WB_IRDEC Infrared Decoder

Summary

Core Reference
CR0173 (v2.0) March 07, 2008

This document provides detailed reference information with respect to the WB_IRDEC peripheral device. This device is used to process modulated IR data transmitted by a remote control device.

The WB_IRDEC peripheral provides the interface between an infrared receiver 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 photodiode circuit or IR receiver where the output signal is passed on, still in modulated form.

The peripheral handles demodulation of the incoming signal and can be configured to operate in one of two modes – either as a dedicated decoder for data transmitted using the NEC IR transmission protocol, or as a raw interface, allowing the reception of data encoded in any other format. In the latter mode the encoded data is received by the processor, to be decoded in software.

Note: The Remote Controller that comes with Altium's Ethernet-USB-IrDA Peripheral Board PB03, transmits data using the NEC IR transmission protocol.

For information on the WB_IRCODER peripheral, used to encode and modulate data for IR transmission, refer to the *CR0183 WB_IRCODER Infrared Encoder* core reference.

Features

- Two operational modes:
  - NEC Decoder mode: full decoding of data transmitted using NEC IR transmission protocol
  - RAW Interface mode: reception of all non-NEC formatted data. Decoding performed by embedded software
- Customizable demodulation of input signal based on specified carrier frequency
- Interrupt-driven, but allows for processor polling also
- 32-bit data interface
- Wishbone-compliant

Available Devices

From a schematic document, the WB_IRDEC 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 Decoder component can be found in the **Peripherals** region of the **OpenBus Palette** panel.

Infrared Communication Background

Before discussing the actual WB_IRDEC 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).

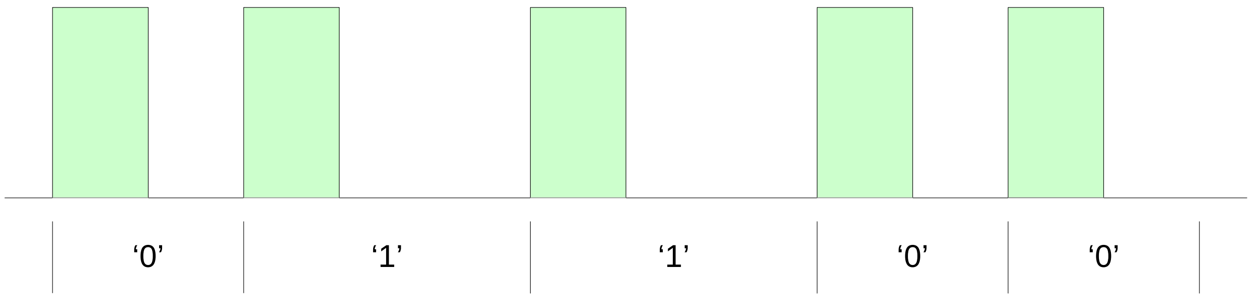

Figure 1. Example of pulse distance encoding.

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.

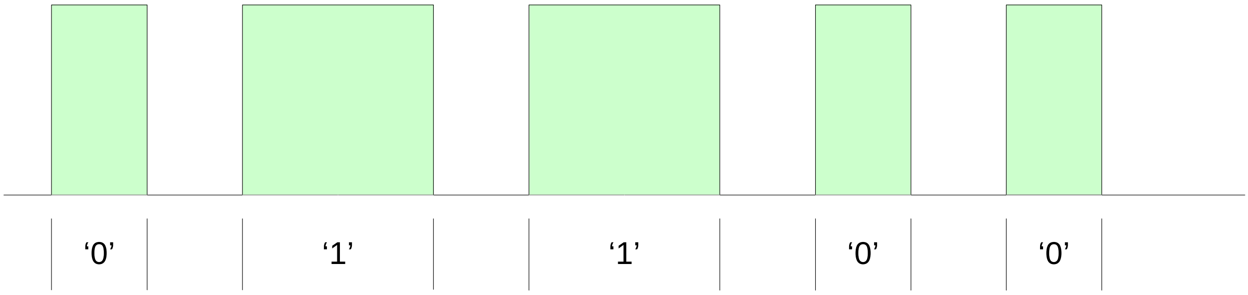

Figure 2. Example of pulse length encoding.

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.

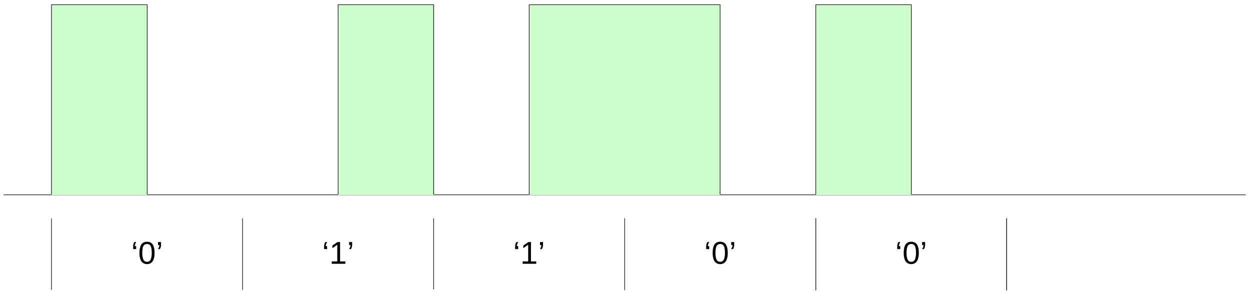

Figure 3. Example of Manchester (or Bi-phase) encoding.

