CY7C924ADX

200-MBaud HOTLink® Transceiver

Features

- **Second-generation HOTLink® technology**
- **Fibre Channel and ESCON® compliant 8B/10B encoder/decoder**
- **10- or 12-bit pre-encoded data path (raw mode)**
- **8- or 10-bit encoded data transport (using 8B/10B coding)**
- **Parity check/generate**
- **Synchronous or asynchronous TTL parallel interface**
- **UTOPIA compatible host bus interface**
- **Embedded/Bypassable 256-character synchronous FIFOs**
- **Integrated support for daisy-chain and ring topologies**
- **Domain or individual destination device addressing**
- **50- to 200-MBaud serial signaling rate**
- **Internal PLLs with no external PLL components**
- **Dual differential PECL serial inputs**
- **Dual differential PECL serial outputs**
- **Compatible with fiber-optic modules and copper cables**
- **Built-In Self-Test (BIST) for link testing**
- **Link Quality Indicator**
- **Single +5.0V ±10% supply**
- **100-pin TQFP**
- **0.35μ CMOS technology**

Functional Description

The 200-MBaud CY7C924ADX HOTLink Transceiver is a point-to-point communications building block allowing the transfer of data over high-speed serial links (optical fiber, balanced, and unbalanced copper transmission lines) at speeds ranging between 50 and 200 MBaud. The transmit section accepts parallel data of selectable width and converts it to serial data, while the receiver section accepts serial data and converts it to parallel data of selectable width. *Figure 1* illustrates typical connections between two independent host systems and corresponding CY7C924ADX parts. As a second generation HOTLink device, the CY7C924ADX provides enhanced levels of technology, functionality, and integration over the field-proven CY7B923/933 HOTLink.

The transmit section of the CY7C924ADX HOTLink can be configured to accept either 8- or 10-bit data characters on each clock cycle, and stores the parallel data into an internal Transmit FIFO. Data is read from the Transmit FIFO and is encoded using an embedded 8B/10B encoder to improve its serial transmission characteristics. These encoded characters are then serialized and output from two Positive ECL (PECL) compatible differential transmission line drivers at a bit-rate of 10 times the input reference clock.

The receive section of the CY7C924ADX HOTLink accepts a serial bit-stream from one of two PECL-compatible differential line receivers and, using a completely integrated PLL Clock Synchronizer, recovers the timing information necessary for data reconstruction. The recovered bit stream is deserialized and framed into characters, 8B/10B decoded, and checked for transmission errors. Recovered decoded characters are reconstructed into either 8- or 10-bit data characters, written to an internal Receive FIFO, and presented to the destination host system.

The integrated 8B/10B encoder/decoder may be bypassed for systems that present externally encoded or scrambled data at the parallel interface. The embedded FIFOs may also be bypassed to create a reference-locked serial transmission link. For those systems requiring even greater FIFO storage capability, external FIFOs may be directly coupled to the CY7C924ADX device through the parallel interface without additional glue-logic.

The TTL parallel I/O interface may be configured as either a FIFO (configurable for UTOPIA emulation or for depth expansion through external FIFOs) or as a pipeline register extender. The FIFO configurations are optimized for transport of time-independent (asynchronous) 8- or 10-bit character-oriented data across a link. A Built-In Self-Test (BIST) pattern generator and checker permits at-speed testing of the high-speed serial data paths in both the transmit and receive sections, and across the interconnecting links.

HOTLink devices are ideal for a variety of applications where parallel interfaces can be replaced with high-speed, point-to-point serial links. Some applications include interconnecting workstations, backplanes, servers, mass storage, and video transmission equipment.

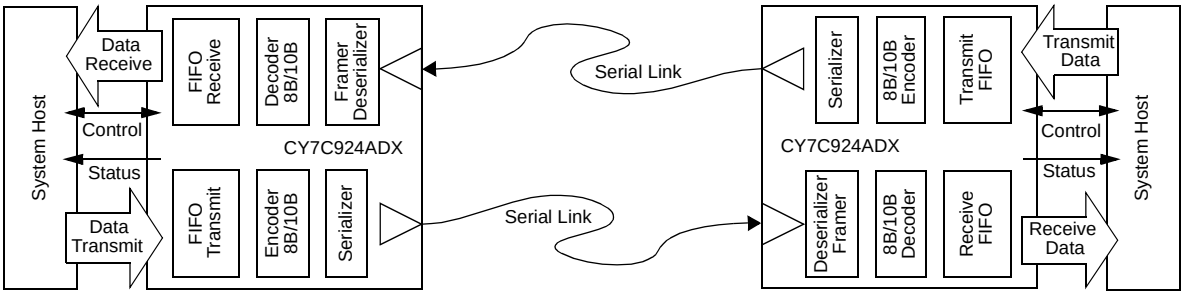
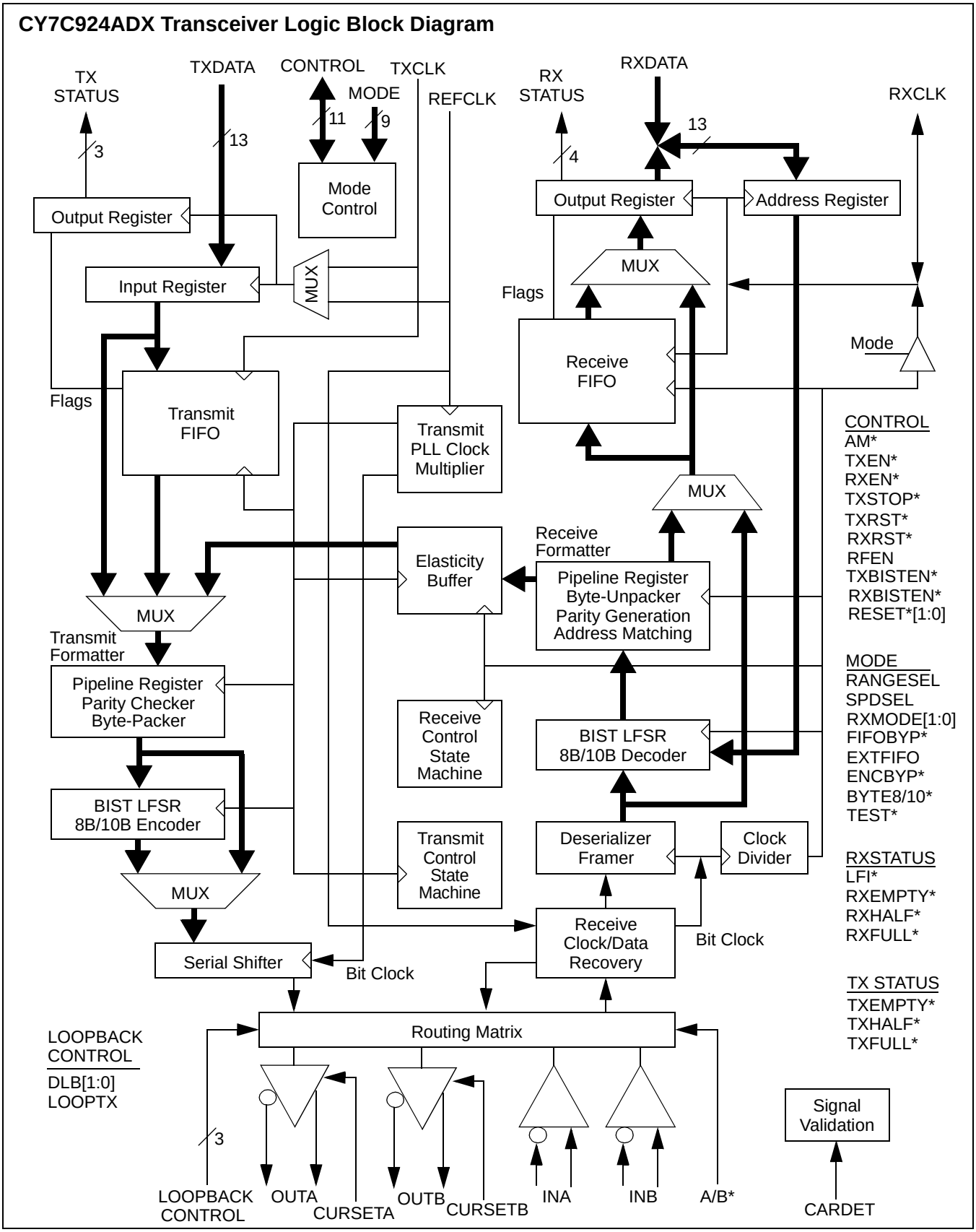
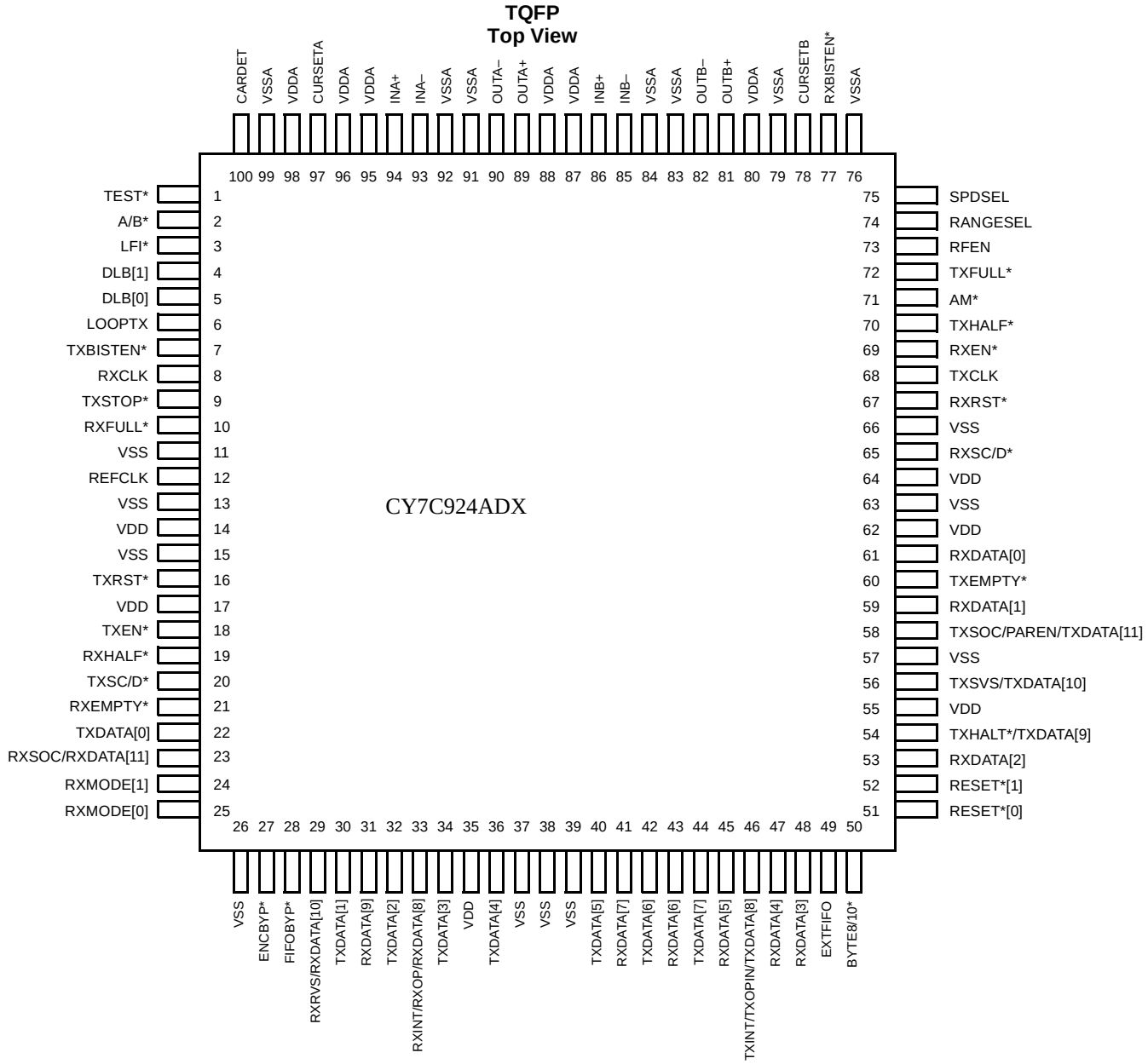


Figure 1. HOTLink System Connections.

CY7C924ADX Transceiver Logic Block Diagram



Pin Configuration



Maximum Ratings

(Above which the useful life may be impaired. For user guidelines, not tested.)

Storage Temperature ..... –65°C to +150°C

Ambient Temperature with Power Applied..... –55°C to +125°C

Supply Voltage to Ground Potential..... –0.5V to +6.5V

DC Voltage Applied to Outputs in High-Z State..... –0.5V to V_{DD}+0.5V

Output Current into TTL Outputs (LOW)..... 30 mA

DC Input Voltage ..... –0.5V to V_{DD}+0.5V

Static Discharge Voltage..... > 2001 V (per MIL-STD-883, Method 3015)

Latch-Up Current..... > 200 mA

Operating Range

Range	Ambient Temperature	V_{CC}
Commercial	0°C to +70°C	5.0V ± 10%
Industrial	–40°C to +85°C	5.0V ± 10%

Pin Descriptions

CY7C924ADX HOTLink Transceiver

Pin #	Name	I/O Characteristics	Signal Description
Transmit Path Signals			
44, 42, 40, 36, 34, 32, 30, 22	TXDATA[7:0]	TTL input, sampled on TXCLK↑ or REFCLK↑, Internal Pull-Up	Parallel Transmit Data Input. Bus width can be configured to accept either 8- or 10-bit characters. When the encoder is bypassed (ENCBYP* is LOW), TXDATA[7:0] functions as the least significant eight bits of the 10- or 12-bit pre-encoded transmit character.
46	TXINT/ TXOPIN/ TXDATA[8]	TTL input, sampled on TXCLK↑ or REFCLK↑, Internal Pull-Up	Transmit Interrupt Input. This input is only interpreted if both the Transmit FIFO and Encoder are enabled. Upon any state-change (0→1 or 1→0) in TXINT, a character is forced into the transmit encoder and shifter prior to accessing the next Transmit FIFO contents. This signal is routed around, not through, the Transmit FIFO. When TXINT transitions from 0→1, a C0.0 (K28.0) special code is sent. When TXINT transitions from 1→0, a C3.0 (K28.3) special code is sent. These special codes force a similar signal transition on the RXINT output of an attached CY7C924ADX HOTLink Transceiver. When the Transmit FIFO is bypassed (FIFOBYP* is LOW) and the encoder is enabled (ENCBYP* is HIGH), this input is the ODD parity input associated with the TXDATA[7:0], TXSC/D*, and TXSVS inputs when parity is enabled, and ignored otherwise. When the encoder is bypassed (ENCBYP* is LOW), TXDATA[8] functions as the 9th bit of the 10- or 12-bit pre-encoded transmit character.
54	TXHALT*/ TXDATA[9]	TTL input, sampled on TXCLK↑ or REFCLK↑, Internal Pull-Up	Transmit FIFO Halt Immediate Input. When TXHALT* is asserted LOW, transmission of data is suspended and the HOTLink transmits pad characters (K28.5). When TXHALT* is deasserted HIGH, normal data processing proceeds. When the encoder is bypassed (ENCBYP* is LOW), TXDATA[9] functions as the 10th bit of the 10- or 12-bit pre-encoded transmit character.
	TXPER	TTL output, changes following TXCLK↑ or REFCLK↑	Transmit Parity Error Output. When the FIFOs are bypassed (FIFOBYP* is LOW) and the Encoder is enabled (ENCBYP* is HIGH) this pin is an output and indicates that parity errors have been found in the TXDATA[7:0], TXSC/D*, TXSVS, and TXOPIN inputs.
56	TXSVS/ TXDATA[10]	TTL input, sampled on TXCLK↑ or REFCLK↑, Internal Pull-Up	Transmit Send Violation Symbol input. When the Transmit FIFO is enabled, this input is interpreted along with TXSOC and TXSC/D* (see *Table 2* for details). When the encoder is bypassed (ENCBYP* is LOW) and in 10-bit mode (BYTE8/10* is LOW), TXDATA[10] functions as the 11th bit of the 12-bit pre-encoded transmit character.
58	TXSOC/ PAREN/ TXDATA[11]	TTL input, sampled on TXCLK↑ or REFCLK↑, Internal Pull-Up	Transmit Start of Cell Input. When the Transmit FIFO is enabled (FIFOBYP* is HIGH), this input is used as a message frame delimiter to indicate the beginning of a data packet. It is interpreted along with TXSVS and TXSC/D* (see *Table 2* for details). When the Transmit FIFO is bypassed (FIFOBYP* is LOW), this input is the Parity Enable input which enables ODD parity checking of the TXDATA[7:0], TXSC/D*, TXSVS, and TXOPIN inputs. When the encoder is bypassed (ENCBYP* is LOW) and in 10-bit mode (BYTE8/10* is LOW), TXDATA[11] functions as the 12th bit (MSB) of the 12-bit pre-encoded transmit character.

Pin Descriptions (continued)

CY7C924ADX HOTLink Transceiver

Pin #	Name	I/O Characteristics	Signal Description
20	TXSC/D*	TTL input, sampled on TXCLK↑ or REFCLK↑, Internal Pull-Up	Transmit Special Character or Data Select Input. When the Transmit FIFO is enabled, this input is interpreted along with TXSVS and TXSOC (see *Table 2* for details). When the encoder is bypassed (ENCBYP* is LOW) TXSC/D* is ignored.
18	TXEN*	TTL input, sampled on TXCLK↑ or REFCLK↑, Internal Pull-Up	Transmit Enable Input. Data enable for the TXDATA[11:0] data bus write operations. Active HIGH when configured for Cascade timing, active LOW when configured for UTOPIA timing.
9	TXSTOP*	TTL input, sampled on TXCLK↑, Internal Pull-Up	Transmit Stop on Start_Of_Cell Input. While the Transmit FIFO is enabled, this signal is used to prevent queued data characters from being serially transmitted. While TXSTOP* is deasserted (HIGH), data flows through the Transmit FIFO without interruption. When TXSTOP* is asserted (LOW), data transfers continue until a TXSOC bit is detected in the character stream, at which point data transmission ceases. If TXSTOP* is momentarily deasserted and then reasserted, a single “cell” (delimited by SOC bits) is transmitted. Stopped transfers and empty FIFO conditions are padded with C5.0 (K28.5) characters. When the transmit FIFO is bypassed (FIFOBYP* = LOW), TXSTOP* has no function. This input can be left open or tied HIGH.
68	TXCLK	TTL clock input, Internal Pull-Up	Transmit FIFO Clock. The input clock for the parallel interface when the Transmit FIFO is enabled. Used to sample all Transmit FIFO related interface signals.
72	TXFULL*	3-state TTL output, changes following TXCLK↑ or REFCLK↑	Transmit FIFO Full Status Flag. Active LOW when configured for UTOPIA timing, active HIGH when configured for Cascade timing. When the Transmit FIFO is enabled (FIFOBYP* is HIGH), TXFULL* Indicates a Transmit FIFO full condition. When TXFULL* is first asserted, the Transmit FIFO can accept a minimum of eight additional write cycles without loss of data. When the Transmit FIFO is bypassed (FIFOBYP* is LOW), with RANGESEL HIGH or SPDSEL LOW, TXFULL* toggles at half the REFCLK rate to provide a character rate indication. Data can be accepted when TXFULL* indicates a non-full condition.
70	TXHALF*	3-state TTL output, changes following TXCLK↑ or REFCLK↑	Transmit FIFO Half-full Status Flag. Active LOW. When the Transmit FIFO is enabled, TXHALF* is asserted LOW when the Transmit FIFO is ≥ half full (128 characters). TXHALF* is only set to High-Z state by the assertion of RESET*[1:0] LOW.
60	TXEMPTY*	3-state TTL output, changes following TXCLK↑ or REFCLK↑	Transmit FIFO Empty Status Flag. Active LOW when configured for UTOPIA timing, active HIGH when configured for Cascade timing. When the Transmit FIFO is enabled, TXEMPTY* is asserted either when no data has been loaded into the Transmit FIFO, or when the Transmit FIFO has been emptied by either a Transmit FIFO reset or by the normal transmission of the FIFO contents. When TXBISTEN* is asserted LOW, TXEMPTY* becomes the transmit BIST-loop counter indicator (regardless of the logic state of FIFOBYP*). In this mode TXEMPTY* is asserted for one TXCLK period at the end of each transmitted BIST sequence. When the Transmit FIFO is bypassed, TXEMPTY* is asserted to indicate that the transmitter can accept data. TXEMPTY* is also used as a BIST progress indicator when TXBISTEN* is asserted.

Pin Descriptions (continued)

CY7C924ADX HOTLink Transceiver

Pin #	Name	I/O Characteristics	Signal Description
16	TXRST*	TTL input, internal pull-up, sampled on TXCLK↑, Internal Pull-Up	Transmit FIFO Reset. When TXRST* is sampled asserted (LOW) for eight or more TXCLK cycles, a reset operation is started on the Transmit FIFO. This input is ignored when the Transmit FIFO is bypassed.
7	TXBISTEN*	TTL input, asynchronous, Internal Pull-Up	Transmitter BIST Enable. When TXBISTEN* is LOW, the transmitter generates a 511-character repeating sequence, that can be used to validate link integrity. The transmitter returns to normal operation when TXBISTEN* is HIGH. All Transmit FIFO read operations are suspended when BIST is active.
Receive Path Signals			
41, 43, 45, 47, 48, 53, 59, 61	RXDATA[7:0]	Bidirectional TTL, changes following RXCLK↑, or sampled by RXCLK↑	Parallel Data Output and Serial Address Register Access. These outputs change following the rising edge of RXCLK, when enabled to output data (the device is addressed by AM* and selected by RXEN*). The contents of this bus are interpreted differently based on the levels present on ENCBYP*, BYTE8/10*, RXSC/D*, and when accessing the Serial Address Register. When the Decoder is bypassed (ENCBYP* is LOW), RXDATA[7:0] functions as the least significant eight bits of the 10- or 12-bit pre-encoded receive character.
33	RXINT/ RXOP/ RXDATA[8]	Bidirectional TTL, changes following RXCLK↑, or sampled by RXCLK↑	Receive Interrupt Output. When the Receive FIFO and Decoder are enabled (FIFOBYP* and ENCBYP* are HIGH) and a C0.0 (K28.0) special code is received, RXINT is set HIGH. When a C3.0 (K28.3) special code is received RXINT is set LOW. These special codes are assumed to be generated in response to equivalent transitions on the TXINT input of an attached CY7C924ADX HOTLink transceiver. This signal is extracted prior to the Receive FIFO and (except for Receive Discard Policy 0) the associated command codes are not considered “data” to be entered into the Receive FIFO and are discarded. When the Receive FIFO is bypassed and Decoder is enabled (FIFOBYP* is LOW and ENCBYP* is HIGH), this output contains the ODD parity of the RXDATA[7:0] and RXSC/D* outputs. When the Decoder is bypassed (ENCBYP* is LOW), RXDATA[8] functions as the 9th bit of the 10- or 12-bit undecoded receive character.
31	RXDATA[9]	Bidirectional TTL, changes following RXCLK↑, or sampled by RXCLK↑	Receive Data Output. When the Decoder is enabled in 10-bit mode (ENCBYP* is HIGH and BYTE8/10* is LOW), this input is the 10th bit (MSB) of the 10-bit decoded and unpacked data character. When the Decoder is enabled and in 8-bit mode this input is ignored. When the Decoder is bypassed (ENCBYP* is LOW), RXDATA[9] functions as the 10th bit of the 10- or 12-bit undecoded receive character.

Pin Descriptions (continued)

CY7C924ADX HOTLink Transceiver

Pin #	Name	I/O Characteristics	Signal Description
29	RXRVS/ RXDATA[10]	Bidirectional TTL, changes following RXCLK↑, or sampled by RXCLK↑, Internal Pull-Up	Received Violation Symbol Indicator. For normal data accesses this signal is used as an output. It is decoded in conjunction with RXSC/D* and RXSOC, per *Table 5*, to indicate the presence of specific Special Character codes in the received data stream. RXRVS is used to report BIST pattern mismatches when RXBISTEN* is LOW. When accessing the Serial Address Register, this signal is used as a "read/write" control input. RXRVS LOW allows the host system to write the Serial Address Register (RXDATA[9:0] and RXSC/D* are inputs). RXRVS HIGH allows the host system to read the Serial Address Register (RXDATA[9:0] and RXSC/D* are outputs). When the Decoder is bypassed (ENCBYP* is LOW) and in 10-bit mode (BYTE8/10* is LOW), RXDATA[10] functions as the 11th bit of the 12-bit undecoded receive character. When in 8-bit mode this output is unused and is driven LOW.
23	RXSOC/ RXDATA[11]	Bidirectional TTL, changes following RXCLK↑, or sampled by RXCLK↑	Receive Start Of Cell. Active HIGH. This output is decoded in conjunction with RXSC/D* and RXRVS, per *Table 5*, to indicate the presence of specific Special Character codes in the received data stream. When the Decoder is bypassed (ENCBYP* is LOW) and in 10-bit mode (BYTE8/10* is LOW), RXDATA[11] functions as the 12th bit (MSB) of the 12-bit undecoded receive character. When in 8-bit mode this output is unused and is driven LOW.
65	RXSC/D*	Bidirectional TTL, changes following RXCLK↑, or sampled by RXCLK↑	Received Special Character or Data Indicator. For normal data accesses this signal is used as an output. It is decoded in conjunction with RXSOC and RXRVS, per *Table 5*, to indicate the presence of specific Special Character codes in the received data stream. When accessing the Serial Address Register, this signal is used as an input to select the addressing mode. RXSC/D* HIGH configures the Serial Address Register for Unicast address matching. RXSC/D* LOW configures the Serial Address Register for Multicast address matching.
69	RXEN*	TTL input, sampled on RXCLK↑, Internal Pull-Up	Receive Enable. Data enable for the RXDATA[11:0] data bus write and read operations. Active HIGH when configured for Cascade timing, active LOW when configured for UTOPIA timing. Used to select the parallel read interface of the device. Also controls the read and write access to the Serial Address Register.
8	RXCLK	Bidirectional TTL clock, Internal Pull-Up	Receive Clock. When the Receive FIFO is enabled, this clock is the Receive interface *input* clock and is used to control Receive FIFO read, reset, and serial register access operations. When the Receive FIFO is bypassed, this clock is *output* continuously at the character rate of the data being received (1/10th the serial bit-rate).
10	RXFULL*	3-state TTL output, changes following RXCLK↑	Receive FIFO Full Flag. Active LOW when configured for UTOPIA timing, active HIGH when configured for Cascade timing. When the Receive FIFO is addressed, RXFULL* is asserted when the Receive FIFO has room for eight or fewer writes. If the RXCLK input is not continuous or the Receive FIFO is accessed at a rate slower than data is being received, RXFULL* may indicate loss of data. When the Receive FIFO is bypassed, RXFULL* and RXHALF* are deasserted to indicate that valid data may be present. RXFULL* is also used as a BIST progress indicator, and pulses asserted once every pass through the 511-character BIST loop.

Pin Descriptions (continued)

CY7C924ADX HOTLink Transceiver

Pin #	Name	I/O Characteristics	Signal Description
19	RXHALF*	TTL output, changes following RXCLK↑	Receive FIFO Half-full Flag. Active LOW. When the Receive FIFO is enabled, this signal is asserted (LOW) when the Receive FIFO is ≥ half full (128 characters). When the Receive FIFO is bypassed, RXHALF* is deasserted (HIGH). RXHALF* is forced to the High-Z state only during a "full-chip" reset (i.e., while RESET*[1:0] are LOW).
21	RXEMPTY*	3-state TTL output, changes following RXCLK↑	Receive FIFO Empty Flag. Active LOW when configured for UTOPIA timing, active HIGH when configured for Cascade timing. When the Receive FIFO is enabled, RXEMPTY* is asserted when no data remains in the Receive FIFO. Any read operation occurring when RXEMPTY is asserted results in no change in the FIFO status, and the data from the last valid read remains on the RXDATA bus. When the Receive FIFO is bypassed but the Decoder is enabled, RXEMPTY* is used as a valid data indicator. When deasserted it indicates that valid data (as selected by RXMODE[1:0]) is present at the RXDATA outputs. When asserted it indicates that a C5.0 (K28.5) is present on the RXDATA output bus. If both the Receive FIFO and the Decoder are bypassed, RXEMPTY* is deasserted to indicate that all received characters are valid.
67	RXRST*	TTL input, sampled on RXCLK↑, Internal Pull-Up	Receive FIFO Reset. When the Receive FIFO is addressed and RXRST* is sampled while asserted (LOW) for eight or more RXCLK cycles, a Receive FIFO reset is initiated. The RXRST* input is also asserted to access the Serial Address Register.
73	RFEN	TTL input, asynchronous, Internal Pull-Up	Reframe Enable. Used to control when the framer is allowed to adjust the character boundaries based on detection of one or more K28.5 characters in the data stream. When HIGH, the framer is allowed to adjust the character boundaries relative to the received serial data stream. When LOW, the boundary is fixed.
77	RXBISTEN*	TTL input, asynchronous, Internal Pull-Up	Receiver BIST Enable. When active, the receiver is configured to perform a character-for-character match of the incoming data stream with a 511-character BIST sequence. The result of character mismatches are indicated on RXRVS. Completion of each 511-character BIST loop is accompanied by an assertion pulse on the RXFULL* flag.
Control Signals			
71	AM*	TTL input, sampled by TXCLK↑, RXCLK↑, and REFCLK↑	Address Match. Active LOW. Used as a qualifier for TXEN*, RXEN*, TXRST*, and RXRST*. Also controls three-state enables for the TXFULL*, TXEMPTY*, RXFULL*, and RXEMPTY* signals.
6	LOOPTX	TTL input, asynchronous, Internal Pull-Down	Serial-in to Serial-out LOOP Select. This input controls the LOOP-through function in which the serial data is recovered by the Clock/Data Recovery PLL and is then retransmitted using the Transmit PLL as the bit-rate reference. It selects between the output of the Transmit FIFO and the output of the Elasticity Buffer as the input to the Transmit Encoder. When LOW, the Transmit FIFO is the source of data for transmission. When HIGH, the Elasticity Buffer is the source of data for transmission.
12	REFCLK	TTL input clock	Reference Clock. This clock input is used as the timing reference for the transmit and receive PLLs. When the Transmit FIFO is bypassed, REFCLK is also used as the clock for the external transmit data interface. See *Table 4* for the relationships between REFCLK, SPDSEL, RANGESEL, and BYTE8/10*.

Pin Descriptions (continued)

CY7C924ADX HOTLink Transceiver

Pin #	Name	I/O Characteristics	Signal Description
75	SPDSEL	Static control input TTL levels Normally wired HIGH or LOW	Speed Select. Used to select one of two operating data rate ranges for the device. When the operating symbol rate is between 100 and 200 MBaud, SPDSEL must be HIGH. When the operating symbol rate is between 50 and 100 MBaud, SPDSEL must be LOW (see *Table 4*).
74	RANGESEL	Static control input TTL levels Normally wired HIGH or LOW	Range Select. Selects the proper prescaler for the REFCLK input. See *Table 4* for the various relationships between REFCLK, SPDSEL, RANGESEL, and BYTE8/10*. When the Transmit FIFO is bypassed (FIFOBYP* is LOW), with RANGESEL HIGH or SPDSEL LOW, TXFULL* toggles at half the REFCLK rate to provide a character rate indication, and to show when data can be accepted.
49	EXTFIFO	Static control input TTL levels Normally wired HIGH or LOW	External FIFO Select. EXTFIFO modifies the active level of the RXEN* and TXEN* inputs and the timing of the Transmitter and Receiver data buses. When not configured for external FIFOs (EXTFIFO is LOW), TXEN* is assumed to be driven as a pipeline register and RXEN* is assumed to be driven by a controller for a pipeline register. In this mode the active data transition for the transmit data bus is within the same clock as the transmit interface is selected by TXEN*. When configured for external FIFOs (EXTFIFO is HIGH), TXEN is assumed to be driven by the empty flag of an attached CY7C42X5 FIFO, and RXEN is assumed to be driven by the Almost Full flag of an attached CY7C42X5 FIFO. In this mode the active data transition for the transmit data bus is in the clock cycle following the clock edge where transmit interface is selected by TXEN*. EXTFIFO also modifies the output state of the Receive and Transmit FIFO flags. When configured for external FIFOs (EXTFIFO is HIGH), the Full and Empty FIFO flags are active HIGH (the Half-full flag is always active LOW). When not configured for external FIFOs (EXTFIFO is LOW), all of the FIFO flags are active LOW.
28	FIFOBYP*	Static control input TTL levels Normally wired HIGH or LOW	FIFO Bypass Select. Active LOW. When LOW, the Transmit and Receive FIFOs are bypassed. In this mode TXCLK is not used. Instead all transmit data must be synchronous to REFCLK. Transmit FIFO status flags are synchronized to REFCLK. All received data is synchronous to the RXCLK output. Receive FIFO status flags are synchronized to RXCLK (the recovered Receive PLL Character clock). When HIGH, the Transmit and Receive FIFOs are enabled. In this mode all Transmit FIFO writes are synchronized to TXCLK**,** and all Receive FIFO reads are synchronous to the RXCLK input.
27	ENCBYP*	Static control input TTL levels Normally wired HIGH or LOW	Encoder Bypass Select. Active LOW. When LOW, both the Encoder and Decoder are bypassed. Data is transmitted in NRZ format, without encoding, LSB first. Received data are presented as parallel characters to the interface without decoding. When HIGH, data is passed through both the 8B/10B Encoder in the Transmit path and the Decoder in the Receive path.
24, 25	RXMODE[1:0]	Static control input TTL levels Normally wired HIGH or LOW	Receive Discard Policy Select. These inputs select between the four data handling and fill-character discard modes in the receiver. See *Table 6*.

