



WB_IRCODER Infrared Encoder


Summary

Core Reference
CR0183 (v2.0) March 06, 2008

This document provides detailed reference information with respect to the WB_IRCODER peripheral device. This device is used to encode and modulate remote controller codes for infrared transmission.

The WB_IRCODER peripheral provides the interface between an infrared transmitter and a processor in an FPGA design. The peripheral has been built primarily to interface to the TFDU6102 Fast Infrared Transceiver (from Vishay Semiconductor) found on Altium's Ethernet-USB-IrDA Peripheral Board PB03. However, it could be used to interface to any IR transmitter where the required input signal is already modulated and simply controls the pulsing of the transmitter's IRLED.

The peripheral encodes remote control codes using the NEC IR transmission protocol and handles the modulation of the signal at the carrier frequency associated with this protocol – 38kHz.

 For information on the WB_IRDEC peripheral, used to process modulated IR data transmitted by a remote control device, refer to the [CR0173 WB_IRDEC Infrared Decoder](#) core reference.

Features

- Encoding of data using NEC IR transmission protocol
- Fixed carrier frequency for modulation of 38kHz
- 32-bit data interface
- Wishbone-compliant

Available Devices

From a schematic document, the WB_IRCODER device can be found in the FPGA Peripherals integrated library (FPGA Peripherals.IntLib), located in the \Library\Fpga folder of the installation.

From an OpenBus System document, the IR Encoder component can be found in the **Peripherals** region of the **OpenBus Palette** panel.

Infrared Communication Background

Before discussing the actual WB_IRCODER peripheral in detail – including its functional and hardware descriptions – it is worth taking a look at how remote control using infrared actually works. A closer look at the encoding schemes used and, in particular, the NEC IR transmission protocol is also a good idea.

IR – an Overview

Infrared remote control devices are abundant in today's gadget-filled world. From the television and video recorder, through the Hi-Fi and on to the garage door that thankfully opens remotely on a rainy day, a remote controller of one form or another is never far from reach.

Why use infrared light to send the control signals? Two reasons in particular stand out. The first is that the diodes used to emit infrared light are quite inexpensive and readily available. The second is the fact that infrared light is at a wavelength outside of the spectrum of visible light – so we can point and shoot our controllers and not get blinded in the process!

So how exactly does infrared remote control work? At the most basic level, the remote controller contains a transmitter circuit, part of which will be an Infrared Light Emitting Diode (IRLED). When a key is pressed on the controller, the command is sent as an IR signal to the device which you are aiming the controller at. The device being controlled will have a receiver circuit, part of which will be a photodiode with which to detect the IR signal and convert it into an electric current.

That's a very simplistic view of IR RC communications. However, when you factor in background infrared "noise" emitted by other heat-generating objects and multiple IR remote-controlled devices located in close proximity to each other, things quickly become more complicated. With simple infrared light, there is now potential for the command not getting to the receiver at all, let alone the receiver in the intended device.

Modulation and Methods of Encoding

To ensure a transmitted IR signal gets to its correct destination, or conversely the target device receives only the signal it is meant to, modulation is used. IR remote control systems utilize Pulse Code Modulation (PCM), where the modulating carrier frequency typically resides in the range 30kHz to 58kHz.

In terms of transmission, modulation means turning the IRLED on and off rapidly in bursts of the carrier frequency. The receiver will typically be tuned to this carrier frequency, ensuring that it receives only the signal required. It then uses this frequency to demodulate the signal.

When the IRLED is not emitting light, the transmitter is in the OFF state, which in terms of the signal is referred to as a 'space'. During IRLED activity, where the light is emitted in pulsed fashion at the carrier frequency, the transmitter is in the ON state, which is referred to as a pulse or 'mark'. At the receiver, a 'space' is output as a High, while a mark is output as a Low.

These spaces and marks are not the '0's and '1's of the command being transmitted, however. The actual data to be sent from the controller is encoded. The method of encoding used determines how to represent the '1's and '0's in terms of the marks and spaces. The following three methods of encoding are typically used in IR remote control systems.

Pulse Distance Encoding

In this method of encoding, the length of the pulse burst (mark) is always the same, but the time between consecutive bursts differs, depending on whether a logical '0' or logical '1' is being transmitted. The time taken to transmit a logical '1' is longer (i.e. transmitter OFF for longer time after the IR burst).

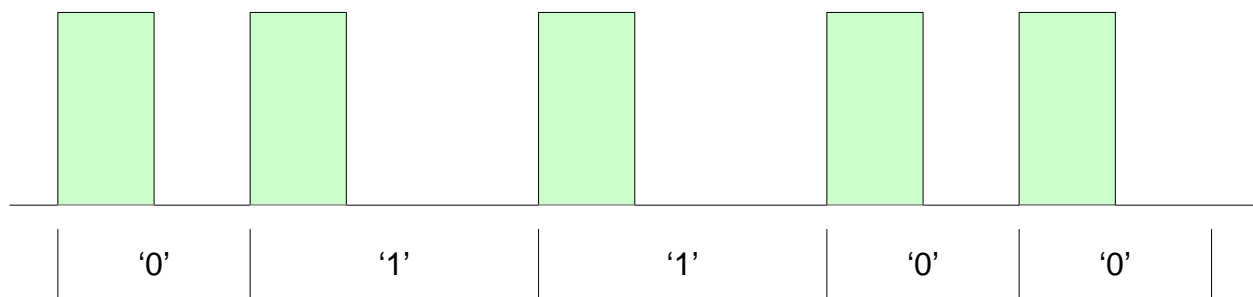


Figure 1. Example of pulse distance encoding.

Pulse Length Encoding

In this method of encoding, the length of the pulse burst (mark) is different for a logical '0' and a logical '1', with logical '1' requiring a longer burst.

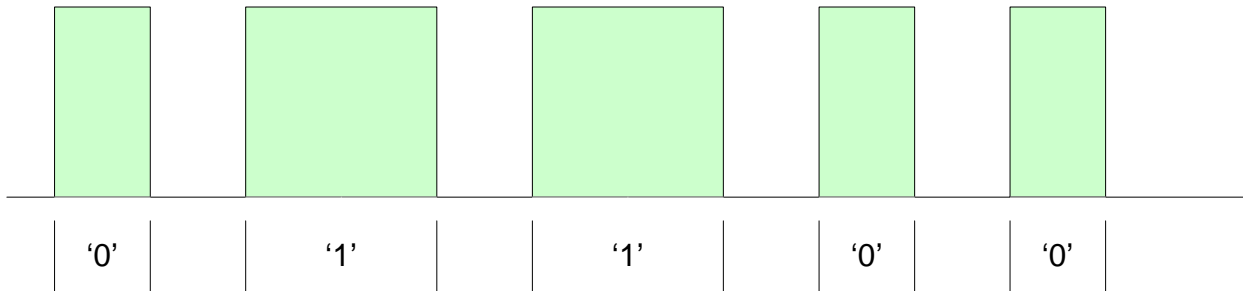


Figure 2. Example of pulse length encoding.

