HOTLink™ Built-In Self-Test (BIST)

The Cypress CY7B923 and CY7B933 HOTLink™ Transmitter and Receiver offer the system integrator a tool to send data from place to place over a high-speed serial transmission link. They have been designed for easy integration into any data communication subsystem and have a data-interface equivalent to that of a data register or FIFO memory. *Figure 1* shows schematically where HOTLink might fit into a typical system. In this example, data is being sent from one data bus to another over a serial link. The controllers used at each end of this representative data link show the generic functions of all data-link controllers. The protocol controller might be simply a PLD state machine, or a VLSI protocol-specific subsystem. The Bus Interface might be an octal register, or a standard CPU bus controller. The data buffer might be a register in the bus interface or a packet-sized FIFO memory. HOTLink connects these typical logical building-blocks with a high-performance serial data link.

Serial data links are notoriously difficult to design and test. HOTLink simplifies the design problem, and offers the system integrator and end user a simple method to test the finished link. The Built-In Self-Test feature described in this application note gives an unambiguous, real-time, offline test of the entire link. Several ways to test serial links, and serial link components are described using BIST as the grading tool.

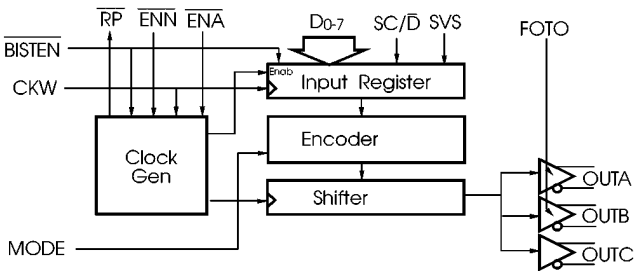

Figure 2. HOTLink Transmitter Block Diagram

The Cypress CY7B923 and CY7B933 HOTLink Transmitter and Receiver block diagrams in *Figures 2* and *3* show the built-in functionality included in these parts. The transmitter uses a fully integrated PLL to multiply the user's byte-rate clock for use as a serial data rate clock. The receiver has a fully integrated PLL that tracks the serial data stream and recovers the bit-rate clock. The bit rate clock is used for internal data decoding and generating the parallel byte-rate clock aligned to the recovered data.

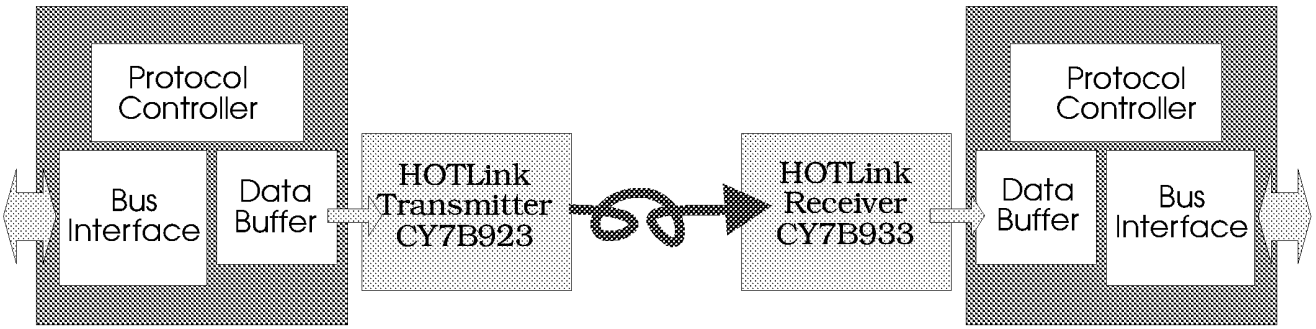

Figure 1. Typical Data-Communication Link using HOTLink

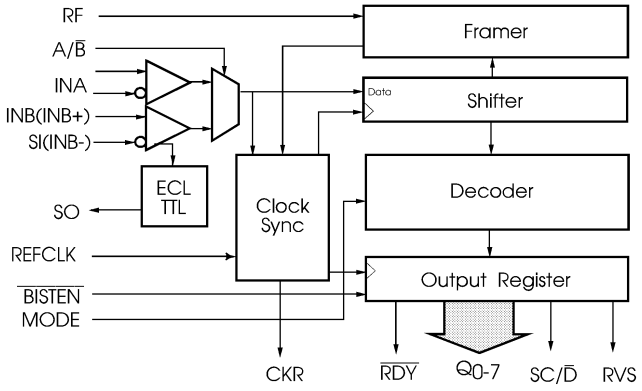

Figure 3. HOTLink Receiver Block Diagram

Both parts have built-in 8B/10B code converters that conform to ANSI X3T11 Fibre Channel and ESCON™ specifications. The encoder and decoder transform 8-bit user data to 10-bit characters more suitable for transmission systems built with fiber-optic cables or any common wire transmission line. Both parts have parallel I/O registers designed to interface with standard FIFO memories without any extra logic. These parts also offer a full-function Built-In Self-Test feature that tests all the circuitry in both ICs and all of the interconnect components that make up the link.

In most traditional serial interface links, the testing of the serial link components and interconnections was done off-line using high-speed analog test equipment and specialized parameter analyzers. If any in-line testing was done, it was done as part of some particular communication protocol or higher-level software functionality. In-line testing usually consisted of only a simple loop-back handshake test that often failed to isolate the problem (if any) to any specific component or connection. And, these simple tests were so superficial that they typically failed to indicate ANY problem at all.

HOTLink products offer the capability to do in-line testing of the entire serial link without adding or removing components from an operational system. The overhead associated with this capability is minimal, and doesn't affect any of the critical analog or high speed interfaces.

Figure 4 shows a link interconnect that is typical for HOTLink systems and illustrates the interface signals used for the BIST operation. This configuration is identical to that of a normally operating data transmission link, except for the simple control features that have been added to facilitate the test function. While the form of the transmission link shown in *Figure 4* illustrates a test being run across an operational communication link, similar testing can be done in either Local-Loop-Back or Long-Loop-Back configurations as shown in *Figures 5* and *6*.

Since there is no required feedback connection between the receiver and transmitter, the BIST function can be used in any configuration that the system designer might find desirable. The previous example (*Figure 4*) illustrates a cooperative test between the two ends of a point-to-point transmission system. The same function can be accomplished as part of the loop-back test of a single communicating node. *Figure 5* shows a typical Local Loop-Back connection wherein the test is run without any involvement of the remote station.