Pin Descriptions (continued)

CY7C924ADX HOTLink Transceiver

Pin #	Name	I/O Characteristics	Signal Description
50	BYTE8/10*	Static control input TTL levels Normally wired HIGH or LOW	Parallel Data Character Size Select. Selects the input data character width. When BYTE8/10* is HIGH, the device is in 8-bit mode. When BYTE8/10* is LOW, the part is in 10-bit mode. If the encoder is enabled (ENCBYP* is HIGH), the data is encoded using the 8B/10B code rules found in *Table 10* and *Table 11*. When ENCBYP* is LOW, the part passes the 10 input bits (when BYTE8/10* is HIGH) or 12 input bits (BYTE8/10* is LOW) directly to the serial stream without encoding or decoding. For affected pin groupings and function see *Table 1* and *Table 7*.
52, 51	RESET*[1:0]	TTL input, asynchronous	Global Logic Reset. These inputs are pulsed LOW for one or more REFCLK periods to reset the internal logic. They must be tied together or driven concurrently to ensure a valid reset.
1	TEST*	TTL input, asynchronous. Normally wired HIGH	Factory Test Mode Select. Used to force the part into a diagnostic test mode used for factory ATE test. This pin is tied HIGH during normal operation.
Analog I/O and Control			
89, 90, 81, 82	OUTA± OUTB±	PECL differential outputs	Differential Serial Data Outputs. These PECL compatible outputs are capable of driving terminated transmission lines or commercial fiber-optic transmitter modules. An unused output pair may be powered down by leaving the outputs unconnected and strapping the associated CURSETx pin to V_{DD}.
97	CURSETA	Analog input	Current-set Resistor Input for OUTA±. A precision resistor is connected between this input and a clean ground to set the output differential amplitude and currents for the OUTA± differential driver.
78	CURSETB	Analog input	Current-set Resistor Input for OUTB±. A precision resistor is connected between this input and a clean ground to set the output differential amplitude and currents for the OUTB± differential driver.
94, 93, 86, 85	INA± INB±	PECL compatible differential input	Differential Serial Data Inputs. These inputs accept the serial data stream for deserialization and decoding. Only one serial stream at a time may be fed to the receiver PLL to extract the data content. This stream is selected using the A/B* input. These inputs may also be routed to the OUTB± serial outputs using the DLB[1:0] inputs.
2	A/B*	TTL input, asynchronous, Internal Pull-Up	Receive Data Input Selector. Determines which internal or external serial bit-stream is passed to the receiver clock and data recovery circuit. See *Table 3* for details.
4,5	DLB[1:0]	TTL input, asynchronous, Internal Pull-Down	Loop-back Select Inputs. Selects connections between serial inputs and outputs. Controls diagnostic loop-back and serial loop-through functions. See *Table 3* for details.
100	CARDET	PECL input, asynchronous	Carrier Detect Input. Used to allow an external device to signify a valid signal is being presented to the high speed PECL compatible input buffers, as is typical on an Optical Module. When CARDET is deasserted LOW, the LFI***** indicator asserts LOW signifying a Link Fault. This input can be tied to V_{DD} for copper media applications.

Pin Descriptions (continued)

CY7C924ADX HOTLink Transceiver

Pin #	Name	I/O Characteristics	Signal Description
3	LFI*	TTL output, changes following RXCLK↑	Link Fault Indication Output. Active LOW. LFI* changes synchronous with RXCLK. This output is driven LOW when the serial link currently selected by A/B* is not suitable for data recovery. This can be caused by 1. Serial Data Amplitude is below acceptable levels. 2. Input transition density is not sufficient for PLL clock recovery. 3. Serial Data stream is outside an acceptable frequency range of operation. 4. CARDET is LOW.
Power			
80, 87, 88, 95, 96, 98	V_{DDA}		Power for PECL I/O signals and internal analog circuits.
76, 79, 83, 84, 91, 92, 99	V_{SSA}		Ground for PECL I/O signals and internal analog circuits.
14, 17, 35, 55, 62, 64	V_{DD}		Power for CMOS I/O signals and internal logic circuits.
11, 13, 15, 26, 37, 38, 39, 57, 63, 66	V_{SS}		Ground for CMOS I/O signals and internal logic circuits.

CY7C924ADX HOTLink Operation

Overview

The CY7C924ADX is designed to move parallel data across both short and long distances with minimal overhead or host system intervention. This is accomplished by converting the parallel characters into a serial bit-stream, transmitting these serial bits at high speed, and converting the received serial bits back into the original parallel data format.

The CY7C924ADX offers a large feature set, allowing it to be used in a wide range of host systems. Some of the configuration options are:

- 8-bit or 10-bit character size
- user definable data packet or frame structure
- 2-octave data rate range
- asynchronous (FIFOed) or synchronous data interface
- 8B/10B encoded or non-encoded (raw data)
- with or without parity check/generate
- embedded or bypassable FIFO data storage
- multi-PHY capability
- point-to-point, point-to-multipoint, or ring data-transport

This flexibility allows the CY7C924ADX to meet the data-transport needs of almost any system.

Transmit Data Path

Transmit Data Interface/Transmit Data FIFO

The transmit data interface to the host system is configurable as either an asynchronous buffered (FIFOed) parallel interface or as a synchronous pipeline register. The bus itself can be configured for operation with 8-bit or 10-bit data.

When configured for asynchronous operation (where the host-bus interface clock operates asynchronous to the serial character and bit stream clocks), the host interface becomes that of a synchronous FIFO clocked by TXCLK. In these configurations an internal 256-character Transmit FIFO is enabled. It allows the host interface to be written at any rate from DC to 50 MHz.

When configured for synchronous operation, the transmit interface is clocked by REFCLK and operates synchronous to the internal character and bit-stream clocks. The input register must be written at the character rate, but REFCLK can operate at the character rate or at twice the character rate.

Both asynchronous and synchronous interface operations support two interface timing models: UTOPIA and Cascade. The UTOPIA timing model is designed to match the active levels, bus timing, and signal sequencing called out in the ATM Forum UTOPIA specification. The Cascade timing model is designed to match a host bus that resembles a synchronous FIFO. These timing models allow the CY7C924ADX to directly couple to host systems, registers, state machines, FIFOs, etc., with minimal and in many cases no external glue logic.

Encoder

Data from the host interface or Transmit FIFO is next passed to an Encoder block. The CY7C924ADX contains an internal 8B/10B encoder that is used to improve the serial transport characteristics of the data. If parity checking is enabled, the characters are checked for valid ODD parity. For those systems that contain their own encoder or scrambler, this Encoder may be bypassed.

Serializer/Line Driver

The data from the Encoder is passed to a Serializer. This Serializer operates at either 2.5, 5, or 10 times the rate of the REFCLK input (or 3, 6, or 12 times when in unencoded 10-bit mode). The serialized data is output from two PECL-compatible differential line drivers configured to drive transmission lines or optical modules.

Receive Data Path

Line Receiver/Deserializer/Framer

Serial data is received at one of two PECL-compatible differential line receivers. The data is passed to both a Clock and Data Recovery PLL (Phase Locked Loop) and to a Deserializer that converts serial data into parallel characters. The Framer adjusts the boundaries of these characters to match those of the original transmitted characters.

Decoder

The parallel characters are passed through a 10B/8B Decoder and returned to their original form. If parity generation is enabled, ODD parity is added to the received characters at this point. For systems that make use of external decoding or descrambling, the decoder may be bypassed.

Receive Data Interface/Receive Data FIFO

Data from the decoder is passed either to a Receive FIFO or is passed directly to the output register. The output register can be configured for operation with 8-bit or 10-bit data.

When configured for an asynchronous buffered (FIFOed) interface, the data is passed through a 256-character Receive FIFO that allows data to be read at any rate from DC to 50 MHz. When configured for synchronous operation (Receive FIFO is bypassed) data is clocked out of the Receive Output register at the byte rate, up to 20 MHz. The receive interface is also configurable for both UTOPIA and Cascade timing models.

CY7C924ADX HOTLink Transceiver Block Diagram Description

Transmit Input/Output Register

The Transmit Input Register, shown in *Figure 2*, captures the data to be processed by the HOTLink Transmitter, and allows the input timing to be made compatible with asynchronous or synchronous host system buses. These buses can take the form of UTOPIA compliant interfaces, external FIFOs, state machines, or other control structures. Data present on the TXDATA[11:0] and TXSC/D* inputs are captured at the rising edge of the selected sample clock. The transmit data bus bit-assignments vary depending on the data encoding, parity, and bus-width selected. These bus bit-assignments are shown in *Table 1*, and list the functional names of these different signals. Note that the function of several of these signals changes in different operating modes. The logical sense of the enable and FIFO flag signals depends on the intended interface convention and is set by the EXTFIFO pin.

The transmit interface supports both synchronous and asynchronous clocking modes, each supporting both UTOPIA and Cascade timing models. The selection of the specific clocking

Table 1. Transmit Input Bus Signal Map

	Transmit Encoder Mode[1]			
	Encoded 8-bit Character Stream	**Pre-encoded 10-bit Character Stream**	**Encoded 10-bit Character Stream**	**Pre-encoded 12-bit Character Stream**
ENCBYP*	HIGH	LOW	HIGH	LOW
BYTE8/10*	HIGH	HIGH	LOW	LOW
TXDATA Bus Input Bit				
TXSC/D*	TXSC/D*		TXSC/D*	
TXDATA[0]	TXD[0]	TXD[0][2]	TXD[0]	TXD[0][2]
TXDATA[1]	TXD[1]	TXD[1]	TXD[1]	TXD[1]
TXDATA[2]	TXD[2]	TXD[2]	TXD[2]	TXD[2]
TXDATA[3]	TXD[3]	TXD[3]	TXD[3]	TXD[3]
TXDATA[4]	TXD[4]	TXD[4]	TXD[4]	TXD[4]
TXDATA[5]	TXD[5]	TXD[5]	TXD[5]	TXD[5]
TXDATA[6]	TXD[6]	TXD[6]	TXD[6]	TXD[6]
TXDATA[7]	TXD[7]	TXD[7]	TXD[7]	TXD[7]
TXINT/TXOPIN/TXDATA[8] (FIFOBYP* = HIGH)	TXINT	TXD[8]	TXD[8]	TXD[8]
TXINT/TXOPIN/TXDATA[8] (FIFOBYP* = LOW)	TXOPIN	TXD[8]	TXD[8]	TXD[8]
TXHALT*/TXPER/TXDATA[9] (FIFOBYP* = HIGH)	TXHALT*	TXD[9]	TXD[9]	TXD[9]
TXHALT*/TXPER/TXDATA[9] (FIFOBYP* = LOW)	TXPER (output)	TXD[9]	TXD[9]	TXD[9]
TXSVS/TXDATA[10]	TXSVS		TXSVS	TXD[10]
TXSOC/TXPAREN/TXDATA[11] (FIFOBYP* = HIGH)	TXSOC		TXSOC	TXD[11]
TXSOC/TXPAREN/TXDATA[11] (FIFOBYP* = LOW)	TXPAREN			TXD[11]

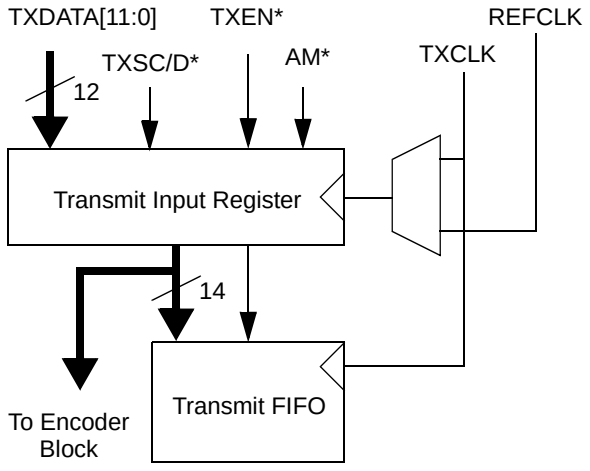


Figure 2. Transmit Input Register.

mode is determined by the RANGESEL and SPDSEL inputs and the FIFO Bypass (FIFOBYP*) signal.

Synchronous Interface

Synchronous interface clocking operates the entire transmit data path synchronous to REFCLK. It is enabled by connecting FIFOBYP* LOW to disable the internal FIFOs.

Asynchronous Interface

Asynchronous interface clocking controls the writing of host bus data into the Transmit FIFO. It is enabled by setting FIFOBYP* HIGH to enable the internal FIFOs. In these configurations, all writes to the Transmit Input Register, and associated transfers to the Transmit FIFO, are controlled by TXCLK. The remainder of the transmit data path is clocked by REFCLK or synthesized derivatives of REFCLK.

Notes:
1. All open cells are ignored.
2. First bit shifted out. Others follow in numerical order creating an NRZ pattern.

UTOPIA Timing Model

The UTOPIA timing model allows multiple CY7C924ADX transmitters to be addressed and accessed from a common host bus, using the protocols defined in the ATM Forum UTOPIA interface standards. It is enabled by setting EXTFIFO LOW.

In UTOPIA timing, the TXEMPTY* and TXFULL* outputs and TXEN* input, are all active LOW signals. If the CY7C924ADX is addressed by AM*, it becomes "selected" when TXEN* is asserted LOW. Following selection, data is written into the Transmit FIFO on every clock cycle where TXEN* remains LOW.

Cascade Timing Model

The Cascade timing model is a variation of the UTOPIA timing model. Here the TXEMPTY* and TXFULL* outputs, and TXEN input, are all active HIGH signals. Cascade timing makes use of the same address and selection sequences as UTOPIA timing, but write data accesses use a delayed write. This delayed write is necessary to allow direct coupling to external FIFOs, or to state machines that initiate a write operation one clock cycle before the data is available on the bus.

Cascade timing is enabled by setting EXTFIFO HIGH.

When used for FIFO depth expansion, Cascade timing allows the size of the internal Transmit FIFO to be expanded to an almost unlimited depth. It allows a CY7C42x5 series synchronous FIFO to be attached to the transmit interface without any extra logic, as shown in *Figure 3*.

Transmit FIFO

The Transmit FIFO is used to buffer data captured in the input register for later processing and transmission. This FIFO is sized to hold 256 14-bit characters. When the Transmit FIFO is enabled, and a Transmit FIFO write is enabled (the device is selected and TXEN* is sampled asserted), data and command are captured in the transmit input register and stored into the Transmit FIFO. All Transmit FIFO write operations are clocked by TXCLK.

The Transmit FIFO presents Full, Half-Full, and Empty FIFO flags. These flags are provided synchronous to TXCLK. When the Transmit FIFO is enabled, it allows operation with a Moore-type external controlling state machine. When configured for Cascade timing, the timing and active levels of these signals are also designed to support direct expansion to Cypress CY7C42x5 synchronous FIFOs.

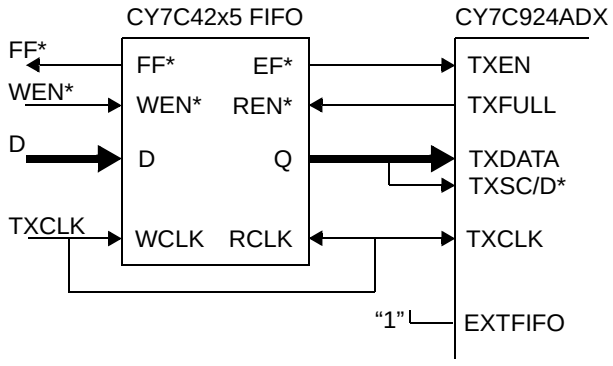


Figure 3. External FIFO Depth Expansion of the CY7C924ADX Transmit Data Path.

Regardless of bus width (8- or 10-bit characters) the Transmit FIFO can be clocked at any rate from DC to 50 MHz. This gives the Transmit FIFO a maximum bandwidth of 50 million characters per second. Since the serial outputs can only move 20 million characters per second at their fastest operating rate, there is ample time to service multiple CY7C924ADX HOTLinks with a single controller.

The read port of the Transmit FIFO is connected to a logic block that performs data formatting and validation. All data read operations from the Transmit FIFO are controlled by a Transmit Control State Machine that operates synchronous to REFCLK.

Transmit Formatter and Validation

The Transmit Formatter and validation logic performs three primary functions:

- Parity checking
- Data format control
- Byte-packing

In addition to these logic functions, this block also controls the timing for the transfer of data from the Transmit Input Register, Transmit FIFO, or Elasticity Buffer.

Parity Checking

Parity checking is enabled in 8-bit encoded mode when the internal FIFOs are disabled (FIFOBYP* is LOW) and PAREN (TXSOC) is HIGH. ODD parity is supported, which requires at least one bit of the data bus to always be a logic-1. The eight data bits (TXDATA[7:0]), TXSC/D*, TXSVS, and TXOPIN (TXDATA[8]) are covered by the parity checking hardware.

If a parity error is detected in a character, that character is not transmitted, but is replaced with the C0.7 Exception character (see Special Character codes in *Table 11*). This prevents bad data from knowingly being transmitted and informs the receiver that an error was detected in the source data stream.

Parity errors are reported to the transmit interface through the TXPER output. This output pulses HIGH for one REFCLK period, one or more cycles after the cycle where the parity error was detected.

Transmit Data Formatting

The CY7C924ADX supports a number of protocol enhancements over a raw physical-layer device. These enhancements are made possible in part through the use of the Transmit and Receive FIFOs. These FIFOs allow the CY7C924ADX to manage the data stream to a much greater extent than was possible before. In addition to the standard 8B/10B encoding used to improve serial data transmission, the CY7C924ADX also supports:

- marking of packet or cell boundaries using TXSOC
- an expanded command set
- ability to address and route packets or frames to specific receivers

All three of these capabilities are supported for both 8- and 10-bit encoded character sizes, and are made possible through use of the TXSOC bit. This bit is interpreted, along with TXSC/D* and TXSVS, in those modes where both the Transmit FIFO and the Encoder are enabled. All three bits determine how the data associated with them is processed for transmission. These operations are listed in *Table 2*.

The entries in *Table 2*, where TXSOC is LOW generate the same characters in the serial data stream as a standard CY7B923 HOTLink Transmitter, which uses the ANSI standard 8B/10B character set. The data, command, and exception character encodings are listed in the Data and Special Character code tables (*Tables 10* and *11*) found near the end of this data sheet.

Table 2. Transmit Data Formatting

TXSOC	TXSC/D*	TXSVS	Data Format Operation
0	0	0	Normal Data Encode
0	0	1	Replace Character with C0.7 Exception
0	1	0	Normal Command Encode
0	1	1	Replace Character with C0.7 Exception
1	0	0	Send Start of Cell Marker (C8.0) + Data Character
1	0	1	Replace Character with C0.7 Exception
1	1	0	Send Extended Command Marker (C9.0) + Data Character
1	1	1	Send Serial Address Marker (C10.0) + Data Character

When the TXSOC bit (as read from the Transmit FIFO) is HIGH, an extra character is inserted into the data stream. This extra character is always a Special Character code (see *Table 11*) that is used to inform the remote receiver that the immediately following character should be interpreted differently from its normal meaning. The associated character present on TXDATA[x:0] is always encoded as a data character.

The 100b combination (TXSOC = 1, TXSC/D* = 0, and TXSVS = 0) is used as a marker for the start of a cell, frame, or packet of data being sent across the interface. When a character is read from the Transmit FIFO with this combination of bits set, a C8.0 Special Character code is sent to the Encoder prior to sending the associated data character.

The 101b character format has the same function as the 001b and 011b normal data modes. It instructs the encoder to discard the associated data character and to replace it with a C0.7 Exception character.

The 110b character format is used to expand the command space beyond that available with the default 8B/10B code. The 8B/10B code normally supports a *data* space of 256 data characters, and a *command* (non-data) space of twelve command characters (C0.0–C11.0 in *Table 11*). For those data links where this is not sufficient, the 110b format can be used to mark the associated data as an extended command. This expands the command space to 256 commands (in addition to some of the present twelve). When a character is read from the Transmit FIFO with these bits set, a C9.0 Special Character code is sent to the Encoder prior to sending the data character.

The 111b character format is used to send a serial addresses to attached receivers. These serial addresses allow a host to direct (the following) data to a specific destination or destinations, when the CY7C924ADX devices are connected in a ring or bus topology.

The Serial Address marker may also be used to send packet identification fields, sequence numbers, or other high-level routing information for those point-to-point connections that do not require physical address capabilities, however, the reporting of the address field contents may be affected by the present receiver discard policy. This marking or tagging capability can be performed with the 100b or 110b character formats without concern for receiver discard policy.

When a character is read from the Transmit FIFO with these bits set, a C10.0 Special Character is sent to the Encoder prior to sending the associated data character.

Byte-Packer

The byte-packer is a logical construct, used to control the efficient segmentation of 10-bit source data into 8-bit characters. This conversion allows these characters to be transported using 8B/10B encoding, with the same encoding overhead (20%) as when sending 8-bit characters. Because the serializer continues to operate using 10-bit transmission characters, this encoding mode can only operate with the Transmit FIFO enabled.

The byte-packer operates by taking pieces of one or more 10-bit characters, combining them into 8-bit groups, and passing these groups to the 8B/10B encoder. It takes exactly five 8-bit characters to transport four 10-bit characters. The allocation is performed, as shown in *Figure 4*, where the low-order eight bits of the first 10-bit character (A[7:0]) are passed to the encoder on the first clock cycle. During the second clock cycle the remaining two bits of the first character are combined with the lower six bits of the second 10-bit character (B[5:0]+A[9:8]). In the third clock cycle the remaining four bits of the second 10-bit character are combined with the lower four bits of the third 10-bit character (C[3:0]+B[9:6]). In the fourth clock cycle the remaining six bits of the third 10-bit character are combined with the lower two bits of the fourth 10-bit character (D[1:0]+C[9:4]). In the fifth clock cycle the remaining eight bits of the fourth 10-bit character are passed to the encoder (D[9:2]).

This process repeats for additional data characters present in the FIFO. If at any time the Transmit FIFO is emptied, and a portion of a 10-bit character has not yet been transmitted, the remaining bits of the 8-bit character are filled with dummy bits before that character is passed to the encoder. The 8-bit character containing these dummy bits is immediately followed by

a C5.0 (K28.5) fill character, which resets the sequencer boundaries to the first character position.

Encoder Block

The Encoder logic block performs two primary functions: encoding the data for serial transmission and generating BIST (Built-In Self Test) patterns to allow at-speed link and device testing.

BIST LFSR

The Encoder logic block operates on data stored in a register. This register accepts information directly from the Transmit FIFO, the Transmit Input Register, the 10/8 Byte-Packer, or from the Transmit Control State Machine when it inserts special characters into the data stream.

This same register is converted into a Linear Feedback Shift Register (LFSR) when the Built-In Self-Test (BIST) pattern generator is enabled (TXBISTEN* is LOW). When enabled, this LFSR generates a 511-character sequence that includes all Data and Special Character codes, including the explicit violation symbols. This provides a predictable but pseudo-random sequence that can be matched to an identical LFSR in the Receiver.

The specific patterns generated are described in detail in the Cypress application note “HOTLink Built-In Self-Test.” The sequence generated by the CY7C924ADX is identical to that in the CY7B923 and CY7C929, allowing interoperable systems to be built when used at compatible serial signaling rates.

Encoder

The data passed through the Transmit FIFO and formatter, or as received directly from the Transmit Input Register, is seldom in a form suitable for transmission across a serial link. The characters must usually be processed or transformed to guarantee:

- a minimum transition density (to allow the serial receiver PLL to extract a clock from the data stream)
- a DC-balance in the signaling (to prevent baseline wander)

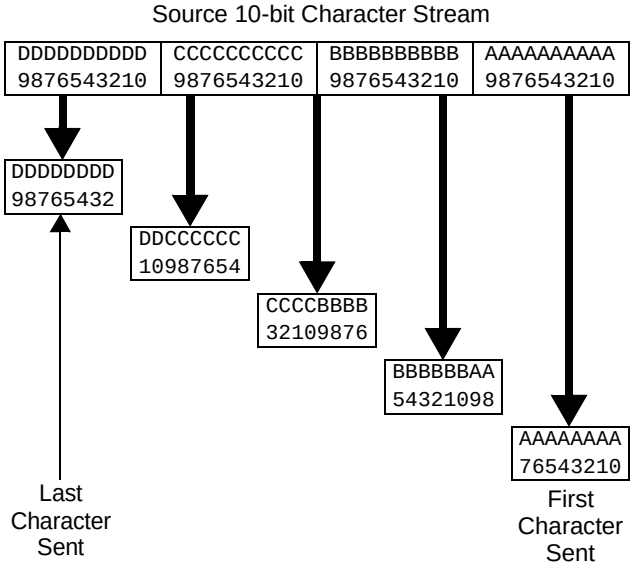


Figure 4. Byte-Packer 10-to-8 Character Mapping.

- run-length limits in the serial data (to limit the bandwidth of the link)
- some way to allow the remote receiver to determine the correct character boundaries (framing).

The CY7C924ADX contains an integrated 8B/10B encoder that accepts 8-bit data characters and converts these into 10-bit transmission characters that have been optimized for transport on serial communications links. The 8B/10B encoder can be bypassed for those system that operate with external 8B/10B encoders, or use alternate forms of encoding or scrambling to ensure good transmission characteristics. The operation of the 8B/10B encoding algorithm is described in detail later in this data sheet, and the complete encoding tables are listed in *Tables 10* and *11*.

When the Encoder is enabled, the transmit data characters (as passed through the Transmit FIFO and formatter) are converted to either a 10-bit Data symbol or a 10-bit Special Character, depending upon the state of the TXSC/D* input. If TXSC/D* is HIGH, the data inputs represent a Special Character code and are encoded using the Special Character encoding rules in *Table 11*. If TXSC/D* is LOW, the data inputs are encoded using the Data Character encoding in *Table 10*.

When operated without parity checking, the MSB of each character is used to override the contents of the remaining bits in the character. If this bit (TXSVS) is HIGH, the respective character is replaced with an SVS (C0.7) character. This can be used to check error handling system-logic in the receiver controller or for proprietary applications.

The 8B/10B encoder is standards compliant with ANSI/NCITS ASC X3.230-1994 (Fibre Channel), IEEE 802.3z (Gigabit Ethernet), the IBM ESCON and FICON channels, and ATM Forum standards for data transport.

The 8B/10B coding function of the Encoder can be bypassed for systems that include an external coder or scrambler function as part of the controller or host system. This is performed by setting ENCBYP* LOW. With the encoder bypassed, each 10-bit character (as captured in the Transmit Input Register) is passed directly to the Transmit Shifter (or Transmit FIFO) without modification.

Transmit Shifter

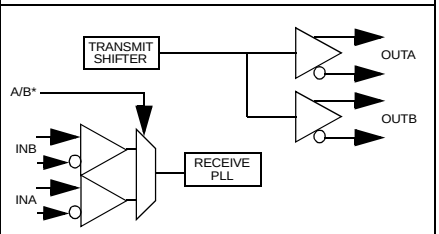
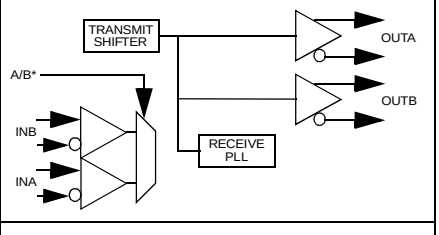
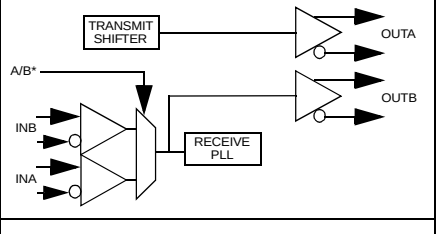
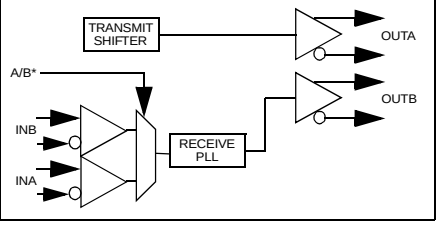
The Transmit Shifter accepts 10-bit parallel data from the Encoder block once each character time, and shifts it out the serial interface output buffers using a PLL-multiplied bit-clock. This bit-clock runs at 2.5, 5, or 10 times the REFCLK rate (3, 6, or 12 times when BYTE8/10* is LOW) as selected by RANGESEL and SPDSEL (see *Table 4*). Timing for the parallel transfer is controlled by the counter and dividers in the Clock Multiplier PLL and is not affected by signal levels or timing at the input pins.

Bits in each character are shifted out LSB first, as required by ANSI and IEEE standards for 8B/10B coded serial data streams.

Routing Matrix

The Routing Matrix is a set of precision multiplexors that allow various combinations of Transmit Shifter, buffered INA± or INB± serial line receiver inputs, or a reclocked serial line receiver input to be transmitted from the OUTB± serial data outputs. The signal routing for the transmit serial outputs is controlled primarily by the DLB[1:0] inputs as listed in *Table 3*.

Table 3. Transmit Data Routing Matrix

DLB[1]	DLB[0]	Data Connections
0	0	
0	1	
1	0	
1	1	

Serial Line Drivers

The serial interface PECL Output Drivers (ECL referenced to +5V) are the transmission line drivers for the serial media. OUTA± receives its data directly from the transmit shifter, while OUTB± receives its data from the Routing Matrix. These two outputs (OUTA± and OUTB±) are capable of direct connection to +5V optical modules, and can also directly drive DC- or AC-coupled transmission lines.

The PECL-compatible Output Drivers can be viewed as programmable current sources. The output voltage is determined by the output current and the load impedance Z_{LOAD}. The desired output voltage swing is therefore controlled by the current-set resistor R_{CURSET} associated with that driver. Different R_{CURSET} values are required for different line impedance/amplitude combinations. The output swing is designed to center around V_{DD}–1.33V. Each output must be externally biased to V_{DD}–1.33V.

This differential output-swing can be specified two ways: either as a peak-to-peak voltage into a single-end load, or as an absolute differential voltage into a differential load.

When specified into a single-ended load (one of the outputs switching into a load), the single output will both source and sink current as it changes between its HIGH and LOW levels. The voltage difference between this HIGH level and LOW level determine the peak-to-peak signal-swing of the output. This amplitude relationship is controlled by the load impedance on the driver, and by the resistance of the R_{CURSET} resistor for that driver, as listed in *Eq. 1*

$$R_{CURSET} = \frac{180 \times Z_{LOAD}}{V_{OPP}} \quad \text{Eq. 1}$$

In *Eq. 1*, V_{OPP} is the difference in voltage levels at one output of the differential driver when that output is driving HIGH and LOW, Z_{LOAD} is that load seen by the one output when it is sourcing and sinking current. With a known load impedance and a desired signal swing, it is possible to calculate the value of the associated CURSETA or CURSETB resistor that sets this current.

Unused differential output drivers should be left open, and can reduce their power dissipation by connecting their respective CURSETx input to V_{DD}.

Transmit PLL Clock Multiplier

The Transmit PLL Clock Multiplier accepts an external clock at the REFCLK input, and multiples that clock by 2.5, 5, or 10 (3, 6, or 12 when BYTE8/10* is LOW and the encoder is disabled) to generate a bit-rate clock for use by the transmit shifter. It also provides a character-rate clock used by the Transmit Controller state machine.