Manchester Encoding

In this method of encoding, all bits are of equal length, with half of the bit-period being a pulse burst (mark) and the other half being a space. A logical '0' is represented by a burst in the first half of the bit-period and a space in the second, giving a mid-period transition from High to Low. A logical '1' is represented by a space in the first half of the bit-period and a burst in the second, giving a mid-period transition from Low to High.

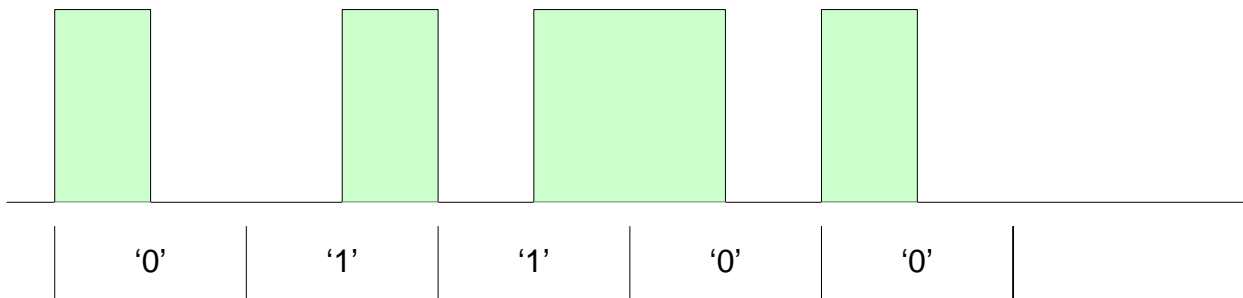


Figure 3. Example of Manchester (or Bi-phase) encoding.

The modulating carrier frequency and method of encoding are a base consideration for any IR RC transmission. The actual format of the transmitted message frame itself varies between manufacturers however. For example, there may be differing numbers of address and command bits, additional pulse bursts before and/or after the address and command bits, built-in error-checking, and so on.

Each of these different encoded message formats can be referred to as distinct infrared transmission protocols. In the next section, we take a closer look at the NEC infrared transmission protocol. This is the protocol used for transmission of commands by the WB_IRCODER, as well as the Altium Remote Controller.

NEC Infrared Transmission Protocol

The NEC IR transmission protocol uses pulse distance encoding of the message bits. Each pulse burst (mark – RC transmitter ON) is 562.5 μ s in length, at a carrier frequency of 38kHz (26.3 μ s). Logical bits are transmitted as follows:

- Logical '0' – a 562.5 μ s pulse burst followed by a 562.5 μ s space, with a total transmit time of 1.125ms
- Logical '1' – a 562.5 μ s pulse burst followed by a 1.6875ms space, with a total transmit time of 2.25ms

When a key is pressed on the remote controller, the message transmitted consists of the following, in order:

- a 9ms leading pulse burst (16 times the pulse burst length used for a logical data bit)
- a 4.5ms space
- the 8-bit address for the receiving device
- the 8-bit logical inverse of the address
- the 8-bit command
- the 8-bit logical inverse of the command
- a final 562.5 μ s pulse burst to signify the end of message transmission.

The four bytes of data bits are each sent least significant bit first. Figure 4 illustrates the format of an NEC IR transmission frame, for an address of 00h (00000000b) and a command of ADh (10101101b).

WB_IRCODER Infrared Encoder

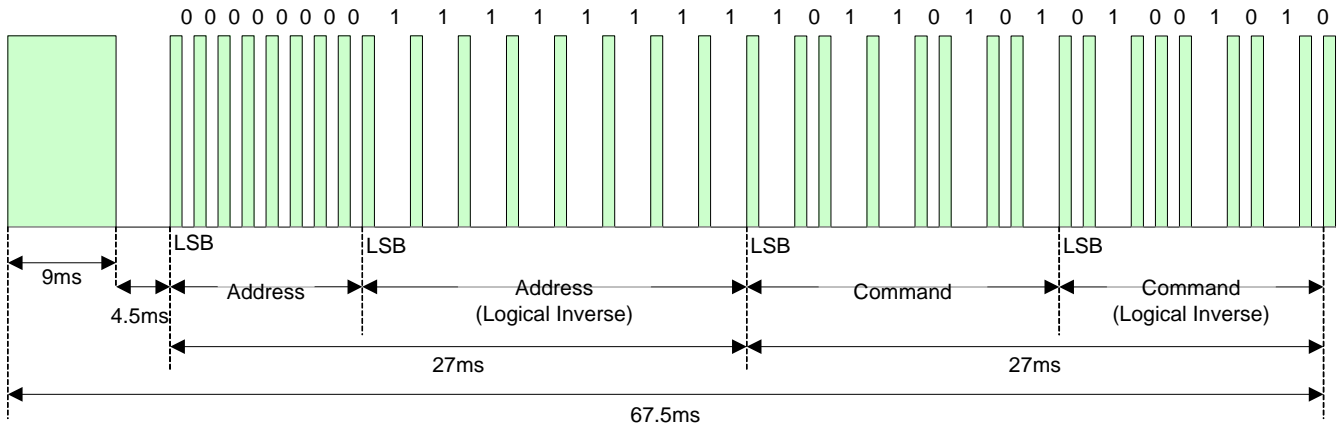


Figure 4. Example message frame using the NEC IR transmission protocol.

Notice from Figure 4 that it takes:

- 27ms to transmit both the 16 bits for the address (address + inverse). This comes from each of the 16 bit blocks ultimately containing eight '0's and eight '1's – giving $(8 * 1.125ms) + (8 * 2.25ms)$.
- 67.5ms to fully transmit the actual message frame (discounting the final 562.5µs pulse burst that signifies the end of message).

Repeat Codes

If the key on the remote controller is kept depressed, a repeat code will be issued, typically around 40ms after the pulse burst that signified the end of the message. A repeat code will continue to be sent out at 108ms intervals, until the key is finally released. The repeat code consists of the following, in order:

- a 9ms leading pulse burst
- a 2.25ms space
- a 562.5µs pulse burst to mark the end of the space (and hence end of the transmitted repeat code).

