAG Data Register which contains **ws_char** data.

Pattern verification for R1 Block

To perform pattern verification on received data for R1 block, follow the steps below:

Select read-only behavior:
*Instruction register access to select the **ws_cfg** instruction;*
22. Write **B"01101101"** to the JTAG Instruction Register to setup an IEEE1500 IR transaction
23. Write **X"00_00_35"** to the JTAG Data Register. The data is passed as Instruction Codes to IEEE1500 Instruction Registers of R0 block and R1 block respectively. R0 blocks receive the no instruction code whereas R1 block receives instruction code **0x35**.

Data register access to set all write enable bits to zero;
24. Write **B" 10011011"** to the JTAG Instruction Register to setup an IEEE1500 DR transaction
25. Write **{00, ws_cfg_head, ws_cfg_rx, ws_cfg_rx, ws_cfg_rx, ws_cfg_rx, ws_cfg_tail}** to the JTAG Data Register. **ws_cfg_head, ws_cfg_rx** and **ws_cfg_tail** contain the **we** bits that need to be set to zero.

Read out the value of the core-side inputs:
*Instruction register access to select the **ws_core** instruction;*
26. Write **B"01101101"** to the JTAG Instruction Register to setup an IEEE1500 IR transaction
27. Write **X"00_00_30"** to the JTAG Data Register.

Data register access to read out the value of all the core-side ports;
28. Write **B" 10011011"** to the JTAG Instruction Register to setup an IEEE1500 DR transaction
29. Read JTAG Data Register which contains **ws_core** data.

Select read-write behaviour:
*Instruction register access to select the **ws_cfg** instruction;*
30. Write **B"01101101"** to the JTAG Instruction Register to setup an IEEE1500 IR transaction
31. Write **X"00_00_35"** to the JTAG Data Register.

Data register access to set all CORE WE bits and CHAR WEi to 1;
32. Write **B" 10011011"** to the JTAG Instruction Register to setup an IEEE1500 DR transaction
33. Write **{00, ws_cfg_head, ws_cfg_rx, ws_cfg_rx, ws_cfg_rx, ws_cfg_rx, ws_cfg_tail}** to the JTAG Data Register. **ws_cfg_head, ws_cfg_rx** and **ws_cfg_tail** contain the **we** bits that need to be set to one.

Select PRBS patterns for all receive channels:
*Instruction register access to select the **ws_core** instruction;*
34. Write **B"01101101"** to the JTAG Instruction Register to setup an IEEE1500 IR transaction
35. Write **X"00_00_30"** to the JTAG Data Register.

Data register access to set CFG OVR to 1, TESTPATTi to 100, and all other bits to the values obtained in step 8;
36. Write **B" 10011011"** to the JTAG Instruction Register to setup an IEEE1500 DR transaction
37. Write **{00, ws_core_head, ws_core_rx, ws_core_rx, ws_core_rx, ws_core_rx, ws_core_tail}** to the JTAG Data Register. The data should contain *CFG OVR* and *TESTPATT* bits set to appropriate values.

Check data is PRBS:
Instruction register access to select the ws_char instruction;
38. Write **B"01101101"** to the JTAG Instruction Register to setup an IEEE1500 IR transaction
39. Write **X"00_00_33"** to the JTAG Data Register.

Data register access to clear false errors from ECOUNTi resulting from pattern verification initialization. Discard the value read out;
40. Write **B" 10011011"** to the JTAG Instruction Register to setup an IEEE1500 DR transaction
41. Read JTAG Data Register which contains **ws_char** data.

Wait the desired gating time;
Data register access to read ECOUNTi;
42. Read JTAG Data Register which contains **ws_char** data.

Revision History:
Version 1.0 – initial draft by Kang Hsia. Reviewed by Bao Nguyen.
Version 1.1 – added information regarding RXDSEL to TESTFAIL and PATTVSYNC.