The clock multiplier PLL can accept a REFCLK input between 8 MHz and 40 MHz, however, this clock range is limited by the operation mode of the CY7C924ADX as selected by the SPDSEL and RANGESEL inputs, and to a limited extent, by the BYTE8/10* and FIFOBYP* signals. The operating serial signalling rate and allowable range of REFCLK frequencies is listed in *Table 4*.

Table 4. Speed Select and Range Select Settings

SPDSEL	RANGESEL	Serial Data Rate (MBaud)	REFCLK[4] Frequency (MHz)
LOW	LOW	50–100	10–20
LOW	HIGH[3]	50–100	20–40
HIGH	LOW	100–200	10–20
HIGH	HIGH	100–200	20–40

Notes:

3. When SPDSEL is LOW and the FIFOs are bypassed (FIFOBYP* is LOW), the RANGESEL input is ignored and is internally mapped to the LOW setting.
4. When configured for 12-bit pre-encoded data (BYTE8/10* and ENCBYP* are both LOW) the allowable REFCLK ranges are 8.33to 16.67 MHz and 16.67 to 33.33 MHz.

Transmit Control State Machine

The Transmit Control State Machine responds to multiple inputs to control the data stream passed to the encoder. It operates in response to:

- the state of the FIFOBYP* and LOOPTX inputs
- the presence of data in the Transmit FIFO
- the contents of the Transmit FIFO
- the contents of the Elasticity Buffer
- the state of the transmitter BIST enable (TXBISTEN*)
- the state of external halt signals (TXHALT* and TXSTOP*)

These signals are used by the Transmit Control State Machine to control the data formatter, read access to the Transmit FIFO and Elasticity Buffer, the Byte-Packer, and BIST. They determine the content of the characters passed to the Encoder and Transmit Shifter.

When the Transmit FIFO is bypassed, the Transmit Control State Machine operates synchronous to REFCLK. In this mode, data from the TXDATA bus (or other source) is passed directly from the Input Register to the Pipeline Register. If no data is enabled into the Input register (TXEN* is deasserted or TXFULL* is asserted) then the Transmit Control State Machine presents a C5.0 Special Character code to the Encoder to maintain link synchronization.

If both the Encoder and Transmit FIFO are bypassed and no data is enabled into the Input Register, the Transmit Control State Machine injects an alternating disparity sequence of pre-encoded (10-bit) forms of the C5.0 characters. This also occurs if the Encoder is bypassed, the Transmit FIFO is enabled, and the Transmit FIFO is empty. However, since disparity tracking is part of the Encoder, the transmitted C5.0 characters may generate a running disparity error at the remote receiver. If the attached receiver has its Decoder enabled, these characters may be reported as a normal C5.0, or as a C1.7 or C2.7 (K28.5 with incorrect running disparity).

External Control of Data Flow

The Transmit Control State Machine supports three different types of external control:

- TXSTOP*
- TXHALT*
- TXINT

These control signal inputs are only interpreted when the Transmit FIFO is enabled. They effect the transmission of data by bringing external signals to the state machine without sending the signals through the Transmit FIFO.

TXSTOP* is used to stop transmission of the next packet or cell of data in the Transmit FIFO. When asserted (LOW) the Transmit Control State Machine continues to read and process characters in the Transmit FIFO until a location is read with the TXSOC bit set. Once a TXSOC is detected, the state machine sends out C5.0 fill characters until TXSTOP* is deasserted (HIGH) for one or more character times. When TXSTOP* is sampled deasserted it allows the next character with TXSOC set to be read from the Transmit FIFO and passed to the Encoder.

When TXSTOP* is used to control the flow of data, it is asserted (LOW) most of the time. To allow a cell or frame to pass, it only needs to be deasserted (HIGH) for one TXCLK cycle (assuming the transmit controller is at a cell boundary). Once the first character of the cell is transmitted the remainder of that cell is also processed. This allows the host system to control the transmission of data across the interface on a cell-by-cell or packet-by-packet basis.

TXHALT* (TXDATA[9]) is an immediate form of TXSTOP*. Instead of continuing to transmit data until a TXSOC is found, the assertion of TXHALT* causes character processing to stop at the next FIFO character location. No additional data is read from the Transmit FIFO until TXHALT* is deasserted (HIGH).

NOTE: If the Encoder is bypassed, TXDATA[9] is a data input and not TXHALT*. Since in this mode the interface does not interpret the TXSOC bit, the TXSTOP* signal assumes the same functionality as TXHALT*.

TXINT is used to send one of two interrupt characters from the local transmitter to a remote receiver. While it also bypasses the Transmit FIFO, it does not stop data transmission (at least not directly).

The Transmit Control State Machine responds to transitions on the TXINT input. When TXINT transitions from 0→1, a C0.0 (K28.0) Special Character code is sent. When TXINT transitions from 1→0, a C3.0 (K28.3) Special Character code is sent. The reception of these characters generates an equivalent action on the attached receiver's RXINT status output.

The combination of RXHALF*, TXINT, RXINT, and TXHALT* may be used to prevent a remote FIFO overflow, which would result in lost data. This back-pressure mechanism can significantly improve data integrity in systems that cannot guarantee the full bandwidth of the host system at all times.

Elasticity Buffer

A short (8-character) FIFO is contained between the receive and transmit paths. This FIFO is used to separate the time domains of the received serial data stream and the outbound transmit data stream. This permits retransmission of received data without worry of jitter gain or jitter transfer. This allows error-free transmission of the same data, when configured in daisy-chain or ring configurations, to an unlimited number of destinations.

This Elasticity Buffer is enabled when the LOOPTX input is asserted HIGH. This directs the receiver to place all non-C5.0 (K28.5) characters into the Elasticity Buffer. LOOPTX also directs the Transmit Control State Machine to read data from the Elasticity Buffer instead of from the Transmit FIFO.

While retransmitting data from the Elasticity Buffer, the Transmit FIFO is available for pre-loading of data to be transmitted. Once LOOPTX is deasserted (LOW), normal data transmission from the Transmit FIFO resumes.

This LOOPTX capability is only possible when sending 8-bit encoded data streams. It cannot be used with byte-packed or non-encoded data streams, and requires that the Transmit and Receive FIFOs are enabled. It also requires that the receiver be configured to process embedded commands (receiver Discard Policy cannot be 0).

Serial Line Receivers

Two differential line receivers, INA± and INB±, are available for accepting serial data streams, with the active input selected using the A/B* input. The DLB[1:0] inputs allow the transmit Serializer output to be selected as a third input serial stream, but this path is generally used only for diagnostic purposes. The serial line receiver inputs are all differential, and will accommodate wire interconnect with filtering losses or transmission line attenuation greater than 9 dB ($V_{DIF} \geq$ 200 mV, or 400 mV peak-to-peak differential) or can be directly connected to +5V fiber-optic interface modules (any ECL logic family, not limited to ECL 100K). The common-mode tolerance of these line receivers accommodates a wide range of signal termination voltages.

As can be seen in *Table 3*, these inputs are configured to allow single-pin control for most applications. For those systems requiring selection of only INA± or INB±, the DLB[1:0] signals can be tied LOW, and the A/B selection can be performed using only A/B*. For those systems requiring only a single input and a local loopback, the

A/B* can be tied HIGH or LOW, DLB[1] signal can be tied LOW and DLB[0] can be used for loopback control.

The level-restored (10b) and reclocked (11b) settings make use of one of the transmit data outputs. When configured for level-restored or reclocked data, the selected input is retransmitted on OUTB±. The level-restored connection simply buffers the input signal allowing a “bus-like” connection to be constructed without concern for multi-drop PECL signal layout issues.

The reclocked connection buffers a PLL-filtered copy of the selected input data stream. This removes most of the high-frequency jitter that accumulates on a signal when sent over long transmission lines. Unlike data retransmitted from the Elasticity Buffer, the output data stream is clocked by a recovered clock, not by a derivative of the local REFLCK input. This allows a data source to provide data to multiple recipients, but can suffer from jitter peaking when communicated through several PLLs. The reclocked connection may be required when sending non-8B/10B coded data streams, or data streams that cannot tolerate the data forwarding policies of the Elasticity Buffer.

This reclocked output stream may also be beneficial in systems requiring very low latency. The internal data delays for a reclocked serial stream are a small number of bits, while data sent through the Elasticity Buffer incurs a delay of a small number of characters.

Signal Detect

The selected Line Receiver (that routed to the clock and data recovery PLL) is simultaneously monitored for:

- analog amplitude (>400 mV pk-pk)
- transition density
- received data stream outside normal frequency range (±400 ppm)
- and carrier detected.

All of these conditions must be valid for the Signal Detect block to indicate a valid signal is present. This status is presented on the LFI* (Link Fault Indicator) output, which changes synchronous to RXCLK. While link status is monitored internally at all times, it is necessary to have transitions on RXCLK to allow this signal to change externally.

Clock/Data Recovery

The extraction of a bit-rate clock and recovery of data bits from the received serial stream is performed within the Clock/Data Recovery (CDR) block. The clock extraction function is performed by a high-performance embedded phase-locked loop (PLL) that tracks the frequency of the incoming bit stream and aligns the phase of its internal bit-rate clock to the transitions in the serial data stream.

The CDR makes use of the clock present at the REFCLK input. It is used to ensure that the VCO (within the CDR) is operating at the correct frequency (rather than some harmonic of the bit rate), to improve PLL acquisition time, and to limit unlocked frequency excursions of the CDR VCO when no data is present at the serial inputs.

Regardless of the type of signal present, the CDR will attempt to recover a data stream from it. If the frequency of the recovered data stream is outside the limits for the range controls, the CDR PLL will track REFCLK instead of the data stream. When the frequency of the selected data stream returns to a valid frequency, the CDR PLL is allowed to track the received data stream. The frequency of REFCLK is required to be within ±400 ppm of the frequency of the clock that drives the REFCLK signal at the remote transmitter to ensure a lock to the incoming data stream.

For systems using multiple or redundant connections, the LFI* output can be used to select an alternate data stream. When an LFI* indication is detected, external logic can toggle selection of the INA± and INB± inputs through the A/B* input. When a port switch takes place, it is necessary for the PLL to reacquire the new serial stream and frame to the incoming characters.

Clock Divider

This block contains the clock division logic, used to transfer the data from the Deserializer/Framer to the Decoder once every character (once every ten or twelve bits) clock. This counter is free running and generates outputs at the bit-rate divided by 10 (12 when the BYTE8/10* and ENCBYP* are LOW). When the Receive FIFO is bypassed, one of these generated clocks is driven out the RXCLK pin.

Deserializer/Framer

The CDR circuit extracts bits from the serial data stream and clocks these bits into the Shifter/Framer at the bit-clock rate. When enabled, the Framer examines the data stream looking for C5.0 (K28.5) characters at all possible bit positions. The location of this character in the data stream is used to determine the character boundaries of all following characters.

The framer operates in one of three different modes, as selected by the RFEN input. When RFEN is first asserted (HIGH), the framer is allowed to reset the internal character boundaries on any detected C5.0 character.

Once RFEN has been HIGH for greater than approximately 2000 character clock cycles, the multi-byte framer is enabled. This requires two C5.0 characters within a span of five characters, with both C5.0 characters located on identical 10-bit character boundary locations, before the framer is allowed to reset the internal character boundary.

If RFEN is LOW, the framer is disabled and no changes are made to character boundaries.

The framer in the CY7C924ADX operates by shifting the internal character position to align with the character clock. This ensures that the recovered clock does not contain any significant phase changes/hops during normal operation or framing, and allows the recovered clock to be replicated and distributed to other circuits using PLL-based logic elements.

Decoder Block

The decoder logic block performs two primary functions: decoding the received transmission characters back into Data and Special Character codes, and comparing generated BIST patterns with received characters to permit at-speed link and device testing.

10B/8B Decoder

The framed parallel output of the Deserializer is passed to the 10B/8B Decoder where, if the Decoder is enabled, it is transformed from a 10-bit transmission character back to the original Data and Special Character codes. This block uses the standard decoder patterns in *Tables 10* and *11* of this data sheet. Data patterns are indicated by a LOW on RXSC/D*, and

Special Character codes are indicated by a HIGH. Invalid patterns or disparity errors are signaled as errors by a HIGH on RXRVS, and by specific Special Character codes.

If the Decoder is bypassed and BYTE8/10* is HIGH, the ten (10) data bits of each transmission character are passed unchanged from the framer to the Pipeline Register.

When the Decoder is bypassed and BYTE8/10* is LOW, the twelve (12) data bits of each transmission character are passed unchanged from the framer to the Pipeline Register.

BIST LFSR

The output register of the Decoder block is normally used to accumulate received characters for delivery to the Receive Formatter block. When configured for BIST mode (RXBISTEN* is LOW), this register becomes a signature pattern generator and checker by logically converting to a Linear Feedback Shift Register (LFSR). When enabled, this LFSR generates a 511-character sequence that includes all Data and Special Character codes, including the explicit violation symbols. This provides a predictable but pseudo-random sequence that can be matched to an identical LFSR in the Transmitter. When synchronized with the received data stream, it checks each character in the Decoder with each character generated by the LFSR and indicates compare errors at the RXRVS output of the Receive Output Register.

The LFSR is initialized by the BIST hardware to the BIST loop start code of D0.0 (D0.0 is sent only once per BIST loop). Once the start of the BIST loop has been detected by the receiver, RXRVS is asserted for pattern mismatches between the received characters and the internally generated character sequence. Code rule violations or running disparity errors that occur as part of the BIST loop do not cause an error indication. RXFULL* pulses asserted for one RXCLK cycle per BIST loop and can be used to check test pattern progress.

The specific patterns checked by the receiver are described in detail in the Cypress application note “HOTLink Built-In Self-Test.” The sequence compared by the CY7C924ADX is identical to that in the CY7B933, allowing interoperable systems to be built when used at compatible serial signaling rates.

If a large number of errors are detected, the receive BIST state machine aborts the compare operations and resets the LFSR to the D0.0 state to look for the start of the BIST sequence again.

Receive Formatter

The Receive Formatter performs four primary functions:

- Data Formatting
- Address Matching
- Byte-Unpacking
- Parity Generation

Receive Data Formatting

The protocol enhancements of the transmit path are mirrored in the receive path logic. The majority of these enhancements require that the Receive FIFO be enabled to allow the CY7C924ADX to manage the data stream. In addition to the standard 10B/8B decoding used for character reception and recovery, the CY7C924ADX also supports:

- marking of packet or cell boundaries using RXSOC
- an expanded control/command character set
- ability to accept or discard data based on an embedded address
- the ability to filter receive data of non-essential information

All of these capabilities are supported for both 8- and 10-bit character sizes, and are made possible through use of the RXSOC bit. RXSOC is generated upon reception of the C8.0, C9.0, or C10.0 Special Character codes, in those modes where both the Receive FIFO and the Decoder are enabled.

The entries in *Table 5* show how the RXSOC, RXSC/D*, and RXRVS bits are formatted to indicate the reception of specific characters and character combinations. Normal Data and Special Character code characters are indicated by RXSOC being LOW (0). This allows the standard Special Characters codes to also be reported and output.

Table 5. Receive Data Formatting

RXSOC	RXSC/D*	RXRVS	Data Format Indication
0	0	0	Normal Data Character
0	0	1	Reserved
0	1	0	Normal Command Character
0	1	1	Received C0.7 Exception Character or Other Character Exception (as listed in *Table 11*)
1	0	0	Received Start of Cell Marker (C8.0) + Data Character
1	0	1	Received Illegal Sequence
1	1	0	Received Extended Command Marker (C9.0) + Data Character (interpreted as a command)
1	1	1	Received Serial Address Marker (C10.0) + Data Character (interpreted as an address)

Individual character errors that are not part of one of the supported sequences (Start of Cell, Extended Command, or Serial Address) are marked by the 011b (RXSOC = 0, RXSC/D* = 1, and RXRVS = 1) decode.

Anytime RXSOC is reported HIGH (1) at least one of the C8.0, C9.0, or C10.0 characters was received as a valid character. If the immediately following character is a valid Data character, then the corresponding combination of RXSOC, RXSC/D*, and RXRVS indicate the type of information received. If the immediately following character is a Special Character code of any type (even a C5.0), then a 101b is posted to indicate an illegal sequence was received.

An illegal sequence can be caused by a remote transmitter sending incorrect information, or by receiving data corrupted during transmission. When such an error is detected, the 101b status bits are posted and the associated data field is set to the Special Character code that was received without error (C8.0, C9.0, or C10.0 reported as D8.0, D9.0, or D10.0 along with the 101b status). This information is provided to assist in debugging link or protocol faults.

The 100b indication is used to mark the associated Data character as the first character of a new frame, packet, cell, or other data construct used by the system. The Data characters and Special Character codes that follow this marker are written to

the Receive FIFO (if the present address matching requirements are satisfied).

The 110b indication is used to mark the associated data character as the first character of an extended command. In reality there is no limit to the number of immediately following data characters that can be considered part of this command. The most common interpretation is based on the configured bus width, such that single-character configurations support the associated character as the extended command, providing up to 256 extended commands for 8-bit data and 1024 for 10-bit byte-packed data.

This marker is treated internally the same as the 100b Start Of Cell indication, which allows it to be used to mark the boundary of any user-specific information. As a boundary or cell marker, the immediately following data can be a data field, a header, a stream identifier, a transaction number, a packet length indicator, or any of a number of pieces of information connected to a data transfer.

> **NOTE:** In reality, the 100b and 110b indicators can be used interchangeably; i.e., the 100b indication can be used to mark extended commands while the 110b indication can be used to mark the start of cells.

The 111b indication is used to mark the start of a Serial Address field. Unlike the Start Of Cell and Extended Command markers, which have no specific data-field length associated with them, the associated Serial Address is always comprised of the immediately following single data character, and supports a fixed 8-bit or 10-bit address field format in 8-bit or 10-bit byte-packed data formats.

When this serial address is received it may be passed to the Receive FIFO or discarded (see *Table 6*)**.**

Address Matching

For those modes where address matching is enabled, the CY7C924ADX's ability to accept or discard data can be controlled by the remote transmitter. This is often useful in configurations with one or more data sources and multiple data destinations.

Each CY7C924ADX contains an 8-bit or 10-bit Serial Address Register that is compared with the first data character received following a Serial Address marker (C10.0). This character constitutes an address, which can be configured for one of two modes for address matching. The first mode is used for multicast addresses, where a bit-wise AND is performed on each bit of the address character received, with the contents of each of the bits in the Serial Address Register. If any of the same bit locations in the register and the received data are both set to '1', a multicast address match is declared and the following data and Special Character codes are interpreted and passed to the Receive FIFO.

If the multicast address field is ever received as all 1s (FFh or 3FFh), the receiver always accepts the data. This all 1s setting is the broadcast address and is used to send data to all receivers.

This all 1s setting also has special meaning when written to the Serial Address Register. When the multicast address field is written to an all 1s (FFh or 3FFh) state, the receiver operates in promiscuous mode, and receives all data, regardless of the contents of any serial address commands received. This is also the default or power-up state of the Serial Address register.

The second mode of operation for address matching is when the Serial Address register contains a unique device address, and is compared with the character received following the C10.0 Serial Address marker. This unicast address requires an exact match between all 8 or 10 bits to declare a match found and allow the following data to pass.

When the Elasticity Buffer is enabled, all received characters (except C5.0) are written to the Elasticity Buffer, regardless of the state or configuration of any present address match. This allows one or more sources to send data to multiple receivers with the receivers connected in a ring or daisy-chain topology. By prefacing cells containing data with an address field, it is possible to have each receiver only process data specifically directed to it.

Byte-Unpacker

The Byte-Unpacker is used to re-assemble 10-bit characters from a received stream of decoded 8-bit characters. This reassembly process is designed to allow transmission of the same embedded commands, serial addresses, and Start of Cell markers that are used with 8-bit data characters. Because of the change in time per received encoded character versus delivered 10-bit data character, this unpacking process is only possible with the Receive FIFO enabled.

The byte-unpacker reverses the character segmentation shown in *Figure 4*. It takes five data characters and combines them into four 10-bit characters. This five-state unpacking process is re-started by the detection of any Special Character code in the Decoder, including the C5.0 (K28.5) fill character. Since usage of the Elasticity Buffer inserts and deletes C5.0 characters (as necessary) to handle the speed differences between the receive and transmit character clocks, it is not possible to send byte-packed data through the Elasticity Buffer. To send 10-bit packed data from one source to multiple destinations it is necessary to either use a star topology of interconnect, or make use of the buffered and reclocked serial input-to-output connections controlled by the Routing Matrix.

Parity Generation

Parity generation is enabled only when the receive FIFO is bypassed (FIFOBYP* is LOW) and the decoder is enabled (ENCBYP* is LOW). In this mode ODD parity is provided on the RXDATA[8] output. This ensures that at least one bit of the data bus is always a logic-1. The generated ODD parity is valid for the RXD[7:0], RXSC/D*, and RXRVS signals.

Receive Control State Machine

The Receive Control State Machine responds to multiple input conditions to control the routing and handling of received characters. It controls the staging of characters across various registers and the Receive FIFO. It also interprets all embedded Special Character codes, and converts the appropriate ones to specific bit combinations in the Receive FIFO. It controls the various discard policies and error control within the receiver, and operates in response to:

- the received character stream
- the detection and validation of serial addresses
- the room for additional data in the Receive FIFO
- the state of the receiver BIST enable (RXBISTEN*)
- the state of LOOPTX
- the state of FIFOBYP*

These signals and conditions are used by the Receive Control State Machine to control the Receive Formatter, write access to the Receive FIFO, write access to the Elasticity Buffer, the Byte-Unpacker, the Receive Output register, and BIST. They determine the content of the characters passed to each of these destinations,

The Receive Control State Machine always operates synchronous to the recovered character clock (bit-clock/10 or bit-clock/12). When the Receive FIFO is bypassed, RXCLK becomes an output that changes synchronous to the internal character clock. RXCLK operates at the same frequency as the internal character clock.

Discard Policies

When the Receive FIFO is enabled, the Receive Control State Machine has the ability to selectively discard specific characters from the data stream that are determined by the present configuration as being unnecessary. When discarding is enabled, it reduces the host system overhead necessary to keep the Receive FIFO from overflowing and losing data.

The discard policy is configured as part of the operating mode and is set using the RXMODE[1:0] inputs. The four discard policies are listed in *Table 6*.

Table 6. Receiver Discard Policies

Policy #	Policy Description
0	Keep all received characters
1	Process Commands, discard all but the last C5.0 character
2	Process Commands, discard all C5.0 characters
3	Process Commands, discard all C5.0 characters, discard serial addresses

Policy 0 is the simplest and also applies for all conditions where the Receive FIFO is bypassed. In this mode, every character that is received is placed into the Receive FIFO (when enabled) or into the Receive Output Register.

In discard policy 1, all Start Of Cell, extended command, and serial address commands are processed as they are received. The C5.0 character, which is automatically transmitted when no data is present in the Transmit FIFO, is treated differently here. In this mode, whenever two or more adjacent C5.0 characters are received, all of them are discarded except the last one received before any other character type. This allows these fill characters to be removed from the data stream, but does not change the data flow for protocols (like Fibre Channel) that use a single C5.0 character as a delimiter.

Policy 2 is identical to policy 1 except that all C5.0 characters are removed from the data stream.

Policy 3 is a super-set of policy 2, where the serial address is also discarded.

When the FIFOs are bypassed (FIFOBYP* LOW), no characters are actually discarded, but the receiver discard policy can be used to control external filtering of the data. The RXEMPTY* FIFO flag is used to indicate if the character on the output bus is valid or not. In discard policy 0, the RXEMPTY* flag is always deasserted to indicate that valid data is always present. In discard policy 1, the RXEMPTY* flag indicates an empty condition for all but the last C5.0 character before any other character is presented. In discard policies 2 or 3, the RXEMPTY* flag indicates an empty condition for all C5.0 characters. When any other character is present, this flag indicates that valid or “interesting” Data or Special Characters are present.

Receive FIFO

The Receive FIFO is used to buffer data captured from the selected serial stream for later processing by the host system. This FIFO is sized to hold 256, 14-bit characters. When the FIFO is enabled, it is written to by the Receive Control State Machine. When data is present in the Receive FIFO (as indicated by the RXFULL*, RXHALF*, and RXEMPTY* Receive FIFO status flags), it can be read from the Output Register by asserting AM* and RXEN*.

The read port on the Receive FIFO may be configured for the same two timing models as the transmit interface: UTOPIA and Cascade. Both are forms of a FIFO interface. The UTOPIA timing model has active LOW RXEMPTY* and RXFULL* status flags, and an active LOW RXEN* enable. When configured for Cascade operation, these same signals are all active HIGH. Either timing model supports connection to various host bus interfaces, state machines, or external FIFOs for depth expansion (see *Figure 5*).

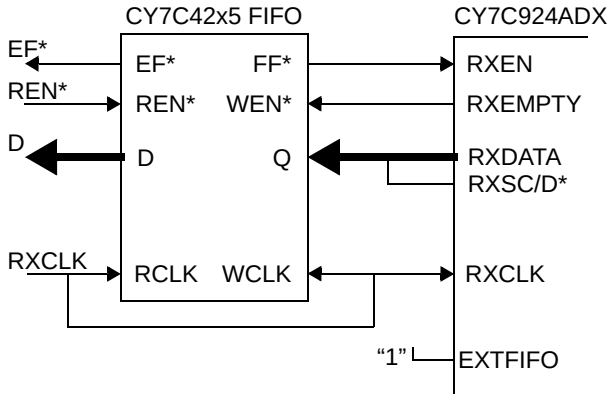


Figure 5. External FIFO Depth Expansion of the CY7C924ADX Receive Data Path.

The Receive FIFO presents Full, Half-Full, and Empty FIFO status flags. These flags are provided synchronous to RXCLK to allow operation with a Moore-type external controlling state machine. When configured with the Receive FIFO enabled, RXCLK is an input. When the Receive FIFO is bypassed (FIFOBYP* is LOW), RXCLK is an output operating at the received character rate.

Receive Input Register

The input register is clocked by the rising edge of RXCLK. It samples numerous signals that control the reading of the Receive FIFO and operation of the Receive Control State Machine.

Receive Output Register

The Receive Output Register changes in response to the rising edge of RXCLK. The Receive FIFO status flag outputs of this register are placed in a High-Z state when the CY7C924ADX is not addressed (AM* is sampled HIGH). The

RXDATA bus output drivers are enabled when the device is selected by RXEN* being asserted in the RXCLK cycle immediately following that in which the device was addressed (AM* is sampled LOW), and RXEN* being sampled by RXCLK. This initiates a Receive FIFO read cycle.

Just as with the TXDATA bus on the Transmit Input Register, the receive outputs are also mapped by the specific decoding, parity, and bus-width selected by the ENCBYP*, BYTE8/10* and FIFOBYP* inputs. These assignments are shown in *Table 7*

When the Decoder and Receive FIFO are both enabled, the Receive Control State Machine interprets and discards (except in discard policy 0) received C0.0 and C3.0 command codes as set and clear directives for the RXINT output. This allows the RXINT output to duplicate the state transitions presented to the TXINT input at the source end of the link. This RXINT output can be used, along with TXHALT*, TXINT, and RXHALF*, to implement a back-pressure mechanism for the Receive FIFO, or for other time dependent signalling.

.

Table 7. Receive Output Bus Signal Map

	Receive Decoder Mode[1]			
	Decoded 10-bit Character Stream (8-bit characters)	**Undecoded 10-bit Character Stream**	**Decoded 10-bit Byte-Packed Character Stream (10-bit characters)**	**Undecoded 12-bit Character Stream**
ENCBYP*	HIGH	LOW	HIGH	LOW
BYTE8/10*	HIGH	HIGH	LOW	LOW
RXDATA Bus I/O Bit				
RXSC/D*	RXSC/D*		RXSC/D*	
RXDATA[0]	RXD[0]	RXD[0][5]	RXD[0]	RXD[0][5]
RXDATA[1]	RXD[1]	RXD[1]	RXD[1]	RXD[1]
RXDATA[2]	RXD[2]	RXD[2]	RXD[2]	RXD[2]
RXDATA[3]	RXD[3]	RXD[3]	RXD[3]	RXD[3]
RXDATA[4]	RXD[4]	RXD[4]	RXD[4]	RXD[4]
RXDATA[5]	RXD[5]	RXD[5]	RXD[5]	RXD[5]
RXDATA[6]	RXD[6]	RXD[6]	RXD[6]	RXD[6]
RXDATA[7]	RXD[7]	RXD[7]	RXD[7]	RXD[7]
RXINT/RXOP/RXDATA[8] (FIFOBYP*=HIGH)	RXINT	RXD[8]	RXD[8]	RXD[8]
RXINT/RXOP/RXDATA[8] (FIFOBYP*=LOW)	RXOP	RXD[8]	RXD[8]	RXD[8]
RXDATA[9]	(LOW)	RXD[9]	RXD[9]	RXD[9]
RXRVS/RXDATA[10]	RXRVS		RXRVS	RXD[10]
RXSOC/RXDATA[11]	RXSOC		RXSOC	RXD[11]

Note:
5. First bit shifted in. Others follow in numerical order interpreted from an NRZ pattern.

If the Receive FIFO and Decoder are bypassed, all received characters are passed directly to the Receive Output Register. If framing is enabled, and K28.5 characters have been detected meeting the present framing requirements, the output characters will appear on proper character boundaries. If framing is disabled (RFEN is LOW) or K28.5 characters have not been detected in the data stream, the received characters may not be output on their proper 10-bit boundaries. In this mode, some form of external framing and decoding/descrambling must be used to recover the original source data.

Serial Address Register

The receiver is capable of selectively accepting or discarding received data based on an address received in the data stream. The address matching capability allows for the choice of matching of either domains (multicast) or exact addresses (unicast). The 8- or 10-bit Serial Address Register represents a single character address field as shown in *Figure 6*.

The multicast mode is bit-specific and allows allocation of up to 8 or 10 separate domains. In the unicast address mode the match is character specific and identifies up to 256 or 1024 destination addresses. A device can either belong to one or more domains, or it can have a single unique address.

When a serial address is received and a match is detected, the address, and all data following that address, is passed to the Receive FIFO (except in discard policy 3 where the address is discarded). This continues until a serial address is received that does not match the contents of the Address Register, whereupon writes to the Receive FIFO are inhibited.

The Serial Address Register has a power-up default state where the multicast field set to an all ones condition (FFh or 3FFh). When set to this value the receiver accepts all data, regardless of the of the presence or content of any received serial address. This “promiscuous” address can also be forced by the momentary assertion of the RESET*[1:0] pair.

Address Register Content

RXDATA[9] or [7] RXDATA[0]

MSB	Serial Address Register	LSB

RXEN*	RXRST*	RXRVS	RXSC/D*	
0	0	0	0	Multicast Address write
0	0	0	1	Unicast Address write
0	0	1	0	Multicast Address read
0	0	1	1	Unicast Address read

Figure 6. Serial Address Register Format and Access.

The Serial Address Register is only used when the receiver is operated with the Receive FIFO enabled (FIFOBYP* is HIGH) and in operating modes where the discard policy is not 0 (see *Table 6* for a list of discard policies).

Serial Address Register Access

The Serial Address Register is accessed through the RXDATA bus. Both reads and writes to the register require the device to be addressed (AM* is LOW) and for RXRST* to be asserted (LOW).

When accessed for write or read operations, the RXRVS signal is used as a read/write selector, and RXSC/D* is used to select the operating mode (multicast or unicast) of the Serial Address Register.