Figure 5 illustrates the transmission of two repeat codes after an initial message frame is sent.

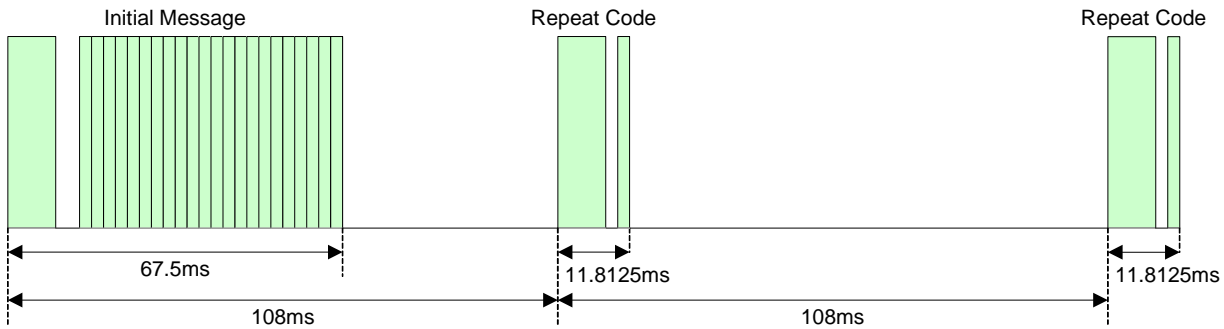


Figure 5. Example transmission of repeat codes using the NEC IR transmission protocol.

Note: Some of the timing values used by the WB_IRCODER when encoding a message frame in NEC format differ slightly from those of the protocol itself. These are:

- a pulse burst length of 560µs is used
- a transmit time of 1.12ms for a logical '0' is used
- a value of 110ms for the repeat code interval is used.

Functional Description

Symbol

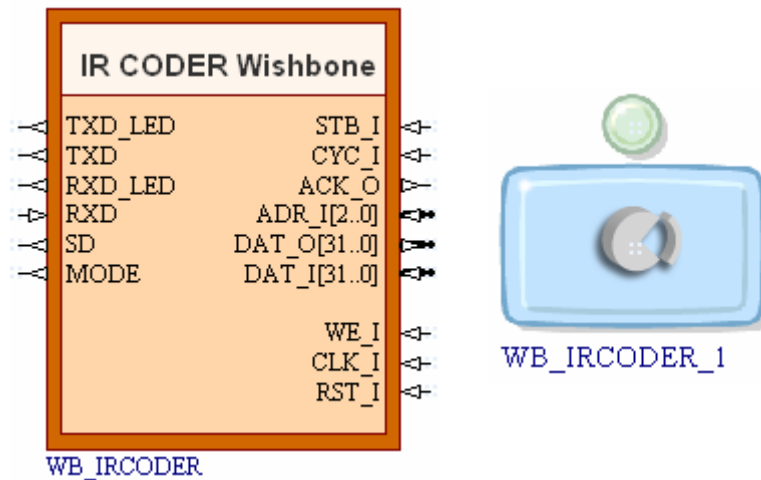


Figure 6. Symbols used for the Infrared Encoder in both schematic (left) and OpenBus System (right).

Pin Description

The following pin description is for the device when used on the schematic. In an OpenBus System, although the same signals are present, the abstract nature of the system hides the pin-level Wishbone interfaces. The external interface signals to the IR Transceiver will be made available as sheet entries, associated with the parent sheet symbol used to reference the underlying OpenBus System.

Table 1. WB_IRCODER pin description

Name	Type	Polarity/ Bus size	Description
Control Signals			
CLK_I	I	Rise	External (system) clock signal
RST_I	I	High	External (system) reset
Host Processor Interface Signals			
STB_I	I	High	Strobe signal. When asserted, indicates the start of a valid Wishbone data transfer cycle
CYC_I	I	High	Cycle signal. When asserted, indicates the start of a valid Wishbone cycle
ACK_O	O	High	Standard Wishbone device acknowledgement signal. When this signal goes high, the WB_IRCODER (Wishbone Slave) has finished execution of the requested action and the current bus cycle is terminated
ADR_I	I	3	Address bus, used to select an internal register of the device for writing to/reading from
DAT_O	O	32	Data to be sent to host processor
DAT_I	I	32	Data received from host processor
WE_I	I	Level	Write enable signal. Used to indicate whether the current local bus cycle is a Read or Write cycle: 0 = Read 1 = Write

WB_IRCODER Infrared Encoder

IR Transceiver Interface Signals ¹			
TXD_LED ²	O	High	Transmit LED Driver. Used to switch an independent LED associated with IR Transmitter on or off, under software control. This output follows the level of the <code>shift</code> bit in the Status register (STATUS.1)
TXD	O	High	Transmit Output. Connects to the TXD input pin of the IR Transceiver. This line can be used during initialization to dynamically set the IR Transceiver for operation in SIR mode. The TXD line is driven directly by the output of the Modulation unit. During a required space, it is Low. During a pulse burst (mark) it is taken High and Low in accordance with the carrier frequency.
RXD_LED ²	O	High	Receiver LED driver. This output is internally tied Low, keeping the LED permanently OFF.
RXD	I	Low	Receive Input. This input is not used.
SD	O	High	Shutdown. Connects to the SD pin of the transceiver. This line can be used to place the IR Transceiver into shut-down mode, in order to conserve power. The SD line can also be used during initialization, in harmony with the TXD line, to dynamically set the IR Transceiver for operation in SIR mode. This output follows the level of the <code>sd</code> bit in the Control register (CTRL.0).
MODE	O	Level	Mode Selection. Connects to the Mode pin of the IR Transceiver. For carrier-based remote control IR communications, the speeds involved are very low, so the IR Transceiver module must be set to operate in SIR mode (slow infrared: 2.4kbit/s to 115.2kbit/s). This output is internally tied Low, resulting in the IR Transceiver being permanently placed in Low Bandwidth, SIR mode. This output is used to 'statically' set the operational mode of the IR Transceiver. If you are setting the operational mode dynamically – using the TXD and SD lines – the Mode output must be left floating. If connected, the Mode output will always override the dynamically-set mode.

¹ Based on connection to the TFDU6102 Fast Infrared Transceiver device found on Altium's Ethernet-USB-IrDA Peripheral Board PB03. This peripheral board plugs into the Desktop NanoBoard NB2DSK01.

² The LEDs to which these signals connect are not part of the TFDU6102 FIR Transceiver device.