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 Altium Remote Controller and the WB_IRDEC has built-in decoding for this particular protocol.

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 562.5µs in length, at a carrier frequency of 38kHz (26.3µs). Logical bits are transmitted as follows:

- Logical '0' – a 562.5µs pulse burst followed by a 562.5µs space, with a total transmit time of 1.125ms
- Logical '1' – a 562.5µs 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 562.5µs 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`).

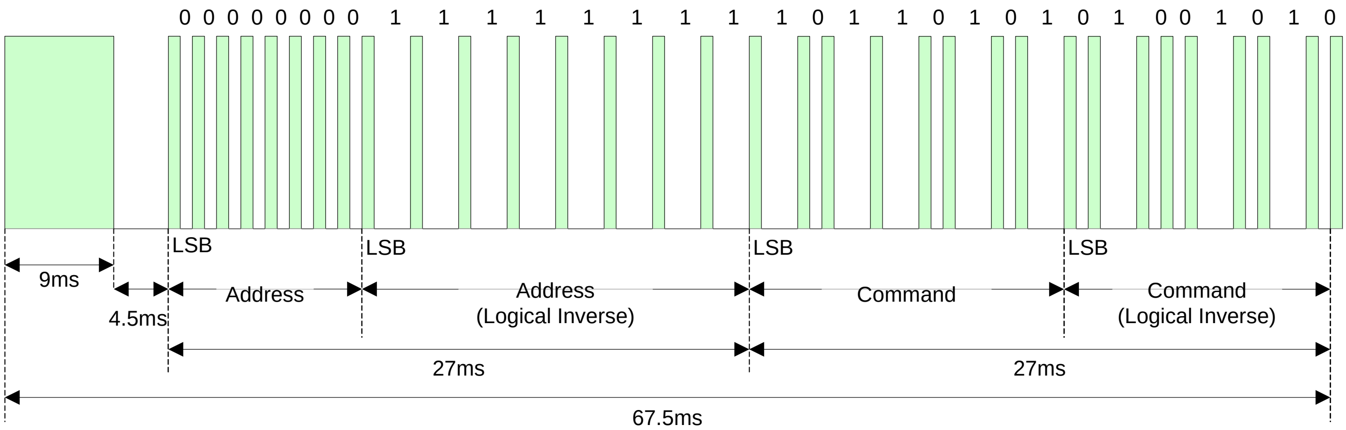

Figure 4. Example message frame using the NEC IR transmission protocol.

Notice from Figure 4 that it takes:

- 27ms to transmit both the 16 bits for the address (address + inverse) and the 16 bits for the command (command + 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.

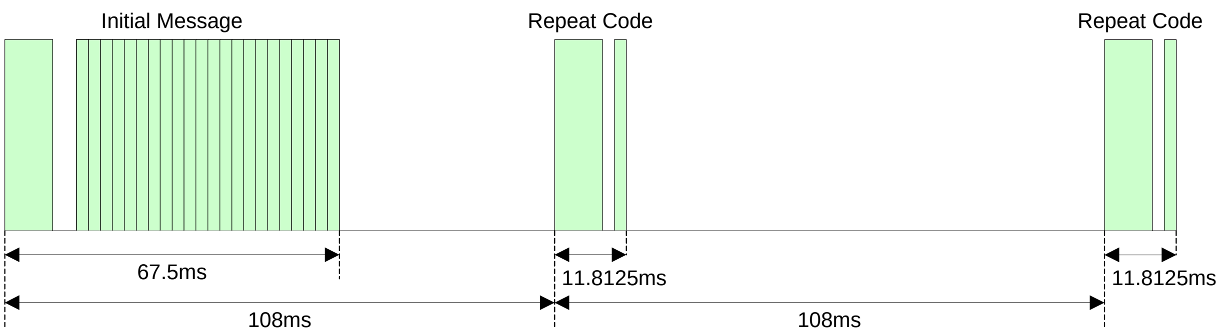

Figure 5. Example repeat codes sent for a key held down on the transmitting remote controller.

Note: Some of the timing values used by the WB_IRDEC when decoding an NEC-formatted message frame 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

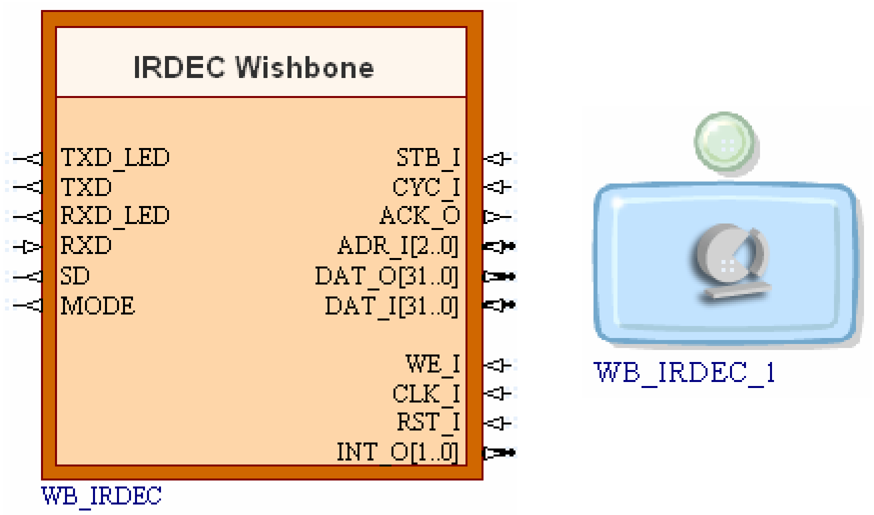

Figure 6. Symbols used for the Infrared Decoder in both schematic (left) and OpenBus System (right).