CY7C924ADX DC Electrical Characteristics Over the Operating Range

Parameter	Description	Test Conditions	Min.	Max.	Unit
TTL Outputs					
V_{OHT}	Output HIGH Voltage	I_{OH} = –2 mA, V_{DD} = Min.	2.4		V
V_{OLT}	Output LOW Voltage	I_{OL} = 8 mA, V_{DD} = Min.		0.4	V
I_{OST}	Output Short Circuit Current	V_{OUT} = 0V[6]	–30	–80	mA
I_{OZL}	High-Z Output Leakage Current		–20	20	µA
TTL Inputs					
V_{IHT}	Input HIGH Voltage		2.0	V_{CC}	V
V_{ILT}	Input LOW Voltage		–0.5	0.8	V
I_{IHT}	Input HIGH Current	$V_{IN} = V_{CC}$		+40	µA
I_{ILT}	Input LOW Current	V_{IN} = 0.0V		–40	µA
I_{IHPD}	Input HIGH Current	$V_{IN} = V_{CC}$, Pins with internal pull-down		+300	µA
I_{ILPU}	Input LOW Current	V_{IN} = 0.0V, Pins with internal pull-up	–300		µA
Transmitter PECL-Compatible Output Pins: OUTA+, OUTA–, OUTB+, OUTB–[7]					
V_{OHE}	Output HIGH Voltage (V_{DD} referenced)	Load = 50Ω to V_{CC} – 1.33V R_{CURSET}= 10k	V_{DD} – 1.03	V_{DD} – 0.83	V
V_{OLE}	Output LOW Voltage (V_{DD} referenced)	Load = 50Ω to V_{CC} – 1.33V R_{CURSET}= 10k	V_{DD} – 2	V_{DD} – 1.62	V
V_{ODIF}	Output Differential Voltage |(OUT+) – (OUT–)|	Load = 50Ω to V_{CC} – 1.33V R_{CURSET}= 10k	600	1100	mV
Receiver Single-ended PECL-Compatible Input Pin: CARDET					
V_{IHE}	Input HIGH Voltage (V_{DD} referenced)		V_{DD} – 1.165	V_{DD}	V
V_{ILE}	Input LOW Voltage (V_{DD} referenced)		2.5	V_{DD} – 1.475	V
I_{IHE}	Input HIGH Current	$V_{IN} = V_{IHE}$(min.)		+40	µA
I_{ILE}	Input LOW Current	$V_{IN} = V_{ILE}$(max.)	–40		µA
Differential Line Receiver Input Pins: INA+, INA–, INB+, INB–					
V_{DIFF}	Input Differential Voltage |(IN+) – (IN–)|		200	2500	mV
V_{IHH}	Highest Input HIGH Voltage			V_{DD}	V
V_{ILL}	Lowest Input LOW Voltage		2.5		V
I_{IHH}	Input HIGH Current	$V_{IN} = V_{IHH}$ Max.		750	µA
I_{ILL}[8]	Input LOW Current	$V_{IN} = V_{ILL}$ Min.	–200		µA
Miscellaneous			**Typ.**	**Max.**	
I_{DD}[9]	Power Supply Current	Freq. = Max.	170	250	mA

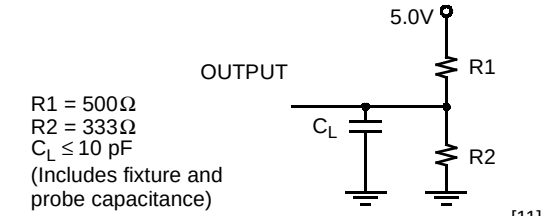
Capacitance[10]

Parameter	Description	Test Conditions	Max.	Unit
C_{INTTL}	TTL Input Capacitance	T_A = 25°C, f_0 = 1 MHz, V_{DD} = 5.0V	7	pF
C_{INPECL}	PECL input Capacitance	T_A = 25°C, f_0 = 1 MHz, V_{DD} = 5.0V	4	pF

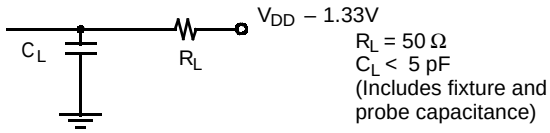
Notes:

6. Tested one output at a time, output shorted for less than one second, less than 10% duty cycle.
7. The output current and (resulting voltage swing) is set using a single resistor between CURSETx and V_{SS}. This CURSET resistor value is calculated as $R_{CURSET} = (100 * Z_O)/V_{ODIF}$, where Z_O is the differential load between the true and compliment outputs of the differential driver.
8. To guarantee positive currents for all PECL voltages, an external pull-down resistor must be present.
9. Maximum I_{DD} is measured with V_{DD} = MAX, RFEN = LOW, and outputs unloaded. Typical I_{DD} is measured with V_{DD} = 5.0V, T_A = 25°C, RFEN = LOW, and outputs unloaded.
10. Tested initially and after any design or process changes that may affect these parameters, but not 100% tested.

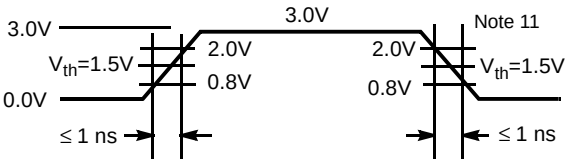
AC Test Loads and Waveforms



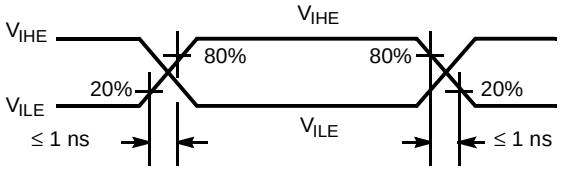
(a) TTL AC Test Load[11]



(b) PECL AC Test Load [11]



(c) TTL Input Test Waveform



(d) PECL Input Test Waveform

CY7C924ADX Transmitter TTL Switching Characteristics, FIFO Enabled Over the Operating Range

Parameter	Description	Min.	Max.	Unit
f_{TS}	TXCLK Clock Cycle Frequency With Transmit FIFO Enabled		50	MHz
t_{TXCLK}	TXCLK Period	20		ns
t_{TXCPWH}	TXCLK HIGH Time	6.5		ns
t_{TXCPWL}	TXCLK LOW Time	6.5		ns
t_{TXCLKR}[10]	TXCLK Rise Time[12]	0.7	5	ns
t_{TXCLKF}[10]	TXCLK Fall Time[12]	0.7	5	ns
t_{TXA}	Flag Access Time From TXCLK↑ to Output	2	15	ns
t_{TXDS}	Transmit Data Set-Up Time to TXCLK↑	4		ns
t_{TXDH}	Transmit Data Hold Time from TXCLK↑	1		ns
t_{TXENS}	Transmit Enable Set-Up Time to TXCLK↑	4		ns
t_{TXENH}	Transmit Enable Hold Time from TXCLK↑	1		ns
t_{TXRSS}	Transmit FIFO Reset (TXRST*) Set-Up Time to TXCLK↑	4		ns
t_{TXRSH}	Transmit FIFO Reset (TXRST*) Hold Time from TXCLK↑	1		ns
t_{TXAMS}	Transmit Address Match (AM*) Set-Up Time to TXCLK↑	4		ns
t_{TXAMH}	Transmit Address Match (AM*) Hold Time from TXCLK↑	1		ns
t_{TXZA}	Sample of AM* LOW by TXCLK↑, Output High-Z to Active HIGH or LOW	0		ns
t_{TXOE}	Sample of AM* LOW by TXCLK↑ to Output Valid	1.5	20	ns
t_{TXAZ}	Sample of AM* HIGH by TXCLK↑ to Output in High-Z	1.5	20	ns

Notes:
11. Cypress uses constant current (ATE) load configurations and forcing functions. This figure is for reference only.
12. Input/output rise and fall time is measured between 0.8V and 2.0V.

CY7C924ADX Receiver TTL Switching Characteristics, FIFO Enabled Over the Operating Range

Parameter	Description	Min.	Max.	Unit
f_{RIS}	RXCLK Clock Cycle Frequency With Receive FIFO Enabled		50	MHz
$t_{RXCLKIP}$	RXCLK Input Period	20		ns
t_{RXCPWH}	RXCLK Input HIGH Time	6.5		ns
t_{RXCPWL}	RXCLK Input LOW Time	6.5		ns
$t_{RXCLKIR}$[10]	RXCLK Input Rise Time[12]	0.7	5	ns
$t_{RXCLKIF}$[10]	RXCLK Input Fall Time[12]	0.7	5	ns
t_{RXENS}	Receive Enable Set-Up Time to RXCLK↑	4		ns
t_{RXENH}	Receive Enable Hold Time from RXCLK↑	1		ns
t_{RXRSS}	Receive FIFO Reset (RXRXT*) Set-Up Time to RXCLK↑	4		ns
t_{RXRSH}	Receive FIFO Reset (RXRXT*) Hold Time from RXCLK↑	1		ns
t_{RXAMS}	Receive Address Match (AM*) Set-Up Time to RXCLK↑	4		ns
t_{RXAMH}	Receive Address Match (AM*) Hold Time from RXCLK↑	1		ns
t_{RXA}[13]	Flag and Data Access Time from RXCLK↑ to Output	1.5	15	ns
t_{RXZA}[13]	Sample of AM* LOW by RXCLK↑, Output High-Z to Active HIGH or LOW, or Sample of RXEN* Asserted by RXCLK↑, Output High-Z to Active HIGH or LOW	0		ns
t_{RXOE}[13]	Sample of AM* LOW by RXCLK↑ to Output Valid,[13] or Sample of RXEN* Asserted by RXCLK↑ to RXDATA Outputs Valid	1.5	20	ns
t_{RXAZ}[13]	Sample of AM* HIGH by RXCLK↑ to Output in High-Z,[13] or Sample of RXEN* Deasserted by RXCLK↑ to RXDATA Outputs in High-Z	1.5	20	ns

CY7C924ADX Transmitter TTL Switching Characteristics, FIFO Bypassed Over the Operating Range

Parameter	Description	Min.	Max.	Unit
t_{TRA}	Flag Access Time From REFCLK↑ to Output	2	15	ns
t_{REFDS}	Write Data Set-Up Time to REFCLK↑	4		ns
t_{REFDH}	Write Data Hold Time from REFCLK↑	2		ns
t_{REFENS}	Transmit Enable Set-Up Time to REFCLK↑	4		ns
t_{REFENH}	Transmit Enable Hold Time from REFCLK↑	2		ns
t_{REFAMS}	Transmit Address Match (AM*) Set-Up Time to REFCLK↑	4		ns
t_{REFAMH}	Transmit Address Match (AM*) Hold Time from REFCLK↑	2		ns
t_{REFZA}	Sample of AM* LOW by REFCLK↑, Output High-Z to Active HIGH or LOW	0		ns
t_{REFOE}	Sample of AM* LOW by REFCLK↑ to Flag Output Valid	1.5	20	ns
t_{REFAZ}	Sample of AM* HIGH by REFCLK↑ to Flag Output High-Z	1.5	20	ns

Note:
13. Parallel data output specifications are only valid if all outputs are loaded with similar DC and AC loads.

CY7C924ADX Receiver TTL Switching Characteristics, FIFO Bypassed Over the Operating Range

Parameter	Description	Min.	Max.	Unit
f_{ROS}[14]	RXCLK Clock Output Frequency—100 to 200 MBaud (RANGESEL is HIGH, ENCBYP* is HIGH or BYTE8/10* is HIGH)	10	20	MHz
	RXCLK Clock Output Frequency—50 to 100 MBaud (RANGESEL is LOW, ENCBYP* is HIGH or BYTE8/10* is HIGH)	5	10	MHz
	RXCLK Clock Output Frequency—100 to 200 MBaud 12-bit Encoder Bypass Operation (RANGESEL is HIGH, ENCBYP* is LOW and BYTE8/10* is LOW)	8.33	16.67	MHz
	RXCLK Clock Output Frequency—50 to 100 MBaud 10-bit Operation (RANGESEL is LOW, ENCBYP* is LOW and BYTE8/10* is LOW)	4.16	8.33	MHz
$t_{RXCLKOP}$	RXCLK Output Period	25	250	ns
$t_{RXCLKOD}$	RXCLK Output Duty Cycle	40	60	%
$t_{RXCLKOR}$[10]	RXCLK Output Rise Time[12]	0.25	2	ns
$t_{RXCLKOF}$[10]	RXCLK Output Fall Time[12]	0.25	2	ns
t_{RXENS}	Receive Enable Set-Up Time to RXCLK↑	4		ns
t_{RXENH}	Receive Enable Hold Time from RXCLK↑	1		ns
t_{RXZA}[13]	Sample of AM* LOW by RXCLK↑, Outputs High-Z to Active Sample of RXEN* Asserted by RXCLK↑ to RXDATA Outputs High-Z to Active	0		ns
t_{RXOE}[13]	Sample of AM* LOW by RXCLK↑ to Flag Output Valid Sample of RXEN* Asserted by RXCLK↑ to RXDATA Output Low-Z	1.5	20	ns
t_{RXAZ}[13]	Sample of AM* HIGH by RXCLK↑ to Flag Output High-Z Sample of RXEN* Deasserted by RXCLK↑ to RXDATA Output High-Z	1.5	20	ns

CY7C924ADX Receiver Switching Characteristics Over the Operating Range

Parameter	Description	Min.	Max.	Unit
t_B[17]	Bit Time	5.0	20.0	ns
t_{IN_J}	IN± Peak-to-Peak Input Jitter Tolerance[15, 16]		0.5	UI
t_{SA}	Static Alignment[10, 18]		600	ps
t_{EFW}	Error Free Window [10, 15, 19]	0.65		UI

Notes:

14. The period of f_{ROS} will match the period of the transmitter PLL reference (REFCLK) when receiving serial data. When data is interrupted, RXCLK may drift to REFCLK ±2500 ppm.
15. Receiver UI (Unit Interval) is calculated as $1/(f_{REF}* 10)$ when operated in 8-bit mode if no data is being received, or $1/(f_{REF}*10)$ of the remote transmitter if data is being received. In an operating link this is equivalent to 10 * t_B. when REFCLK = 1x the character rate. These equations change to $1/(f_{REF}* 12)$ when operated in 10-bit mode with the Encoder bypassed. At alternate multiply ratios (2x or 4x, as selected by SPDSEL and RANGESEL), the numerator is multiplied by 2 or 4 respectively.
16. The specification is sum of 25% Duty Cycle Distortion (DCD), 10% Data Dependant Jitter (DDJ), 15% Random Jitter (RJ).
17. The PECL switching threshold is the midpoint between the V_{OHE}, and V_{OLE} specifications (approximately V_{DD} – 1.33V).
18. Static alignment is a measure of the alignment of the Receiver sampling point to the center of a bit. Static alignment is measured by the absolute difference of the left and right edge shifts ($|t_{SH\_L} - t_{SH\_R}|$) of one bit until a character error occurs.
19. Error Free Window is a measure of the time window between bit centers where a transition may occur without causing a bit sampling error. EFW is measured over the operating range, input jitter < 50% Dj.

CY7C924ADX Transmitter Switching Characteristics Over the Operating Range

Parameter	Description	Min.	Max.	Unit
t_B[17]	Bit Time	5.0	20.0	ns
t_{RISE}	PECL Output Rise Time 20–80% (PECL Test Load)[10]	200	1700	ps
t_{FALL}	PECL Output Fall Time 80–20% (PECL Test Load)[10]	200	1700	ps
t_{DJ}	Deterministic Jitter (peak-peak)[10, 20]		0.02	UI
t_{RJ}	Random Jitter (σ)[10, 21]		0.008	UI
t_{JT}	Transmitter Total Output Jitter (pk-pk)[10]		0.08	UI

CY7C924ADX REFCLK Input Switching Characteristics Over the Operating Range

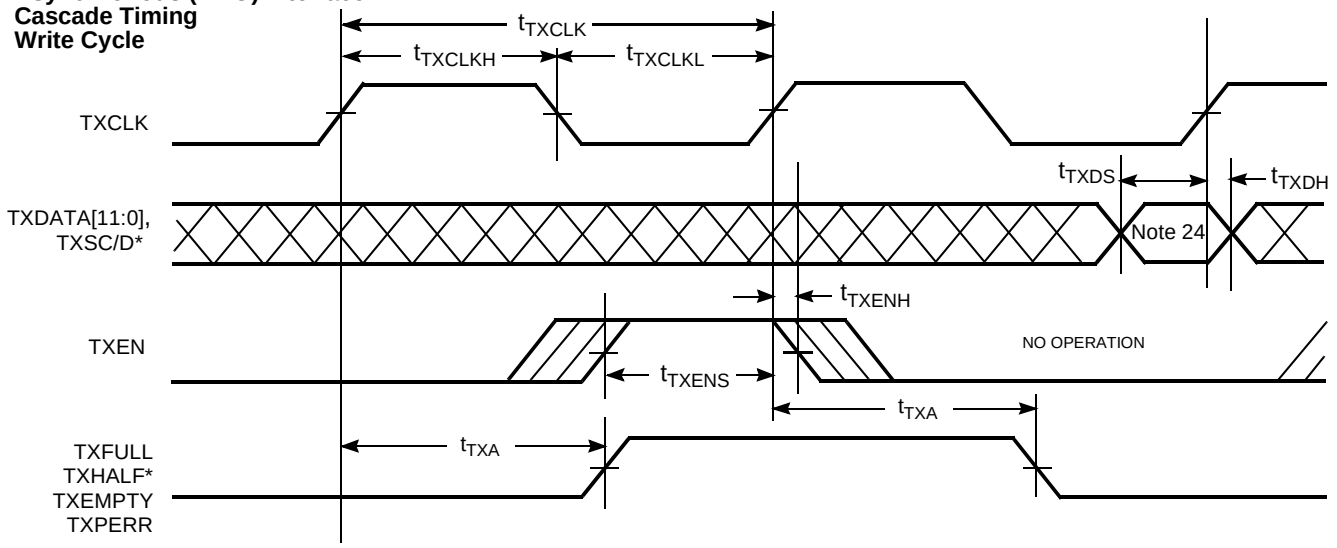
Parameter	Description	Conditions			Min.	Max	Unit
		SPDSEL	RANGESEL	BYTE8/10*			
f_{REF}	REFCLK Clock Frequency—50 to 100 MBaud, 10-bit mode, encoder bypass, REFCLK = 2x character rate	0	0	0	8.33	16.67	MHz
	REFCLK Clock Frequency—50 to 100 MBaud, 8-bit mode, REFCLK = 2x character rate	0	0	1	10	20	MHz
	REFCLK Clock Frequency—50 to 100 MBaud, 10-bit mode, encoder bypass, REFCLK = 4x character rate	0	1[22]	0	16.67	33.33	MHz
	REFCLK Clock Frequency—50 to 100 MBaud, 8-bit mode, REFCLK = 4x character rate	0	1[22]	1	20	40	MHz
	REFCLK Clock Frequency—100 to 200 MBaud, 10-bit mode, encoder bypass, REFCLK = character rate	1	0	0	8.33	16.67	MHz
	REFCLK Clock Frequency—100 to 200 MBaud, 8-bit mode, REFCLK = character rate	1	0	1	10	20	MHz
	REFCLK Clock Frequency—100 to 200 MBaud, 10-bit mode, encoder bypass, REFCLK = 2x character rate	1	1	0	16.67	33.3	MHz
	REFCLK Clock Frequency—100 to 200 MBaud, 8-bit mode, REFCLK = 2x character rate	1	1	1	20	40	MHz
t_{REFCLK}	REFCLK Period				25	120	ns
t_{REFH}	REFCLK HIGH Time				6.5		ns
t_{REFL}	REFCLK LOW Time				6.5		ns
t_{REFRX}	REFCLK Frequency Referenced to Received Clock Period[23]				–0.04	+0.04	%

Notes:

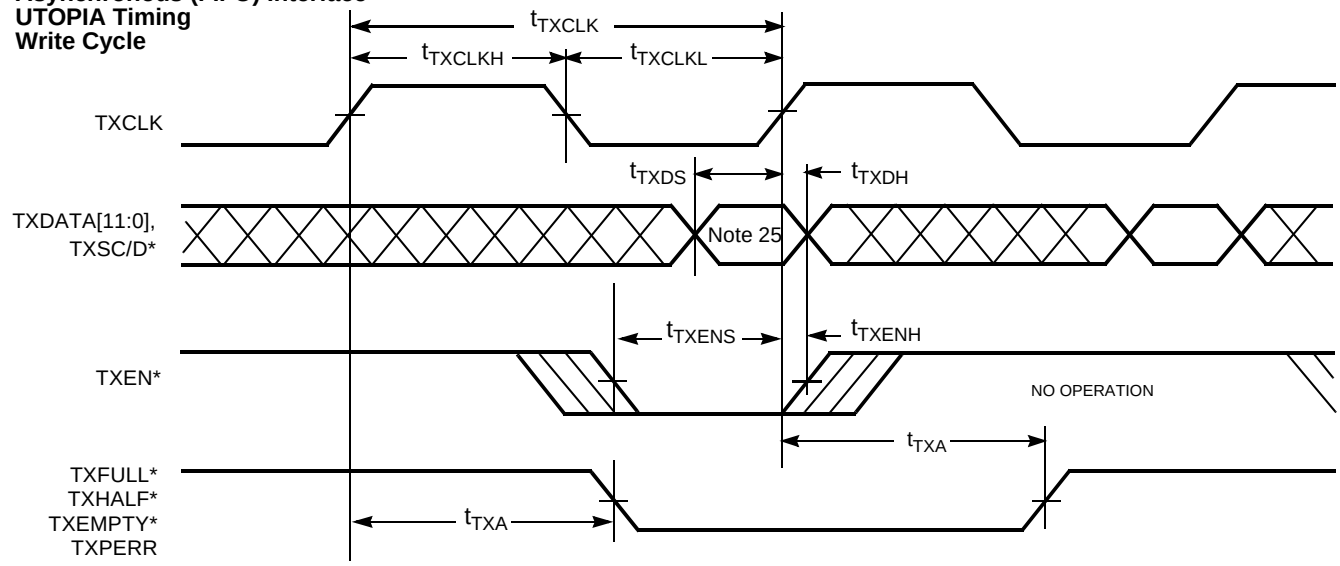
20. While sending continuous K28.5s, outputs loaded to 50Ω to V_{DD}–1.33V, over the operating range.
21. While sending continuous K28.7s, after 100,000 samples measured at the cross point of differential outputs, time referenced to REFCLK input, over the operating range.
22. When configured for synchronous operation with the FIFOs bypassed (FIFOBYP* is LOW), if RANGESEL is HIGH the SPDSEL input is ignored and operation is forced to the 100–200 MBaud range.
23. REFCLK has no phase or frequency relationship with RXCLK and only acts as a centering reference to reduce clock synchronization time. REFCLK must be within ±0.04% of the transmitter PLL reference (REFCLK) frequency, necessitating a ±200-PPM crystal.

CY7C924ADX HOTLink Transmitter Switching Waveforms

Asynchronous (FIFO) Interface
Cascade Timing
Write Cycle



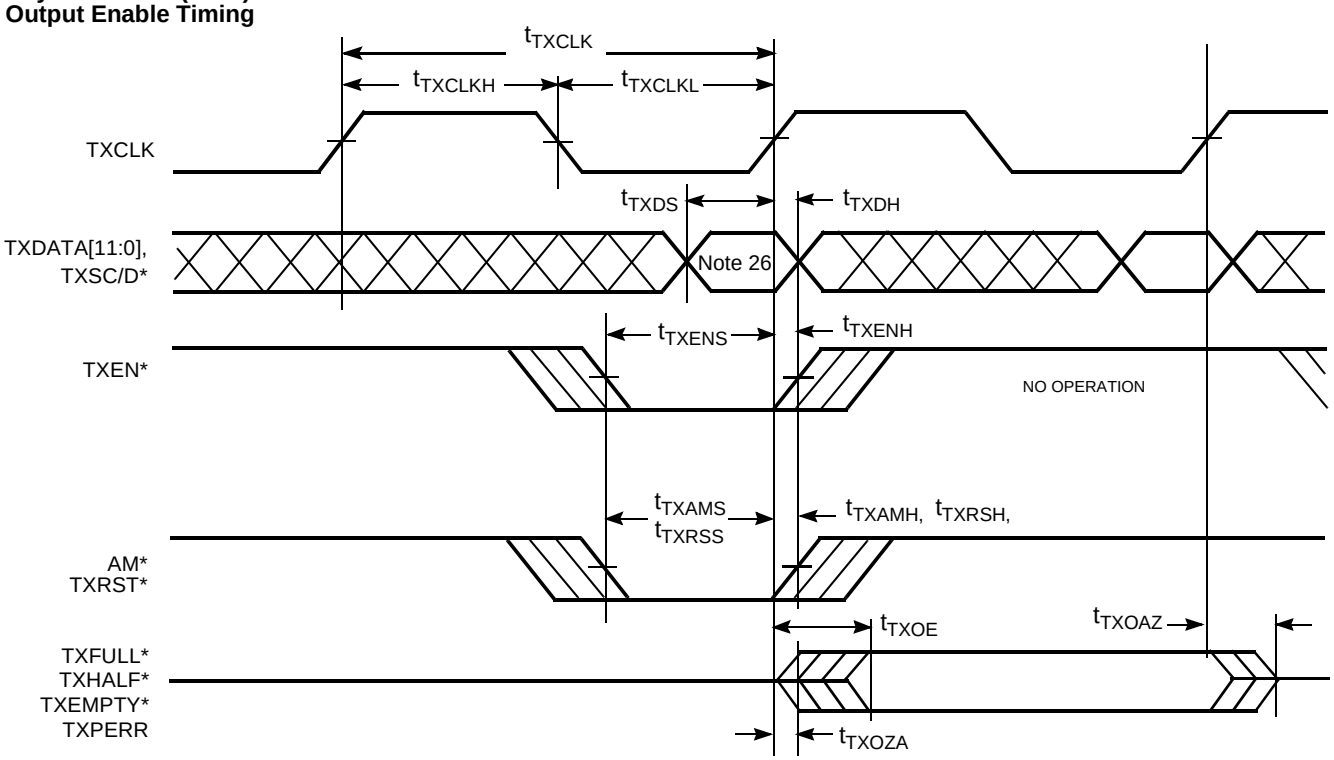
Asynchronous (FIFO) Interface
UTOPIA Timing
Write Cycle



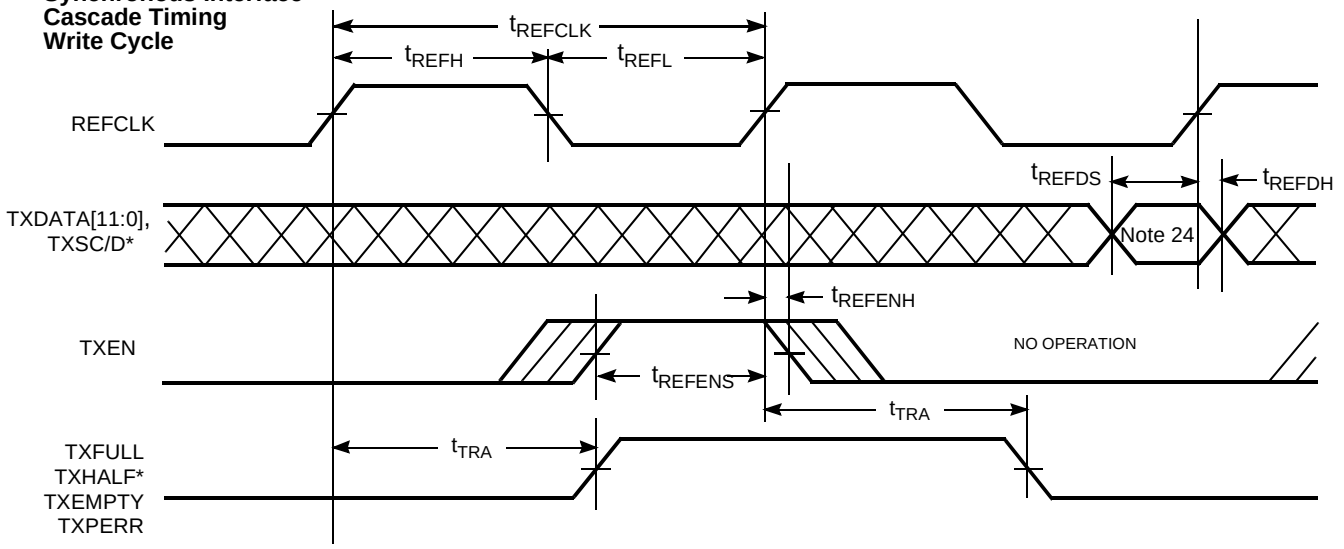
Notes:
24. When transferring data to the Transmit FIFO from a depth expanded external FIFO, the data is captured from the external FIFO one clock cycle following the actual enable.
25. When writing data from a UTOPIA compliant interface, the write data is captured on the same clock cycle as the data.

CY7C924ADX HOTLink Transmitter Switching Waveforms (continued)

Asynchronous (FIFO) Interface
Output Enable Timing



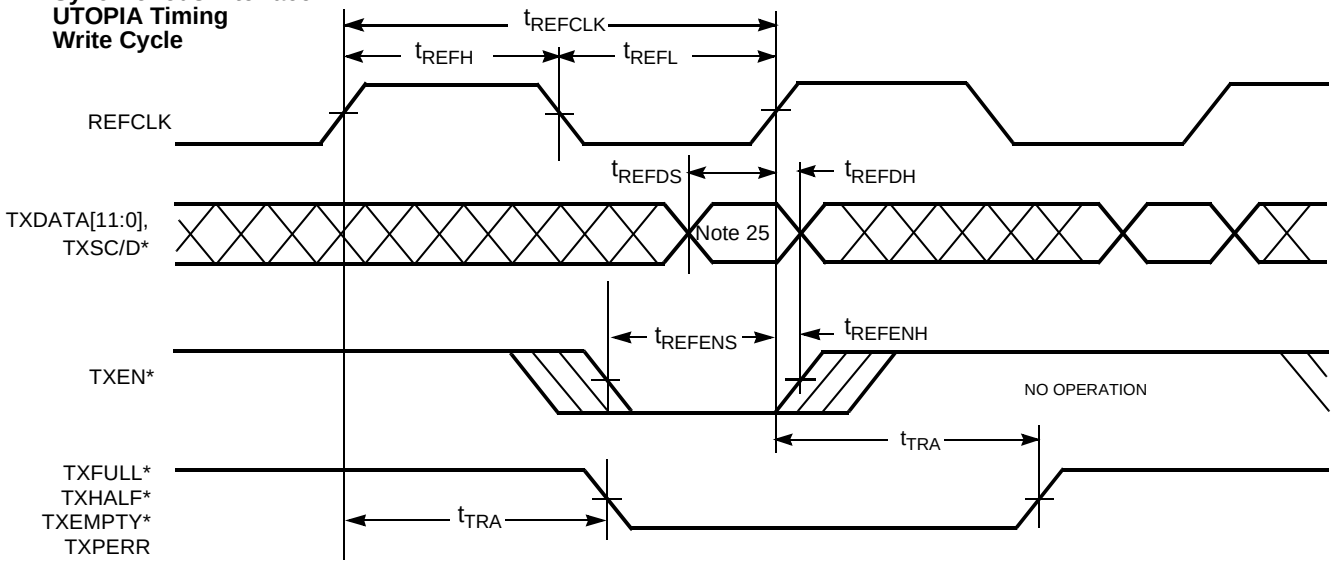
Synchronous Interface
Cascade Timing
Write Cycle



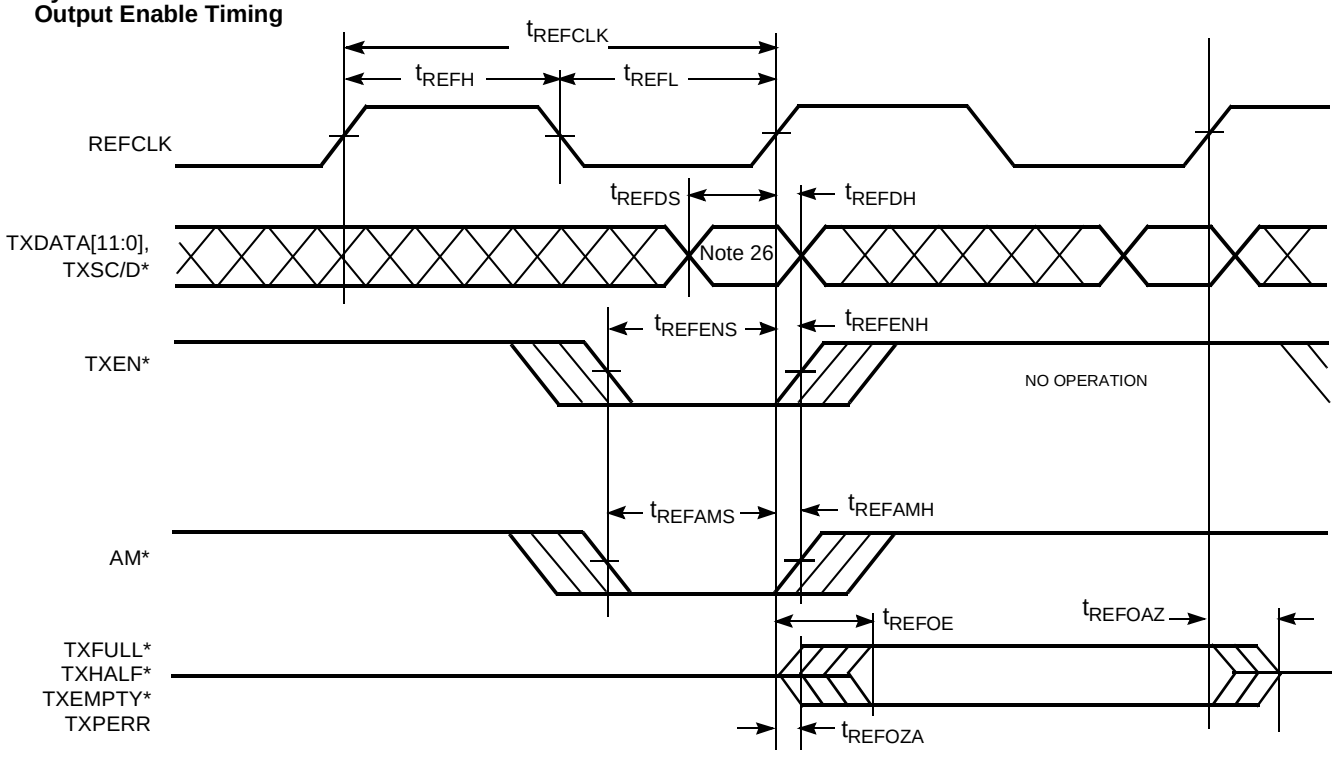
Note:
26. Transmit FIFO Writes are permitted while the status flag outputs are High-Z, however operation in this mode is not encouraged since this may mask a FIFO full condition, causing data to be lost.