Hardware Description

Block Diagram

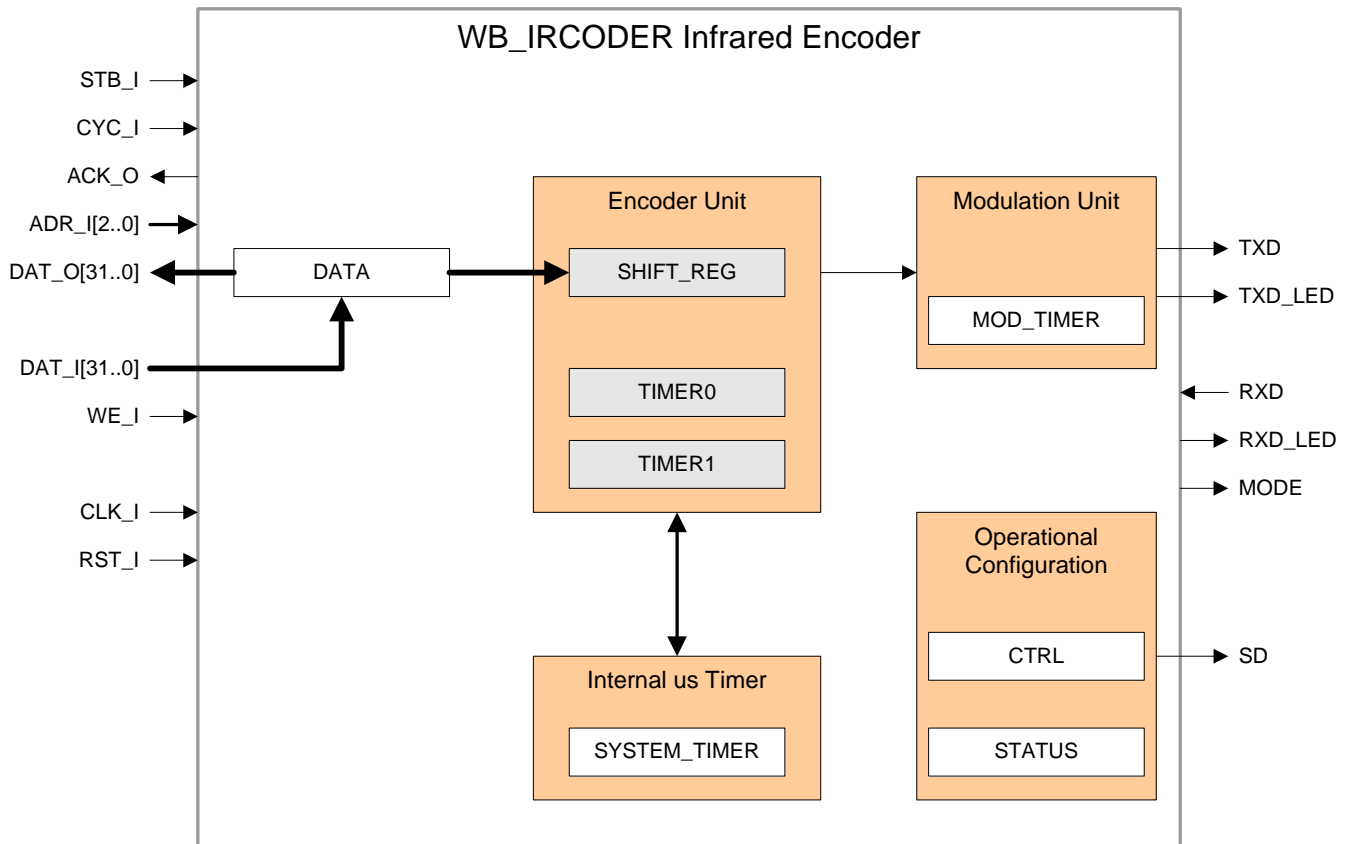


Figure 7. WB_IRCODER block diagram.

Internal Registers

The following sections detail the internal registers for the WB_IRCODER that can be accessed from the host processor.

Control Register (CTRL)

Address: 0h

Access: Read and Write

Value after Reset: 0000_0000h

This register is used to control the level on the SD output line.

Table 2. The CTRL register

MSB			LSB
31		1	0
	-		sd

WB_IRCODER Infrared Encoder

Table 3. The CTRL register bit functions

Bit	Symbol	Function
CTRL.31..CTRL.1	-	Not Used.
CTRL.0	sd	SD output bit. This bit controls the signal level on the SD pin of the device: 0 = SD is Low 1 = SD is High. Use this bit, along with the TXD line to dynamically set the operational mode of the TFDU6102 FIR Transceiver (on the peripheral board PB03) during initialization.

Status Register (STATUS)

Address: 1h

Access: Read only

Value after Reset: 0000_0000h

This register is used to determine the current state of the WB_IRCODER.

Table 4. The STATUS register

MSB		LSB	
31	3	1	0
-		shift	data

Table 5. The STATUS register bit functions

Bit	Symbol	Function
STATUS.31..STATUS.2	-	Not Used.
STATUS.1	shift	Shift Data flag. This bit is set as long as there is data present in the internal Shift register (SHIFT_REG). It is automatically cleared when no data is present.
STATUS.0	data	New Data flag. This bit is set if new data is present in the Data register (DATA). It is automatically cleared once the new data has been moved into the Shift register (SHIFT_REG).

Data Register (DATA)

Address: 2h

Access: Read and Write

Value after Reset: 0000_0000h

The remote control code to be transmitted is sent to this register from the host processor. Once the register is full, the `data` bit in the Status register (STATUS.0) is set, to flag the presence of new data. The data is then loaded into the internal Shift register (SHIFT_REG) and the `shift` bit in the Status register (STATUS.1) is set, indicating that data is ready for transmission.

Note: If new data is sent to the Data register before the previous remote control code transmission has ended (i.e. the internal Shift register still contains data, STATUS.1 = '1'), a repeat code will be transmitted instead of the actual data.

Modulation Timer Register (MOD_TIMER)

Address: 3h

Access: Read and Write

Value after Reset: 0000_0000h

This register is used to hold a timeout value relating to the period of the required modulated signal (i.e. the length of the IR pulse). This timeout value will be used to modulate the encoded message. The value must be entered in terms of the corresponding number of cycles of external system clock signal CLK_I, required to achieve the pulse length.