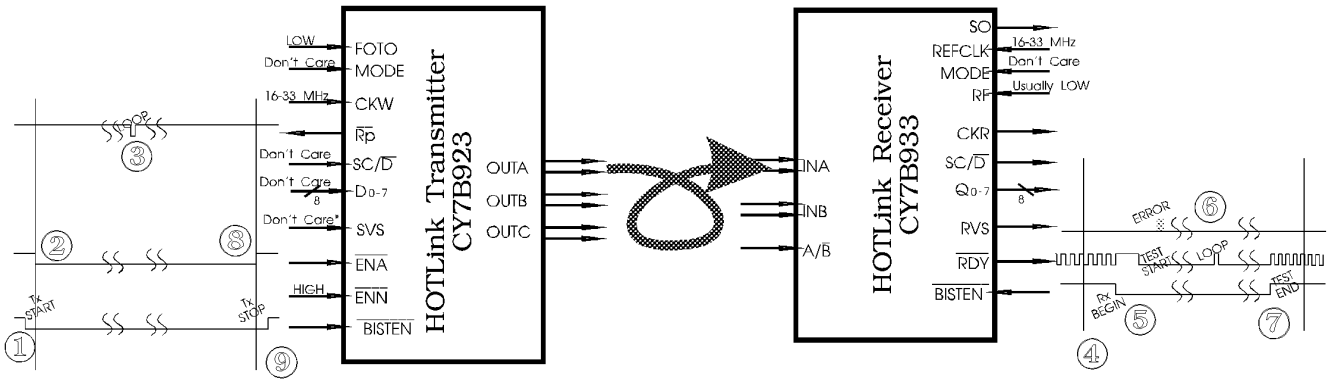

Figure 4. HOTLink BIST Connections

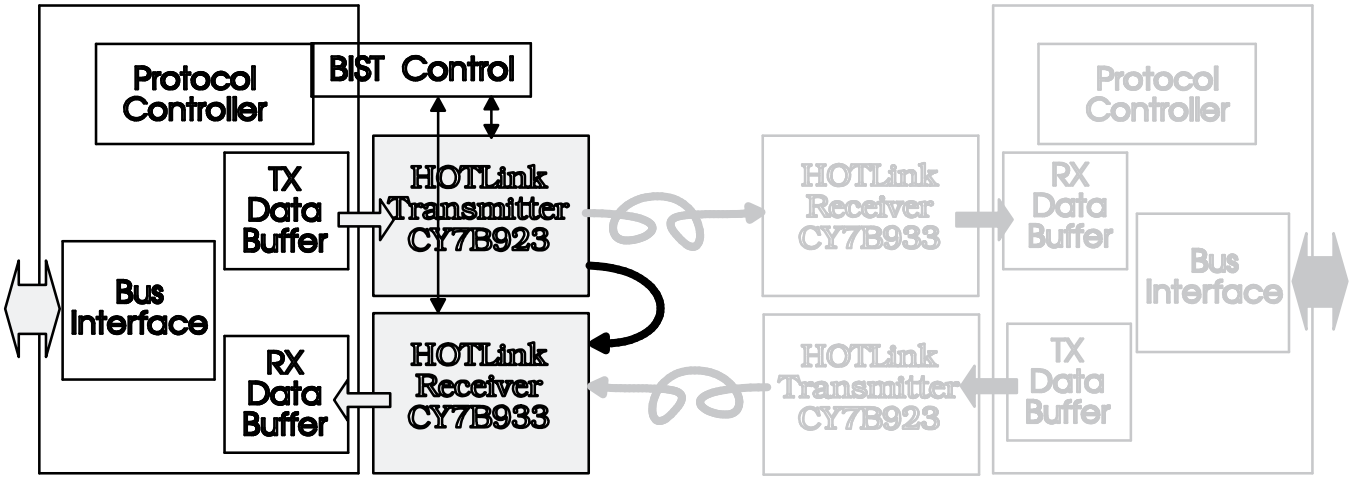

Figure 5. Local Loop Back Testing

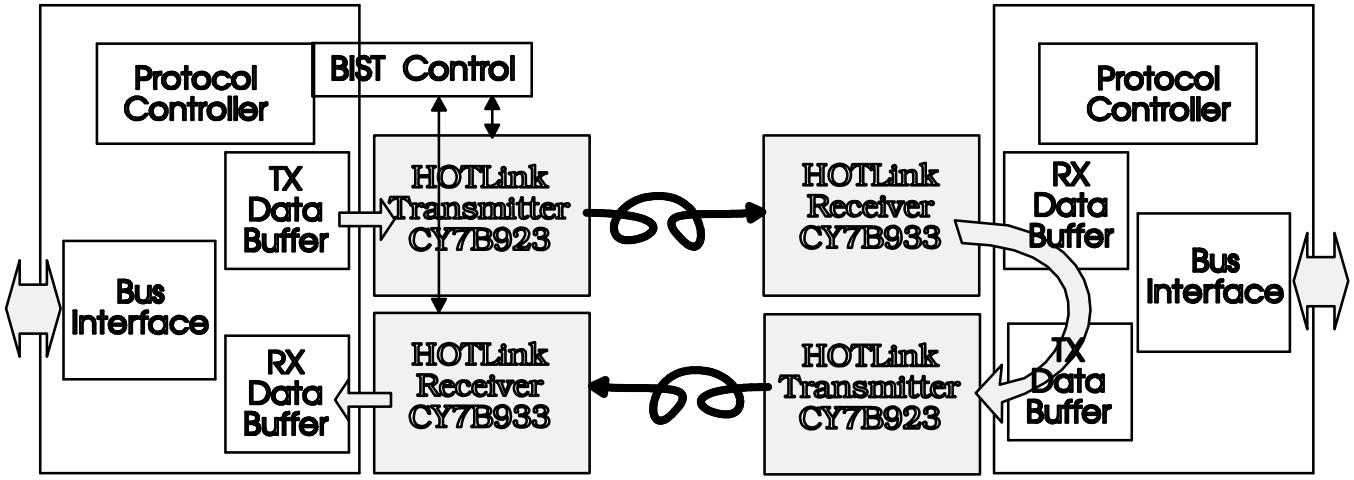

Figure 6. Long Loop Back Testing

A more complete test can be made if it is possible to gain the cooperation of the station at the remote end of the communication link. In the example shown in *Figure 6*, the remote station becomes a simple flow-through repeater. This connection is typical of many network systems, and can be implemented in many simple and obvious ways. This test takes advantage of the orthogonality built into HOTLink products. The data patterns that emerge from the parallel data outputs of the receiver will send equivalent codes when put into the parallel data inputs of the transmitter. This is true in either Bypass Mode or Encoded Mode. The only restriction is that the receiver and transmitter, which are connected via the parallel data path, must both be in the same mode. In the illustration shown in *Figure 6*, the pair of HOTLink chips on the left should be in the same mode, and the pair on the right should be in the same mode. Each pair could be in either Bypass or Encoded mode.

These BIST specific features do not affect the normal message transaction function of the HOTLink's host system. *Figure 7* shows the logical functions that would be added to the protocol controller to support BIST. The timer shown in both the transmitter and receiver BIST control functions might be an analog timer (one-shot) or digital logic that counts BIST-Loops. It could also be simply a user-controlled time-out that performs BIST while a TEST button is pressed.

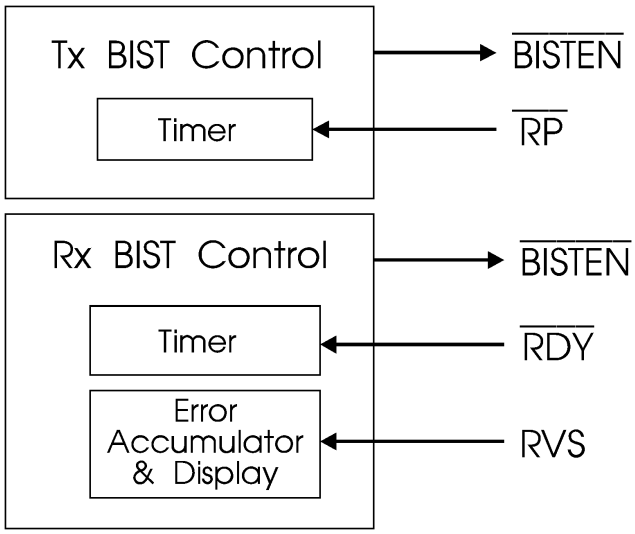

Figure 7. Control Functions for Built-In Self-Test

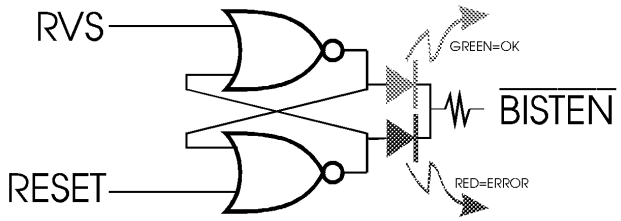

Figure 8. A Simple BIST Error Indicator

The error accumulator and display shown in the example could be as simple or as complex as desired. For a software supported BIST function, the error-count accumulator would be machine readable. For example, the CY9266 HOTLink Evaluation Board provides a two-digit counter that displays the accumulated error count, but in general any indicator will work. *Figure 8* shows a simple RS latch connected to trap errors and display any rare and random RVS indication.

The simple circuit shown above provides an unambiguous indication of errors detected by the BIST comparator and indicated by the RVS output of the HOTLink Receiver. If the $\overline{\text{BISTEN}}$ control is LOW, the LED output indicates the state of the R-S latch that records any pulses on RVS. After $\overline{\text{BISTEN}}$ is set LOW, and the circuit is RESET, the GREEN-LED is illuminated indicating no detected errors. If RVS ever pulses HIGH, then the RED-LED will light and stay illuminated until RESET is pulsed HIGH (probably by a momentary push-button switch or a control signal from the controller). When $\overline{\text{BISTEN}}$ is set HIGH, the LEDs will be extinguished.

The four areas that must be addressed to build BIST functionality into a system involve control of $\overline{\text{BISTEN}}$ pins, selection of transmitter enable pins ($\overline{\text{ENA}}$ or $\overline{\text{ENN}}$), test-progress supervision and error reporting.

The $\overline{\text{BISTEN}}$ control pins enable only the BIST function logic and can be externally controlled without affecting any system resources or timing. The BIST function performs its test sequence automatically, regardless of the encoding mode selected for the operational data link. No consideration needs to be made for this aspect of a HOTLink system.

The HOTLink Transmitter includes two separate enable inputs. The $\overline{\text{ENA}}$ input is used for systems that simultaneously present data and its validation signal (i.e., traditional controllers, and asynchronous FIFOs). The $\overline{\text{ENN}}$ input is used for systems that have a request function and expect the data to be presented on the next clock cycle (i.e., typical of read-request controllers and clocked FIFOs). The system integrator selects the appropriate enable pin to accommodate system functionality and timing. The other enable can be used to control BIST functionality, thus reducing the burden of additional control logic. When $\overline{\text{BISTEN}}$ is enabled (LOW) and one of the enables (either $\overline{\text{ENA}}$ or $\overline{\text{ENN}}$) is asserted (LOW) the HOTLink Transmitter creates a continuous 511 byte (2^9-1 bytes) pseudo-random stream of 8B/10B-encoded data-patterns, which the HOTLink Receiver checks byte-by-byte. The 256 data patterns are sent once each and the 12 special characters and the 4 error codes are sent sixteen times each (except C0.0 which is sent only 15 times) for a total of another 255 data patterns. For a complete list of codes used in the 8B/10B encoder and the special character and error codes, see the CY7B923/933 HOTLink Transmitter/Receiver datasheet.

The system control logic can monitor the test duration at the transmitter by observing the $\overline{RP}$ pin output. This pin is normally used as the byte-rate read-pulse for asynchronous FIFOs, but during the BIST function $\overline{RP}$ will pulse once per loop. By using a counter or timer that observes $\overline{RP}$ while $\overline{BISTEN}$ is LOW, the number of transmitter test-loops can be monitored. (See timing illustration in *Figure 4*, times 1, 2, 3, 8, and 9.)

The $\overline{RDY}$ output of the receiver serves a similar function as $\overline{RP}$ in the transmitter. In normal modes this output is a status indicator used to control the data flow to an external controller or FIFO. $\overline{RDY}$ pulses once per byte in Encoded Mode, and once per SYNC in Bypass Mode. In BIST Mode, it will rest HIGH while the receiver awaits the start of the BIST sequence, and then rest LOW for all but one byte-time of the BIST loop. A counter or timer similar to that described for the transmitter can be used to count when $\overline{RDY}$ pulses HIGH, recording the number of times that the receiver has executed the BIST loop. (See timing illustration in *Figure 4*, times 4, 5, 6 and 7.)

If errors are discovered in the received sequence, received running disparity or received transmission codes, they are flagged by the RVS output. (See timing illustration in *Figure 4*, time 6.) In Encoded Mode this output is not used as part of the communication data path. In Bypass Mode this output serves as the Qj output, but in either case the external BIST control engine will use RVS as an input with a minimal impact on the rest of the operating system.