CY7C924ADX HOTLink Transmitter Switching Waveforms (continued)

Synchronous Interface
UTOPIA Timing
Write Cycle

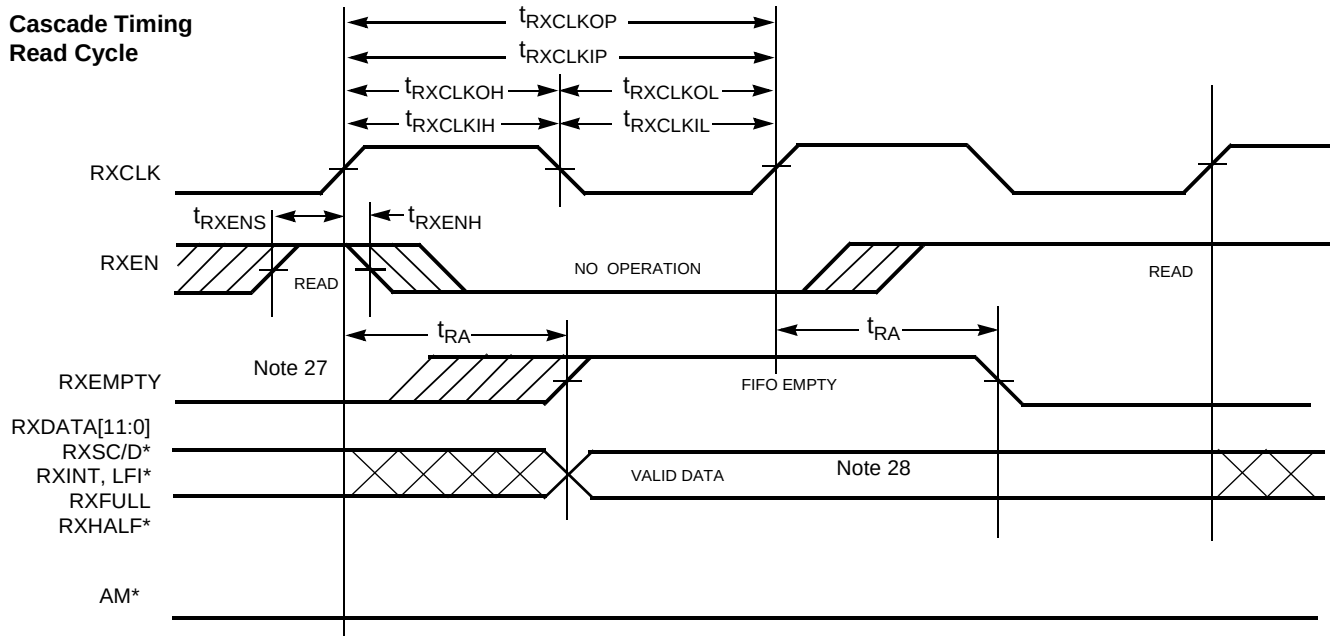


Synchronous Interface
Output Enable Timing

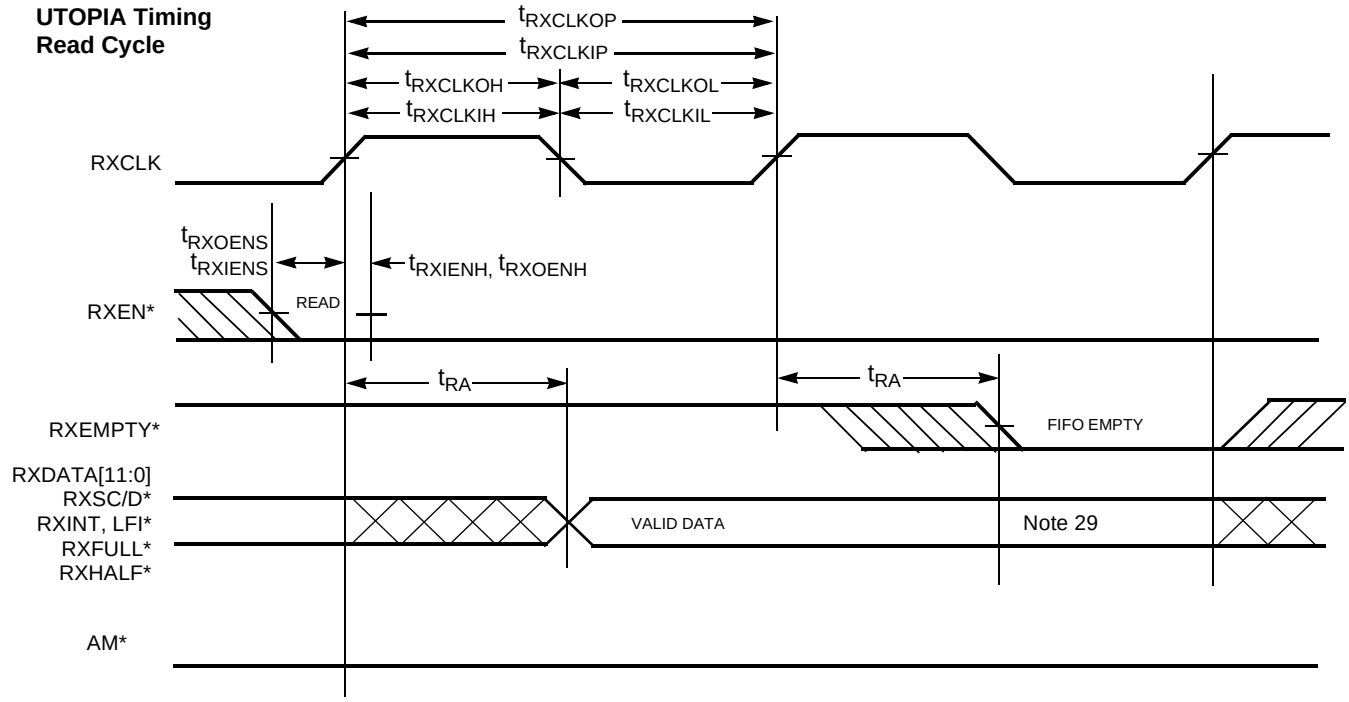


CY7C924ADX HOTLink Receiver Switching Waveforms

Cascade Timing
Read Cycle



UTOPIA Timing
Read Cycle

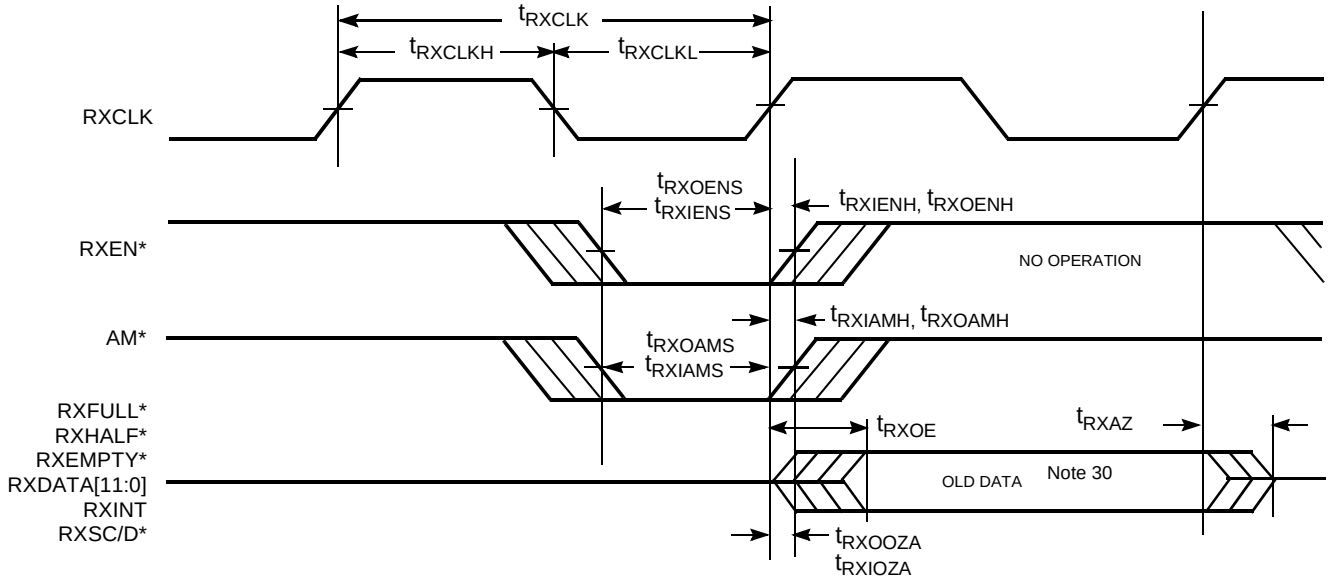


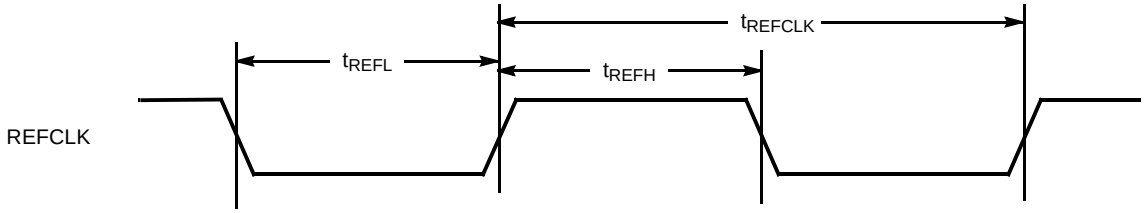
Notes:

27. When transferring data from the Receive FIFO to a depth expanded external FIFO, the data is sent to the external FIFO on the same clock cycle an RXEMPTY indicates the data is available.
28. On inhibited reads, or if the Receive FIFO goes empty, the data outputs do not change.
29. When reading data from a UTOPIA compliant interface, the data is captured on the same clock cycle as the FIFO flag indicates data available, and not when the FIFO indicates empty.

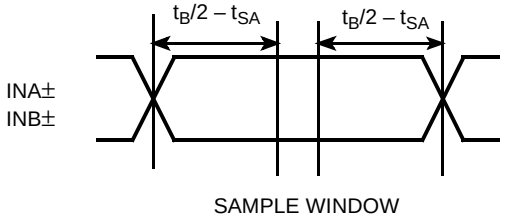
CY7C924ADX HOTLink Receiver Switching Waveforms (continued)

Output Enable Timing

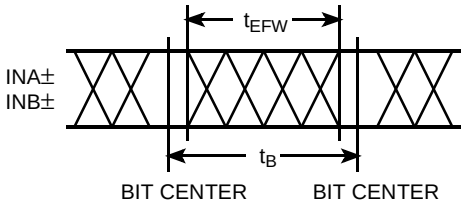




Static Alignment



Error-Free Window



Note:
30. Receive FIFO Reads are inhibited while the outputs are High-Z or RXBISTEN* is active.

CY7C924ADX HOTLink Transceiver Operation

The interconnection of two or more CY7C924ADX Transceivers form a general-purpose communications subsystem capable of transporting user data at up to 20 MBytes per second over several types of serial interface media. The CY7C924ADX is highly configurable with multiple modes of operation.

In the transmit section of the CY7C924ADX, data moves from the input register, through the Transmit FIFO, to the 8B/10B Encoder. The encoded data is then shifted serially out the OUTx± differential PECL compatible drivers. The bit-rate clock is generated internally from a 2.5x, 5x, or 10x PLL clock multiplier. A more complete description is found in the section *CY7C924ADX HOTLink Transmit-Path Operating Mode Description*.

In the receive section of the CY7C924ADX, serial data is sampled by the receiver on one of the INx± differential line receiver inputs. The receiver clock and data recovery PLL locks onto the selected serial bit stream and generates an internal bit-rate sample clock. The bit stream is deserialized, decoded, and presented to the Receive FIFO, along with a character clock. The data in the FIFO can then be read either slower or faster than the incoming character rate. A more complete description is found in the section *CY7C924ADX HOTLink Receive-Path Operating Mode Description*.

The Transmitter and Receiver parallel interface timing and functionality can be configured to Cascade directly to external FIFOs for depth expansion, to emulate a UTOPIA interface, couple directly to registers, or couple directly to state machines. These interfaces can accept or output either:

- 8-bit characters
- 10-bit characters (for byte-packed encoded transport)
- 10-bit pre-encoded characters (pre-scrambled or pre-encoded)
- 12-bit pre-encoded characters (pre-scrambled or pre-encoded)

The bit numbering and content of the parallel transmit interface is shown in *Table 1*. When operated with the 8B/10B Encoder bypassed, the TXSC/D* and RXSC/D* bits are ignored.

The HOTLink Transceiver serial interface provides a seamless interface to various types of media. A minimal number of external passive components are required to properly terminate transmission lines and provide LVPECL loads. For power supply decoupling, a single capacitor (in the range of 0.02 µF to 0.1 µF) is required per power/ground pair. Additional information on interfacing these components to various media can be found in the *HOTLink Design Considerations* application note.

CY7C924ADX HOTLink Transmit-Path Operating Mode Descriptions

The HOTLink Transmitter can be configured into several operating modes, each providing different capabilities and fitting different transmission needs. These modes are selected using the FIFOBYP*, ENCBYP* and BYTE8/10* inputs on the CY7C924ADX Transceiver. These modes can be reduced to five primary classes:

- Synchronous Encoded
- Synchronous Pre-encoded
- Asynchronous Encoded
- Asynchronous Byte-Packed
- Asynchronous Pre-encoded

Synchronous Encoded

In this mode, the Transmit FIFO is bypassed, while the 8B/10B encoder is enabled. One character is accepted at the Transmit Input Register at the rising edge of REFCLK, and passed to the Encoder where it is encoded for serial transmission. The Serializer operates synchronous to REFCLK, which is multiplied by 10 or 5 to generate the serial data bit-clock. In this mode the TXSOC, TXRST*, TXINT, TXHALT*, and TXSTOP* inputs (when they are not used for data bits) are not interpreted and may be tied either HIGH or LOW. To place the CY7C924ADX into synchronous modes, FIFOBYP* must be LOW.

This mode is usually used for products that must meet specific predefined protocol requirements, and cannot tolerate the uncontrolled insertion of C5.0 fill characters. The host system is required to provide new data at every rising edge of REFCLK (along with TXEN*) to maintain the data stream. If TXEN* is not asserted, the Encoder is loaded with C5.0 (K28.5) sync characters. Because the Encoder is enabled, the transmitted C5.0 characters follow all 8B/10B encoding rules.

Input Register Mapping

In Encoded modes, the bits of the TXDATA input bus are mapped into characters (as shown in *Table 1*), including a TXSVS bit, eight bits of data, and a TXSC/D* bit to select either Special Character codes or Data characters. When the internal FIFO is enabled, TXINT and TXSOC bits are associated with the TXDATA bus. When the internal FIFO is disabled TXPAREN and TXOPIN are associated with the TXDATA bus.

When parity generation is enabled (TXPAREN is HIGH) the TXOPIN bit is interpreted as the ODD parity for the remaining bits of the character. When parity generation is disabled (TXPAREN is LOW) the TXOPIN bit is ignored

If the TXSVS bit is HIGH, an SVS (C0.7) character is passed to the encoder, regardless of the contents of the other TXDATA inputs. If the TXSVS bit is LOW, the associated TXDATA character is encoded per the remaining bits in that character.

The TXSC/D* bit controls the encoding of the TXDATA[7:0] or TXDATA[9:0] bits of each character. It is used to identify if the input character represents a Data Character or a Special Character code. If TXSC/D* is LOW, the character is encoded using the Data Character codes listed in *Table 10*. If TXSC/D* is HIGH, the character is encoded using the Special Character codes listed in *Table 11*.

Parity

Synchronous Encoded operation allows parity checking of all characters written to the Transmit Input Register. If a parity error is detected the TXPERR output goes HIGH for one REFCLK period.

The specific character or characters having parity errors are replaced at the Encoder with the SVS (C0.7) character, which may then be detected at the remote receiver.

Synchronous Pre-encoded

In synchronous pre-encoded mode, both the Transmit FIFO and the 8B/10B encoder are bypassed, and data passes di-

rectly from the Transmit Input Register to the Serializer. The Serializer operates synchronous to REFCLK, which is multiplied by 10 or 5 when BYTE8/10* is HIGH (as selected by the SPDSEL and RANGESEL inputs) to generate the serial data bit-clock. In this mode the TXINT, TXHALT*, TXSVS and TXSOC inputs are used as part of the data input bus. To place the CY7C924ADX into synchronous modes, FIFOBYP* must be LOW.

This mode is usually used for products containing external encoders or scramblers, that must meet specific protocol requirements. The host system is required to provide new data at every rising edge of the REFCLK (along with TXEN*) to maintain the data stream. If TXEN* is not asserted, the Serializer is loaded with C5.0 (K28.5) sync characters. However, because the bypassed encoder is not able to track the running disparity of the previously transmitted character, the transmitted C5.0 characters may be received with a running disparity code-rule violation.

In this mode the LSB of each input character (TXDATA[0]) is shifted out first, followed sequentially by TXDATA[1] through TXDATA[9].

Asynchronous Encoded

Asynchronous Encoded mode is the most powerful operating mode of the CY7C924ADX. Both the Transmit FIFO and the Encoder are enabled. This allows transmission of normal data streams, while offering the added benefits of embedded cell or packet markers, an expanded command set, serial addressing, and in-band bypass-signaling (for flow control or other purposes). The Serializer operates synchronous to REFCLK, which is multiplied by 2.5, 5, or 10 to generate the serial data bit-clock (as selected by SPDSEL and RANGESEL). In this mode the TXSOC, TXRST*, TXINT, TXHALT*, and TXSTOP* inputs are interpreted.

This mode supports the same Input Register mapping as Synchronous Encoded mode, with the addition of the TXSOC bit. Because both the Transmit FIFO and Encoder are enabled, the input FIFO may be loaded at any rate supported by the FIFO (up to 50 MHz), without generating any decoder errors at the receive end of the link. All characters added to the data stream to support these additional capabilities may be automatically extracted by the Receive Control State Machine in the CY7C924ADX Receiver.

Embedded Cell Marker

An embedded cell marker is used to mark the start of cells or frames of information passed from one end of the link to the other. This marker is set by asserting TXSOC HIGH, with TXSC/D* and TXSVS both LOW, along with the remaining data on the TXDATA bus. When the character (or characters) accompanying this marker is read from the output end of the Transmit FIFO, a C8.0 (K23.7) character is inserted into the data stream prior to the following data characters being read from the Transmit FIFO.

Expanded Commands

The standard 8B/10B Character set contains all 256 possible data characters, but only twelve special or *command* characters. To allow use of a larger selection of command codes, a Special Character code was selected to expand the command set.

An expanded command marker is used to mark the associated data as any one of 256 (2^8) possible commands codes. This marker is generated by asserting both TXSOC and TXSC/D* HIGH, with TXSVS being LOW, along with the associated data on the TXDATA bus. When the character accompanying this marker is read from the output end of the Transmit FIFO, a C9.0 (K27.7) character is inserted into the data stream prior to the data characters read from the Transmit FIFO.

Serial Addressing

The CY7C924ADX receiver has the ability to accept or reject data based on an internal address-controlled switch. This switch is turned on when a serial address (matching the receiver address settings) is received.

A serial address is transmitted by asserting TXSOC, TXSC/D*, and TXSVS all HIGH. When the character accompanying this marker is read from the output end of the Transmit FIFO, a C10.0 (K29.7) character is inserted into the data stream prior to the data characters read from the Transmit FIFO. The serial address is either 8 or 10.bits depending on the level on BYTE8/10*.

In-Band Bypass-Signaling

In-band bypass-signaling allows a signal to be sent to the remote receiver without that signal having to pass through the Transmit (or Receive) FIFO.

When TXINT transitions from 0→1, a C0.0 (K28.0) special character is sent. When TXINT transitions from 1→0, a C3.0 (K28.3) special code is sent. These special codes may be used to force a similar signal transition on the RXINT output of an attached CY7C924ADX HOTLink Receiver.

This input may be used to transport a low data rate signal (like a serial RS-232/UART signal) across the interface, without any significant impact on the actual data being transported across the link. It may also be used to transparently propagate FIFO flow control information across the link by directly connecting the RXHALF* flag of the local receiver to the TXINT of the local transmitter. The RXINT at the remote end of the link can then be connected to the TXHALT* input to halt data transfers at the remote end of the link until the local Receive FIFO has sufficient room to continue.

Asynchronous Byte-Packed

Asynchronous byte-packed mode contains the same features as asynchronous encoded, but with support for 10-bit source data. This data is byte-packed through the 8B/10B encoder to deliver the data across the interface.

When sending extended commands, the larger 10-bit character size enlarges the extended command space to 1024 (2^{10}) possible commands codes.

Asynchronous Pre-encoded

In Asynchronous pre-encoded modes, the Transmit FIFO is enabled. This means that all words clocked into the input register are written to the Transmit FIFO before being sent to the Serializer. The Serializer operates synchronous to REFCLK, which is multiplied by 10 or 5 (or 6 or 12 when BYTE8/10* is LOW) to generate the serial data bit-clock. In this mode the TXINT and TXHALT* inputs are used as part of the 10-bit input character. TXSVS, TXSOC and TXSTOP* are still available to stop reading of data from the Transmit FIFO.

These modes are usually used for products containing external encoders or scramblers, that must meet specific protocol requirements. The host system must provide new data at every rising edge of TXCLK (along with TXEN*) to maintain the data stream (without overfilling the Transmit FIFO). If the Transmit FIFO ever goes empty, the Serializer is loaded with an alternating disparity string of C5.0 (K28.5) sync characters (when BYTE8/10* is HIGH) or the bit pattern 0110000100011 (when BYTE8/10* is LOW).

Depending on the system implementation this can be a significant issue. If the remote receiver is configured to decode 8B/10B coded characters, it will probably detect running disparity errors because the bypassed Encoder is not able to track the running disparity of the previously transmitted character. However, since these pre-encoded modes are generally used with alternate forms of scrambling or encoding, this is not generally an issue.

To maintain a data stream without adding these C5.0 SYNC codes, it is necessary that the Transmit FIFO be loaded at or faster than the rate that data is read from that FIFO.

CY7C924ADX HOTLink Receive-Path Operating Mode Descriptions

The HOTLink Receiver can be configured into several operating modes, each providing different capabilities and fitting different reception needs. These modes are selected using the FIFOBYP*, ENBYP*, BYTE8/10* inputs on the CY7C924ADX Transceiver. These modes can be reduced to five primary classes:

- Synchronous Decoded
- Synchronous Undecoded
- Asynchronous Decoded
- Asynchronous Byte-Packed
- Asynchronous Undecoded

In all these modes, serial data is received at one of the differential line receiver inputs and routed to the Deserializer and Framer. The PLL in the clock and data recovery block is used to extract a bit-rate clock from the transitions in the data stream, and uses that clock to capture bits from the serial stream. These bits are passed to the Deserializer where they are formed into 10- or 12-bit characters.

To align the incoming bit stream to the proper character boundaries, the Framer must be enabled by asserting RFEN HIGH. The Framer logic-block checks the incoming bit stream for the unique pattern that defines the character boundaries. This logic filter looks for the ANSI X3.230 symbol defined as a “Special Character Comma” (K28.5 or C5.0). Once a K28.5 is found, the Framer captures the offset of the data stream from the present character boundaries, and resets the boundary to reflect this new offset, thus framing the data to the correct character boundaries.

Since noise induced errors can cause the incoming data to be corrupted, and since many combinations of corrupt and legal data can create an aliased K28.5, the framer may also be disabled by deasserting RFEN LOW.

An option exists in the framer to require multiple K28.5 characters, meeting specific criteria, before the character boundaries are reset. This multi-byte mode of the Framer is enabled by keeping RFEN asserted HIGH for greater than approximately 2000 character clock cycles. For multi-byte framing, the receiver must find a pair of K28.5 characters, both on identical 10-bit boundaries, within a 5-character span (50 bits).

Synchronous Decoded

In these modes, the Receive FIFO is bypassed, while the 10B/8B Decoder is enabled. Framed characters output from the Deserializer are decoded, and passed direct to the Receive Output Register. The Deserializer operates synchronous to the recovered bit-clock, which is divided by 10 generate the output RXCLK clock. In this mode the RXRST* input is not interpreted and may be biased either HIGH or LOW.

These modes are usually used for products that must meet specific protocol requirements. New decoded characters are provided at the RXDATA outputs once every rising edge of RXCLK. If RXEMPTY is asserted HIGH along with the data, the characters in the output register is a C5.0 (K28.5) sync character, and the discard policy is set to non-0. Because the decoder is now enabled, all received characters are checked for compliance to the 8B/10B decoding rules.

Output Register Mapping

The RXDATA[11:0] output bus is mapped into a character consisting of eight bits of data, and two bits that carry Violation and Parity information. An accompanying RXSC/D* bit identifies the character as either control or data.

The bits accompanying each character are interpreted differently depending on the configuration selected. When parity generation is enabled, the parity bit contains the ODD parity of the RXDATA bus.

When parity generation is disabled, these bits have combinations that identify the meaning of the remaining bits of the character. If RXRVS is HIGH and RXSC/D* is LOW, the decoder outputs a C0.7, C1.7, C2.7 or C4.7 in response to reception of either an SVS (C0.7) character or other invalid character.

Parity

Synchronous Decoded modes allows ODD parity generation on all decoded characters. If characters are detected with parity errors in the transmitter, they will have been replaced with the SVS (C0.7) character. If a C0.7 character is received it may be an indication of a parity error at the transmitter.

Synchronous Undecoded

In this mode, both the Receive FIFO and the 10B/8B Decoder are bypassed, and data passes directly from the Deserializer to the output register. The Deserializer operates synchronous to the recovered bit-clock, which is divided by 10 to generate the output RXCLK clock. In this mode the RXRST* input is not interpreted and may be biased either HIGH or LOW.

This mode is usually used for products containing external decoders or descramblers that must meet specific protocol requirements. New data is provided at the RXDATA outputs once every rising edge of RXCLK. Received characters are not checked for any specific coding requirements and no decoding errors are reported.

Asynchronous Decoded

Asynchronous Decoded modes are the most powerful operating modes of the CY7C924ADX HOTLink Receiver. Both the Receive FIFO and the Decoder are enabled. This allows reception of normal data streams, while offering the added benefits of embedded cell markers, an expanded command set,

serial address support, and in-band bypass-signaling (for flow control or other purposes). All characters added to the data stream to support these additional capabilities may be automatically extracted by the Receive Control State Machine in the CY7C924ADX Transceiver.

The deserializer operates synchronous to the recovered bit-clock, which is divided by 10 to generate the Receive FIFO write clock. Characters are read from the Receive FIFO, using the external RXCLK input, when addressed by AM* and selected by RXEN*. In this mode the RXRST* input is interpreted.

Asynchronous Decoded modes support the same Output Register mapping and Parity generation capabilities as the Synchronous Decoded modes. Because both the Receive FIFO and Decoder are enabled, the output FIFO may be read at any rate supported by the FIFO, however, if the Receive FIFO ever indicates a full condition (RXFULL* is asserted), data may be lost.

Embedded Cell Marker

An embedded cell marker is used to mark the start of cells or frames of information passed from one end of the link to the other. When a C8.0 (K23.7) character is detected in the data stream, the following character is written to the Receive FIFO along with RXSOC set HIGH, and RXSC/D* and RXRVS set LOW. When the character accompanying this marker is read from the Receive FIFO with these same bits set, it indicates the start of a cell or frame.

Expanded Command

The standard 8B/10B Character set contains all 256 possible data characters, but only twelve Special Character codes. To allow use of a larger selection of command codes, one Special Character code was selected to expand the command set.

An Expanded Command marker is used to mark the associated data as any one of 256 (2^8) possible commands codes. When a C9.0 (K27.7) character is detected in the data stream, the following character is written to the Receive FIFO along with both RXSOC and RXSC/D* set HIGH, and RXRVS set LOW. When the character accompanying this marker is read from the Receive FIFO with these same bits set, it may be used to indicate that the data on the RXDATA bus is an Expanded Command.

Serial Addressing

The CY7C924ADX receive path can be directed to accept all characters, or to only accept that data specifically addressed to it. This address control is managed through an embedded Address Compare Register in the receiver logic. This register supports either domain (multicast) or exact-match (unicast) based compares on an address field received across the serial link. When a C10.0 (K29.7) special code is received, the immediately following data character contains the address field that is compared with the receiver Serial Address Register contents.

When the CY7C924ADX is configured for multicast address matching, the received address field is compared as an OR of a bit-wise AND with the Serial Address Register. A valid match between any of the bits sets the switch to allow the following data to be written into the Receive FIFO.

When the CY7C924ADX is configured for unicast address matching, the received address field is compared for an exact match with the Serial Address Register. If an exact match is found, a switch is set in the receiver to accept all following data until the next serial address marker is found.

In-Band Bypass-Signaling

In-band bypass-signaling allows a signal to be received at the local receiver without that signal having to pass through the Receive (or Transmit) FIFO.

When a C0.0 (K28.0) character is received, the RXINT output is set HIGH. When a C3.0 (K28.3) character is received, the RXINT output is set LOW. These special codes are generated by forcing similar transitions into the TXINT input of the CY7C924ADX HOTLink Transmitter sourcing the data stream.

This output may be used to transport a low data-rate signal (like a serial RS-232/UART signal) across the interface, without any significant impact on the actual data being transported across the link. It may also be used to transparently propagate FIFO flow-control information across the link by directly connecting the RXHALF* flag of the local receiver to the TXINT of the local transmitter. The RXINT at the remote end of the link can then be connected to the TXHALT* input to halt data transfers at the remote end of the link until the local Receive FIFO has sufficient room to continue.

Asynchronous Byte-Packed

Asynchronous byte-packed mode contains the same features as asynchronous decoded, but with support for 10-bit source data. The received characters are decoded first back into 8-bit data characters, which are then re-assembled into 10-bit source data.

Because of the time difference involved with the packing and unpacking operations, this mode can only be used with the internal FIFOs enabled.

When receiving extended commands, the larger 10-bit character size enlarges the extended command space to 1024 (2^{10}) possible commands codes.

When receiving a serial address, the larger 10-bit character size also increases the Serial Address Register to 10 bits. This allows up to 10 separate domains for multicast addressing or 1024 unique addresses for unicast addressing.

Asynchronous Undecoded

In Asynchronous Undecoded modes, the Receive FIFO is enabled. This means that all characters received from the serial interface are written to the Receive FIFO before being passed to the output register. The Deserializer operates synchronous to the recovered bit-clock, which is divided by 10 (or 12) to generate the Receive FIFO write clock. Data is read from the Receive FIFO, using the RXCLK input clock, when addressed by AM* and selected by RXEN*.

These modes are usually used for products containing external decoders or descramblers, that must meet specific protocol requirements. New data may be read from the Receive FIFO any time that the FIFO status flags indicate a non-empty condition (RXEMPTY* is deasserted). To ensure that data is not lost, the Receive FIFO must be read faster than data is loaded into the Receive FIFO.

If the receiver is to provide framed characters, it is necessary for the transmit end to include C5.0 (K28.5) characters in the data stream. This can be done by:

- operating the transmitter in encoded mode and writing C5.0 characters into the data stream,
- operating the transmitter in pre-encoded mode and writing the 10-bit value for an encoded C5.0 character to the data stream (`1100000101` or `0011111010`)
- not enabling the transmitter when it is operated in synchronous mode, or by allowing the transit FIFO to go empty when it is operated in asynchronous mode.