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_IRDEC 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_IRDEC (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

Name	Type	Polarity/ Bus size	Description
INT_O	O	2/High	Interrupt output lines. Two interrupts are sent to the connected processor on this 2-bit bus: • bit 0 = Goes High if the `inten` bit in the Control register (CTRL.0) is set and the `int0` bit in the Status register (STATUS.0) becomes set • bit 1 = Goes High if the `inten` bit in the Control register (CTRL.0) is set and the `int1` bit in the Status register (STATUS.1) becomes set.
			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 `txdled` bit in the Control register (CTRL.4).
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. This output follows the level of the `txd` bit in the Control register (CTRL.5). During normal operation, the TXD line will be held Low by keeping the value of the `txd` bit in the Control register (CTRL.5) '0'.
RXD_LED[2]	O	High	Receiver LED driver. Used to switch an independent LED associated with IR Receiver on or off, under software control. This output follows the level of the `rxdled` bit in the Control register (CTRL.3).
RXD	I	Low	Receive Input. Connects to the RXD output of the IR Transceiver, which provides the modulated data signal.
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. This output follows the level of the `sd` bit in the Control register (CTRL.1). 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.
MODE	O	Level	Mode Selection. Connects to the Mode pin of the IR Transceiver. This line allows you to set the operational mode of the IR Transceiver as follows: 0 = Low speed mode (SIR) 1 = High speed mode (MIR and FIR) This output follows the level of the `mode` bit in the Control register (CTRL.2). 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 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

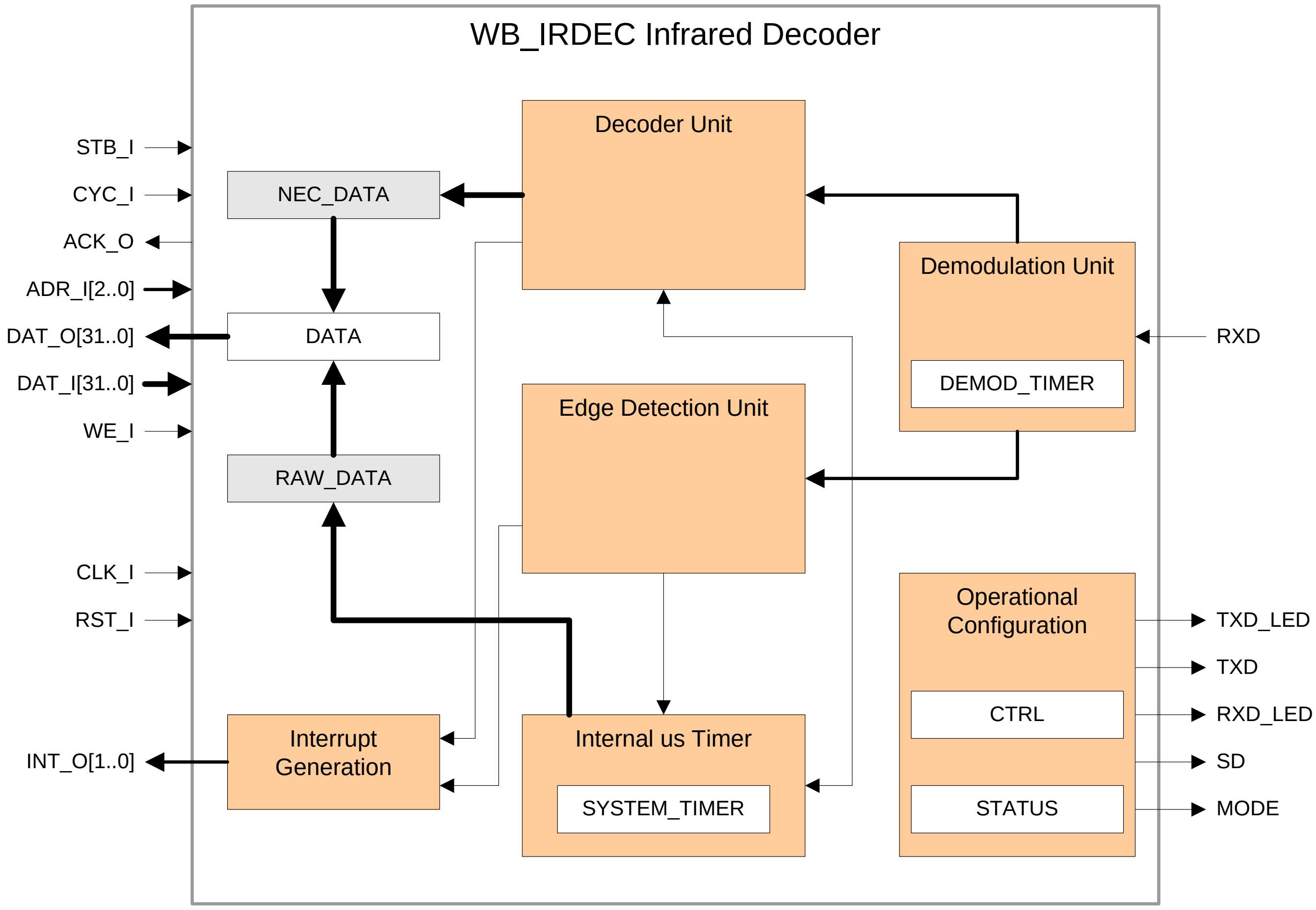

Figure 7. WB_IRDEC block diagram.