Transmitter BIST Generator

The BIST generator in the transmitter is the Input register reconfigured into a nine-bit Linear Feedback Shift register (LFSR). In this configuration it makes every possible combination of nine bits (minus 1). The 256 data codes are sent once each, and the 12 special characters and 4 error codes shown in the datasheet are sent multiple times to complete the 511-byte pattern. This is the pattern that is sent regardless of the HOTLink Transmitter and Receiver encoding mode being used by the system. If the system is using the Bypass Mode (MODE input = V_{CC}) the Tx-Encoder and the Rx-Decoder is enabled upon the beginning of BIST Mode ($\overline{BISTEN}$ = LOW and $\overline{ENA}$ or $\overline{ENN}$ = LOW). When BIST Mode ends, the encoder and decoder are bypassed (if MODE = V_{CC}) and user data is directly serialized and deserialized.

Table 1 is a complete list of the codes sent by the transmitter when it has been enabled (either $\overline{ENA}$ or $\overline{ENN}$ held LOW) in BIST Mode ($\overline{BISTEN}$ input LOW). This sequence is a continuously repeating data loop that may start at any point and then continue through all of the codes. There are several places in the sequence where running disparity is explicitly forced, and as a result, each pass through the sequence will repeat exactly. (It is possible to precondition the BIST generator so that, on the first time through, the initial few codes will be sent using the wrong running disparity.) Running disparity will be corrected when one of the force codes (i.e., C1.7 or C2.7) is encountered and will certainly be correct before the next “Start of BIST”.

Once per loop (synchronous with the D0.0 at the beginning of the table), $\overline{RP}$ will pulse (see *Figure 4*, time 3). This BIST specific behavior allows an external system monitor state machine to count the number of times that the transmitter has sent its BIST data. In normal non-BIST operation, $\overline{RP}$ pulses once per byte when $\overline{ENA}$ is asserted, and doesn’t pulse at all otherwise.

Table 1 shows the transmitter BIST sequence expressed in codes that would be commonly sent by a controller connected to the transmitter input pins. This table would be used if a controller was programmed to send the BIST pattern as an external system function or as a test of the BIST pattern checker in the receiver.

In normal operation it is unnecessary for the user to initialize the transmitter BIST sequence since the receiver automatically aligns its code generator to the pattern being sent by the transmitter. If there is some ambiguity in the exact start of the transmitter BIST loop, the receiver will patiently wait for a good start and begin checking from there. If an erroneous code causes the receiver to make a false start, it will soon detect the error and resume waiting for the proper starting code.

Table 1. HOTLink Transmitter BIST Sequence

Start here ---->															
D0.0	C0.0	C4.0	C1.7	C4.7	C0.7	D27.7	D23.3-->	D29.1	D30.0	C0.7	D15.4	D11.2	D3.1	D17.0	D4.0
C8.0	C6.0	C2.7	D14.7	C11.0	D19.5	D21.2	D12.1	C10.0	C3.0	D5.6	D8.3	C2.0	C1.0	D0.6	C0.0
C4.0	C8.0	C2.0	C5.0	D24.7	C6.0	C9.0	D18.6	C5.0	D28.5	C4.7	C11.0	D23.6	D13.3	D26.1	C7.0
D9.4	D2.2	C1.0	D20.4	C1.7	C4.7	C0.7	D11.7	D19.3	D21.1	D28.0	C4.7	C0.7	D15.6	D11.3	D19.1
D21.0	D12.0	C10.0	C7.0	D13.6	D10.3	C3.0	D17.4	D4.2	C8.0	C6.0	C9.0	D6.7	C9.0	D18.5	C5.0
D8.5	C2.0	C1.0	D20.6	C1.7	C4.7	C11.0	D3.7	D17.3	D20.1	C1.7	C10.0	C7.0	D13.7	D26.3	C7.0
D25.4	D6.2	C9.0	D22.4	C2.7	D14.5	C11.0	D3.5	D17.2	D4.1	C8.0	C2.0	C5.0	D12.7	C10.0	C3.0
D17.6	D4.3	C8.0	C2.0	C1.0	D4.7	C8.0	C2.0	C1.0	D0.7	C0.0	C0.0	C0.0	C0.0	C4.0	C8.0
C2.0	***C1.0***	D16.7	C4.0	C8.0	C2.0	C1.0	D20.7	C1.7	C10.0	C3.0	D1.7	D16.3	C4.0	C8.0	C2.0
C5.0	D28.7	C4.7	C11.0	D19.6	D5.3	D24.1	C6.0	C9.0	D6.6	C9.0	D22.5	C2.7	D10.5	C3.0	D1.5
D16.2	C4.0	C1.7	C10.0	C7.0	D29.7	D30.3	C0.7	D27.4	D7.2	D9.1	D18.0	C5.0	D12.4	C10.0	C7.0
D29.6	D14.3	C11.0	D19.4	D5.2	D8.1	C2.0	C1.0	D4.6	C8.0	C6.0	C9.0	D2.7	C1.0	D16.5	C4.0
C8.0	C6.0	C9.0	D22.7	C2.7	D26.5	C7.0	D9.5	D18.2	C5.0	D28.4	C4.7	C0.7	D31.6	D15.3	D27.1
D23.0	D13.0	D10.0	C3.0	D5.4	D8.2	C2.0	C5.0	D8.6	C2.0	C5.0	D24.6	C6.0	C2.7	D26.6	C7.0
D29.5	D30.2	C0.7	D31.4	D15.2	D11.1	D19.0	D5.0	D8.0	C2.0	C5.0	D12.6	C10.0	C7.0	D25.6	D6.3
C9.0	D18.4	C5.0	D12.5	C10.0	C3.0	D21.6	D12.3	C10.0	C3.0	D1.6	D0.3	C0.0	C0.0	C0.0	C4.0
C1.7	C10.0	C3.0	D17.7	D20.3	C1.7	C10.0	C3.0	D5.7	D24.3	C6.0	C9.0	D2.6	C1.0	D20.5	C1.7
C10.0	C7.0	D9.7	D18.3	C5.0	D24.4	C6.0	C2.7	D30.6	C0.7	D31.5	D31.2	D15.1	D27.0	D7.0	D9.0
D2.0	C1.0	D4.4	C8.0	C6.0	C2.7	D10.7	C3.0	D17.5	D20.2	C1.7	C4.7	C11.0	D7.7	D25.3	D22.1
C2.7	D10.4	C3.0	D5.5	D24.2	C6.0	C2.7	D10.6	C3.0	D21.5	D28.2	C4.7	C0.7	D11.6	D3.3	D17.1
D20.0	C1.7	C4.7	C0.7	D15.7	D27.3	D23.1	D29.0	D14.0	C11.0	D7.4	D9.2	D2.1	C1.0	D0.4	C0.0
C4.0	C1.7	C10.0	C7.0	D25.7	D22.3	C2.7	D26.4	C7.0	D13.5	D26.2	C7.0	D29.4	D14.2	C11.0	D23.4
D13.2	D10.1	C3.0	D1.4	D0.2	C0.0	C4.0	C8.0	C6.0	C2.7	D26.7	C7.0	D25.5	D22.2	C2.7	D30.4
C0.7	D15.5	D27.2	D7.1	D25.0	D6.0	C9.0	D6.4	C9.0	D6.5	C9.0	D2.5	C1.0	D0.5	C0.0	C0.0
C4.0	C8.0	C6.0	C9.0	D18.7	C5.0	D24.5	C6.0	C9.0	D22.6	C2.7	D30.5	C0.7	D11.5	D19.2	D5.1
D24.0	C6.0	C2.7	D14.6	C11.0	D23.5	D29.2	D14.1	C11.0	D3.4	D1.2	D0.1	C0.0	C0.0	C4.0	C1.7
C4.7	C11.0	D19.7	D21.3	D28.1	C4.7	C11.0	D7.6	D9.3	D18.1	C5.0	D8.4	C2.0	C5.0	D28.6	C4.7
C0.7	D27.6	D7.3	D25.1	D22.0	C2.7	D14.4	C11.0	D7.5	D25.2	D6.1	C9.0	D2.4	C1.0	D4.5	C8.0
C2.0	C5.0	D8.7	C2.0	C1.0	D16.6	C4.0	C1.7	C10.0	C3.0	D21.7	D28.3	C4.7	C11.0	D3.6	D1.3
D16.1	C4.0	C8.0	C6.0	C2.7	D30.7	C0.7	D27.5	D23.2	D13.1	D26.0	C7.0	D13.4	D10.2	C3.0	D21.4
D12.2	C10.0	C7.0	D9.6	D2.3	C1.0	D16.4	C4.0	C1.7	C4.7	C11.0	D23.7	D29.3	D30.1	C0.7	D11.4
D3.2	D1.1	D16.0	C4.0	C1.7	C4.7	C0.7	D31.7	D31.3	D31.1	D31.0	D15.0	D11.0	D3.0	D1.0	GOTO Start

If the user wants to initialize the BIST pattern, there are two methods available. In the Encoded Mode (MODE input=GND) the SVS input overrides the BIST sequence and forces the transmitter to send the code indicating a Code Rule Violation (see the CY7B923/933 HOTLink datasheet for a complete list of 8B/10B data codes and special characters). It also resets the BIST LFSR to its initial state (D0.0). (Running disparity is not explicitly set by SVS, and the first few bytes after its release may be sent with the wrong disparity. The fourth byte of the sequence, C1.7, will explicitly set running disparity for the rest of the patterns.)

Alternatively, the transmitter BIST LFSR can be forced to start its pattern from any other point in the sequence by noting that the BIST sequence proceeds from the code that was in the Input register when BIST was asserted. (Note that the BIST sequence generator state sequence is expressed in En-

coded-Mode terms. If the transmitter is in Bypass Mode, the Next State can be deciphered by converting the actual bit pattern on the inputs to the code that the transmitter would send if it were in the Encoded Mode and that were its input pattern, and then looking for that code in the table.) In most cases this will be sufficient to initialize the sequence, except for the state of the internal running disparity flip-flop. If running disparity must also be assured, the codes for +K28.5 (C2.7) or −K28.5 (C1.7) should be used to initialize the LFSR. The C2.7 and C1.7 starting locations are shown in *Table 1* (row 9, items 1 and 2 respectively). The user would put, for example, C2.7 on the transmitter inputs for one or more byte times, then start the BIST test. The pattern would start from row 9, item 1 in this example. This technique can be useful if the user wants only a portion of the transmitter BIST pattern for some oscilloscope test.