The WB_IRCODER outputs an IR message frame in accordance with the NEC IR transmission protocol, using modulation based on a carrier frequency of 38kHz (period 1/38000). The value entered into the register would simply be the integer result of the carrier period divided by the period of the CLK_I signal. So for a system clock of 50MHz (period 1/50000000), you would have:

$\text{cycles of CLK_I required} = (1/38000) / (1/50000000) = 1315.78947$

MOD_TIMER value = 1315 (or 0000_0523h).

The same value can be reached simply by dividing the frequency used for the system clock by the frequency of the carrier signal (38000), giving a more generic expression of:

MOD_TIMER value = integer value of $f_{\text{CLK_I}} / 38000$

System Timer Register (SYSTEM_TIMER)

Address: 4h

Access: Read and Write

Value after Reset: 0000_0000h

This register holds a value that is used to divide the incoming system clock (CLK_I) to produce a 1MHz clock signal required to drive the two internal timers located in the peripheral's Encoder Unit – TIMER0 and TIMER1. These timers are used to count out the required times (in microseconds) needed for the various transmission stages of the encoded IR message frame.

For example, if the system clock is 50MHz, then to achieve the 1MHz clock, a value of 50 (or 0000_0032h) would need to be entered into the System Timer register.

WB_IRCODER Infrared Encoder

Interfacing to a 32-bit Processor

How the WB_IRCODER is placed and wired within an FPGA design depends on the method used to build that design. The main processor-based system can be defined purely on the schematic sheet, or it can be contained as a separate OpenBus System, referenced from the top-level schematic. The following sections take a look at using the WB_IRCODER in both of these design arenas.

Design using a Schematic Only

Figure 8 illustrates how a WB_IRCODER device can be wired into a schematic-based design that uses a 32-bit processor – in this case a TSK3000A. A configurable Wishbone Interconnect device (WB_INTERCON) is used to simplify connection and also handle the address mapping – taking the 24-bit address line from the processor and mapping it to the 3-bit address line used to drive the WB_IRCODER.

The WB_IRCODER's IR Transceiver interface signals are connected to the IRDA port component, which represents the pins of the physical FPGA device.

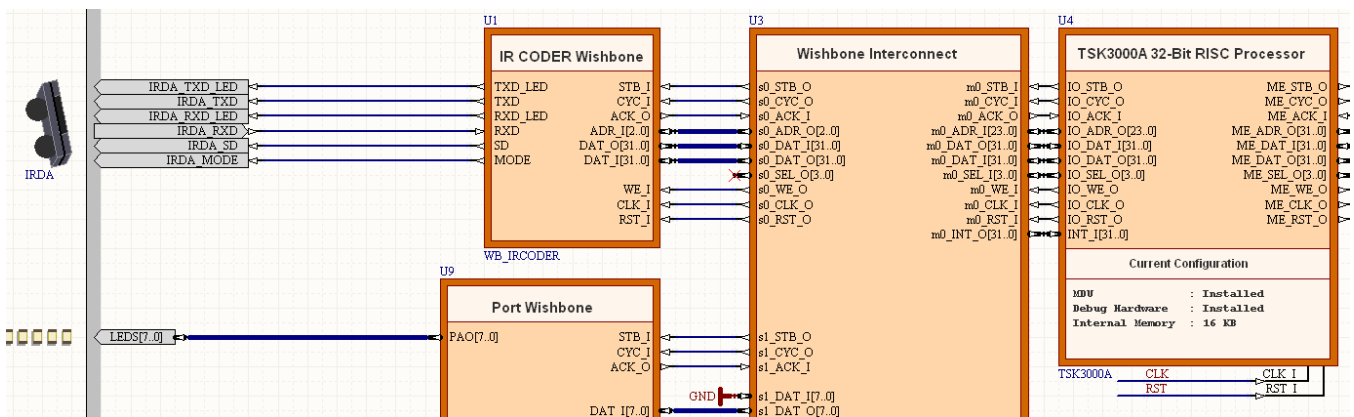


Figure 8. Example interfacing between a 32-bit processor (TSK3000A) and a WB_IRCODER.

When configuring the WB_INTERCON device – in particular the WB_IRCODER slave interface – ensure that the Address Bus Mode is set to Word Addressing – $ADR_O(0) \leq ADR_I(1 \text{ or } 2)$. As the WB_IRCODER's data bus width is 32-bit, the two lowest address bits are not connected to the slave device. $ADR_I(2)$ of the master is mapped to $ADR_O(0)$ of the slave, providing sequential word addresses (or addresses at every 4 bytes). Bits 4..2 of the output address line from the host processor (IO_ADR_O) are therefore mapped, through the WB_INTERCON, to bits 2..0 of the WB_IRCODER's input address line (ADR_I).

The actual 24-bit address sent from the processor on its IO_ADR_O line is constructed as follows:

$$WB_IRCODER \text{ Base Address} + (\text{Internal Register Address} \& \text{"00"})$$

The Base Address for the WB_IRCODER is specified as part of the peripheral's definition when adding it as a slave to the Wishbone Interconnect. For example, if the base address entered for the device is 100000h (mapping it to address FF10_0000h in the processor's address space), and you want to write to the Data register (DATA) with address 2h, the value entered on the processor's IO_ADR_O line would be:

$$100000h + 08h = 100008h$$

For further information on the Wishbone Interconnect, refer to the [CR0150 WB_INTERCON Configurable Wishbone Interconnect](#) core reference.

For further information on the TSK3000A processor, refer to the [CR0121 TSK3000A 32-bit RISC Processor](#) core reference. Similar references can be found for other 32-bit processors supported by Altium Designer, by using the lower section of the **Knowledge Center** panel and navigating to the *Documentation Library* » *Embedded Processors and Software Development* » *FPGA Based and Discrete Processors* section.

Design Featuring an OpenBus System

Figure 9 illustrates identical use of the WB_IRCODER peripheral within a design where the main processor system has been defined as an OpenBus System. The IR Encoder peripheral (as it is referred to in the OpenBus System world) is connected to the TSK3000A processor through an Interconnect component. The OpenBus System environment is a much more abstract and intuitive place to create a design, where the interfaces are reduced to single ports and connection is made courtesy of single links.

Much of the configuration is handled for you – there is no addressing mode to specify, no data width to enter – the IR Encoder peripheral is automatically added as a slave to the Interconnect component by virtue of its link. The Interconnect contains information regarding the device's address bus size and a default decoder address width. All that is really needed is specification of the peripheral's base address – where in the TSK3000A's address space it is to be mapped.

An OpenBus System is defined on an OpenBus System Document (*.OpenBus). This document is referenced from the FPGA design's top-level schematic sheet through a sheet symbol. Figure 10 illustrates the interface circuitry between the IR Encoder's external interface and the physical pins of the target FPGA device – represented by the IRDA port component.