Internal Registers

The following sections detail the internal registers for the WB_IRDEC 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 set the operational mode of the WB_IRDEC, as well as providing several control signals used to configure the IR transceiver.

Note: Certain bits in this register relate specifically to control of the TFDU6102 FIR Transceiver device (and associated independent LEDs), found on Altium's Ethernet-USB-IrDA Peripheral Board PB03.

Table 2. The CTRL register

MSB LSB

31 8	7	6	5	4	3	2	1	0
-	dec1	dec0	txd	txdled	rxdled	mode	sd	inten

Table 3. The CTRL register bit functions

Bit	Symbol	Function
CTRL.31..CTRL.8	-	Not Used.
CTRL.7	dec1	Decoder selection bits. These bits (`dec1:dec0`) are used to set the operational mode of the peripheral: 00 = RAW Interface mode. Use this mode when the data from the remote transmitter has been encoded using any IR transmission protocol other than NEC. 01 = NEC Decoder mode. Use this mode when the data from the remote transmitter has been encoded using the NEC IR transmission protocol. 10 = Not used 11 = Not used
CTRL.6	dec0	
CTRL.5	txd	TXD output bit. This bit controls the signal level on the TXD pin of the device: 0 = TXD is Low 1 = TXD is High. Use this bit, along with the `sd` bit (CTRL.1), to dynamically set the operational mode of the TFDU6102 FIR Transceiver (on the peripheral board PB03) during initialization.
CTRL.4	txdled	TXD_LED output bit. This bit controls the signal level on the TXD_LED pin of the device and, subsequently, the independent LED associated with IR transmission (on the peripheral board PB03). 0 = Switch LED OFF 1 = Switch LED ON.
CTRL.3	rxdled	RXD_LED output bit. This bit controls the signal level on the RXD_LED pin of the device and, subsequently, the independent LED associated with IR reception (on the peripheral board PB03). 0 = Switch LED OFF 1 = Switch LED ON.
CTRL.2	mode	MODE output bit. This bit controls the signal level on the MODE pin of the device: 0 = MODE is Low 1 = MODE is High. Use this bit to statically set the operational mode of the TFDU6102 FIR Transceiver (on the peripheral board PB03).
CTRL.1	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` bit (CTRL.5), to dynamically set the operational mode of the TFDU6102 FIR Transceiver (on the peripheral board PB03) during initialization.
CTRL.0	inten	Interrupt Enable bit. Set this bit High to enable generation of external interrupts to the processor.

Status Register (STATUS)

Address: 1h

Access: Read only except where mentioned

Value after Reset: 0000_0000h

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

Table 4. The STATUS register

MSB				LSB
31	3	2	1	0
-		rxddemod	int1	int0

Table 5. The STATUS register bit functions

Bit	Symbol	Function
STATUS.31..STATUS.3	-	Not Used.
STATUS.2	rxddemod	Demodulated Receive bit. This bit reflects the value of the data signal appearing at the RXD input, after it has been demodulated. If RXD is High, reflecting a space, then `rxddemod` will be '0'. Conversely, if RXD is Low, reflecting a mark (or pulse), then `rxddemod` will be '1'.
STATUS.1	int1	Interrupt Line 1 flag. This bit is set if a falling edge is detected on the demodulated data signal (when in RAW Interface mode), or a valid repeat code is received (when in NEC Decoder mode). Writing a '1' to this bit clears the flag.
STATUS.0	int0	Interrupt Line 1 flag. This bit is set if a rising edge is detected on the demodulated data signal (when in RAW Interface mode), or if valid command data is received (when in NEC Decoder mode). Writing a '1' to this bit clears the flag.

Demodulation Timer Register (DEMOD_TIMER)

Address: 2h

Access: Read and Write

Value after Reset: 0000_0000h

This register is used to hold a 16-bit timeout value relating to the period of the received modulated signal (i.e. the length of the IR pulse). This value will be used to demodulate the IR signal, stripping away the carrier and leaving just the encoded data.

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. For example, if you want to receive an IR signal modulated using 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:

cycles of CLK_I required = (1/38000) / (1/50000000) = 1315.78947

DEMOD_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, giving a more generic expression of:

DEMOD_TIMER value = integer value of $f_{CLK\_I} / f_{carrier}$

System Timer Register (SYSTEM_TIMER)

Address: 3h

Access: Read and Write

Value after Reset: 0000_0000h

This register is used to hold an 8-bit value used to divide the incoming system clock (CLK_I) to produce a 1MHz clock signal required for the peripheral's decoder unit. For example, if the system clock is 50MHz, then to achieve the 1MHz clock for demodulation, a value of 50 (or 0000_0032h) would need to be entered into the System Timer register.