Receiver BIST Comparator

The BIST generator in the receiver is the Output Register reconfigured into a nine-bit Linear-Feedback-Shift-Register (LFSR) that exactly matches the one in the transmitter. In this configuration it puts all possible combinations of nine bits on the receiver outputs (Q_{0-7} and SC/$\overline{D}$). *Table 2* is a complete list of the codes that must appear on the receiver serial inputs during a BIST test-loop if the receiver is to indicate "no-errors". These codes are slightly different than those shown in *Table 1* (e.g., the fourth element in *Table 1* = C1.7 = K28.5 with forced negative running disparity, the fourth element in *Table 2*, = C5.0 = correct K28.5) because of the way that the running disparity affects the interpretation made by the receiver when it decodes certain characters. This table shows the codes that would be sent by a controller sending the BIST sequence without depending upon the LFSR in the transmitter, or by a transmitter connected to an Encoded Mode receiver that was receiving a BIST sequence while not itself in BIST mode.

It should be noted that the first K28.5 (C5.0) (the fourth byte in the BIST loop at *1 in *Table 2*) may cause a false RVS indication on the first pass through the BIST loop, because the transmitter sends it as a C1.7 (−K28.5) to force the state of running disparity (RD). Depending upon the actual RD state in the receiver as BIST starts, this forced RD might appear to the receiver as a running disparity error on the first time through the BIST loop. All subsequent loops are interpreted correctly.

The SVS character (C0.7) combined with the adjacent D11.5 (*2 in *Table 2*, row 25 bytes 13 and 14) encoded as (011000 **0 0111 11010**0 1010) creates an alias K28.5 (001111 1010) which will cause an erroneous reframe if RF is HIGH for short periods of time (less than 2048 bytes). This alias sync can be used to check the system response to clock stretching, a topic that will be covered later. This error (normal single-byte reframe behavior) will not occur when Multi-Byte-Sync is enabled (i.e., RF = HIGH for more than 2048 byte times).

Note that there are several intentional code rule violations and incorrect running disparity transitions included in the receiver sequence. RVS (during BIST) will indicate when an error in the expected code has been detected; it does not indicate that an illegal code is present. *Figure 9* illustrates this behavior and shows the timing of RVS when the receiver detects an error in the sequence. RVS will pulse HIGH for the byte time following detection of a mismatch between the received-decoded pattern and the internally generated code.

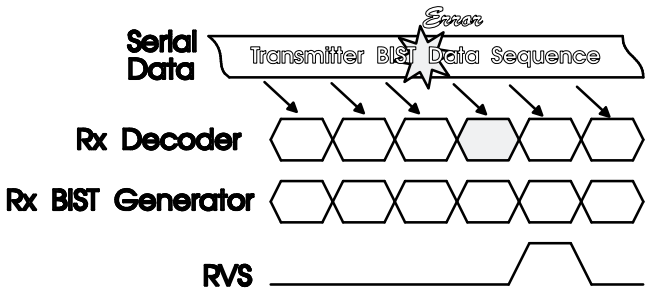

Figure 9. RVS Indicates Errors in Received Sequence

Table 2. HOTLink Receiver Input BIST Sequence

Start here −−−−>															
D0.0	C0.0	C4.0	**C5.0**[*1]	C4.7	C0.7	D27.7	D23.3 −−>	D29.1	D30.0	C0.7	D15.4	D11.2	D3.1	D17.0	D4.0
C8.0	C6.0	C2.7	D14.7	C11.0	D19.5	D21.2	D12.1	C10.0	C3.0	D5.6	D8.3	C2.0	C1.0	D0.6	C0.0
C4.0	C8.0	C2.0	C5.0	D24.7	C6.0	C9.0	D18.6	C5.0	D28.5	C4.7	C11.0	D23.6	D13.3	D26.1	C7.0
D9.4	D2.2	C1.0	D20.4	C5.0	C4.7	C0.7	D11.7	D19.3	D21.1	D28.0	C4.7	C0.7	D15.6	D11.3	D19.1
D21.0	D12.0	C10.0	C7.0	D13.6	D10.3	C3.0	D17.4	D4.2	C8.0	C6.0	C9.0	D6.7	C9.0	D18.5	C5.0
D8.5	C2.0	C1.0	D20.6	C1.7	C4.7	C11.0	D3.7	D17.3	D20.1	C1.7	C10.0	C7.0	D13.7	D26.3	C7.0
D25.4	D6.2	C9.0	D22.4	C2.7	D14.5	C11.0	D3.5	D17.2	D4.1	C8.0	C2.0	C5.0	D12.7	C10.0	C3.0
D17.6	D4.3	C8.0	C2.0	C1.0	D4.7	C8.0	C2.0	C1.0	D0.7	C0.0	C0.0	C0.0	C0.0	C4.0	C8.0
C2.0	C1.0	D16.7	C4.0	C8.0	C2.0	C1.0	D20.7	C1.7	C10.0	C3.0	D1.7	D16.3	C4.0	C8.0	C2.0
C5.0	D28.7	C4.7	C11.0	D19.6	D5.3	D24.1	C6.0	C9.0	D6.6	C9.0	D22.5	C5.0	D10.5	C3.0	D1.5
D16.2	C4.0	C1.7	C10.0	C7.0	D29.7	D30.3	C0.7	D27.4	D7.2	D9.1	D18.0	C5.0	D12.4	C10.0	C7.0
D29.6	D14.3	C11.0	D19.4	D5.2	D8.1	C2.0	C1.0	D4.6	C8.0	C6.0	C9.0	D2.7	C1.0	D16.5	C4.0
C8.0	C6.0	C9.0	D22.7	C2.7	D26.5	C7.0	D9.5	D18.2	C5.0	D28.4	C4.7	C0.7	D31.6	D15.3	D27.1
D23.0	D13.0	D10.0	C3.0	D5.4	D8.2	C2.0	C5.0	D8.6	C2.0	C5.0	D24.6	C6.0	C2.7	D26.6	C7.0
D29.5	D30.2	C0.7	D31.4	D15.2	D11.1	D19.0	D5.0	D8.0	C2.0	C5.0	D12.6	C10.0	C7.0	D25.6	D6.3
C9.0	D18.4	C5.0	D12.5	C10.0	C3.0	D21.6	D12.3	C10.0	C3.0	D1.6	D0.3	C0.0	C0.0	C0.0	C4.0
C1.7	C10.0	C3.0	D17.7	D20.3	C1.7	C10.0	C3.0	D5.7	D24.3	C6.0	C9.0	D2.6	C1.0	D20.5	C1.7
C10.0	C7.0	D9.7	D18.3	C5.0	D24.4	C6.0	C2.7	D30.6	C0.7	D31.5	D31.2	D15.1	D27.0	D7.0	D9.0
D2.0	C1.0	D4.4	C8.0	C6.0	C2.7	D10.7	C3.0	D17.5	D20.2	C5.0	C4.7	C11.0	D7.7	D25.3	D22.1
C5.0	D10.4	C3.0	D5.5	D24.2	C6.0	C2.7	D10.6	C3.0	D21.5	D28.2	C4.7	C0.7	D11.6	D3.3	D17.1
D20.0	C1.7	C4.7	C0.7	D15.7	D27.3	D23.1	D29.0	D14.0	C11.0	D7.4	D9.2	D2.1	C1.0	D0.4	C0.0
C4.0	C5.0	C10.0	C7.0	D25.7	D22.3	C2.7	D26.4	C7.0	D13.5	D26.2	C7.0	D29.4	D14.2	C11.0	D23.4
D13.2	D10.1	C3.0	D1.4	D0.2	C0.0	C4.0	C8.0	C6.0	C2.7	D26.7	C7.0	D25.5	D22.2	C5.0	D30.4
C0.7	D15.5	D27.2	D7.1	D25.0	D6.0	C9.0	D6.4	C9.0	D6.5	C9.0	D2.5	C1.0	D0.5	C0.0	C0.0
C4.0	C8.0	C6.0	C9.0	D18.7	C5.0	D24.5	C6.0	C9.0	D22.6	C5.0	D30.5	**C0.7**[*2]	D11.5	D19.2	D5.1
D24.0	C6.0	C2.7	D14.6	C11.0	D23.5	D29.2	D14.1	C11.0	D3.4	D1.2	D0.1	C0.0	C0.0	C4.0	C1.7
C4.7	C11.0	D19.7	D21.3	D28.1	C4.7	C11.0	D7.6	D9.3	D18.1	C5.0	D8.4	C2.0	C5.0	D28.6	C4.7
C0.7	D27.6	D7.3	D25.1	D22.0	C2.7	D14.4	C11.0	D7.5	D25.2	D6.1	C9.0	D2.4	C1.0	D4.5	C8.0
C2.0	C5.0	D8.7	C2.0	C1.0	D16.6	C4.0	C5.0	C10.0	C3.0	D21.7	D28.3	C4.7	C11.0	D3.6	D1.3
D16.1	C4.0	C8.0	C6.0	C5.0	D30.7	C0.7	D27.5	D23.2	D13.1	D26.0	C7.0	D13.4	D10.2	C3.0	D21.4
D12.2	C10.0	C7.0	D9.6	D2.3	C1.0	D16.4	C4.0	C5.0	C4.7	C11.0	D23.7	D29.3	D30.1	C0.7	D11.4
D3.2	D1.1	D16.0	C4.0	C5.0	C4.7	C0.7	D31.7	D31.3	D31.1	D31.0	D15.0	D11.0	D3.0	D1.0	GOTO Start

When the receiver $\overline{\text{BISTEN}}$ input is set LOW, it initializes its BIST pattern generator and begins searching for the start of the transmitter BIST pattern (see *Figure 4*, time 5). While it is awaiting this start code, $\overline{\text{RDY}}$ and RVS will be HIGH. When it finds the beginning of the pattern (a D1.0 followed by a D0.0, with the proper running disparity), first RVS falls then $\overline{\text{RDY}}$ falls. $\overline{\text{RDY}}$ will remain LOW for the next 510 bytes of the BIST loop and then pulse HIGH for one byte time (see *Figure 4*, time 6). This BIST specific behavior of the $\overline{\text{RDY}}$ output allows an external system monitor state machine to count the number of times that the receiver has checked the BIST data. In non-BIST modes, $\overline{\text{RDY}}$ pulses once per byte in Encoded mode, or once per K28.5 in Bypass mode.