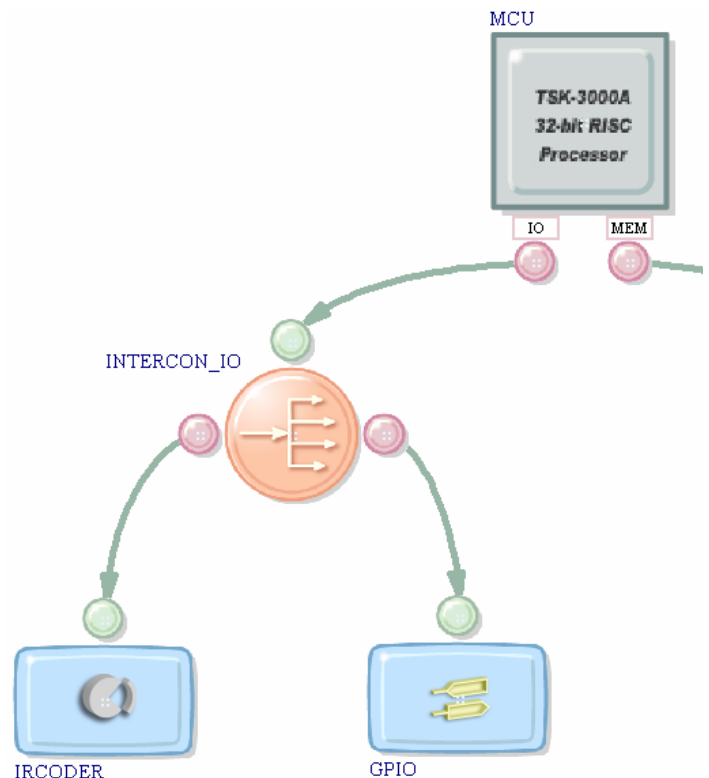


Figure 9. Example interfacing between a 32-bit processor (TSK3000A) and an IR Encoder device, as part of an OpenBus System.

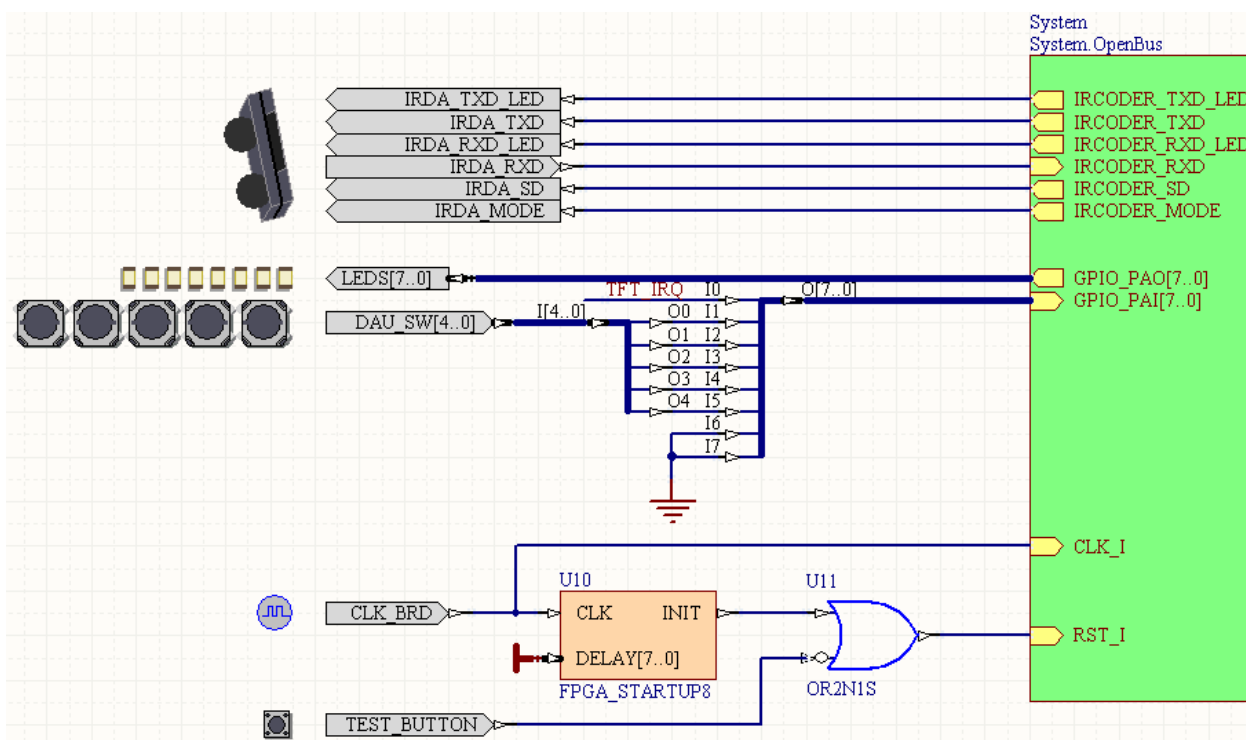


Figure 10. Wiring the OpenBus System-based IR Encoder to the physical pins of the FPGA device.

For further information on the Interconnect component, refer to the document [TR0170 OpenBus Interconnect Component Reference](#).

For more information on the concepts and workings of the OpenBus System, refer to the article [AR0144 Streamlining Processor-based FPGA design with the OpenBus System](#).

Host to Controller Communications

Communications between a 32-bit host processor and the WB_IRCODER are carried out over a standard Wishbone bus interface. The following sections detail the communication cycles involved between host and peripheral device for writing to/reading from the internal registers.

Writing to an Internal Register

Data is written from the host processor to an internal register in the WB_IRCODER, in accordance with the standard Wishbone data transfer handshaking protocol. The write operation occurs on the rising edge of the CLK_I signal and can be summarized as follows:

- The host presents the required 24-bit address based on the register to be written on its IO_ADR_O output and valid data on its IO_DAT_O output. It then asserts its IO_WE_O signal, to specify a write cycle
- The WB_IRCODER receives the 3-bit address on its ADR_I input and, identifying the addressed register, prepares to receive data into that register
- The host asserts its IO_STB_O and IO_CYC_O outputs, indicating that the transfer is to begin. The WB_IRCODER, which monitors its STB_I and CYC_I inputs on each rising edge of the CLK_I signal, reacts to this assertion by latching the data appearing at its DAT_I input into the target register and asserting its ACK_O signal – to indicate to the host that the data has been received
- The host, which monitors its IO_ACK_I input on each rising edge of the CLK_I signal, responds by negating the IO_STB_O and IO_CYC_O signals. At the same time, the WB_IRCODER negates the ACK_O signal and the data transfer cycle is naturally terminated.