Data Register (DATA)

Address: 4h

Access: Read only

Value after Reset: 0000_0000h

This is not actually a register in the true sense of the word, but rather is a single address that is used to access two internal storage registers – RAW_DATA and NEC_DATA respectively. The internal register accessed and value read back depends on the WB_IRDEC's current operational mode:

- RAW Interface mode – the RAW_DATA register is read, providing the time (in microseconds) since the last transition of the demodulated RXD signal, which is reflected by the `rxddemod` bit in the Status register (STATUS.2). The time is retrieved from the internal microsecond timer.
- NEC Decoder mode – the NEC_DATA register is read, providing the output of the NEC Decoder unit. Note that the 32-bit value will contain all bits sent from the remote controller (8 bit address, 8 bit address (inversed), 8 bit command, 8 bit command (inversed)).

Interrupts

The WB_IRDEC generates two interrupt flags – `int0` and `int1`, which are reflected in bits 0 and 1 of the Status register, respectively. The source of these two interrupts depends on the current operational mode set for the peripheral:

- NEC Decoder mode (CTRL[7..6] = "01") – the levels of `int0` and `int1` follow the levels of internal interrupt signals `nec_int0` and `nec_int1`, which are generated within the Decoder Unit. The `nec_int0` signal goes High if valid command data is received. The `nec_int1` signal goes High if a valid repeat code is received.
- RAW Interface mode (CTRL[7..6] = "00") – the levels of `int0` and `int1` follow the levels of internal interrupt signals `raw_int0` and `raw_int1`, which are generated within the Edge Detection Unit. The `raw_int0` signal goes High if a rising edge is detected on the demodulated signal. The `raw_int1` signal goes High if a falling edge is detected.

These interrupts can be exported to the processor on the INT_O[1..0] bus, provided the `inten` bit in the Control register (CTRL.0) is set. Interrupt `int0` is output as INT_O[0], while `int1` is output as INT_O[1].

The `int0` and `int1` flags (and their associated internal source interrupt signals) are cleared by writing a '1' to the respective bit of the Status register.

Figure 8 summarizes the mechanics of interrupt generation for the WB_IRDEC.

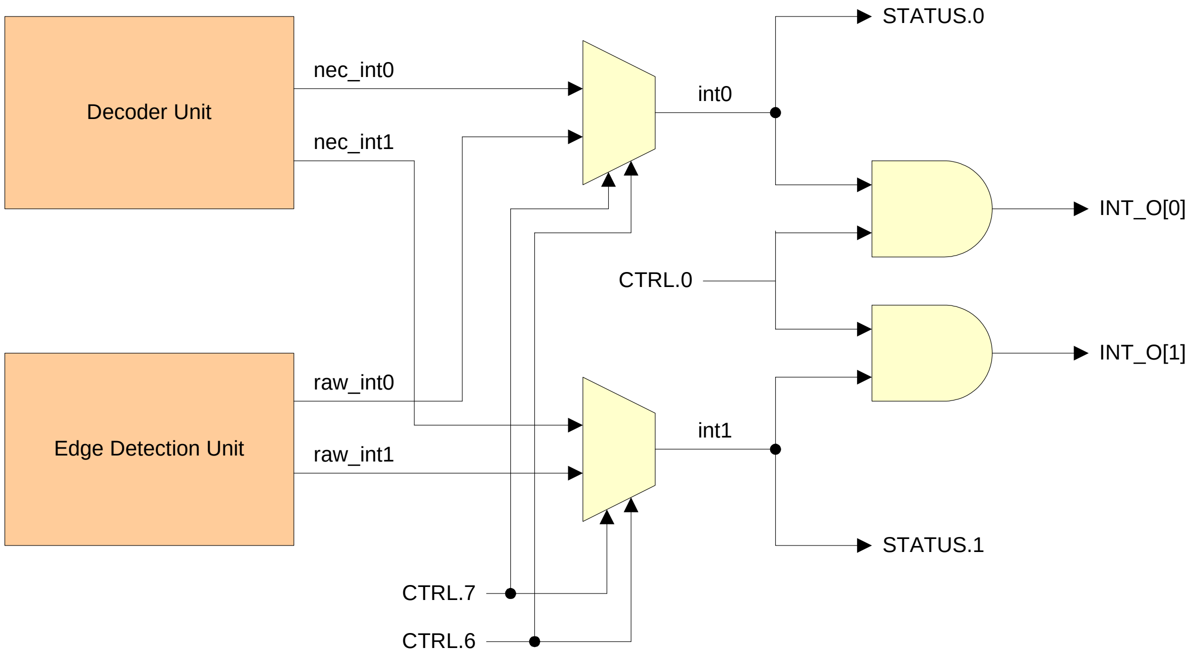

Figure 8. Interrupt generation for the WB_IRDEC.

Interfacing to a 32-bit Processor

How the WB_IRDEC 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, which is then referenced from the top-level schematic. The following sections take a look at using the WB_IRDEC in both of these design arenas.

Design using a Schematic only

Figure 9 illustrates how a WB_IRDEC 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_IRDEC.