BIST Operation and Reporting

The CY7C924ADX HOTLink Transceiver incorporates the same Built-In Self-Test (BIST) capability used with the CY7B923/ CY7B933 and CY7C929 HOTLink components. This link diagnostic uses a Linear Feedback Shift Register (LFSR) to generate a 511-character repeating sequence that is compared, character-for-character, at the receiver.

BIST mode is intended to check the entire high-speed serial link at full link-speed, without the use of specialized and expensive test equipment. The complete sequence of characters used in BIST are documented in the “HOTLink Built-In Self-Test” application note.

BIST Enable Inputs

There are separate BIST enable inputs for the transmit and receive paths of the CY7C924ADX. These inputs are both active LOW; i.e., BIST is enabled in its respective section of the device when the BIST enable input is determined to be at a logic-0 level. Both BIST enable inputs are asynchronous; i.e., they are synchronized inside the CY7C924ADX to the internal state machines.

BIST Transmit Path

The transmit path operation with BIST is controlled by the TXBISTEN* input and overrides most other inputs (see *Figure 7*). When the Transmit FIFO is enabled (not bypassed) and TXBISTEN* is recognized internally, all reads from the Transmit FIFO are suspended and the BIST generator is enabled to sequence out the 511 character repeating BIST sequence. If the Transmit Control State Machine was in the middle of an atomic operation (e.g., sending an extended command) the Data Character associated with the Special Character code is transmitted prior to recognition of the TXBISTEN* signal and suspension of FIFO data processing. If the recognition occurs in the middle of a data field, the following data is not transmitted at that time, but remains in the Transmit FIFO. Once the TXBISTEN* signal is removed, the data in the Transmit FIFO is again available for transmission. To ensure proper data handling at the destination, the transmit host controller should either use TXHALT* or TXSTOP* to prevent transmission of data at specific boundaries, or allow the Transmit FIFO to completely empty before enabling BIST.

With transmit BIST enabled, the Transmit FIFO remains available for loading of data. It may be written up to its normal maximum limit while the BIST operation takes place. To allow removal of stale data from the Transmit FIFO, it may also be

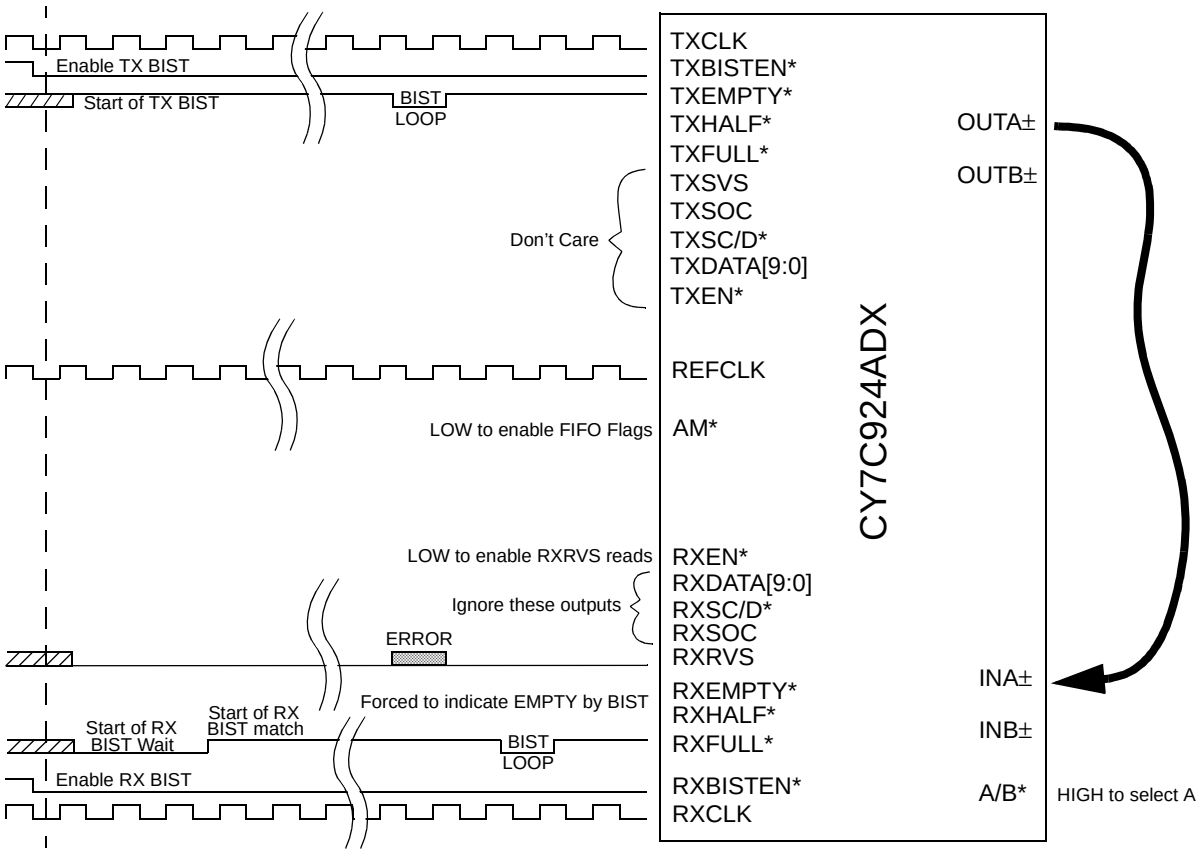


Figure 7. Built-In Self-Test Illustration.

reset during a BIST operation. The reset operation proceeds as documented, with the exception of the information presented on the TXEMPTY* FIFO status flag. Since this flag is used to present BIST loop status, it continues to reflect the state of the transmit BIST loop status until TXBISTEN* is no longer recognized internally. The completion of the reset operation may still be monitored through the TXFULL* FIFO status flag.

The TXEMPTY* flag, when used for transmit BIST progress indication, continues to reflect the active HIGH or active LOW settings determined by the UTOPIA or Cascade timing model selected by EXTFIFO; i.e., when configured for the Cascade timing model, the TXEMPTY* and TXFULL* FIFO flags are active HIGH, when configured for the UTOPIA timing model the TXEMPTY* and TXFULL* FIFO flags are active LOW. The illustration in *Figure 7* uses the UTOPIA conventions.

When TXBISTEN* is first recognized, the TXEMPTY* flag is clocked to a reset state, regardless of the addressed state of the Transmit FIFO (if AM* is LOW or not), but is not driven out of the part unless AM* has been sampled asserted (LOW). Following this, on each completed pass through the BIST loop, the TXEMPTY* flag is set for one interface clock period (TXCLK or REFCLK).

The TXEMPTY* flag remains set until the interface is addressed and the state of TXEMPTY* has been observed. If the device is not addressed (AM* is not sampled LOW), the flag remains set internally regardless of the number of TXCLK clock cycles that are processed. If the device status is not polled on a sufficiently regular basis, it is possible for the host system to miss one or more of these BIST loop indications.

A pass through the loop is defined as that condition where the Encoder generates the D0.0 state. Depending on the initial state of the BIST LFSR, the first pass through the loop may occur at substantially less than 511 character periods. Following the first pass, as long as TXBISTEN* remains LOW, all remaining passes are exactly 511 characters in length.

When the Transmit FIFO is bypassed, the interface is clocked by the REFCLK signal instead of TXCLK. While the active or asserted state of the TXEMPTY* signal is still controlled by the EXTFIFO, the state of any completed BIST loops is no longer preserved. Instead, the TXEMPTY* flag reflects the dynamic state of the BIST loop progress, and is asserted only once every 511 character periods. If the interface is not addressed at the time that this occurs, then the FIFO status flags remain in a High-Z state and the loop event is lost.

BIST Receive Path

The receive path operation in BIST is similar to that of the transmit path. While the Receive FIFO is enabled (not bypassed) and RXBISTEN* is recognized internally, all writes to the Receive FIFO are suspended. If the receiver had a previous serial address match and was accepting data, no additional characters are written to the Receive FIFO. If the receive data state machine was in the middle of processing a multi-character sequence or other atomic operation (e.g., a start of cell marker and its associated data), the characters associated with the atomic operation are discarded and not written to the Receive FIFO.

Upon internal recognition of RXBISTEN*, the serial address match flag is cleared such that once BIST has been disabled and data is again being received, all received data is rejected until a new serial address is again received that matches the address match criteria.

NOTE: if the CY7C924ADX is set to match all data (all 1s in the multicast match field), then it is not necessary to get an address match before receiving data following the termination of BIST. When reset or programmed to this state, the device ignores all serial address commands and matches all data.

Any data present in the Receive FIFO when RXBISTEN* is recognized remains in the FIFO and cannot be read until the BIST operation is complete. The data in the Receive FIFO remains valid, but is NOT available for reading through the host parallel interface. This is because the error output indicator for receive BIST operations is the RXRVS signal, which is normally part of the RXDATA bus. To prevent read operations while BIST is in operation, the RXEMPTY* and RXHALF* flags are forced to indicate an Empty condition. Once RXBISTEN* has been removed and recognized internally, the Receive FIFO status flags are updated to reflect the current content status of the Receive FIFO.

To allow removal of stale data from the Receive FIFO, it may be reset during a BIST operation. The reset operation proceeds as documented, with the exception that the RXEMPTY* and RXHALF* status flags already indicate an empty condition. The RXFULL* flag is used to present BIST progress. The active (asserted) state on RXFULL* (and RXEMPTY*) remain controlled by the present operating mode and interface timing model (UTOPIA or Cascade).

When RXBISTEN* has been recognized, RXFULL* becomes the receive BIST loop indicator (regardless of the logic state of FIFOBYP*). When RXBISTEN* is first recognized, the RXFULL* flag is clocked to a set state, regardless of the addressed state of the Receive FIFO (if AM* is sampled LOW or not). Following this, RXFULL* remains set until the receiver detects the start of the BIST pattern. Then RXFULL* is deasserted for the duration of the BIST pattern, pulsing asserted for one RXCLK period on the last symbol of each BIST loop. If 14 of 28 consecutive characters are received in error, RXFULL* returns to the set state until the start of a BIST sequence is again detected.

Just like the BIST status flag on the transmit data path, the RXFULL* flag captures the asserted states, and keeps them until they are read. This means that if the status flag is not read on a regular basis, events may be lost.

The detection of errors is presented on the RXRVS output. Unlike the RXFULL* FIFO status flag, the active state of this output is not controlled by the EXTFIFO input. With the Receive FIFO enabled, these outputs should operate the same as the RXFULL* flag, with respect to preserving the detection state of an error until it is read.

Unlike the RXFULL* flag, which only needs the CY7C924ADX to be addressed (AM* sampled LOW by RXCLK) to enable the RXFULL* three-state driver, and an RXCLK to “read” the flag, the RXRVS output requires a selection (assertion of RXEN* while addressed) to enable the RXDATA bus three-state drivers. The selection process is necessary to ensure that a multi-PHY implementation does not enable multiple RXRVS drivers at the same time.

When the Receive FIFO is bypassed, the interface is clocked by the RXCLK output signal. While the active or asserted state of the RXFULL* signal is still controlled by the EXTFIFO input, the state of any completed BIST loops or detected errors are no longer preserved. Instead, the RXFULL* flag reflects the dynamic state of the BIST loop progress, and is asserted only

once every 511 character periods. If the interface is not addressed at the time that this occurs, then the FIFO status flags remain in a high-Z state and the loop event is lost. This is also true of the RXRVS output, such that if the CY7C924ADX receive path is not selected to enable the RXDATA bus three-state drivers, the detection of a BIST miscompare is lost.

BIST Three-state Control

When BIST is enabled on either the transmitter or the receiver, the three-state enable signals for the BIST status flags and error indicators work the same as for normal data processing. The output drivers for the BIST status that is presented on FIFO status flags are only enabled when AM* has been sampled asserted (LOW) by the respective clock (TXCLK, RXCLK, or REFCLK).

To access the BIST error information, it is necessary to perform a read cycle of the addressed receiver. This means that AM* must be LOW to allow a receiver address match (Rx_Match) to exist, and RXEN* must be asserted to select the device. Because the part is in BIST, no data is read from the FIFO, but the data bus is driven. This allows the RXRVS indicator to be driven onto the RXDATA bus. So long as RXEN* remains asserted, the receiver stays selected, the data bus remains driven, and RXRVS has meaning.

Bus Interfacing

The parallel transmit and receive host interfaces to the CY7C924ADX are configurable for either synchronous or asynchronous operation. Each of these configurations supports two selectable timing and control models of UTOPIA or Cascade.

All asynchronous bus configurations have the internal Transmit and Receive FIFOs enabled. This allows data to be written or read from these FIFOs at any rate up to the maximum 50-MHz clock rate of the FIFOs. All internal operations of the CY7C924ADX do not use the external TXCLK or RXCLK, but instead make use of synthesized derivatives of REFCLK for transmit path operations and a recovered character clock for receive path operations.

All synchronous bus configurations require the bus interface operations to be synchronous to REFCLK on the transmit path and the recovered clock (output as RXCLK) on the receive path. The internal FIFOs are bypassed in all synchronous modes.

The two supported timing and control models are UTOPIA and Cascade. These timing models take their name from their default configuration. The UTOPIA timing model is based on the ATM Forum UTOPIA interface standards. This timing model is that of a FIFO with active LOW FIFO status flags and read/write enables.

The Cascade timing model is a modification of the UTOPIA configuration that changes the flags and FIFO read/write enables to active HIGH. This model is present primarily to allow depth expansion of the internal FIFO by direct coupling to external CY7C42x5 synchronous FIFOs. To allow this direct coupling, the cycle-to-cycle timing between the transmit and receive enables (TXEN* and RXEN*) are also modified to ensure correct data transfer.

These four configurations of bus operation and timing/control can all be used with or without external FIFOs. Depending on the specific mode selected, the amount of external hardware necessary to properly couple the CY7C924ADX to state machines or external FIFOs is minimal in all cases, and may be zero if the proper configuration is selected.

With only minor exceptions, all configurations rely on the common UTOPIA concepts of addressing and selection to control the enabled/disabled state of the output drivers, and when data can be written to or read from the part.

UTOPIA Interface Background

The UTOPIA interface is defined by the ATM Forum as the bus interface between the ATM and PHY layer devices of an ATM system. This interface is defined as 8 or 16 bits wide, with the latter reserved mainly for high-speed physical interfaces (PHYs) such as 622 Mbps OC-12. Due to the limited speed range of the CY7C924, only the 8-bit interface is implemented.

UTOPIA-1 was the original UTOPIA specification (created in 1993) which covers transport of:

- 155.52 Mbps (scrambled SONET/OC-3)
- 155.52 Mbps (8B/10B block coded at 194.4 MBaud)
- 100 Mbps (4B/5B encoded TAXI)
- 44.736 Mbps (DS-3/T3)
- 51.84 Mbps (OC-1)

The UTOPIA-1 interface has a maximum clock rate of 25 MHz. All AC-timing and pin descriptions are covered in the UTOPIA-1 Specification, Version 2.01.

UTOPIA-2 was created as an addendum to the UTOPIA-1 specification. In this revision, the parallel interface was extended to both 33 MHz and 50 MHz to accommodate PCI bus architectures in ATM designs. A method of addressing was added to allow multiple devices (PHYs) to share a common host bus. Also, a description of a management interface was added (not supported by this device).

The CY7C924ADX contains all pins necessary to support the UTOPIA-1 and, through use of an external address decoder, can emulate the multi-PHY capability of a UTOPIA-2 interface. The maximum bus speed supports the full 50 MHz I/O rate for emerging high-performance systems.

UTOPIA Address Match and Selection

All actions on a UTOPIA-2 interface are controlled by the Address Match and selection states of the interface. These states control the read and write access to the Receive and Transmit FIFOs, access to the FIFO status flags, reset of the Transmit and Receive FIFOs, and read and write access to the Serial Address Register. The CY7C924ADX supports the concept of an “address match” through a single Address Match (AM*) input.

Address Match and FIFO Flag Access

The CY7C924ADX makes use of a single active-LOW Address Match (AM*) to generate address-match conditions. When this input is LOW it is equivalent to an ATM address compare on both the TXADDR and RXADDR buses. This allows multiple CY7C924ADX devices to share a common bus, with device output three-state controls being managed by either an address match condition (AM* sampled LOW), or by a selection state.

The Transmit and Receive FIFO flag output drivers are enabled in any TXCLK, REFCLK, or RXCLK cycle following AM* being sampled asserted (LOW) by the rising edge of the re-

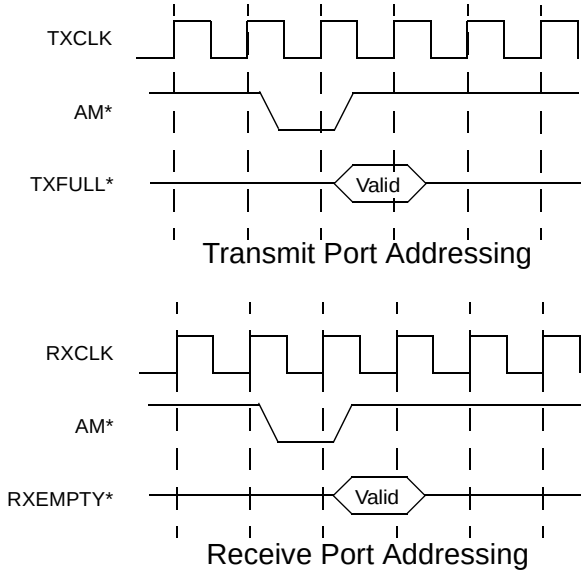


Figure 8. FIFO Flag Driver Enables.

spective clock. The AM* input is sampled separately by the clocks for the transmit and receive interfaces, which allows these clocks to be both asynchronous to each other, and to operate at different clock rates. An example of both Transmit and Receive FIFO flag access is shown in *Figure 8*.

When the Transmit FIFO is enabled (FIFOBYP* is HIGH) or in Byte-Packed mode) and AM* is sampled LOW by the rising edge of TXCLK, the output drivers for the TXFULL* and TXEMPTY* FIFO flags are enabled. When AM* is sampled HIGH by the rising edge of TXCLK, these same output drivers are disabled.

When the Transmit FIFO is bypassed (FIFOBYP* is LOW and not in byte-packed mode) and AM* is sampled LOW by the rising edge of REFCLK, the output drivers for the TXFULL* and TXEMPTY* FIFO flags are enabled. When AM* is sampled HIGH by the rising edge of REFCLK, the FIFO flag output drivers are disabled.

When AM* is sampled LOW by the rising edge of RXCLK (input or output), the output drivers for the RXFULL* and RXEMPTY* FIFO flags are enabled. When AM* is sampled HIGH by the rising edge of RXCLK, the FIFO flag output drivers are disabled.

Device Selection

The concept of selection is used to control the access to the transmit and receive parallel-data ports of the device. There are three primary types of selection:

- Transmit data selection (with and without internal Transmit FIFO)
- Receive data selection (with and without internal Receive FIFO)
- Continuous selection (for either or both transmit and receive interfaces)

In addition to these normal selection types, there are two additional sequences that are used to control the internal Transmit and Receive FIFOs reset operations, and to control read/write access to the Serial Address Register:

- Transmit reset sequence
- Receive reset sequence (includes access to the Serial Address Register)

Of these operations, the transmit data selection and transmit reset sequence are mutually exclusive and cannot exist at the same time. The receive data selection and receive reset sequence are also mutually exclusive and cannot exist at the same time. Either transmit operation can exist at the same time as either receive operation.

All normal forms of selection require that an Address Match condition must exist (AM* sampled LOW) either at the same time as the selection control signal being sampled asserted, or one or more clock cycles prior to the selection control signal being sampled asserted.

Transmit Data Selection

Asynchronous With UTOPIA Timing and Control (Transmit FIFO Enabled)

When AM* is sampled LOW and TXRST* is sampled HIGH by the rising edge of TXCLK, a Tx_Match condition is generated. This Tx_Match condition continues until AM* is sampled HIGH or TXRST* is sampled LOW at the rising edge of TXCLK. When a Tx_Match (or Tx_RstMatch) condition is present, the TXEMPTY* and TXFULL* output drivers are enabled. When a Tx_Match (or Tx_RstMatch) condition is not present, these same drivers are disabled (HIGH-Z).

The selection state of the Transmit FIFO is entered when a Tx_Match condition is present, and TXEN* transitions from HIGH to LOW. Once selected, the Transmit FIFO remains selected until TXEN* is sampled HIGH by the rising edge of TXCLK. In the selected state, data present on the TXDATA inputs is captured and stored in the Transmit FIFO. This transmit interface selection process is shown in *Figure 9*.

Synchronous With UTOPIA Timing and Control (Transmit FIFO Bypassed)

When the Transmit FIFO is bypassed (FIFOBYP* is LOW and not in byte-packed mode), the CY7C924ADX must still be selected to write data into the Transmit Input Register.

When AM* is sampled LOW and TXRST* is sampled HIGH by the rising edge of REFCLK, a Tx_Match condition is generated. This Tx_Match condition continues until AM* is sampled HIGH or TXRST* is sampled LOW at the rising edge of TXCLK. When a Tx_Match (or Tx_RstMatch) condition is present, the TXEMPTY* and TXFULL* output drivers are enabled (with the Transmit FIFO bypassed, the status flags normally indicate an Empty condition). When a Tx_Match (or Tx_RstMatch) condition is not present, these same drivers are disabled (High-Z).

The selection state of the Transmit Input Register is entered when a Tx_Match condition is present, and TXEN* transitions from HIGH to LOW. Once selected, the transmit input register remains selected until TXEN* is sampled HIGH by the rising edge of REFCLK. In the selected state, data present on the TXDATA inputs is captured in the Transmit Input Register and passed to the Serializer or Encoder (as selected by the ENCBYP* input). This transmit interface selection process is shown in *Figure 10*.

When data is not written to the Transmit Input Register, the data stream is automatically padded with C5.0 (K28.5) SYNC characters. If the 8B/10B Encoder is enabled, disparity track-

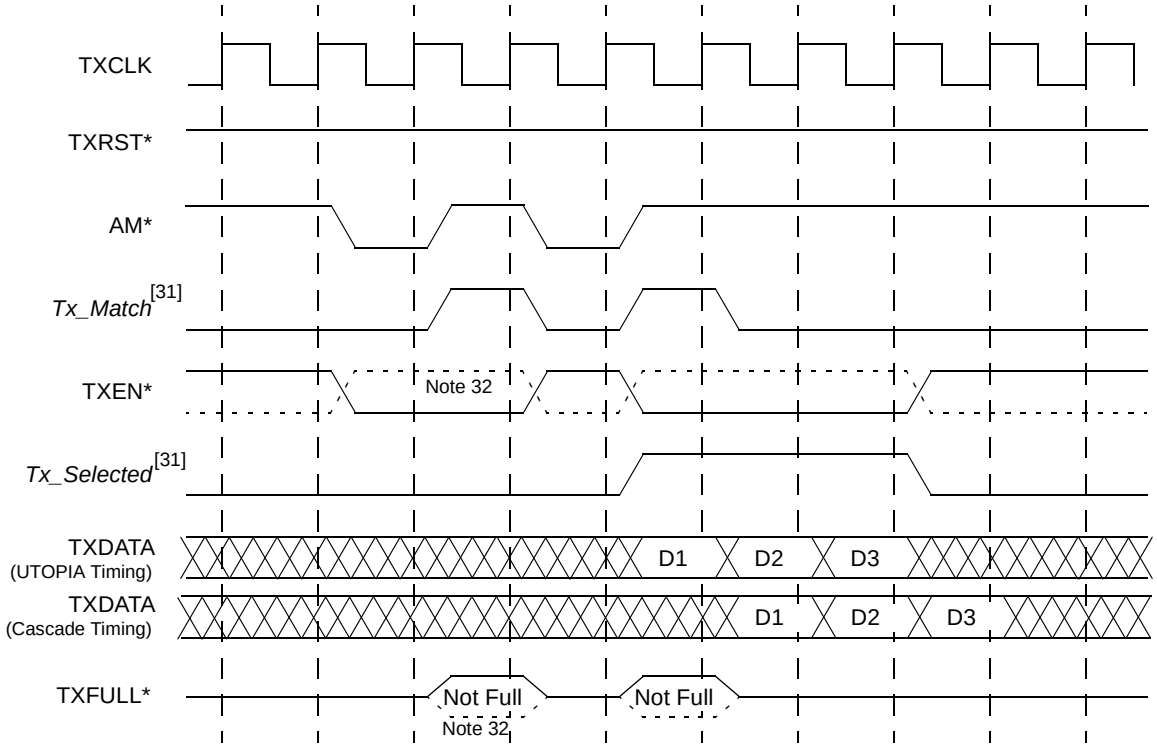


Figure 9. Transmit Selection with Transmit FIFO Enabled.

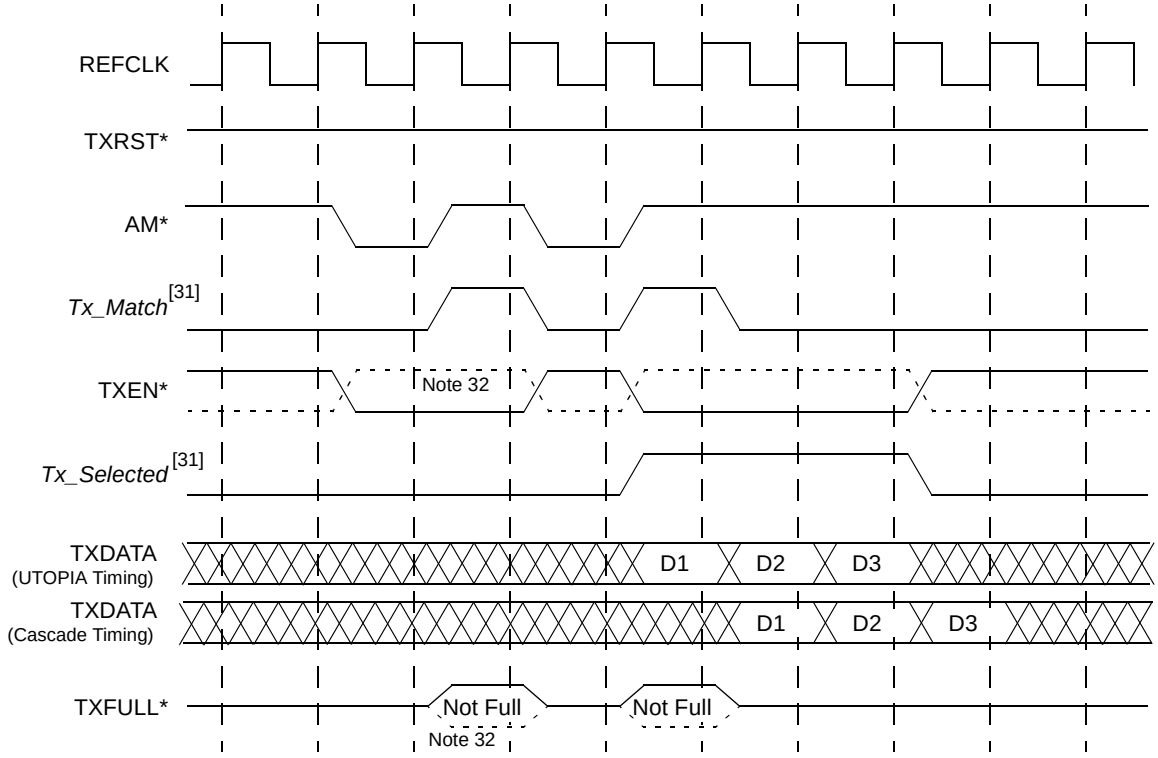


Figure 10. Transmit Selection with Transmit FIFO Bypassed.

Notes:

31. Signals labeled in italics are internal to the CY7C924ADX.
32. Signals shown as dotted lines represent the differences in timing and active state of signals when operated in Cascade Timing.

ing allows the added C5.0 fill characters to follow all 8B/10B encoding rules. If the 8B/10B encoder is bypassed, disparity tracking is disabled, and the transition between externally encoded data and internally generated C5.0 characters may generate a running disparity error at the attached receiver. The same error may occur at the transition between the internal C5.0 characters and the resumption of externally encoded data. When strings of contiguous C5.0 characters are generated, each C5.0 has alternating running disparity with the previous C5.0 character.

Receive Data Selection

Asynchronous With UTOPIA Timing and Control (Receive FIFO Enabled)

When AM* is sampled LOW and RXRST* is sampled HIGH by the rising edge of RXCLK input, an Rx_Match condition is generated. This Rx_Match condition continues until AM* is sampled HIGH or RXRST* is sampled LOW at the rising edge of RXCLK input. When an Rx_Match (or Rx_RstMatch) condition is present, the RXEMPTY* and RXFULL* output drivers are enabled. When an Rx_Match (or Rx_RstMatch) condition is not present, these same drivers are disabled (High-Z).

The selection state of the Receive FIFO is entered when an Rx_Match condition is present, and RXEN* transitions from HIGH to LOW. Once selected, the Receive FIFO remains selected until RXEN* is sampled HIGH by the rising edge of RXCLK input. The selected state initiates a read cycle from the Receive FIFO and enables the Receive FIFO data onto the RXDATA bus. This receive interface selection process is shown in *Figure 11*.

Synchronous With UTOPIA Timing and Control (Receive FIFO Bypassed)

When the Receive FIFO is bypassed (FIFOBYP* is LOW and not in a byte-packed mode), the CY7C924ADX must still be selected to enable the output drivers for the RXDATA bus. With the Receive FIFO bypassed, RXCLK becomes a synchronous output clock operating at the character rate.

When AM* is sampled LOW and RXRST* is sampled HIGH by the rising edge of RXCLK output, an Rx_Match condition is generated. This Rx_Match condition continues until AM* is sampled HIGH or RXRST* is sampled LOW at the rising edge of RXCLK.

When an Rx_Match (or Rx_RstMatch) condition is present, the RXEMPTY* and RXFULL* output drivers are enabled. With the Receive FIFO bypassed, these flags normally indicate a non-empty condition but may indicate empty if a C5.0 fill character is present in the output register and the receiver discard policy is non-0. When an Rx_Match (or Rx_RstMatch) condition is not present, these same drivers are disabled (High-Z).

The selection state of the Receive Output Register is entered when an Rx_Match condition is present, and RXEN* transitions from HIGH to LOW. Once selected, the Receive Output Register remains selected until RXEN* is sampled HIGH by the rising edge of RXCLK output. In the selected state, the output drivers for the RXDATA outputs are enabled, and new data is presented to the RXDATA bus on every clock cycle.

Continuous Selection

Continuous Selection is a specialized form of selection which does not require sequenced assertion of AM* and TXEN* or RXEN* to select the device for data transfers. In this Continuous Selection mode, the AM* and associated TXEN* or RXEN* enable signal must be asserted when the device is

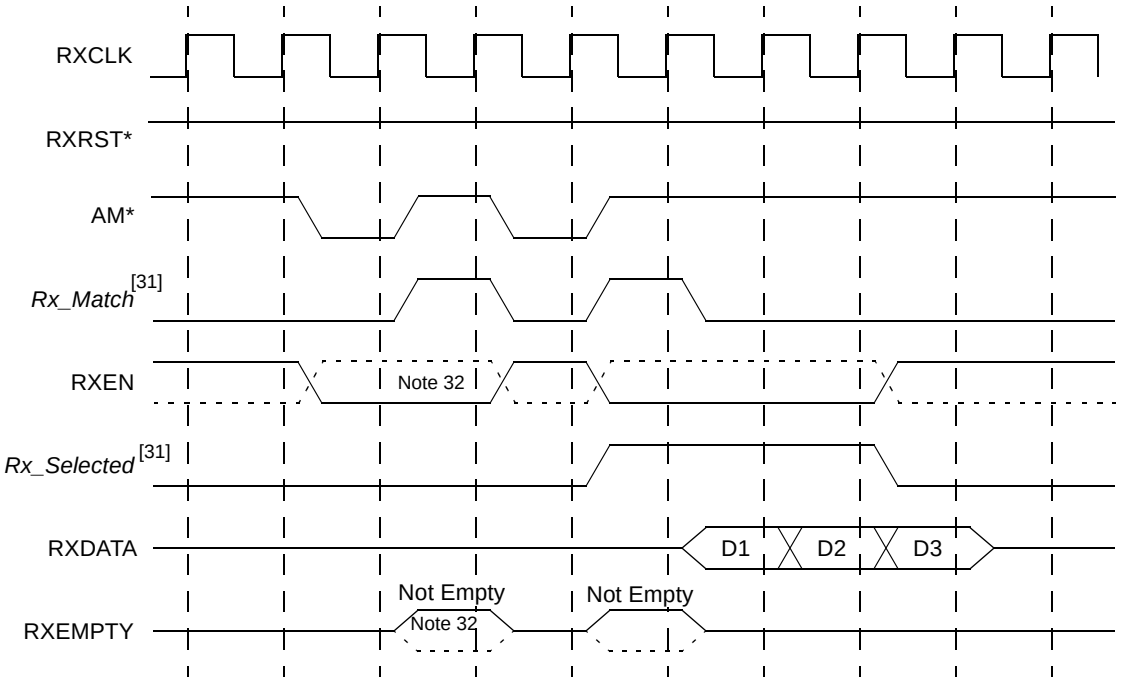


Figure 11. Receive Selection with Receive FIFO Enabled.

powered up or during assertion of RESET*[1:0]. So long as these signals remain asserted, the device remains selected and data is accepted and presented on every clock cycle.