Table 6 summarizes how the 32-bit data word from the host processor is used by each of the internal registers.

Table 6. Values written to internal registers during a write.

Writing to...	Results in...
CTRL	DAT_I(0) loaded into the Control register
DATA	the entire 32-bit value arriving on DAT_I loaded into the Data register
MOD_TIMER	the entire 32-bit value arriving on DAT_I loaded into the Modulation Timer register
SYSTEM_TIMER	the entire 32-bit value arriving on DAT_I loaded into the System Timer register

Reading from an Internal Register

Data is read from an internal register in accordance with the standard Wishbone data transfer handshaking protocol. The read operation, which occurs on the rising edge of the CLK_I signal, can be summarized as follows:

- The host presents the required 24-bit address based on the register to be read on its IO_ADR_O output. It then negates its IO_WE_O signal, to specify a read cycle
- The WB_IRCODER receives the 3-bit address on its ADR_I input and, identifying the addressed register, prepares to transmit data from the selected register
- The host asserts its IO_STB_O and IO_CYC_O outputs, indicating that the transfer is to begin. The WB_IRCODER, which monitors its STB_I and CYC_I inputs on each rising edge of the CLK_I signal, reacts to this assertion by presenting the valid data on its DAT_O output and asserting its ACK_O signal – to indicate to the host that valid data is present
- The host, which monitors its IO_ACK_I input on each rising edge of the CLK_I signal, responds by latching the data appearing at its IO_DAT_I input and negating the IO_STB_O and IO_CYC_O signals. At the same time, the WB_IRCODER negates the ACK_O signal and the data transfer cycle is naturally terminated.

Table 7 summarizes the 'make-up' of the 32-bit data word that is read back from each register.

Table 7. Values read from internal registers during a read.

Reading from...	Presents (to host processor)...
CTRL	"00000000000000000000000000000000" & 1-bit value currently in the CTRL register
STATUS	"00000000000000000000000000000000" & 2-bit value currently in the STATUS register
DATA	32-bit value currently in the DATA register
MOD_TIMER	32-bit value currently in the MOD_TIMER register
SYSTEM_TIMER	32-bit value currently in the SYSTEM_TIMER register

Operational Overview

Operation of the WB_IRCODER can be broken down into two main areas – initialization and data transmission. The following sections take a closer look at these areas.

Initialization

After an external reset, you will need to initialize the WB_IRCODER. This should be carried out in accordance with design requirements and can include:

- If interfacing to the TFDU6102 FIR Transceiver on peripheral board PB03, ensuring that the transceiver is set to operate in SIR mode.
- Loading the Modulation Timer register with an integer value, in terms of cycles of CLK_I, that represents the length of the IR pulse.
- Loading the System Timer register with the integer value equal to $f_{CLK_I} / 1000000$.

Setting the TFDU6102 in SIR Mode

When the TFDU6102 is powered on, it is set to operate in SIR mode by default. Therefore, cycling the power of the NB2DSK01 into which the peripheral board PB03 is plugged, can achieve the required mode for infrared remote control communications.

Aside from this, there are two methods for changing the operating mode of the IR Transceiver – either dynamically or statically.

Dynamic Mode Change

The operational mode of the IR Transceiver is changed dynamically using the TXD and SD lines, and generating a falling edge on TXD in the middle of a 400nsec shutdown cycle. From the WB_IRCODER, this is achieved as follows:

- Set the `sd` bit in the Control register (CTRL.0) – takes the SD line High and puts the IR Transceiver in shutdown mode.
- The TXD line must now be taken Low. This is already the case, as the TXD line is internally tied to '0'
- Wait for 200nsec
- Clear the `sd` bit in the Control register (CTRL.0) – the state of the IR Transceiver's TXD input is latched on this falling edge of the SD line, determining the speed of the device to be low bandwidth, SIR mode.
- Wait a further 200nsec.

Note: Although the MODE pin of the IR Transceiver can be read to determine the resulting mode after a dynamic change has been performed, the WB_IRCODER does not support reading of the MODE line. There is therefore no way of telling if the dynamic setting of the mode went as expected. Setting the operational mode of the IR Transceiver statically is therefore the most reliable method (see next section).

Static Mode Change

The operational mode of the IR Transceiver is changed statically using the MODE line. The transceiver is placed into SIR mode when this line is Low. This is already the case, as the MODE line is internally tied to '0'.

Note: Use of the MODE line to control IR Transceiver operational mode will always override any dynamic mode change. Should you wish to set SIR mode purely using the dynamic method, you would have to leave the MODE pin of the WB_IRCODER unconnected.

WB_IRCODER Infrared Encoder

Data Transmission

Once the WB_IRCODER has been initialized, the actual data can be transmitted, using the NEC IR transmission protocol.

To transmit a new remote control code, simply poll the Status register. When both `shift` (STATUS.1) and `data` (STATUS.0) bits are cleared, write the 32-bit message data to the Data register.

Remember that when transmitting data using the NEC IR transmission protocol, the 32-bit message value will contain the actual address and command bytes, as well as their inverses, and that the data is transmitted least significant bit first. The word written to the Data register must therefore be comprised as follows:

MSB				LSB			
31	24	23	16	15	8	7	0
Command Inverted		Command		Address Inverted		Address	

Consider, for example, using the WB_IRCODER to send the command 10101101 to some target receiver device. The address used to target that remote device is, let's say, 00000000.

The 32-bit value written to the Data register would be:

01010010 10101101 11111111 00000000

Once the value is loaded into the Data register, the following transmission process occurs:

- The `data` bit in the Status register (STATUS.0) is set, to flag the presence of new data to be transmitted.
- The data is moved into the Shift register and the `shift` bit in the Status register (STATUS.1) is set. This flags that data is present in the Shift register. The `data` bit in the Status register is cleared at this time. The TXD_LED line, which follows the level of the `shift` bit, is taken High – lighting the transmit-related LED on the peripheral board PBO3.
- The message is then transmitted, in accordance with the NEC IR transmission protocol, with the TXD input line to the IR Transceiver being pulsed, when required, at the associated modulating carrier frequency – 38kHz.
- If no new data is present in the Data register after the end of message transmission (110ms), then the `shift` bit in the Status register is cleared. The processor is then free to load the Data register with the next new remote control code to be transmitted. The TXD_LED line is taken Low upon clearance of the `shift` bit, turning the transmit-related LED on the peripheral board PB03 OFF.