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

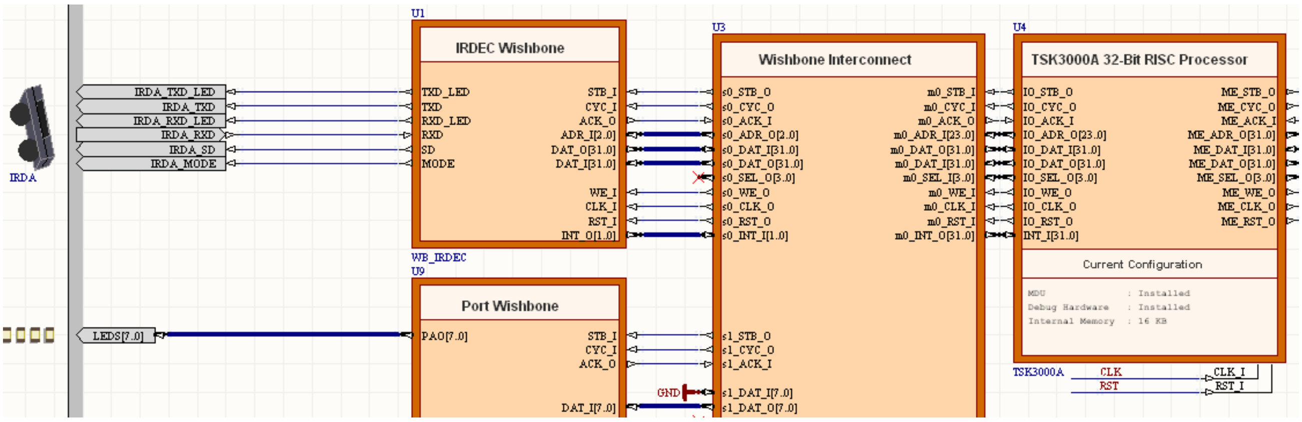

Figure 9. Example interfacing between a 32-bit processor (TSK3000A) and a WB_IRDEC.

When configuring the WB_INTERCON device – in particular the WB_IRDEC slave interface – ensure that the Address Bus Mode is set to `Word Addressing - ADR_O(0) <= ADR_I(1 or 2)`. As the WB_IRDEC'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_IRDEC'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_IRDEC Base Address + (Internal Register Address & "00")

The Base Address for the WB_IRDEC 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 Demodulation Timer register (DEMOD_TIMER) 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.

For an example schematic-based FPGA design featuring a WB_IRDEC device, used to receive IR data encoded using the NEC IR transmission protocol, refer to the example project: `\Examples\NB2DSK1 Examples\DSF Infrared RC\DSF_Infrared_RC.PrjFpg`. This illustrates use of the device in NEC Decoder mode.

For an example schematic-based FPGA design featuring a WB_IRDEC device, used to receive IR data encoded using the Philips RC5 IR transmission protocol, refer to the example project: `\Examples\NB2DSK1 Examples\DSF Infrared RC5\DSF_Infrared_RC5.PrjFpg`. This illustrates use of the device in RAW Interface mode.

Design Featuring an OpenBus System

Figure 10 illustrates identical use of the WB_IRDEC peripheral within a design where the main processor system has been defined as an OpenBus System. The IR Decoder 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 Decoder 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.

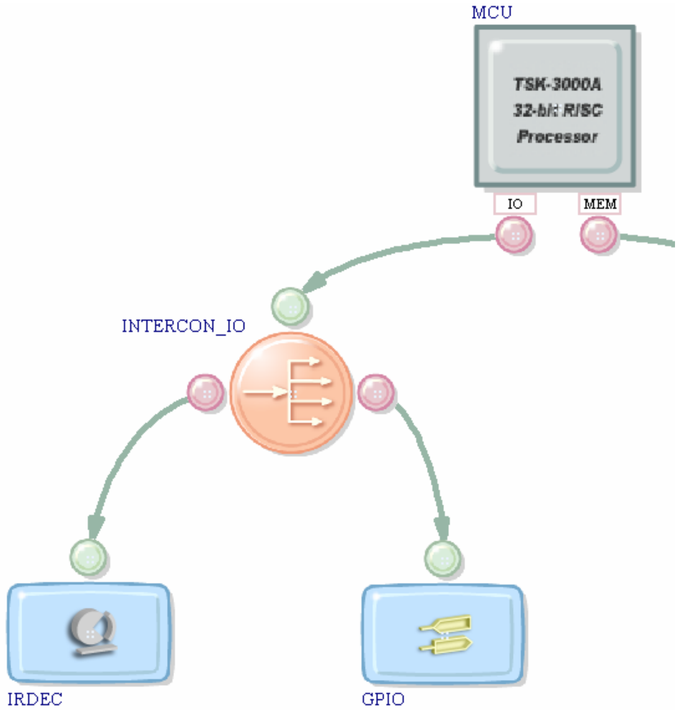

Figure 10. Example interfacing between a 32-bit processor (TSK3000A) and an IR Decoder device, as part of an OpenBus System.

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 11 illustrates the interface circuitry between the IR Decoder's external interface and the physical pins of the target FPGA device – represented by the IRDA port component.