NOTE: The use of continuous selection makes it impossible to reset the respective internal FIFOs, or to access the Serial Address Register.

FIFO Reset Address Match

When AM* and TXRST* are both LOW, and this condition is sampled by the rising edge of TXCLK, a Tx_RstMatch condition is generated. This Tx_RstMatch condition continues until AM* or TXRST* is sampled HIGH by the rising edge of TXCLK. When a Tx_RstMatch (or Tx_Match) condition is present, the TXEMPTY* and TXFULL* output drivers are enabled (just as in a normal Tx_Match condition). When a Tx_RstMatch (or Tx_Match) condition is not present, these same drivers are disabled (High-Z). The Transmit FIFO reset Address Match is shown in *Figure 12*. Note that although TXRST* remains LOW for more than one clock cycle, the Tx_RstMatch does not because the AM* signal is no longer asserted (LOW).

When AM* and RXRST* are both LOW, and this condition is sampled by the rising edge of RXCLK, an Rx_RstMatch condition is generated. This Rx_RstMatch condition continues until AM* or RXRST* is sampled HIGH, at the rising edge of RXCLK. When an Rx_RstMatch (or Rx_Match) condition is present, the RXEMPTY* and RXFULL* output drivers are enabled. When an Rx_RstMatch (or Rx_Match) condition is not present, these same drivers are disabled (High-Z). The Receive FIFO reset Address Match is shown in *Figure 13*. Note that while the FIFO flags remain asserted for more than one clock cycle, this is due to an Rx_Match condition, not a continuation of the Rx_RstMatch.

FIFO Reset Sequence

On power-up, the Transmitter and Receiver FIFOs are cleared automatically. If the usage of the FIFOs in specific operating modes results in stale or unwanted data, this data can be cleared by resetting the respective FIFO. Data in the Transmit FIFO will empty automatically if it is enabled to read the FIFO (assuming TXHALT* is not LOW). Stale received data can be “flushed” by reading it, or the Receive FIFO can be reset to remove the unwanted data.

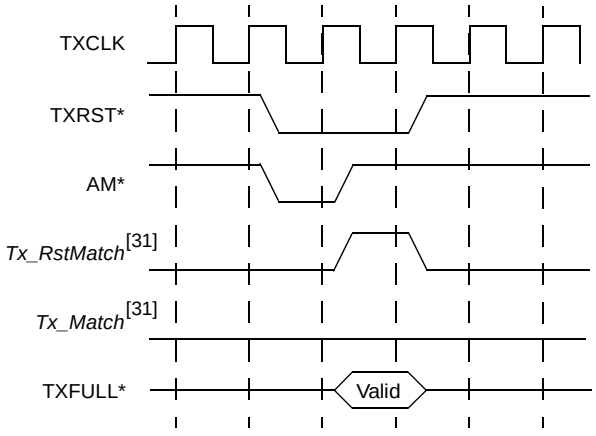


Figure 12. Transmit FIFO Reset Address Match.

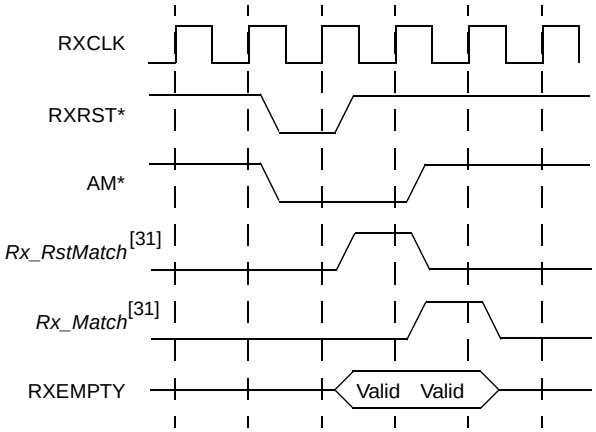


Figure 13. Receive FIFO Reset Address Match.

The Transmit and Receive FIFOs are reset when the Tx_RstMatch or Rx_RstMatch condition remains present for eight consecutive clock cycles. Any disruption of the reset sequence prior to reaching the eight cycle count, either by removal of AM* or the respective TXRST* or RXRST*, or assertion of the associated TXEN* or RXEN*, terminates the sequence and does not reset the FIFO. Because AM* must remain asserted during the reset sequence, the addressed FIFO flags remain driven during the entire sequence.

Transmit FIFO Reset Sequence

The Transmit FIFO reset sequence is started when TXRST* and AM* are first sampled LOW by the rising edge of TXCLK. However, if TXEN* is asserted, the reset sequence is inhibited until it is removed (TXEN* is sampled HIGH for UTOPIA timing or LOW for Cascade timing). Because a Tx_RstMatch condition is present, the Transmit FIFO flags are asserted and can be used to track the status of any Transmit FIFO reset in progress. Once the reset sequence has reached its maximum count, the Transmit FIFO flags are forced to indicate a FULL* condition (TXEMPTY* is deasserted, and both TXHALF* and TXFULL* are asserted). This indicates that the Transmit FIFO reset has been recognized by the Transmit Control State Machine and that a reset has been started.

NOTE: The FIFO Full state forced by the reset operation is different from a Full state caused by normal FIFO data writes. For normal FIFO write operations, when Full is first asserted, the Transmit FIFO must still accept up to eight additional writes of data. When a Full state is asserted due to a Transmit FIFO reset operation, the FIFO will not accept any additional data.

When a Transmit FIFO reset sequence is enabled and has been active for at least eight TXCLK cycles, a Transmit FIFO reset operation is started. This FIFO reset operation is not allowed to progress within the device until the associated TXRST* is sampled deasserted (HIGH). Following deassertion of TXRST* (which starts the FIFO reset operation), selection of the device for normal data transfers is inhibited during the immediately following TXCLK clock cycle. If a selection of the transmit interface is attempted during this immediately following cycle (by asserting TXEN*), the selection is ignored, and the device remains unselected until TXEN* is deasserted, and reasserted in a following TXCLK cycle.

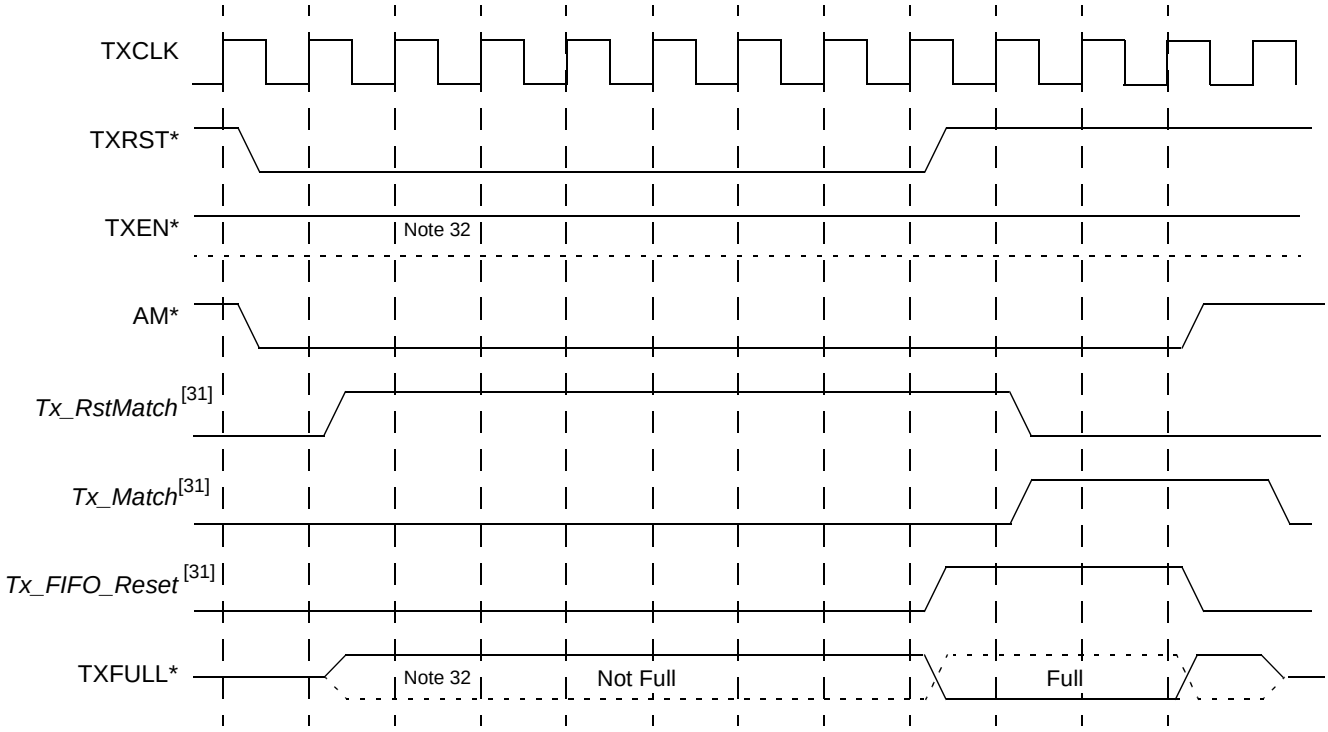


Figure 14. Transmit FIFO Reset Sequence.

The Transmit FIFO reset does not complete until the external reset condition is removed. This can be removed by deassertion of either TXRST* or AM*. If AM* is deasserted (HIGH) to remove the reset condition, the Transmit FIFO flag's drivers are disabled, and the Transmit FIFO must be addressed at a later time to validate completion of the Transmit FIFO reset. If TXRST* is deasserted (HIGH) to remove the reset condition, the Tx_RstMatch is changed to a Tx_Match, and the Transmit FIFO status flags remain driven. The Transmit FIFO reset operation is complete when the Transmit FIFO flags indicate an Empty state (TXEMPTY* is asserted and both TXHALF* and TXFULL* are deasserted). A valid Transmit FIFO reset sequence is shown in *Figure 14*.

Here the TXRST* and AM* are asserted (LOW) at the same time. When these signals are both sampled LOW by TXCLK, a Tx_RstMatch condition is present. With TXEN* deasserted (HIGH), the Transmit FIFO is not selected for data transfers. This Tx_RstMatch condition remains for eight TXCLK cycles to generate the Tx_FIFO_Reset. Following this the TXFULL* FIFO status flag is asserted to indicate that the Transmit FIFO reset sequence has completed and that a Transmit FIFO reset is in progress.

When the TXRST* signal is deasserted (HIGH), AM* remains LOW to allow the FIFO status flags to be driven. This allows the completion of the reset operation to be monitored. To allow better multi-tasking on multi-PHY implementations, it is possible to deassert AM* (HIGH) as soon as the Full state is indicated. The FIFO reset operation will complete and the Empty state (indicating completion of the reset operation) can be detected during a separate polling operation.

For those links implemented with a single PHY, it is possible to tie AM* LOW and still perform normal accesses and reset operations, as shown in *Figure 15*. In a single-PHY implementation with AM* always LOW, a Transmit FIFO reset can never be initiated with TXEN* asserted at the same time as TXRST*. Since AM* is always LOW, any assertion of TXEN* prevents the reset sequence counter from advancing.

Figure 16 shows a sequence of input signals which does not produce a FIFO reset. In this case TXEN* was asserted to select the a Transmit FIFO for data transfers. Because TXEN* remains active, the assertion of AM* and TXRST* does not initiate a reset operation. This is shown by the TXFULL* flag remaining HIGH (deasserted) following what would be the normal expiration of the eight-state reset counter.

Receive FIFO Reset Sequence

The Receive FIFO reset sequence operates (for the most part) the same as the Transmit FIFO reset sequence. The same requirements exist for the assertion state of RXRST* and selection of the interface. A sample Receive FIFO reset sequence is shown in *Figure 17*. Upon recognition of a Receive FIFO reset, the Receive FIFO flags are forced to indicate an Empty state to prohibit additional reads from the FIFO. Unlike the Transmit FIFO, where the internal completion of the reset operation is shown by first going Full and later going Empty when the internal reset is complete, there is no secondary indication of the completion of the internal reset of the Receive FIFO. The Receive FIFO is usable as soon as new data is placed into it by the Receive Control State Machine.

When a Receive FIFO reset sequence is enabled and has been active for at least eight RXCLK cycles, a Receive FIFO reset operation is started. This FIFO reset operation is not allowed to progress within the device until the associated RXRST* is sampled deasserted (HIGH). Following deassertion of RXRST* (which starts the FIFO reset operation), selection of the device for normal data transfers is inhibited during

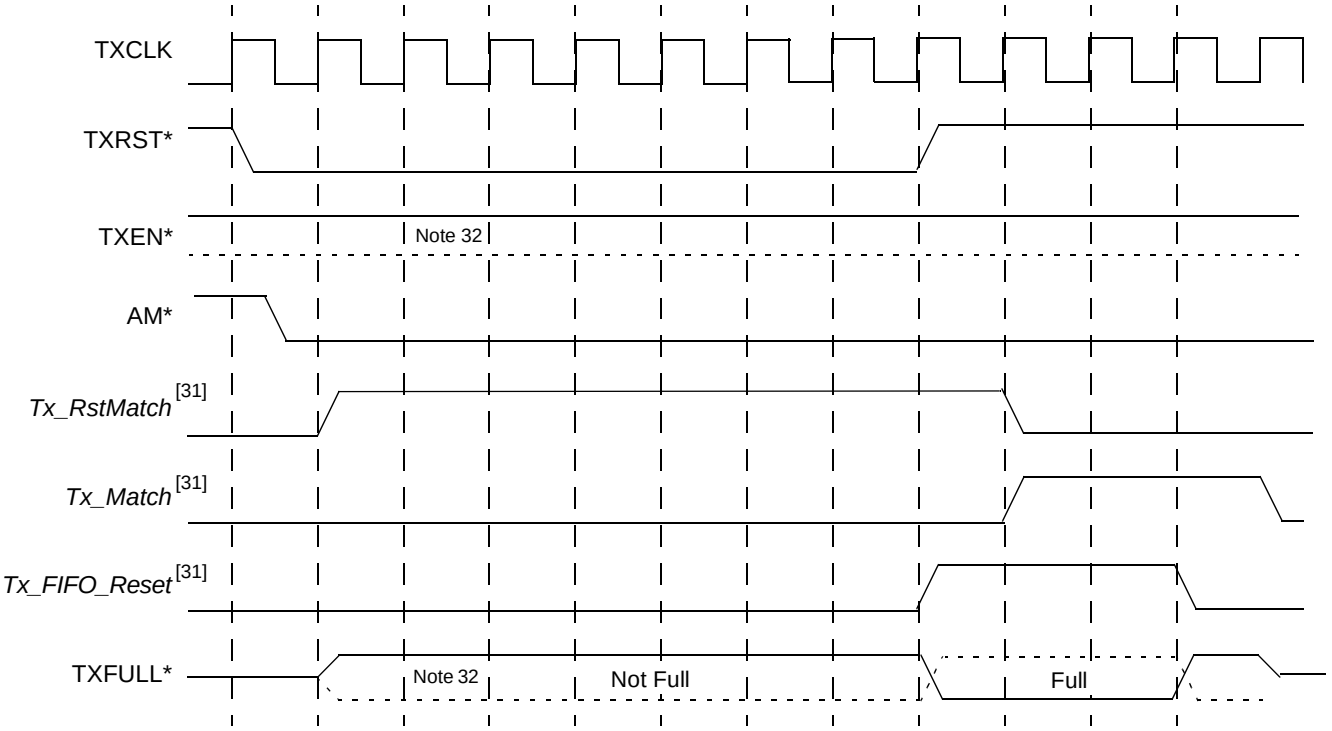


Figure 15. Transmit FIFO Reset Sequence with Constant AM*.

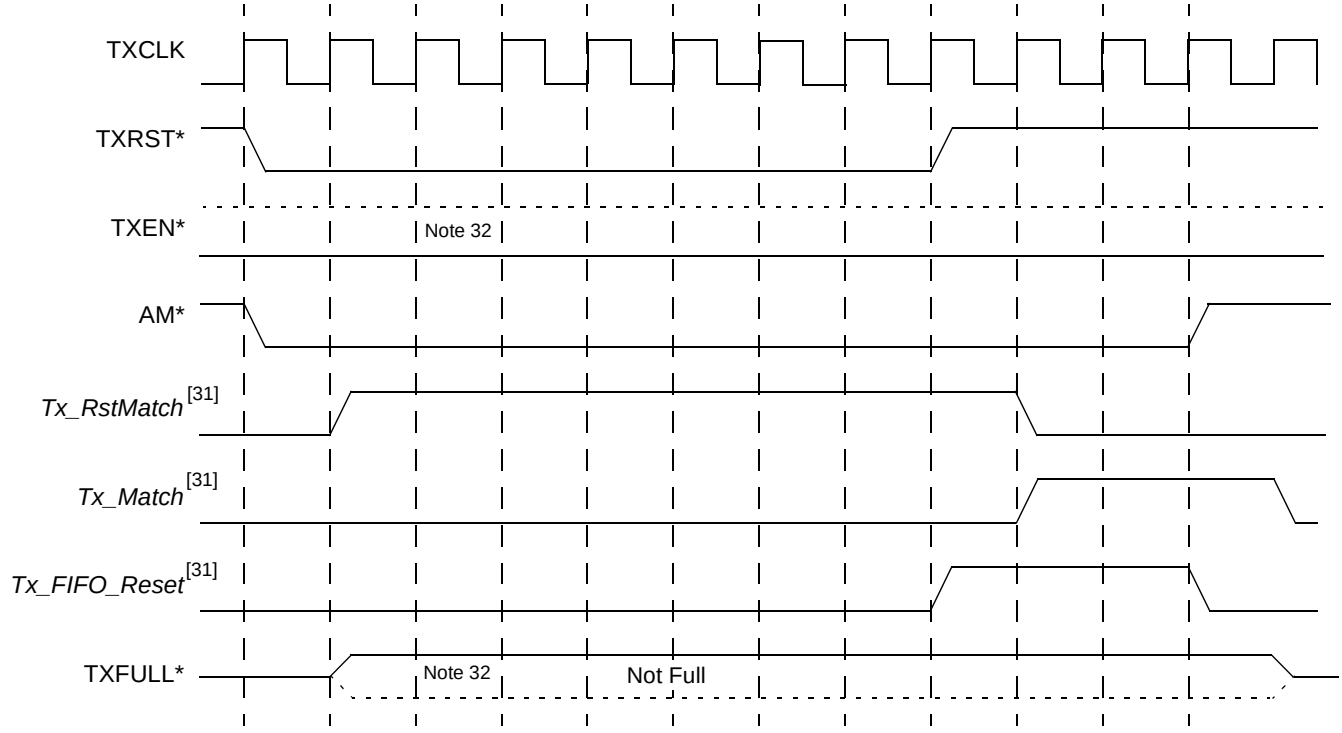


Figure 16. Invalid Transmit FIFO Reset Sequence with TXEN* Asserted.

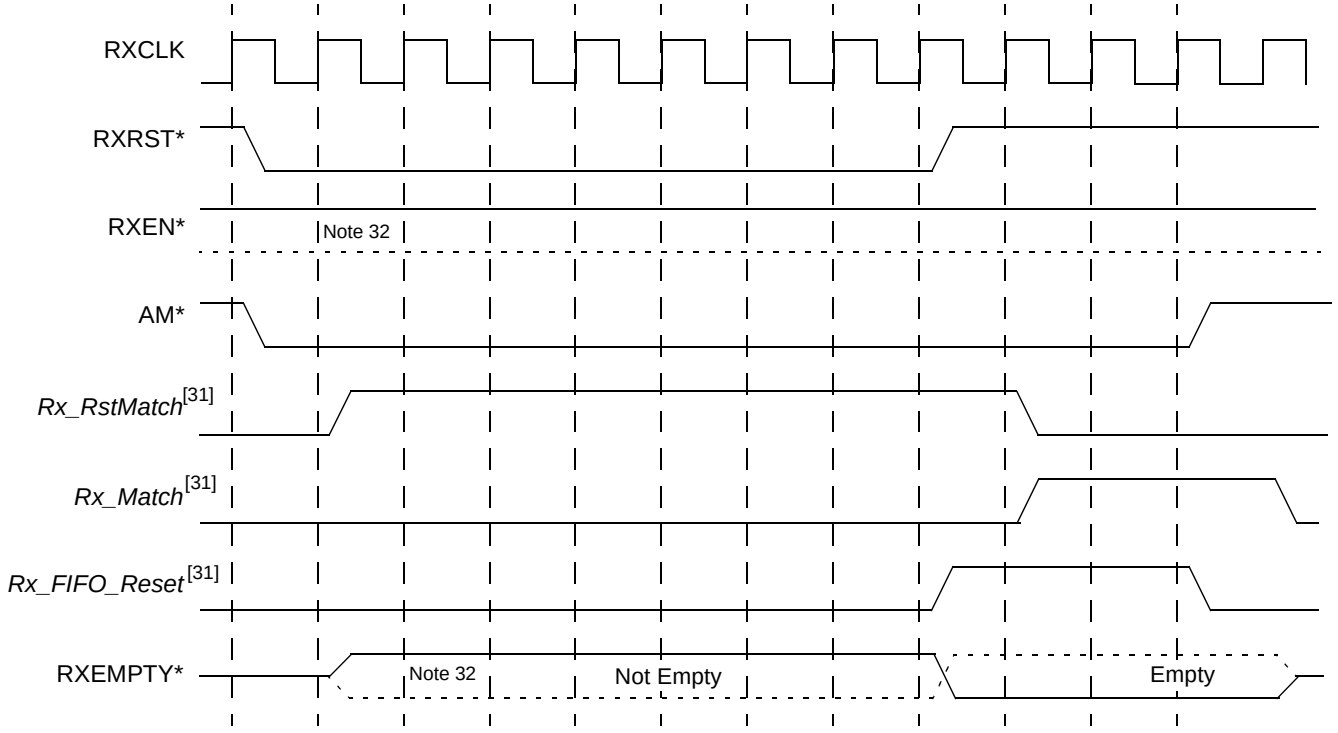


Figure 17. Receive FIFO Reset Sequence.

the immediately following RXCLK clock cycle. If a selection of the transmit interface is attempted during this immediately following cycle (by asserting RXEN*), the selection is ignored, and the device remains unselected until RXEN* is deasserted, and reasserted in a following RXCLK cycle.

Serial Address Register Access

The Serial Address Register in the CY7B924DX is accessed through the RXDATA bus. This register can be both written and read, and is accessed by asserting RXRST* to address the register in the device instead of the normal Receive FIFO data. Within this alternate address space, the RXRVS signal is an input at all times, and is used to select between read (RXRVS is HIGH) and write (RXRVS is LOW) operations on the Serial Address Register.

The Serial Address Register is the same size as the 8- or 10-bit data width selected by BYTE8/10*. It can be set to match domain or multicast addresses by the level on RXSC/D*. If RXS/D* is LOW when the Serial Address Register is written, it becomes the Multicast address register and declares a match if any single bit (or multiple bits) matches the equivalent bit in the incoming Address character. If RXSC/D* is HIGH when the Serial Address Register is written, it becomes the Unicast address register and defines a match only if all of the bits match the incoming Address character.

This register mapping is shown in *Figure 6*.

Accessing Serial Address Register

To access the Serial Address Register in the CY7B924DX, an Rx_RstMatch condition must first be generated by the combined assertion of AM* and RXRST*. RXEN* is then used as the data strobe signal to initiate either a read or write cycle to the RXDATA bus. If RXRVS = 1 (HIGH) at the time of the RXEN* data strobe, a register read operation takes place. If RXRVS = 0 (LOW) at the time of the RXEN* data strobe, a register write operation takes place.

The RXSC/D* input is used in conjunction with RXDATA[9:0] or RXDATA[7:0] to select the operational mode of the Serial Address Register (Unicast or Multicast).

Register write and read operations are shown in *Figure 18*. If the serial address register write and read operations are both performed without deasserting RXRST*, then RXRST* will still not extend to the eight-cycle requirement for a reset and the Receive FIFO will not be reset.

FIFO Reset and Continuous Selection

When configured for continuous selection (AM* asserted with TXEN* always enabled, or AM* asserted with RXEN* always enabled), it is not possible to reset the Transmit and Receive FIFOs. It is also not possible to write to the Serial Address Register without deselecting the Receive FIFO interface.

X3.230 Codes and Notation Conventions

Information to be transmitted over a serial link is encoded eight bits at a time into a 10-bit Transmission Character and then sent serially, bit by bit. Information received over a serial link is collected ten bits at a time, and those Transmission Characters that are used for data (Data Characters) are decoded into the correct eight-bit codes. The 10-bit Transmission Code supports all 256 8-bit combinations. Some of the remaining Transmission Characters (Special Characters) are used for functions other than data transmission.

The primary rationale for use of a Transmission Code is to improve the transmission characteristics of a serial link. The encoding defined by the Transmission Code ensures that suf-

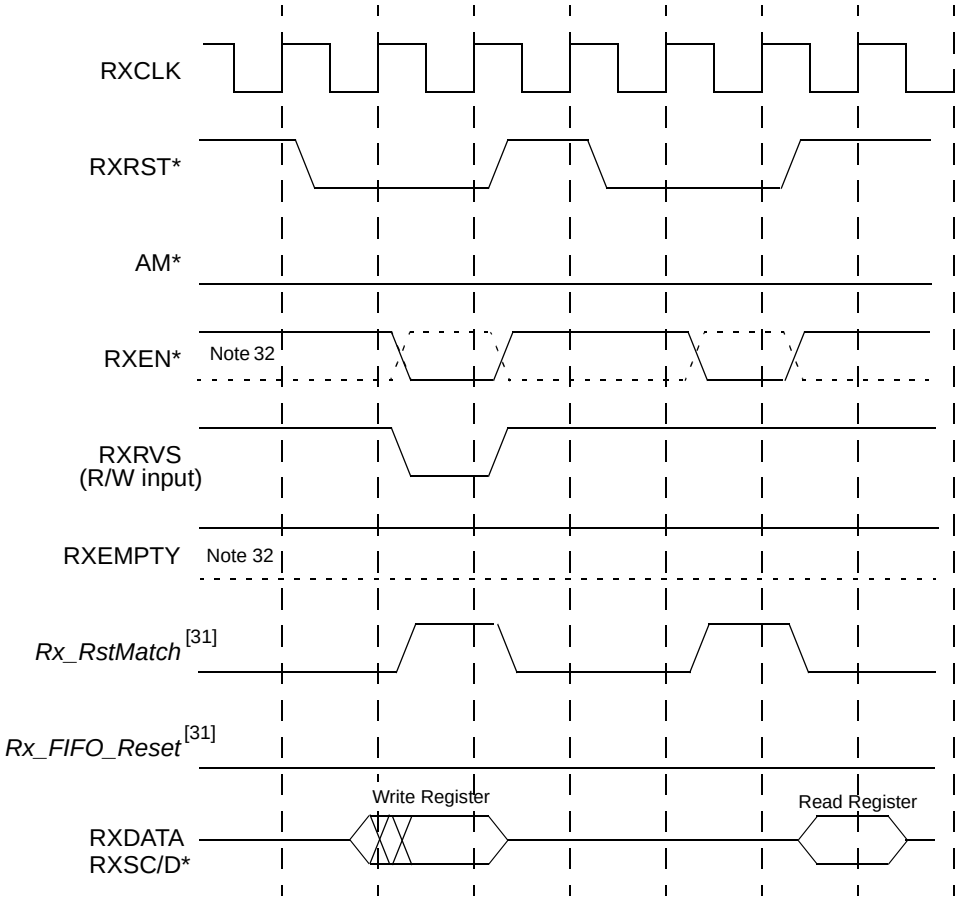


Figure 18. Serial Address Register Access.

ficient transitions are present in the serial bit stream to make clock recovery possible at the Receiver. Such encoding also greatly increases the likelihood of detecting any single or multiple bit errors that may occur during transmission and reception of information. In addition, some Special Characters of the Transmission Code selected by Fibre Channel Standard consist of a distinct and easily recognizable bit pattern (the Special Character Comma) that assists a Receiver in achieving word alignment on the incoming bit stream.

Notation Conventions

The documentation for the 8B/10B Transmission Code uses letter notation for the bits in an 8-bit byte. Fibre Channel Standard notation uses a bit notation of A, B, C, D, E, F, G, H for the 8-bit byte for the raw 8-bit data, and the letters a, b, c, d, e, i, f, g, h, j for encoded 10-bit data. There is a correspondence between bit A and bit a, B and b, C and c, D and d, E and e, F and f, G and g, and H and h. Bits i and j are derived, respectively, from (A,B,C,D,E) and (F,G,H).

The bit labeled A in the description of the 8B/10B Transmission Code corresponds to bit 0 in the numbering scheme of the FC-2 specification, B corresponds to bit 1, as shown here:

FC-2 bit designation—	7	6	5	4	3	2	1	0
HOTLink TX/RX designation—	7	6	5	4	3	2	1	0
8B/10B bit designation—	H	G	F	E	D	C	B	A

To clarify this correspondence, the following example shows the conversion from an FC-2 Valid Data Byte to a Transmission Character (using 8B/10B Transmission Code notation):

```
FC-2  45
      Bits: 7654 3210
            0100 0101
```

Converted to 8B/10B notation (note carefully that the order of bits is reversed):

```
Data Byte Name   D5.2
                 Bits: ABCDE  FGH
                       10100  010
```

Translated to a transmission Character in the 8B/10B Transmission Code:

```
      Bits: abcdeifghj
            1010010101
```

Each valid Transmission Character of the 8B/10B Transmission Code has been given a name using the following convention: cxx.y, where c is used to show whether the Transmission Character is a Data Character (c is set to D, and the SC/D* pin is LOW) or a Special Character (c is set to K, and the SC/D* pin is HIGH). When c is set to D, xx is the decimal value of the binary number composed of the bits E, D, C, B, and A in that order, and the y is the decimal value of the binary number composed of the bits H, G, and F in that order. When c is set to K, xx and y are derived by comparing the encoded bit patterns of the Special Character to

those patterns derived from encoded Valid Data bytes and selecting the names of the patterns most similar to the encoded bit patterns of the Special Character.

Under the above conventions, the Transmission Character used for the examples above, is referred to by the name D5.2. The Special Character K29.7 is so named because the first six bits (abcdei) of this character make up a bit pattern similar to that resulting from the encoding of the unencoded 11101 pattern (29), and because the second four bits (fghj) make up a bit pattern similar to that resulting from the encoding of the unencoded 111 pattern (7).

Note: This definition of the 10-bit Transmission Code is based on (and is in basic agreement with) the following references, which describe the same 10-bit transmission code.

A.X. Widmer and P.A. Franaszek. “A DC-Balanced, Partitioned-Block, 8B/10B Transmission Code” *IBM Journal of Research and Development*, 27, No. 5: 440-451 (September, 1983).

U.S. Patent 4,486,739. Peter A. Franaszek and Albert X. Widmer. “Byte-Oriented DC Balanced (0.4) 8B/10B Partitioned Block Transmission Code” (December 4, 1984).

Fibre Channel Physical and Signaling Interface (ANS X3.230–1994 ANSI FC–PH Standard).

IBM Enterprise Systems Architecture/390 ESCON I/O Interface (document number SA22–7202).

8B/10B Transmission Code

The following information describes how the tables are used for both generating valid Transmission Characters (encoding) and checking the validity of received Transmission Characters (decoding). It also specifies the ordering rules to be followed when transmitting the bits within a character and the characters within the higher-level constructs specified by the standard.

Transmission Order

Within the definition of the 8B/10B Transmission Code, the bit positions of the Transmission Characters are labeled a, b, c, d, e, i, f, g, h, j. Bit “a” is transmitted first followed by bits b, c, d, e, i, f, g, h, and j in that order. (Note that bit i is transmitted between bit e and bit f, rather than in alphabetical order.)