The actual bit pattern appearing at the receiver outputs (Q0−7, and SC/$\overline{\text{D}}$) will match the decoder output for all data patterns, but may not match the data-

sheet pattern for all of the Special Character codes being received. *Table 3* shows the patterns which will appear at the outputs of the receiver while BIST is running. Many of the codes shown do not appear in the datasheet and no correlation should be inferred between these output patterns and the reserved codes mentioned therein. Likewise, if these codes are presented to a transmitter, it will not send the codes necessary to create a good BIST pattern. These codes will typically be monitored by a logic analyzer, and might assist in debugging a particular serial link error phenomenon.

Table 3. HOTLink Receiver Output Patterns during a BIST Sequence

D0.0	C16.0	C20.4	C28.6	C30.7	C15.7	D27.7	D23.3	D29.1	D30.0	C31.0	D15.4	D11.2	D3.1	D17.0	D4.0
C24.0	C22.4	C29.6	D14.7	C11.3	D19.5	D21.2	D12.1	C10.0	C19.4	D5.6	D8.3	C2.1	C1.4	D0.6	C16.3
C4.5	C8.6	C18.7	C5.7	D24.7	C6.3	C9.5	D18.6	C21.3	D28.5	C14.2	C27.5	D23.6	D13.3	D26.1	C7.0
D9.4	D2.2	C17.1	D20.4	C28.2	C30.5	C15.6	D11.7	D19.3	D21.1	D28.0	C30.0	C31.4	D15.6	D11.3	D19.1
D21.0	D12.0	C26.0	C23.4	D13.6	D10.3	C3.1	D17.4	D4.2	C24.1	C6.4	C25.6	D6.7	C9.3	D18.5	C5.2
D8.5	C2.2	C17.5	D20.6	C28.3	C14.5	C11.6	D3.7	D17.3	D20.1	C12.0	C26.4	C23.6	D13.7	D26.3	C7.1
D25.4	D6.2	C25.1	D22.4	C29.2	D14.5	C11.2	D3.5	D17.2	D4.1	C8.0	C18.4	C21.6	D12.7	C10.3	C3.5
D17.6	D4.3	C8.1	C2.4	C17.6	D4.7	C8.3	C2.5	C1.6	D0.7	C0.3	C0.5	C0.6	C16.7	C4.7	C8.7
C2.7	C1.7	D16.7	C4.3	C8.5	C2.6	C17.7	D20.7	C12.3	C10.5	C3.6	D1.7	D16.3	C4.1	C8.4	C18.6
C21.7	D28.7	C14.3	C11.5	D19.6	D5.3	D24.1	C6.0	C25.4	D6.6	C25.3	D22.5	C13.2	D10.5	C3.2	D1.5
D16.2	C20.1	C12.4	C26.6	C23.7	D29.7	D30.3	C15.1	D27.4	D7.2	D9.1	D18.0	C21.0	D12.4	C26.2	C23.5
D29.6	D14.3	C11.1	D19.4	D5.2	D8.1	C2.0	C17.4	D4.6	C24.3	C6.5	C9.6	D2.7	C1.3	D16.5	C4.2
C24.5	C6.6	C25.7	D22.7	C13.3	D26.5	C7.2	D9.5	D18.2	C21.1	D28.4	C30.2	C31.5	D31.6	D15.3	D27.1
D23.0	D13.0	D10.0	C19.0	D5.4	D8.2	C18.1	C5.4	D8.6	C18.3	C5.5	D24.6	C22.3	C13.5	D26.6	C23.3
D29.5	D30.2	C31.1	D31.4	D15.2	D11.1	D19.0	D5.0	D8.0	C18.0	C21.4	D12.6	C26.3	C7.5	D25.6	D6.3
C9.1	D18.4	C21.2	D12.5	C10.2	C19.5	D21.6	D12.3	C10.1	C3.4	D1.6	D0.3	C0.1	C0.4	C16.6	C20.7
C12.7	C10.7	C3.7	D17.7	D20.3	C12.1	C10.4	C19.6	D5.7	D24.3	C6.1	C9.4	D2.6	C17.3	D20.5	C12.2
C26.5	C7.6	D9.7	D18.3	C5.1	D24.4	C22.2	C29.5	D30.6	C31.3	D31.5	D31.2	D15.1	D27.0	D7.0	D9.0
D2.0	C17.0	D4.4	C24.2	C22.5	C13.6	D10.7	C3.3	D17.5	D20.2	C28.1	C14.4	C27.6	D7.7	D25.3	D22.1
C13.0	D10.4	C19.2	D5.5	D24.2	C22.1	C13.4	D10.6	C19.3	D21.5	D28.2	C30.1	C15.4	D11.6	D3.3	D17.1
D20.0	C28.0	C30.4	C31.6	D15.7	D27.3	D23.1	D29.0	D14.0	C27.0	D7.4	D9.2	D2.1	C1.0	D0.4	C16.2
C20.5	C12.6	C26.7	C7.7	D25.7	D22.3	C13.1	D26.4	C23.2	D13.5	D26.2	C23.1	D29.4	D14.2	C27.1	D23.4
D13.2	D10.1	C3.0	D1.4	D0.2	C16.1	C4.4	C24.6	C22.7	C13.7	D26.7	C7.3	D25.5	D22.2	C29.1	D30.4
C31.2	D15.5	D27.2	D7.1	D25.0	D6.0	C25.0	D6.4	C25.2	D6.5	C9.2	D2.5	C1.2	D0.5	C0.2	C16.5
C4.6	C24.7	C6.7	C9.7	D18.7	C5.3	D24.5	C6.2	C25.5	D22.6	C29.3	D30.5	C15.2	D11.5	D19.2	D5.1
D24.0	C22.0	C29.4	D14.6	C27.3	D23.5	D29.2	D14.1	C11.0	D3.4	D1.2	D0.1	C0.0	C16.4	C20.6	C28.7
C14.7	C11.7	D19.7	D21.3	D28.1	C14.0	C27.4	D7.6	D9.3	D18.1	C5.0	D8.4	C18.2	C21.5	D28.6	C30.3
C15.5	D27.6	D7.3	D25.1	D22.0	C29.0	D14.4	C27.2	D7.5	D25.2	D6.1	C9.0	D2.4	C17.2	D4.5	C8.2
C18.5	C5.6	D8.7	C2.3	C1.5	D16.6	C20.3	C12.5	C10.6	C19.7	D21.7	D28.3	C14.1	C11.4	D3.6	D1.3
D16.1	C4.0	C24.4	C22.6	C29.7	D30.7	C15.3	D27.5	D23.2	D13.1	D26.0	C23.0	D13.4	D10.2	C19.1	D21.4
D12.2	C26.1	C7.4	D9.6	D2.3	C1.1	D16.4	C20.2	C28.5	C14.6	C27.7	D23.7	D29.3	D30.1	C15.0	D11.4
D3.2	D1.1	D16.0	C20.0	C28.4	C30.6	C31.7	D31.7	D31.3	D31.1	D31.0	D15.0	D11.0	D3.0	D1.0	GOTO Start

BIST Auto-Abort and Restart

When the receiver detects an error in the received expected sequence of transmission codes it will assert RVS during the byte-time following the error. A normally operating system will rarely experience one error per hour (a bit error rate of $1x10^{-12} \approx$ 1 error/hour @ 266 Mbaud), and systems doing some kind of design tolerance or performance limit testing will usually run with less than a few errors per second (BER of $1x10^{-8} \approx$ 3 error/second @ 266 Mbaud) even during link length testing. At these rates, it can be assumed that each error flagged by RVS was caused by an error that corrupts a single bit. It is impossible to distinguish between single-bit errors and multiple-bit errors within a single byte, since errors are only reported on a byte-by-byte basis. Further, since many kinds of errors change a legal data-byte into another legal byte many errors will be reported at times unrelated to when the error occurred. Single-bit errors can cause changes in the data stream running disparity, and will be detected as errors in the forced running disparity codes.

In extreme cases, where the errors cause PLL cycle slipping, or loss of framing, it is possible to create ambiguous error indications and seemingly endless running error sequences. Once the bit sequence has been corrupted, or after the PLL has bit-slipped, the BIST comparator will indicate a 100% error rate (except for the 32 expected violations that occur as part of the BIST pattern).

Since the BIST generator is a free-running counter that is only initialized while it awaits the start of the transmitter BIST sequence, errors of any kind don't affect the LFSR sequence. This feature can be used to advantage for several types of testing that generate long sequences of errors, since when the errors are removed, the receiver BIST generator predicted data will eventually match the received serial digital data without having to be realigned. Unfortunately, if the error causes the PLL to slip a bit, the received stream will never match.

To account for any loss of BIST sequence condition, the BIST logic included in the receiver will abort an extremely damaged sequence. It will abandon the current sequence and search for the start-of-BIST character and then resume comparisons from the beginning. When this auto-abort happens, $\overline{RDY}$ will go HIGH and remain there until the beginning of a new sequence is detected. While the receiver is waiting, RVS will also remain HIGH. The criteria for Auto-abort requires that there be $\geq$16 RVS indications within 32 contiguous bytes, and is checked every 64 bytes.