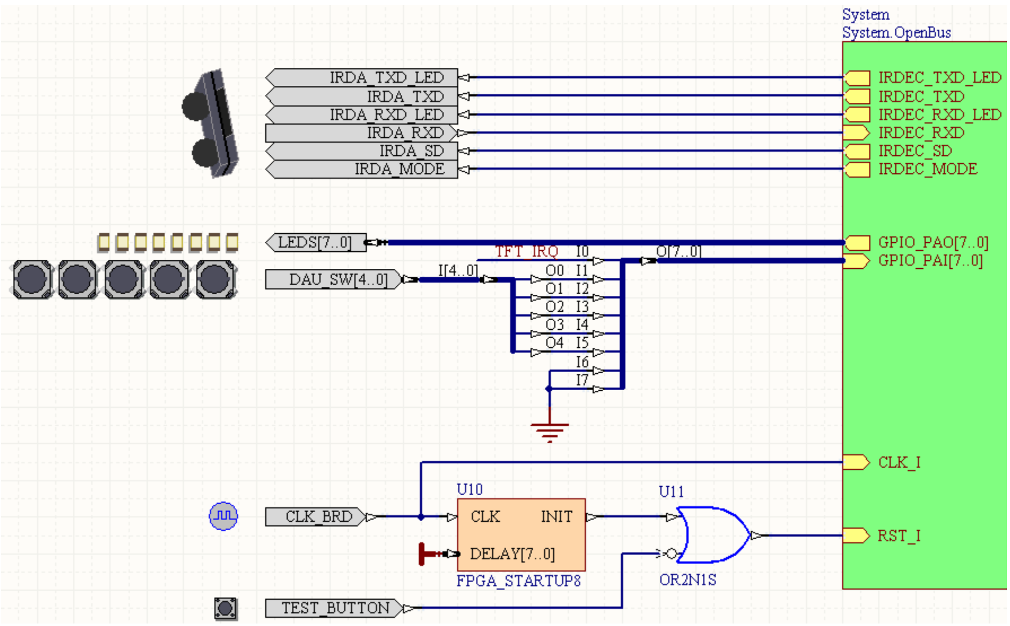

Figure 11. Wiring the OpenBus System-based IR Decoder to the physical pins of the FPGA device.

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*.

For an example OpenBus System-based FPGA design featuring an IR Decoder device, used to receive IR data encoded using the NEC IR transmission protocol, refer to the example project: `\Examples\NB2DSK1 Examples\OpenBus Infrared RC\DSF_Infrared_RC.PrjFpg`. This illustrates use of the device in NEC Decoder mode.

For an example OpenBus System-based FPGA design featuring an IR Decoder device, used to receive IR data encoded using the Philips RC5 IR transmission protocol, refer to the example project: `\Examples\NB2DSK1 Examples\OpenBus DSF Infrared RC5\DSF_Infrared_RC5.PrjFpg`. This illustrates use of the device in RAW Interface mode.

Host to Controller Communications

Communications between a 32-bit host processor and the WB_IRDEC 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_IRDEC, 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_IRDEC 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_IRDEC, 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_IRDEC 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(7..0) loaded into the Control register
STATUS	DAT_I(1..0) loaded into the Status register
DEMOD_TIMER	DAT_I(15..0) loaded into the Demodulation Timer register
SYSTEM_TIMER	DAT_I(7..0) 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_IRDEC 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_IRDEC, 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_IRDEC 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	"000000000000000000000000" & 8-bit value currently in the CTRL register
STATUS	"00000000000000000000000000000" & 3-bit value currently in the STATUS register
DEMOD_TIMER	"0000000000000000" & 16-bit value currently in the DEMOD_TIMER register
SYSTEM_TIMER	"000000000000000000000000" & 8-bit value currently in the SYSTEM_TIMER register
DATA	The internal register and value read depends on the current operating mode of the WB_IRDEC: • RAW Interface mode: 32-bit value currently in the RAW_DATA register • NEC Decoder mode: 32-bit value currently in the NEC_DATA register

Operational Overview

Operation of the WB_IRDEC can be broken down into two main areas – initialization and data reception. For the latter, the way in which data is received and processed depends on the operational mode set for the peripheral. The following sections take a closer look at these areas.

Initialization

After an external reset, you will need to initialize the WB_IRDEC. 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 Demodulation 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.
- Enabling external interrupts to the processor, if required, by setting the `inten` bit in the Control register (CTRL.0).
- Configuring the mode of the peripheral – and therefore switching it on. To operate in NEC Decoder mode, ensure bit `dec0` in the Control register (CTRL.6) is set. To operate in RAW Interface mode, ensure that this bit is cleared.

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_IRDEC, this is achieved as follows:

- Set the `sd` bit in the Control register (CTRL.1) – takes the SD line High and puts the IR Transceiver in shutdown mode.
- Clear the `txd` bit in the Control register (CTRL.5) – takes the TXD line Low.
- Wait for 200nsec
- Clear the `sd` bit in the Control register (CTRL.1) – 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.

As the WB_IRDEC will not be used for transmission in any way during normal operation, the `txd` bit can remain '0', tying the TXD line Low.

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_IRDEC 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. Simply ensure that the `mode` bit in the Control register (CTRL.2) is cleared, in order to tie the MODE line Low and place the transceiver in SIR mode.

Note: Use of the `mode` bit to control IR Transceiver operational mode always overrides 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_IRDEC unconnected.