Transmitting a Repeat Code

Should you wish to transmit a repeat code – identical to pressing and holding down the same key on a remote control handset – simply write to the Data register while the Shift register still contains data being transmitted. This requires data to be loaded when:

- the `data` bit in the Status register (STATUS.0) is cleared
- the `shift` bit in the Status register (STATUS.1) is still set.

At the end of the normal IR frame transmission cycle (at 110ms), the WB_IRCODER checks to see if there is any new data (i.e. `data` bit is '1'). At this stage, the `shift` bit is still '1'. If new data is present, a repeat code will be sent. The new data will not be transmitted.

The new data will be loaded across into the Shift register and the `data` bit cleared. At the end of the 110ms cycle, the `data` flag will again be checked. If you have again written to the Data register, the `data` bit will be '1' again and another repeat code will be sent.

The process of sending repeat codes will continue until the `data` bit is found to be '0', at which time the WB_IRCODER will cease sending repeat codes and the `shift` bit will finally be cleared.

Data from the Data register that was last loaded into the Shift register will be effectively overwritten when you next write to the Data register.

Bidirectional IR Communications

Should you wish to transmit and receive IR remote control codes in the same FPGA design – using the WB_IRCODER and WB_IRDEC peripheral devices respectively – this can be achieved, using a wiring layout as shown in Figure 11.

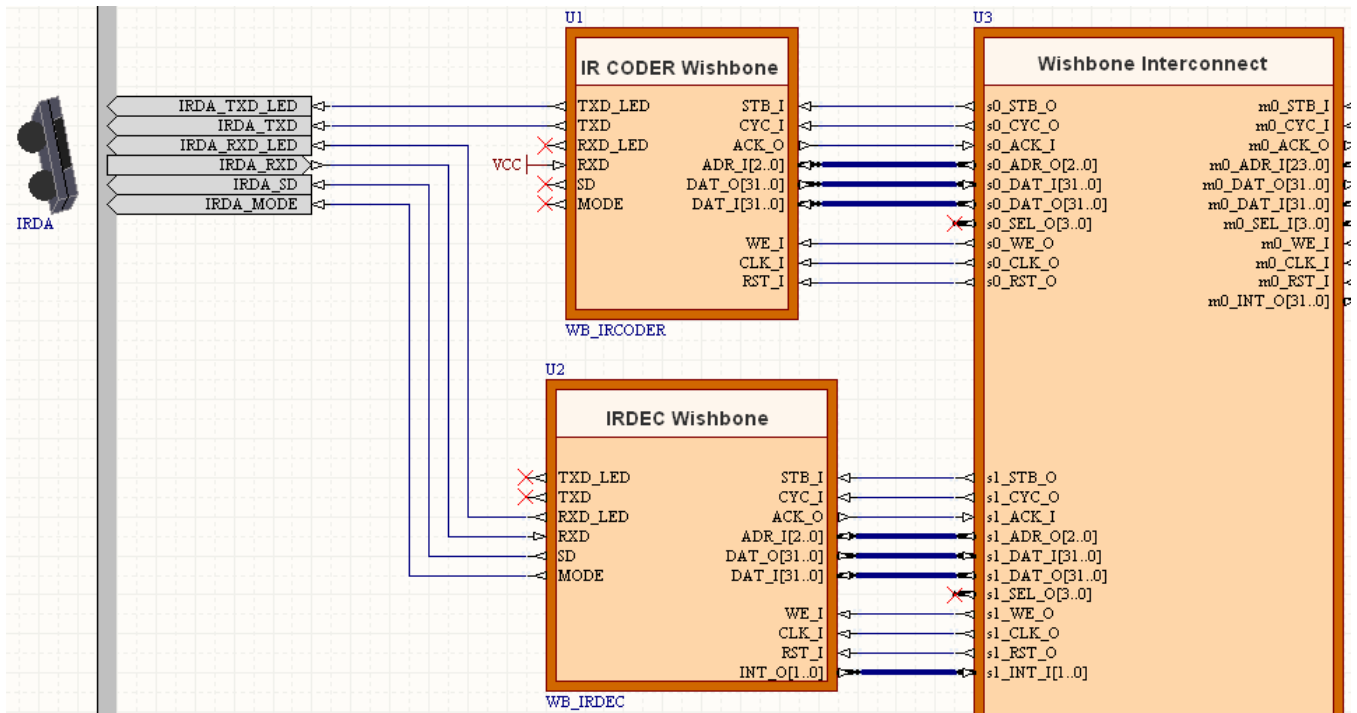



Figure 11. Transmitting and receiving IR remote control codes in the same FPGA design.

Looking at the IRDA port component in Figure 11, we can see:

- TXD and TXD_LED are wired from the WB_IRCODER peripheral.
- RXD and RXD_LED are wired from the WB_IRDEC peripheral.
- The SD and MODE lines can be wired from either of the peripherals (but not both). In Figure 11, they have been wired from the WB_IRDEC simply for neater wiring. If the ability to set the operational mode of the IR Transceiver is required, the MODE line would be left unconnected.

Note also that the RXD line of the WB_IRCODER is tied High, disabling this active-Low input.

 Care should be taken when using these devices in an FPGA design, and interfacing to the TFDU6102 FIR Transceiver on the peripheral board PB03. Feedback at the transceiver results in transmissions made using the WB_IRCODER being received directly by the WB_IRDEC.

Revision History

Date	Version No.	Revision
18-Oct-2007	1.0	Initial release
06-Mar-2008	2.0	Updated for Altium Designer Summer 08

Software, hardware, documentation and related materials:

Copyright © 2008 Altium Limited.

All rights reserved. You are permitted to print this document provided that (1) the use of such is for personal use only and will not be copied or posted on any network computer or broadcast in any media, and (2) no modifications of the document is made. Unauthorized duplication, in whole or part, of this document by any means, mechanical or electronic, including translation into another language, except for brief excerpts in published reviews, is prohibited without the express written permission of Altium Limited. Unauthorized duplication of this work may also be prohibited by local statute. Violators may be subject to both criminal and civil penalties, including fines and/or imprisonment. Altium, Altium Designer, Board Insight, Design Explorer, DXP, LiveDesign, NanoBoard, NanoTalk, P-CAD, SimCode, Situs, TASKING, and Topological Autorouting and their respective logos are trademarks or registered trademarks of Altium Limited or its subsidiaries. All other registered or unregistered trademarks referenced herein are the property of their respective owners and no trademark rights to the same are claimed.