For system tests where the user wants to use the BIST comparator to check for longer running errors (and receiver PLL recovery without slipping) it is possible to disable the auto-abort function. The counter that is used to sample the error counter runs on REFCLK. By disconnecting the REFCLK input from the receiver after the PLL has reached the correct operating frequency, the internal counters that manage the error monitor are disabled. (There is a 50/50 probability that when REFCLK is disabled, the auto-abort counter will still be enabled, but by reconnecting, and then disconnecting the REFCLK the auto-abort function can be disabled. The function is controlled by an internal REFCLK divided-by-64 counter. For the first 32-byte times, auto-abort is enabled, for the other 32-byte times it is disabled.)

Tests Using BIST

Built-In Self-Test is a valuable and versatile tool for performing offline-test in any system. It also offers an unambiguous method to examine the performance of HOTLink products and other serial link components. The following short test descriptions are intended to introduce the reader to the capabilities of HOTLink and BIST as an evaluation tool. The tests described are typical of those required to evaluate most physical layer components.

Transmission Line Length

To check for the maximum transmission line length over which HOTLink can communicate, it is only necessary to connect the selected transmission line between a HOTLink Transmitter and Receiver. Most transmission line testing uses arbitrary data patterns that represent typical communication patterns. The HOTLink transmitter and receiver BIST function serves this purpose so the user can check

for an acceptable error rate without extra test equipment and without reconfiguration of an operational link just to perform this test. Transmission lines can be extended or modified until RVS indicates an unacceptable error rate. Tests that might use BIST to indicate system margin include;

- fiber-optic optical attenuation budget and optical or electrical margin testing;
- wire transmission-line attenuation, crosstalk, emissions and noise susceptibility testing;
- electrical interface connections and signal-margin testing;
- data sources for serial interconnect hardware testing.

Rx Jitter Tolerance

The ultimate performance of any serial link is determined by the performance of the receiver. The function of the receiver is to recover data from a (seemingly arbitrary) serial data stream. This data stream is translated several times, coupled to and though several non-linear devices and subjected to all manners of distortion. The receiver must accept this serial pulse train and recover a high-speed bit synchronous clock, de-jitter it, and then separate the DATA from the CLOCK. Jitter tolerance is the typical term used for this function. HOTLink receiver jitter tolerance can be measured by connecting a suitable transmission media between the transmitter and receiver, and inserting a jitter generation source similar to that shown in *Figure 10*. By inserting measured jitter amplitudes and watching the RVS output of the receiver, jitter tolerance can be measured.

There are two basic types of jitter that must be accommodated, deterministic jitter and random jitter. Deterministic jitter is comprised of data dependent jitter (DDJ) and duty cycle distortion (DCD). DDJ is caused by imperfections in the serial link that cause signal corruption that is proportional (or at least a strong function of) the particular data stream. DCD is caused by imperfection or imbalances in the serial link that cause signal corruption that is related to the timing of the rising or the falling edges. Random jitter is unrelated to the data stream, the edge rates, or the link quality. It is typically caused by external noise events or by thermal noise in the optical components. Random jitter is uncorrelated to the data stream and is difficult to reproduce experimentally.

DCD creates high-frequency jitter at about the bit rate of the serial data stream since bit placement errors are complemented within the pulse that is distorted. DDJ creates high-frequency jitter at about the bit rate of the serial data stream since bit-placement errors are usually complemented within a bit or two. These high-frequency jitter components should be filtered by the PLL filter, and should

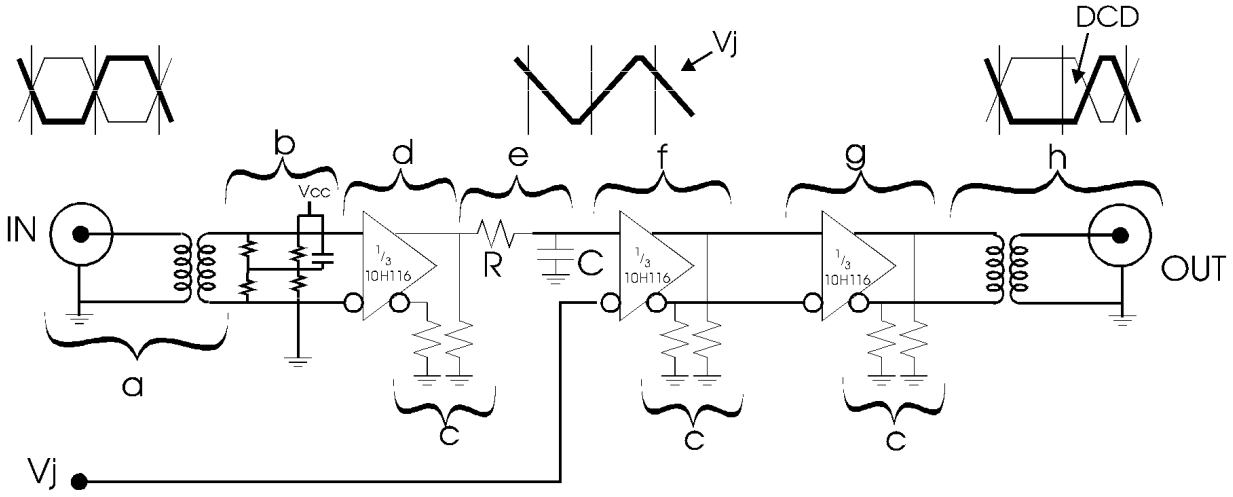

Figure 10. Jitter Generator Schematic

cause no significant jitter at the CKR output of the receiver. DDJ can cause baseline-wander at about the byte rate of the serial data stream, but since the 8B/10B code is balanced over multiple bytes, there should be little or no low-frequency components in the jitter. Random jitter has both high- and low-frequency components, and will cause output jitter as it causes the PLL to attempt to track a corrupted data stream. All three types of jitter must be accommodated by the receiver as it captures the data and aligns its serial clock.

Data dependent jitter can be generated by a suitable length of coaxial cable. If DDJ and input amplitude must be separately measured, an external line receiver and level restoration circuit might be needed. Duty cycle distortion can be generated by the circuit shown in *Figure 10*. This circuit uses the stages in a 10H116 (ECL triple-differential amplifier) to perform; (1) Differential-to-single-ended transformation; (2) Ramp generation; (3) Threshold shifting; (4) Level restoration; (5) Differential buffering. In this circuit the transmitter data stream is fed through the jitter generator while the receiver monitors and checks for correct operation. As the control voltage (Vj) input is varied between the 10KH Vil and Vih levels, the duty cycle of the data stream is corrupted in a repeatable and measurable manner.

Serial-data input to the jitter generator can use any appropriate connector, and coupling circuit. The connector and transformer shown at (a & h) will work with coaxial cable or STP cables. For fiber-optic interfaces, these could be eliminated by direct coupling to fiber-optic receiver/transmitter modules. Transmission-line termination and DC threshold adjustments are performed by the simple network shown at (b). The first differential stage of the ’116 (d) is used as a differential-to-single-ended converter with a controlled output impedance and symmetrical rise and fall times. The ECL output termination resistors shown at the outputs of each differential stage (c) may be replaced with parallel termination resistors if better impedance control or closer edge-rate matching is required. The R-C ramp generator at (e) must be tuned to each data rate, to insure that 100% voltage swing is maintained for the narrowest pulses expected. If the Ramp is too long, it will be possible to raise Vj above the level of some data bits, thus losing data. The second differential stage of the ’116 (f) serves as a voltage comparator between the control voltage (Vj) level and the level of the signal at the output of the ramp-generator. Additional DC-filtering may be required between the Vj input and its input to (f) to insure that high-frequency, single-ended noise does not corrupt the data flow. The third differential stage of the ’116 (g) is used to restore crisp-edged, full-swing levels to the serial data, and to drive the subsequent transmission line. Further details on the fabrication of the jitter generator and the measurement techniques required for accurate measurement of this injected jitter is beyond the scope of this note.

Receiver Error-Free-Window Test

A normally operating receiver PLL will adjust its internal clock such that incoming data transitions are placed at the maximum distance from the data-separator flip-flop sampling window. This placement allows misplaced transitions (jittered edges) the maximum margin before data-misinterpretation occurs. The width of this error free zone is commonly called error-free-window. It is less than the actual bit width (expressed in nanoseconds) by the sum of maximum peak-to-peak receiver PLL jitter, data-separator flip-flop sampling window width, and absolute misalignment of the internal PLL sampling clock. (Actual test results will be additionally affected by clock source jitter and test equipment trigger and measurement inaccuracies.)

To measure the error-free-window (EFW) in the HOTLink Receiver, it is only necessary to connect a HOTLink Transmitter and receiver in BIST mode, while controlling the serial data stream with the transmitter FOTO pin. The FOTO input to the transmitter causes the OUTA+ and OUTB+ outputs to be LOW, and the OUTA− and OUTB− outputs to be HIGH for the time that FOTO is HIGH. (For purposes of this example it will be assumed that Tx−OUTA+ is connected directly to Rx−INA+ and that Tx−OUTA− is connected directly to Rx−INA−.) Since FOTO is an asynchronous TTL input, it is possible to use it as a Controlled Data Corrupter that can move an edge away from its nominal

position. The limits of the EFW will be signaled by an indication on RVS.

To set up the test, the user would connect a pulse generator to the transmitter FOTO pin. This generator would be triggered by the $\overline{RP}$ output and would be controllable in both delay and pulse width. Since $\overline{RP}$ pulses once each BIST loop, the generator would make pulses that were phase aligned with the serial data stream. By careful adjustment of pulse width (VERY narrow, adjustable-width pulses) and delay (alignment such that the forced LOW is placed in a position that is naturally LOW) it is possible to measure the EFW.

To perform the test, the user should first adjust the generator so that it causes no corruption in the actual data stream. Then, by carefully adjusting the delay and/or width of the generator a specific edge in the data stream can be realigned until the RVS output indicates a BIST error. By noting the position of the realigned edge relative to its nominal position, the early and late limits of the EFW can be measured.