Valid and Invalid Transmission Characters

The following tables define the valid Data Characters and valid Special Characters (K characters), respectively. The tables are used for both generating valid Transmission Characters (encoding) and checking the validity of received Transmission Characters (decoding). In the tables, each Valid-Data-byte or Special-Character-code entry has two columns that represent two (not necessarily different) Transmission Characters. The two columns correspond to the current value of the running disparity (“Current RD–” or “Current RD+”). Running disparity is a binary parameter with either the value negative (–) or the value positive (+).

After powering on, the Transmitter may assume either a positive or negative value for its initial running disparity. Upon transmission of any Transmission Character, the transmitter will select the proper version of the Transmission Character based on the current running disparity value, and the Transmitter calculates a new value for its running disparity based on the contents of the transmitted character. Special Character codes C1.7 and C2.7 can be used to force the transmission of a specific Special Character with a specific running disparity as required for some special sequences in X3.230.

After powering on, the Receiver may assume either a positive or negative value for its initial running disparity. Upon reception of any Transmission Character, the Receiver decides whether the Transmission Character is valid or invalid according to the following rules and tables and calculates a new value for its Running Disparity based on the contents of the received character.

The following rules for running disparity are used to calculate the new running-disparity value for Transmission Characters that have been transmitted (Transmitter’s running disparity) and that have been received (Receiver’s running disparity).

Running disparity for a Transmission Character is calculated from sub-blocks, where the first six bits (abcdei) form one sub-block and the second four bits (fghj) form the other sub-block. Running disparity at the beginning of the 6-bit sub-block is the running disparity at the end of the previous Transmission Character. Running disparity at the beginning of the 4-bit sub-block is the running disparity at the end of the 6-bit sub-block. Running disparity at the end of the Transmission Character is the running disparity at the end of the 4-bit sub-block.

Running disparity for the sub-blocks is calculated as follows:

1. Running disparity at the end of any sub-block is positive if the sub-block contains more ones than zeros. It is also positive at the end of the 6-bit sub-block if the 6-bit sub-block is 000111, and it is positive at the end of the 4-bit sub-block if the 4-bit sub-block is 0011.
2. Running disparity at the end of any sub-block is negative if the sub-block contains more zeros than ones. It is also negative at the end of the 6-bit sub-block if the 6-bit sub-block is 111000, and it is negative at the end of the 4-bit sub-block if the 4-bit sub-block is 1100.
3. Otherwise, running disparity at the end of the sub-block is the same as at the beginning of the sub-block.

Use of the Tables for Generating Transmission Characters

The appropriate encoding for the Valid Data byte or the Special Character byte into a Transmission Character is listed in the character encoding *Tables 10* and *11*. The current value of the Transmitter’s running disparity is used to select the Transmission Character from its corresponding column. For each Transmission Character transmitted, a new value of the running disparity is calculated. This new value is used as the Transmitter’s current running disparity for the next Valid Data byte or Special Character byte to be encoded and transmitted. *Table 8* shows naming notations and examples of valid transmission characters.

Use of the Tables for Checking the Validity of Received Transmission Characters

The column corresponding to the current value of the Receiver’s running disparity is searched for the received Transmission Character. If the received Transmission Character is found in the proper column, then the Transmission Character is valid and the associated Data byte or Special Character code is determined (decoded). If the received Transmission Character is not found in that column, then the Transmission Character is invalid. This is called a code violation. Independent of the Transmission Character’s validity, the received Transmission Character is used to calculate a new value of

running disparity. The new value is used as the Receiver's current running disparity for the next received Transmission Character.

Table 8. Valid Transmission Characters

Data			
	TX_{IN} or RX_{OUT}		
Byte Name	765	43210	Hex Value
D0.0	000	00000	00
D1.0	000	00001	01
D2.0	000	00010	02
. .	. .	. .	. .
D5.2	010	000101	45
. .	. .	. .	. .
D30.7	111	11110	FE
D31.7	111	11111	FF

Detection of a code violation does not necessarily show that the Transmission Character in which the code violation was detected is in error. Code violations may result from a prior error that altered the running disparity of the bit stream which did not result in a detectable error at the Transmission Character in which the error occurred. *Table 9* shows an example of this behavior.

Table 9. Code Violations Resulting from Prior Errors

	RD	Character	RD	Character	RD	Character	RD
Transmitted data character	–	D21.1	–	D10.2	–	D23.5	+
Transmitted bit stream	–	101010 1001	–	010101 0101	–	111010 1010	+
Bit stream after error	–	101010 1011	+	010101 0101	+	111010 1010	+
Decoded data character	–	D21.0	+	D10.2	+	Code Violation	+

Table 10. Valid Data Characters (TXSC/D* = LOW, RXSC/D* = LOW)

Data Byte Name	Bits	Current RD–	Current RD+	Data Byte Name	Bits	Current RD–	Current RD+
	HGF EDCBA	abcdei fghj	abcdei fghj		HGF EDCBA	abcdei fghj	abcdei fghj
D0.0	000 00000	100111 0100	011000 1011	D0.1	001 00000	100111 1001	011000 1001
D1.0	000 00001	011101 0100	100010 1011	D1.1	001 00001	011101 1001	100010 1001
D2.0	000 00010	101101 0100	010010 1011	D2.1	001 00010	101101 1001	010010 1001
D3.0	000 00011	110001 1011	110001 0100	D3.1	001 00011	110001 1001	110001 1001
D4.0	000 00100	110101 0100	001010 1011	D4.1	001 00100	110101 1001	001010 1001
D5.0	000 00101	101001 1011	101001 0100	D5.1	001 00101	101001 1001	101001 1001
D6.0	000 00110	011001 1011	011001 0100	D6.1	001 00110	011001 1001	011001 1001
D7.0	000 00111	111000 1011	000111 0100	D7.1	001 00111	111000 1001	000111 1001
D8.0	000 01000	111001 0100	000110 1011	D8.1	001 01000	111001 1001	000110 1001
D9.0	000 01001	100101 1011	100101 0100	D9.1	001 01001	100101 1001	100101 1001
D10.0	000 01010	010101 1011	010101 0100	D10.1	001 01010	010101 1001	010101 1001
D11.0	000 01011	110100 1011	110100 0100	D11.1	001 01011	110100 1001	110100 1001
D12.0	000 01100	001101 1011	001101 0100	D12.1	001 01100	001101 1001	001101 1001
D13.0	000 01101	101100 1011	101100 0100	D13.1	001 01101	101100 1001	101100 1001
D14.0	000 01110	011100 1011	011100 0100	D14.1	001 01110	011100 1001	011100 1001
D15.0	000 01111	010111 0100	101000 1011	D15.1	001 01111	010111 1001	101000 1001
D16.0	000 10000	011011 0100	100100 1011	D16.1	001 10000	011011 1001	100100 1001
D17.0	000 10001	100011 1011	100011 0100	D17.1	001 10001	100011 1001	100011 1001
D18.0	000 10010	010011 1011	010011 0100	D18.1	001 10010	010011 1001	010011 1001
D19.0	000 10011	110010 1011	110010 0100	D19.1	001 10011	110010 1001	110010 1001
D20.0	000 10100	001011 1011	001011 0100	D20.1	001 10100	001011 1001	001011 1001
D21.0	000 10101	101010 1011	101010 0100	D21.1	001 10101	101010 1001	101010 1001
D22.0	000 10110	011010 1011	011010 0100	D22.1	001 10110	011010 1001	011010 1001
D23.0	000 10111	111010 0100	000101 1011	D23.1	001 10111	111010 1001	000101 1001
D24.0	000 11000	110011 0100	001100 1011	D24.1	001 11000	110011 1001	001100 1001
D25.0	000 11001	100110 1011	100110 0100	D25.1	001 11001	100110 1001	100110 1001
D26.0	000 11010	010110 1011	010110 0100	D26.1	001 11010	010110 1001	010110 1001
D27.0	000 11011	110110 0100	001001 1011	D27.1	001 11011	110110 1001	001001 1001
D28.0	000 11100	001110 1011	001110 0100	D28.1	001 11100	001110 1001	001110 1001
D29.0	000 11101	101110 0100	010001 1011	D29.1	001 11101	101110 1001	010001 1001
D30.0	000 11110	011110 0100	100001 1011	D30.1	001 11110	011110 1001	100001 1001
D31.0	000 11111	101011 0100	010100 1011	D31.1	001 11111	101011 1001	010100 1001

Table 10. Valid Data Characters (TXSC/D* = LOW, RXSC/D* = LOW) (continued)

Data Byte Name	Bits	Current RD–	Current RD+	Data Byte Name	Bits	Current RD–	Current RD+
	HGF EDCBA	abcdei fghj	abcdei fghj		HGF EDCBA	abcdei fghj	abcdei fghj
D0.2	010 00000	100111 0101	011000 0101	D0.3	011 00000	100111 0011	011000 1100
D1.2	010 00001	011101 0101	100010 0101	D1.3	011 00001	011101 0011	100010 1100
D2.2	010 00010	101101 0101	010010 0101	D2.3	011 00010	101101 0011	010010 1100
D3.2	010 00011	110001 0101	110001 0101	D3.3	011 00011	110001 1100	110001 0011
D4.2	010 00100	110101 0101	001010 0101	D4.3	011 00100	110101 0011	001010 1100
D5.2	010 00101	101001 0101	101001 0101	D5.3	011 00101	101001 1100	101001 0011
D6.2	010 00110	011001 0101	011001 0101	D6.3	011 00110	011001 1100	011001 0011
D7.2	010 00111	111000 0101	000111 0101	D7.3	011 00111	111000 1100	000111 0011
D8.2	010 01000	111001 0101	000110 0101	D8.3	011 01000	111001 0011	000110 1100
D9.2	010 01001	100101 0101	100101 0101	D9.3	011 01001	100101 1100	100101 0011
D10.2	010 01010	010101 0101	010101 0101	D10.3	011 01010	010101 1100	010101 0011
D11.2	010 01011	110100 0101	110100 0101	D11.3	011 01011	110100 1100	110100 0011
D12.2	010 01100	001101 0101	001101 0101	D12.3	011 01100	001101 1100	001101 0011
D13.2	010 01101	101100 0101	101100 0101	D13.3	011 01101	101100 1100	101100 0011
D14.2	010 01110	011100 0101	011100 0101	D14.3	011 01110	011100 1100	011100 0011
D15.2	010 01111	010111 0101	101000 0101	D15.3	011 01111	010111 0011	101000 1100
D16.2	010 10000	011011 0101	100100 0101	D16.3	011 10000	011011 0011	100100 1100
D17.2	010 10001	100011 0101	100011 0101	D17.3	011 10001	100011 1100	100011 0011
D18.2	010 10010	010011 0101	010011 0101	D18.3	011 10010	010011 1100	010011 0011
D19.2	010 10011	110010 0101	110010 0101	D19.3	011 10011	110010 1100	110010 0011
D20.2	010 10100	001011 0101	001011 0101	D20.3	011 10100	001011 1100	001011 0011
D21.2	010 10101	101010 0101	101010 0101	D21.3	011 10101	101010 1100	101010 0011
D22.2	010 10110	011010 0101	011010 0101	D22.3	011 10110	011010 1100	011010 0011
D23.2	010 10111	111010 0101	000101 0101	D23.3	011 10111	111010 0011	000101 1100
D24.2	010 11000	110011 0101	001100 0101	D24.3	011 11000	110011 0011	001100 1100
D25.2	010 11001	100110 0101	100110 0101	D25.3	011 11001	100110 1100	100110 0011
D26.2	010 11010	010110 0101	010110 0101	D26.3	011 11010	010110 1100	010110 0011
D27.2	010 11011	110110 0101	001001 0101	D27.3	011 11011	110110 0011	001001 1100
D28.2	010 11100	001110 0101	001110 0101	D28.3	011 11100	001110 1100	001110 0011
D29.2	010 11101	101110 0101	010001 0101	D29.3	011 11101	101110 0011	010001 1100
D30.2	010 11110	011110 0101	100001 0101	D30.3	011 11110	011110 0011	100001 1100
D31.2	010 11111	101011 0101	010100 0101	D31.3	011 11111	101011 0011	010100 1100

Table 10. Valid Data Characters (TXSC/D* = LOW, RXSC/D* = LOW) (continued)

Data Byte Name	Bits	Current RD–	Current RD+	Data Byte Name	Bits	Current RD–	Current RD+
	HGF EDCBA	abcdei fghj	abcdei fghj		HGF EDCBA	abcdei fghj	abcdei fghj
D0.4	100 00000	100111 0010	011000 1101	D0.5	101 00000	100111 1010	011000 1010
D1.4	100 00001	011101 0010	100010 1101	D1.5	101 00001	011101 1010	100010 1010
D2.4	100 00010	101101 0010	010010 1101	D2.5	101 00010	101101 1010	010010 1010
D3.4	100 00011	110001 1101	110001 0010	D3.5	101 00011	110001 1010	110001 1010
D4.4	100 00100	110101 0010	001010 1101	D4.5	101 00100	110101 1010	001010 1010
D5.4	100 00101	101001 1101	101001 0010	D5.5	101 00101	101001 1010	101001 1010
D6.4	100 00110	011001 1101	011001 0010	D6.5	101 00110	011001 1010	011001 1010
D7.4	100 00111	111000 1101	000111 0010	D7.5	101 00111	111000 1010	000111 1010
D8.4	100 01000	111001 0010	000110 1101	D8.5	101 01000	111001 1010	000110 1010
D9.4	100 01001	100101 1101	100101 0010	D9.5	101 01001	100101 1010	100101 1010
D10.4	100 01010	010101 1101	010101 0010	D10.5	101 01010	010101 1010	010101 1010
D11.4	100 01011	110100 1101	110100 0010	D11.5	101 01011	110100 1010	110100 1010
D12.4	100 01100	001101 1101	001101 0010	D12.5	101 01100	001101 1010	001101 1010
D13.4	100 01101	101100 1101	101100 0010	D13.5	101 01101	101100 1010	101100 1010
D14.4	100 01110	011100 1101	011100 0010	D14.5	101 01110	011100 1010	011100 1010
D15.4	100 01111	010111 0010	101000 1101	D15.5	101 01111	010111 1010	101000 1010
D16.4	100 10000	011011 0010	100100 1101	D16.5	101 10000	011011 1010	100100 1010
D17.4	100 10001	100011 1101	100011 0010	D17.5	101 10001	100011 1010	100011 1010
D18.4	100 10010	010011 1101	010011 0010	D18.5	101 10010	010011 1010	010011 1010
D19.4	100 10011	110010 1101	110010 0010	D19.5	101 10011	110010 1010	110010 1010
D20.4	100 10100	001011 1101	001011 0010	D20.5	101 10100	001011 1010	001011 1010
D21.4	100 10101	101010 1101	101010 0010	D21.5	101 10101	101010 1010	101010 1010
D22.4	100 10110	011010 1101	011010 0010	D22.5	101 10110	011010 1010	011010 1010
D23.4	100 10111	111010 0010	000101 1101	D23.5	101 10111	111010 1010	000101 1010
D24.4	100 11000	110011 0010	001100 1101	D24.5	101 11000	110011 1010	001100 1010
D25.4	100 11001	100110 1101	100110 0010	D25.5	101 11001	100110 1010	100110 1010
D26.4	100 11010	010110 1101	010110 0010	D26.5	101 11010	010110 1010	010110 1010
D27.4	100 11011	110110 0010	001001 1101	D27.5	101 11011	110110 1010	001001 1010
D28.4	100 11100	001110 1101	001110 0010	D28.5	101 11100	001110 1010	001110 1010
D29.4	100 11101	101110 0010	010001 1101	D29.5	101 11101	101110 1010	010001 1010
D30.4	100 11110	011110 0010	100001 1101	D30.5	101 11110	011110 1010	100001 1010
D31.4	100 11111	101011 0010	010100 1101	D31.5	101 11111	101011 1010	010100 1010

Table 10. Valid Data Characters (TXSC/D* = LOW, RXSC/D* = LOW) (continued)

Data Byte Name	Bits	Current RD–	Current RD+	Data Byte Name	Bits	Current RD–	Current RD+
	HGF EDCBA	abcdei fghj	abcdei fghj		HGF EDCBA	abcdei fghj	abcdei fghj
D0.6	110 00000	100111 0110	011000 0110	D0.7	111 00000	100111 0001	011000 1110
D1.6	110 00001	011101 0110	100010 0110	D1.7	111 00001	011101 0001	100010 1110
D2.6	110 00010	101101 0110	010010 0110	D2.7	111 00010	101101 0001	010010 1110
D3.6	110 00011	110001 0110	110001 0110	D3.7	111 00011	110001 1110	110001 0001
D4.6	110 00100	110101 0110	001010 0110	D4.7	111 00100	110101 0001	001010 1110
D5.6	110 00101	101001 0110	101001 0110	D5.7	111 00101	101001 1110	101001 0001
D6.6	110 00110	011001 0110	011001 0110	D6.7	111 00110	011001 1110	011001 0001
D7.6	110 00111	111000 0110	000111 0110	D7.7	111 00111	111000 1110	000111 0001
D8.6	110 01000	111001 0110	000110 0110	D8.7	111 01000	111001 0001	000110 1110
D9.6	110 01001	100101 0110	100101 0110	D9.7	111 01001	100101 1110	100101 0001
D10.6	110 01010	010101 0110	010101 0110	D10.7	111 01010	010101 1110	010101 0001
D11.6	110 01011	110100 0110	110100 0110	D11.7	111 01011	110100 1110	110100 1000
D12.6	110 01100	001101 0110	001101 0110	D12.7	111 01100	001101 1110	001101 0001
D13.6	110 01101	101100 0110	101100 0110	D13.7	111 01101	101100 1110	101100 1000
D14.6	110 01110	011100 0110	011100 0110	D14.7	111 01110	011100 1110	011100 1000
D15.6	110 01111	010111 0110	101000 0110	D15.7	111 01111	010111 0001	101000 1110
D16.6	110 10000	011011 0110	100100 0110	D16.7	111 10000	011011 0001	100100 1110
D17.6	110 10001	100011 0110	100011 0110	D17.7	111 10001	100011 0111	100011 0001
D18.6	110 10010	010011 0110	010011 0110	D18.7	111 10010	010011 0111	010011 0001
D19.6	110 10011	110010 0110	110010 0110	D19.7	111 10011	110010 1110	110010 0001
D20.6	110 10100	001011 0110	001011 0110	D20.7	111 10100	001011 0111	001011 0001
D21.6	110 10101	101010 0110	101010 0110	D21.7	111 10101	101010 1110	101010 0001
D22.6	110 10110	011010 0110	011010 0110	D22.7	111 10110	011010 1110	011010 0001
D23.6	110 10111	111010 0110	000101 0110	D23.7	111 10111	111010 0001	000101 1110
D24.6	110 11000	110011 0110	001100 0110	D24.7	111 11000	110011 0001	001100 1110
D25.6	110 11001	100110 0110	100110 0110	D25.7	111 11001	100110 1110	100110 0001
D26.6	110 11010	010110 0110	010110 0110	D26.7	111 11010	010110 1110	010110 0001
D27.6	110 11011	110110 0110	001001 0110	D27.7	111 11011	110110 0001	001001 1110
D28.6	110 11100	001110 0110	001110 0110	D28.7	111 11100	001110 1110	001110 0001
D29.6	110 11101	101110 0110	010001 0110	D29.7	111 11101	101110 0001	010001 1110
D30.6	110 11110	011110 0110	100001 0110	D30.7	111 11110	011110 0001	100001 1110
D31.6	110 11111	101011 0110	010100 0110	D31.7	111 11111	101011 0001	010100 1110

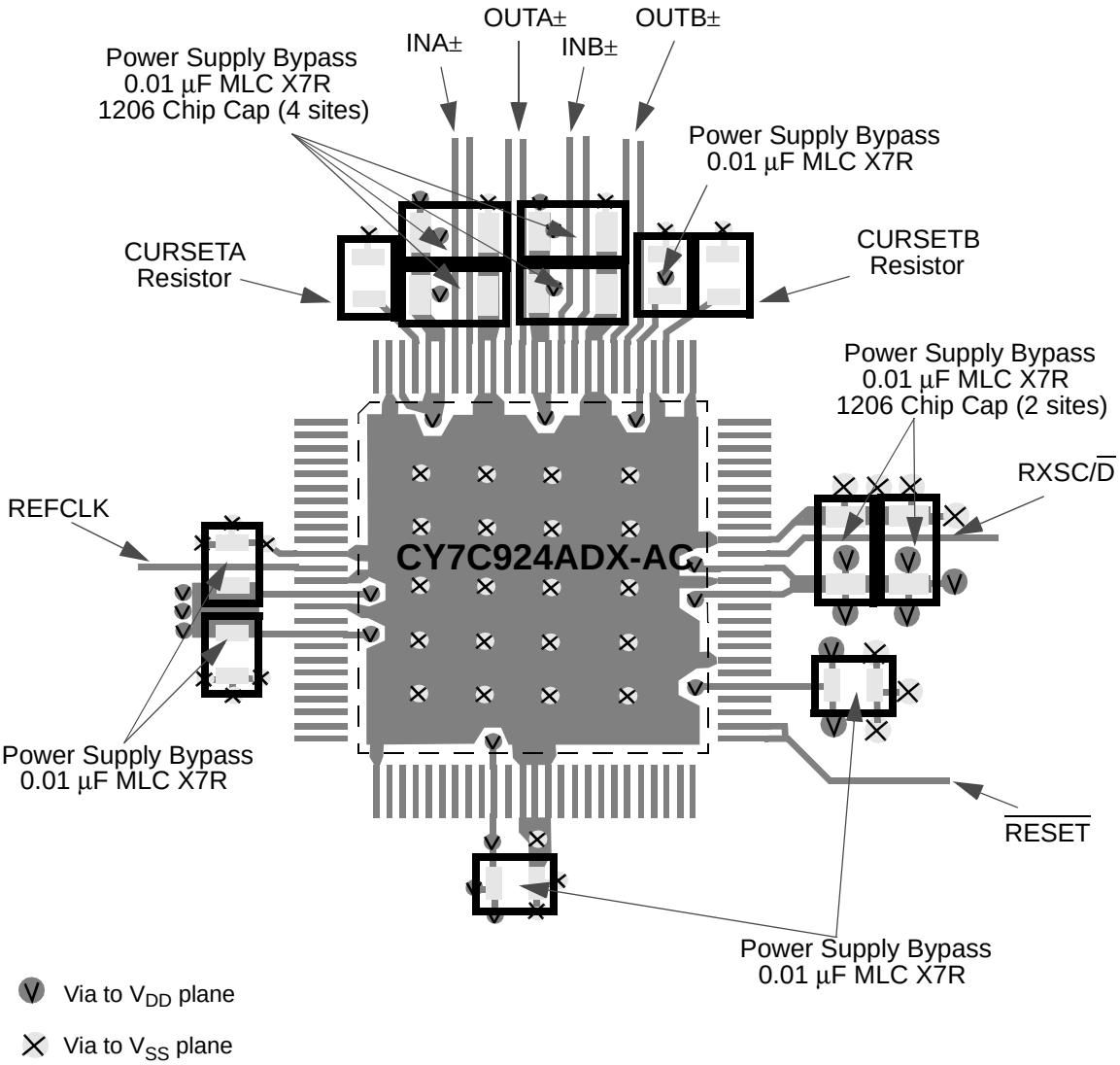
Table 11. Valid Special Character Codes and Sequences (TXSC/D* = HIGH or RXSC/D* = HIGH)[33, 34]

S.C. Byte Name	S.C. Code Name		Bits		Current RD–		Current RD+	
			HGF	EDCBA	abcdei	fghj	abcdei	fghj
K28.0	C0.0[35]	(C00)	000	00000	001111	0100	110000	1011
K28.1	C1.0[36]	(C01)	000	00001	001111	1001	110000	0110
K28.2	C2.0[36]	(C02)	000	00010	001111	0101	110000	1010
K28.3	C3.0[35]	(C03)	000	00011	001111	0011	110000	1100
K28.4	C4.0[36]	(C04)	000	00100	001111	0010	110000	1101
K28.5	C5.0[36, 37]	(C05)	000	00101	001111	1010	110000	0101
K28.6	C6.0[36]	(C06)	000	00110	001111	0110	110000	1001
K28.7	C7.0[36, 38]	(C07)	000	00111	001111	1000	110000	0111
K23.7	C8.0[35]	(C08)	000	01000	111010	1000	000101	0111
K27.7	C9.0[35]	(C09)	000	01001	110110	1000	001001	0111
K29.7	C10.0[35]	(C0A)	000	01010	101110	1000	010001	0111
K30.7	C11.0	(C0B)	000	01011	011110	1000	100001	0111
					End of Frame Sequence			
EOFxx	C2.1[39]	(C22)	001	00010	–K28.5,Dn.xxx0		+K28.5,Dn.xxx1	
					Code Rule Violation and SVS Tx Pattern			
Exception	C0.7[38, 40]	(CE0)	111	00000	100111	1000	011000	0111
–K28.5	C1.7[41]	(CE1)	111	00001	001111	1010	001111	1010
+K28.5	C2.7[42]	(CE2)	111	00010	110000	0101	110000	0101
					Running Disparity Violation Pattern			
Exception	C4.7[43]	(CE4)	111	00100	110111	0101	001000	1010

Notes:

33. All codes not shown are reserved.
34. Notation for Special Character Code Name is consistent with Fibre Channel and ESCON naming conventions. Special Character Code Name is intended to describe binary information present on I/O pins. Common usage for the name can either be in the form used for describing Data patterns (i.e., C0.0 through C31.7), or in hex notation (i.e., Cnn where nn = the specified value between 00h and FFh).
35. These characters have reserved meanings when command processing is enabled. This includes all operating modes where the FIFOs are enabled and the discard policy is not 0.
36. These characters are used for control of ESCON interfaces. They can be sent as embedded commands or other markers when not operating using ESCON protocols.
37. The K28.5 character is used for framing operations by the receiver. It is also the pad or fill character transmitted to maintain the serial link when no user data is available.
38. Care must be taken when using this Special Character code. When a C7.0 is followed by a D11.x or D20.x, or when an SVS (C0.7) is followed by a D11.x, and alias K28.5 sync character is created. These sequences can cause erroneous framing and should be avoided while RFEN is HIGH.
39. C2.1 = Transmit either –K28.5+ or +K28.5– as determined by Current RD and modify the following Transmission Character by setting its least significant bit to 1 or 0. If Current RD at the start of the following character is plus (+) the LSB is set to 0, and if Current RD is minus (–) the LSB becomes 1. This modification allows construction of X3.230:1994 “EOF” frame delimiters wherein the second data byte is determined by the Current RD.
For example, to send “EOFdt” the controller could issue the sequence C2.1–D21.4– D21.4–D21.4, and the HOTLink Transmitter will send either K28.5–D21.4–D21.4–D21.4 or K28.5–D21.5– D21.4–D21.4 based on Current RD. Likewise to send “EOFdti” the controller could issue the sequence C2.1–D10.4–D21.4–D21.4, and the HOTLink Transmitter will send either K28.5–D10.4–D21.4– D21.4 or K28.5–D10.5–D21.4– D21.4 based on Current RD.
The receiver will never output this Special Character, since K28.5 is decoded as C5.0, C1.7, or C2.7, and the subsequent bytes are decoded as data.
40. C0.7 = Transmit a deliberate code rule violation. The code chosen for this function follows the normal Running Disparity rules. Transmission of this Special Character has the same effect as asserting TXSVS = HIGH. The receiver outputs this Special Character only if the Transmission Character being decoded is not found in the tables.
41. C1.7 = Transmit Negative K28.5 (–K28.5+) disregarding Current RD. The receiver will only output this Special Character if K28.5 is received with the wrong running disparity. The receiver will output C1.7 if –K28.5 is received with RD+, otherwise K28.5 is decoded as C5.0 or C2.7.
42. C2.7 = Transmit Positive K28.5 (+K28.5–) disregarding Current RD. The receiver will only output this Special Character if K28.5 is received with the wrong running disparity. The receiver will output C2.7 if +K28.5 is received with RD–, otherwise K28.5 is decoded as C5.0 or C1.7.
43. C4.7 = Transmit a deliberate code rule violation to indicate a Running Disparity violation. The receiver will only output this Special Character if the Transmission Character being decoded is found in the tables, but Running Disparity does not match. This may indicate that an error occurred in a prior byte.

Printed Circuit Board Layout Suggestions



This is a typical printed circuit board layout showing example placement of power supply bypass components and other components mounted on the same side as the CY7C924ADX.

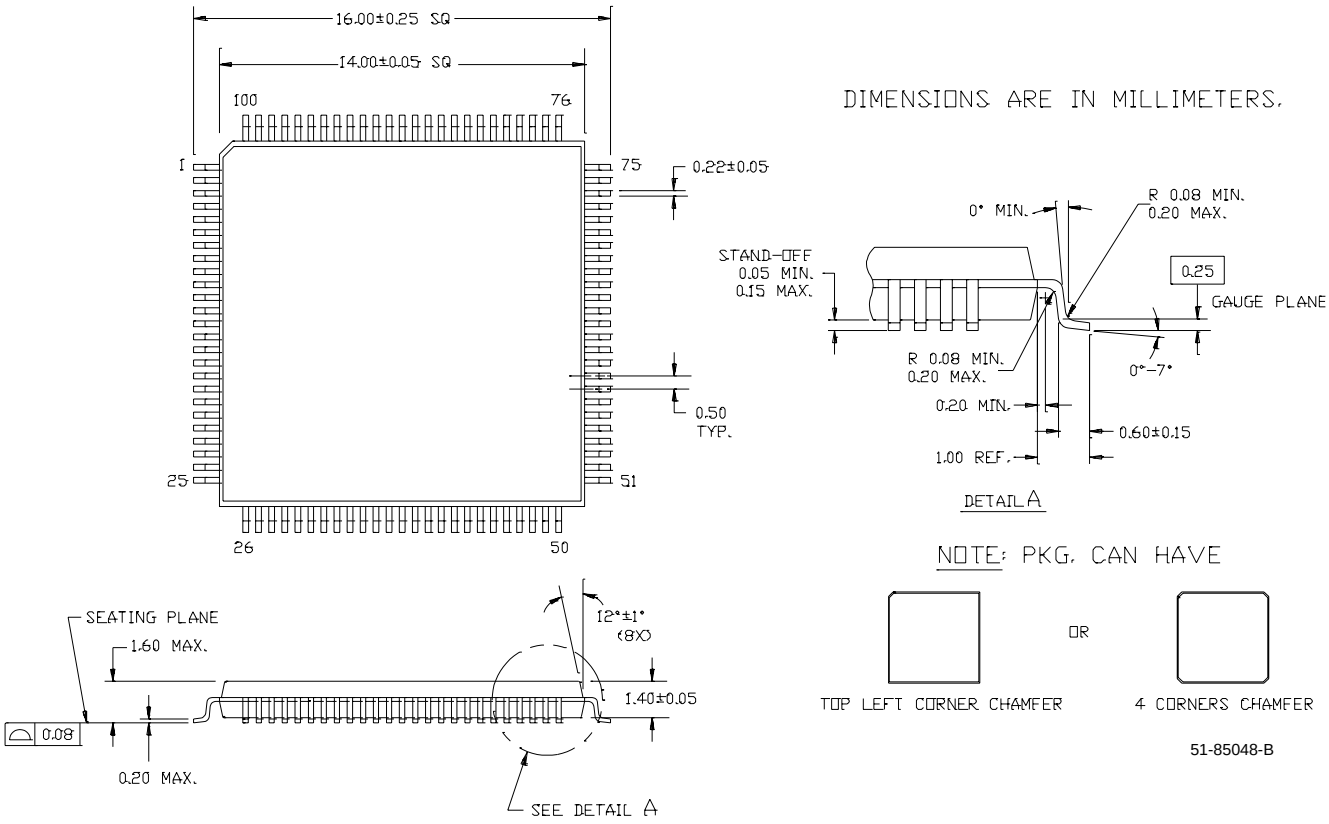
Other layouts, including cases with components mounted on the reverse side would work as well.

Ordering Information

Ordering Code	Package Name	Package Type	Operating Range
CY7C924ADX-AC	A100	100-Lead Thin Quad Flat Pack	Commercial
CY7C924ADX-AI	A100	100-Lead Thin Quad Flat Pack	Industrial

Package Diagram

100-Pin Thin Plastic Quad Flat Pack (TQFP) A100



Document Title: CY7C924ADX 200-MBaud HOTLink® Transceiver Document Number: 38-02008				
REV.	**ECN NO.**	**Issue Date**	**Orig. of Change**	**Description of Change**
**	105846	03/26/01	SZV	Change from Spec number: 38-00770 to 38-02008
*A	107878	07/09/01	KET	Changed part number: CY7C924DX to CY7C924ADX