The relationship between the FOTO pulse and its effect on the OUTA transition must be measured empirically. The OUTA+ expanded waveform shown in *Figure 11* illustrates the control that can be effected by FOTO. The vertical lines (Internal Rx Sampling Locations) indicate the location of ideal receiver sampling points, and the shaded regions around them indicate the built-in errors that limit EFW.

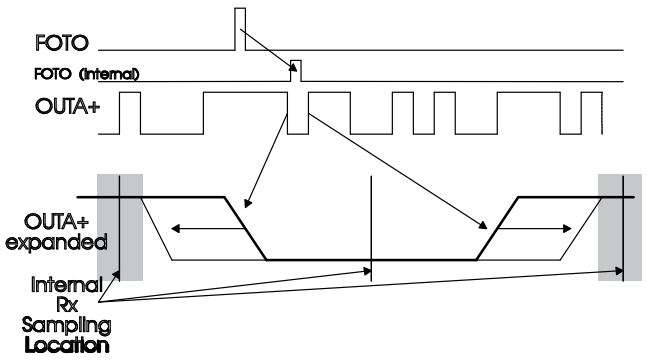

Figure 11. Example of Error Free Window Testing

Since FOTO only forces the OUTA+ & B+ output to a LOW (and OUTA− & OUTB− to a HIGH), it is not possible to check for rising and falling edge symmetry with this test. A falling edge can only be forced to an earlier LOW, and a rising edge can only be forced to a later HIGH. By careful adjustment of the FOTO generator, it is possible to adjust the position of all of the various 1, 2, 3, 4, and 5 bit-length pulses found in the 8B/10B code.

Rx Run-Length Tolerance Test

An extension of the EFW test will allow the user to measure the receiver's tolerance to missing pulses. If the pulse width of the FOTO generator described above is increased beyond a few bits, the resulting data stream will have missing pulses beyond the 5-bit run-length of the 8B/10B code. These missing transitions allow the PLL control voltage to drift, causing an arbitrary phase change. When the transitions resume, the PLL realigns to the incoming bit stream. However, if the phase drift has gone beyond the jitter limit, the PLL may align to a different bit position than the one to which it was previously aligned (see *Figure 12*). This realignment is commonly called cycle slip and equates to the loss or addition of a bit to a serial data stream.

Obviously the RVS output will indicate an error, as shown in *Figure 13*, while the data is masked, but since the indicated error is bounded (i.e., recovers within a few bytes), the BIST detector shows that the receiver is able to continue finding good data within a few bits or bytes of resumption of the sequence.

As the FOTO pulse width increases from a few bits to a few bytes, the RVS indication widens proportionately. There may be positions where a minor change in width causes multiple byte errors and others where multi-bit width changes cause the RVS to show apparently good data. The former is an indication of a running disparity error which might run for several bytes before being terminated by a code in the BIST sequence. The latter is an indication that at that particular position in the sequence, BIST was already expecting a violation, so would not flag an error for this type of data corruption.

When the FOTO pulse width (or the RVS pulse width) approaches 16 bytes, the BIST-Auto-abort

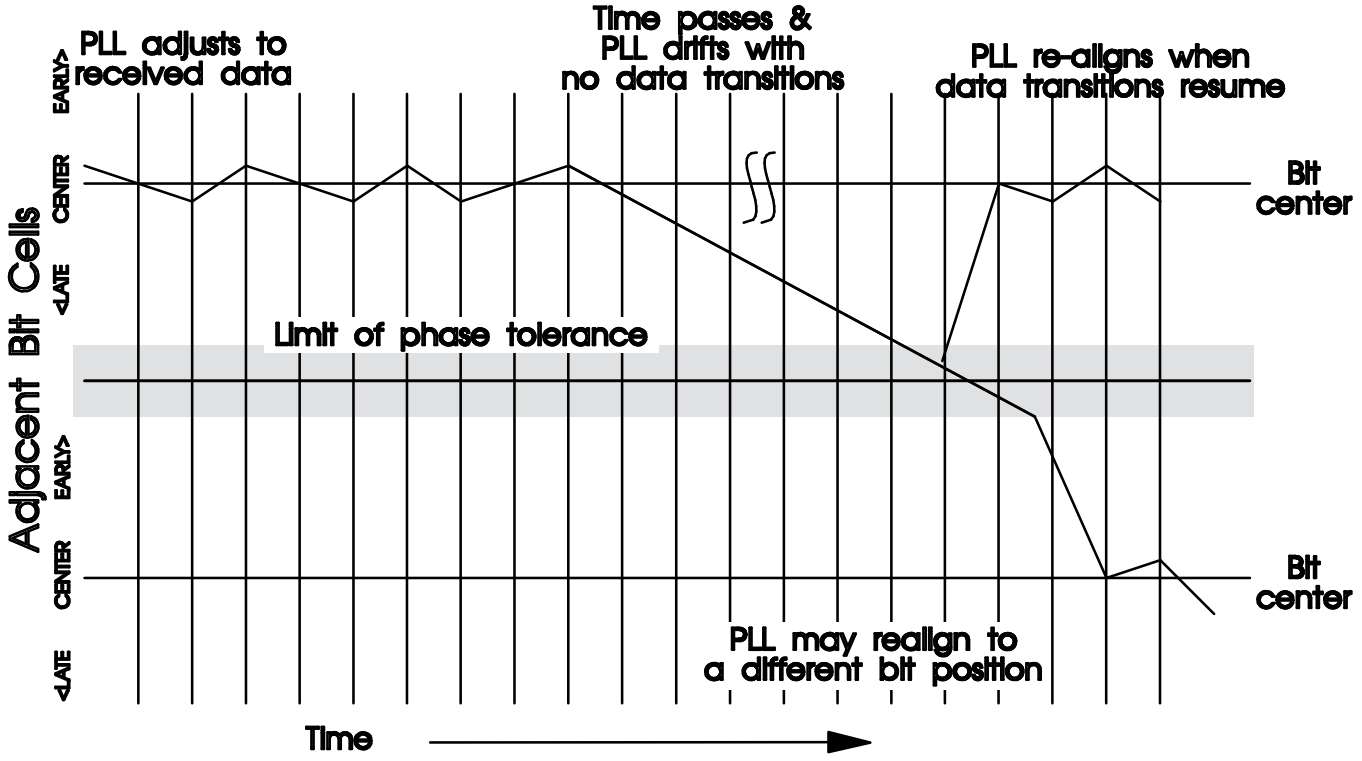

Figure 12. Long Spaces without Transitions May Cause Cycle Slip

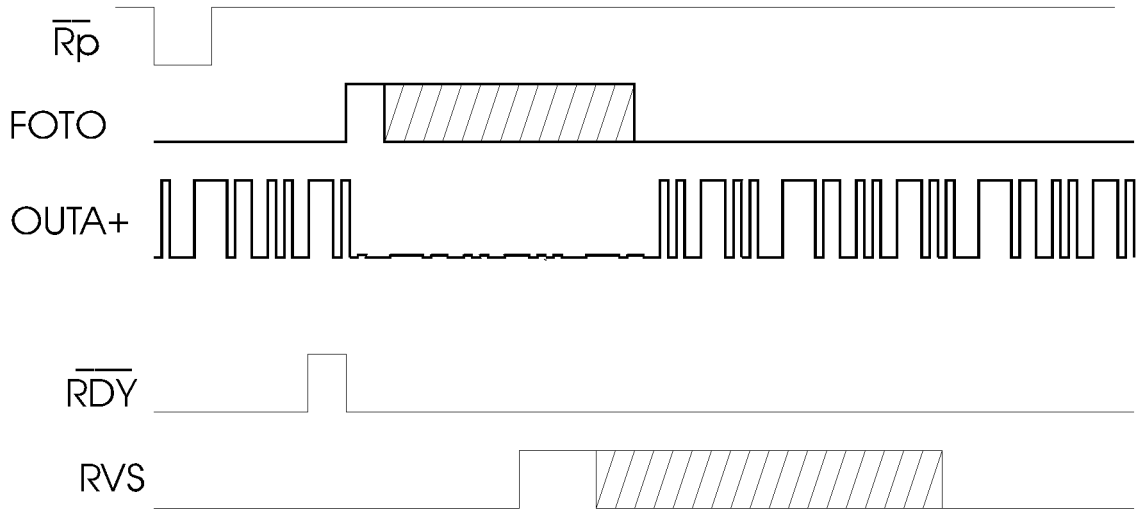

Figure 13. Missing-Transition Test Timing Diagram

mechanism described in an earlier section will begin to obscure the real receiver tolerance. When the error run length approaches 16 bytes, the RVS width will become almost continuous. $\overline{\text{RDY}}$ will cease its normal pulse-once-per-loop operation and will rise during the RVS pulse, and stay HIGH for the remainder of the BIST loop as the receiver BIST checking circuit waits for a start-of-BIST pattern. If $\overline{\text{RP}}$, RVS and $\overline{\text{RDY}}$ are all simultaneously displayed on the oscilloscope screen, it will be noted that the

BIST loop appears to start correctly, but that after the FOTO pulse, all the data is corrupted. This is the automatic-restart behavior and is characteristic of the BIST-auto-abort logic, not an indication of a real corrupted data stream.

HOTLink Receiver can tolerate nearly 100 transitionless bytes without cycle-slipping so meaningful testing requires suppression of the auto-abort function. As described earlier, this BIST-auto-abort logic can be suppressed by removing the REFCLK input from the receiver.

Even with the BIST-auto-abort logic disabled the pulse width of the FOTO generator (and thus the number of missing data transitions) will ultimately become long enough that the receiver PLL will cycle slip before data resumes. Since the BIST comparator requires an absolutely perfect data stream and cannot realign without external assistance, the BIST checker will show that the all of the received data is incorrect. Once the RVS indication becomes continuous it will be necessary to either reconnect REFCLK (and allow the Auto-abort logic to reinitialize the receiver BIST generator) or to toggle the $\overline{\text{BISTEN}}$ input on the receiver (which forces the BIST generator to begin from the beginning). The FOTO-generator pulse width (expressed in bit times) that causes an unrecoverable error is the missing-transition limit of the HOTLink receiver (i.e., run length tolerance).

Figure 13 illustrates signals involved in the run length tolerance test. The actual time measurement will be affected by the timing of the FOTO pulse, and the jitter added by the interconnect link. Care should be taken to assure that the first missing pulse comes after a normally placed pulse (i.e., make sure that FOTO takes effect while the data stream is naturally LOW and that it does not disturb the position of the last transition before it kills the data stream). If the receiver PLL is recovering from a large phase correction at the time it is left to float, reduced run-length tolerances will result.

Reframe-CKR Stretch

In normal systems it is difficult to cause the HOTLink Receiver to reframe off established byte boundaries using normal transmission data. The HOTLink BIST sequence includes one occurrence of a bit pattern that mimics a K28.5 aligned to incorrect byte boundaries. To view this clock stretch behavior, it will be necessary to synchronize the oscilloscope with the $\overline{\text{RP}}$ of the transmitter, and delay the display to the area of the alias sync. *Figure 14* shows the effect of an alias sync (five bits misaligned). In this example, taken from the BIST sequence, the C0.7−D11.5 cause an alias-sync realignment. The next several bytes are corrupted because of this misframing. When the C2.7 (+K28.5) arrives, it realigns the data to the proper boundaries. In the six byte times between the C0.7 and the C2.7 there have been two clock-stretching events and only five bytes have come out of the receiver (all bad without any RVS indication). Please note that this illustration shows the function of the receiver and is not intended to show actual timing with respect to the serial data stream.

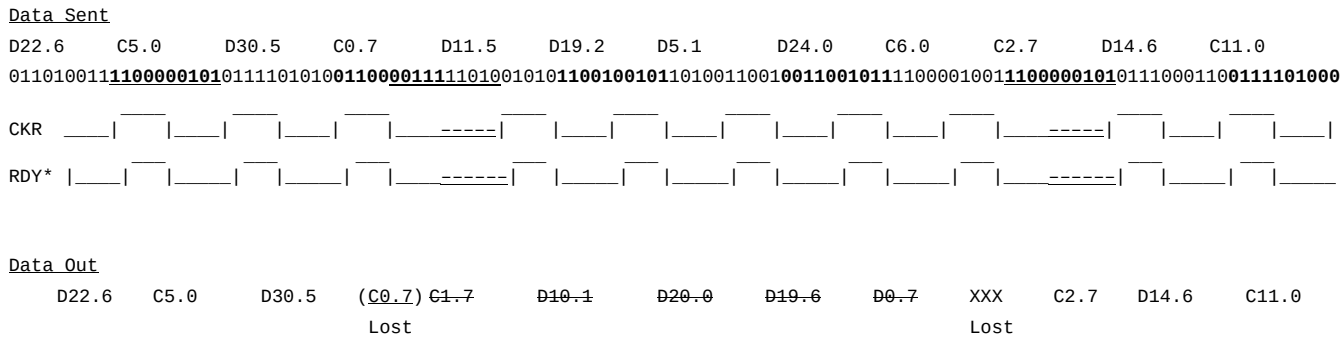

Figure 14. Illustration of Receiver Behavior during Reframe Clock Stretch

If the receiver RF input has been HIGH for less than 2048 bytes, this single alias-K28.5 (double-underlined in *Figure 8*) will cause a byte realignment (reframe) to the incorrect byte boundary (five bits off of the real byte alignment) and thus a stretch of CKR, $\overline{RDY}$ and the position of Q_{0-7}, SC/$\overline{D}$ and RVS until the next properly aligned K28.5 (approximately five bytes later). In the illustration, the C0.7 indication is lost because of the reframe caused by the alias sync and the adjacent clock edges are separated by fifteen bits. Similarly, when the real K28.5 arrives (C2.7 in the example), the Q_{0-7}, SC/$\overline{D}$ outputs will change twice between adjacent clocks (i.e., internal bit 0) between the D0.7 and the C2.7 (i.e., once on the old bit 2 and again on the new bit 2). These adjacent clock edges are separated by fifteen bits and the specified set-up and hold times for subsequent logic will be assured.

After the receiver RF has been HIGH for more than 2048 bytes, the internal byte framer changes from requiring a single K28.5 to re-align the byte, to requiring two K28.5s to reframe. To keep the receiver in single byte-framing mode, and to perform this test it will be necessary to pulse the RF input at a rate less than once per 2048 bytes (maybe triggered by $\overline{RP}$/4).

An alternative method to show byte realignment and CKR stretching involves sending a string of data that includes a positive running disparity K28.7 (C7.0) followed by D11.x or D20.x or by sending a positive running disparity SVS (C0.7) followed by D11.x. (e.g., C0.7 = 0110000111 or 1001111000 and D11.x = 110100xxxx or 001011xxxx so if the correct running disparity SVS is followed by the correct running disparity D11.x, a five bit misaligned-alias-sync is created as follows; 01100**0011111010**0xxxx)

Receiver Offset Frequency

Differences in frequency between the transmitter crystal oscillator and receiver REFCLK crystal oscillator might limit performance of the data communication link. The HOTLink datasheet specifies that the receiver and transmitter frequencies can be different by ±0.1% (1000 ppm) without compromise to the reliability of the data link. This parameter is conveniently checked by operating a transmitter CKW on one generator or crystal oscillator, and the receiver REFCLK on another. If both HOTLink parts are operating in BIST mode the RVS output will indicate the quality of the link.

As the generator frequency is adjusted (slowly and smoothly) the RVS should stay LOW indicating correct operation. RVS may show errors when the generator frequency is adjusted, though it is unlikely. If this happens, it is probable that the frequency change is being made too abruptly. The test is still possible, if RVS is checked only after the generators stabilize at each new frequency.

Tolerance to Phase Changes in Received Data

Two transmitters operated from the same clock source will run at exactly the same data rate. If they are both in BIST mode and synchronized by simultaneous assertion of the SVS input, they will also be sending exactly the same serial data. If their respective clocks are phase adjusted over a narrow delay range, they can be used as a source of synchronized serial data with a known phase relationship.

The receiver has two equivalent serial inputs (INA± and INB±) which can be independently selected. If the two transmitters are each connected to one of the serial data inputs, and if a synchronized source alternately selects one, then the other (using A/$\overline{B}$ Select), the receiver's phase adjustment behavior can be examined. (See *Figure 15*.) Synchronized switching is easily accomplished by using the $\overline{RP}$ output of one of the transmitters to trigger a long-pulse-width ECL generator (200−300 bytes pulse width, carefully aligned so that the change happens during a quiescent portion of the serial stream).

As the two transmitters are alternately selected and as the delay between them is increased, the receiver sees a continuous BIST data stream containing instantaneous phase changes equal to the difference in transmitter-to-transmitter, clock-to-clock skew. It must adjust to the new data phase and realign its internal clock to correctly recover the data. The theoretical maximum phase adjustment range is slightly less than ±0.5 bit time (i.e., ±0.5 bit less Rx PLL jitter, static alignment, and flip-flop set-up/hold times). When the phase difference reaches the

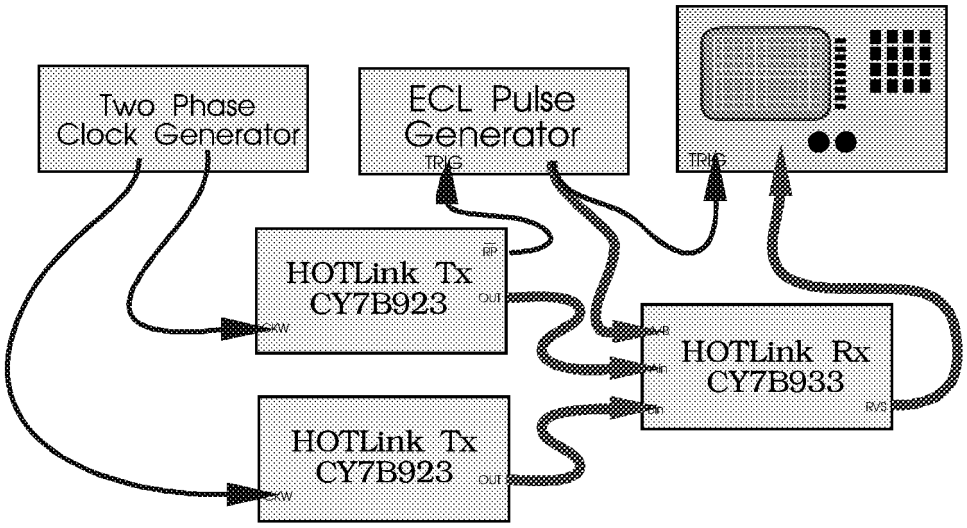

Figure 15. Receiver Phase Tolerance Test Setup

limit, errors will be indicated by pulses on RVS that are one or more bytes wide. (Even though the actual error might involve only one bit, in one byte, the RVS indication may run for several byte times because of running disparity corruption.) As the data phase hop increases, the RVS pulses will increase in width proportional to the time taken to adjust the phase of the internal PLL.

Eventually RVS will stay high continually from the time of the A/B switch to the next $\overline{\text{RDY}}$ pulse (i.e., the start of the next BIST loop). As the magnitude of transmitter clock-to-clock phase difference approaches the point where the PLL phase alignment slips from one bit to the next (i.e., at approximately 180° phase difference) the BIST loop will become irreversibly corrupted and will auto-abort-restart after each phase hop.

Conclusion

HOTLink BIST capability should help system integrators add features to high-performance communication links. These features can be made to enhance usability and improve reliability of the link.

Test methods that use BIST will aid in evaluation of HOTLink products and other link support hardware. The HOTLink built-in test features allow an unambiguous indication of data quality, many of which require only inexpensive test equipment.