Data Reception – NEC Decoder Mode

First, let us consider the reception of IR RC data, transmitted using the NEC IR transmission protocol. We will assume that the WB_IRDEC has been initialized and set to operate in NEC Decoder mode.

- The modulated IR data signal from the IR Transceiver is first demodulated. The RXD input is active Low, which means that a space will appear as a logical '1' and a mark will appear as a logical '0'. The output of the demodulator is an inverted logic data stream where a space is represented as logical '0' and a pulse is represented as logical '1'. The current value of the demodulated signal is reflected in the `rxddemod` bit of the Status register (STATUS.2).
- If it is a valid new data command from the remote controller, the data will be decoded and the 32-bit value stored in the internal NEC_DATA register. The `int0` bit in the Status register (STATUS.0) will be set to flag the availability of new command data.
- If it is a repeat code (the same button on the remote controller has been held down), then the `int1` bit in the Status register (STATUS.1) will be set to flag a valid repeat code has arrived.
- Provided the `inten` bit in the Control register (CTRL.0) is enabled, the processor can wait for the respective interrupt line to be taken High, signifying valid remote command (INT_I(0) = High) or repeat code (INT_I(1) = High). Alternatively, it can actively poll the interrupt flags in the Status register.
- The processor can jump to it's respective software routine for handling the new command or the repeat code. If the former, it should perform a read of the WB_IRDEC's Data register address, and access the 32-bit data word stored in the internal NEC_DATA register.
- As part of the handling routine, the processor should, after the data has been retrieved or repeat code noted, send an interrupt acknowledgement – writing a '1' to the `int0` or `int1` bit in the Status register to clear the respective interrupt flag. Optionally, a '1' could be written to the `rxdled` bit in the Control register (CTRL.3), to take the RXD_LED line High and therefore light the associated Receive LED on the peripheral board PB03. By applying an appropriate timeout, the LED can be switched off again to signify completion of data reception.

Remember that when reading data stored in the NEC_DATA register, the 32-bit value will contain the actual address and command bytes, as well as their inverses. The word will 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 Altium Remote Controller to send the "STOP" command (10101101) to the design running in a daughter board FPGA on the Desktop NanoBoard NB2DSK01. The address used to target the IR Transceiver on the attached peripheral board PB03, is 00000000.

The decoded content of the NEC_DATA register will be:

01010010 10101101 11111111 00000000

The code running on the processor within the design may take just the command value from this word, for further processing or, more typically, use the entire 32-bit value and compare it against a pre-determined table of values – one of which will be designated the value associated with the "STOP" command. The code will then proceed with the relevant action to take.

For an example FPGA design featuring a WB_IRDEC device used to receive IR data encoded using the NEC IR transmission protocol, refer to the example project: `\Examples\NB2DSK1 Examples\DSF Infrared RC\DSF_Infrared_RC.PrjFpg`. The file `main.c` in the associated embedded software project (`DSF_Infrared_RC.PrjEmb`) contains examples of routines to handle received data and repeat codes. Where calls are made to functions used to initialize the peripheral, read data, acknowledge interrupts, and so on, these functions can be found in the file `llpi_wb_irdec.c`. This file can be found in the `\System\Tasking\dsf\llpi\src` folder of the installation.

Data Reception – RAW Interface Mode

Now let us consider the reception of IR RC data, transmitted using an IR transmission protocol other than NEC. We will assume that the WB_IRDEC has been initialized and set to operate in RAW Interface mode.

- The modulated IR data signal from the IR Transceiver is first demodulated. The RXD input is active Low, which means that a space will appear as a logical '1' and a mark will appear as a logical '0'. The output of the demodulator is a data stream where a space is represented as logical '0' and a pulse is represented as logical '1'. The current value of the demodulated signal is reflected in the `rxddemod` bit of the Status register (STATUS.2).
- The demodulated signal is polled internally and an interrupt is generated for the next rising edge (`int0` = '1') or falling edge (`int1` = '1') respectively.
- The internal RAW_DATA register is updated at each edge transition, with the contents of an internal microsecond counter and the counter subsequently reset. This gives the time elapsed since the last edge transition. This value can be used by edge handling routines in software to determine whether the full command data has been received.
- Provided the `inten` bit in the Control register (CTRL.0) is enabled, the processor can wait for the respective interrupt line to be taken High, signifying rising edge transition (INT_I(0) = High) or falling edge transition (INT_I(1) = High). Alternatively, it can actively poll the interrupt flags in the Status register.
- The processor can then jump to it's respective software routine for handling either a rising edge or falling edge transition. These routines together will build the data word – still in encoded format. In both cases, it should perform a read of the WB_IRDEC's Data register address, and access the 32-bit data word stored in the internal RAW_DATA register.
- As part of the handling routine, the processor should, after the time data has been retrieved, send an interrupt acknowledgement – writing a '1' to the `int0` or `int1`bit in the Status register to clear the respective interrupt flag.
- The encoded data should then be passed to a decoding routine to generate the final decoded message.

For an example FPGA design featuring a WB_IRDEC device used to receive IR data encoded using the Philips RC5 IR transmission protocol, refer to the example project: `\Examples\NB2DSK1 Examples\DSF Infrared RC5\DSF_Infrared_RC5.PrjFpg`. The file `main.c` in the associated embedded software project (`DSF_Infrared_RC5.PrjEmb`) contains examples of edge handling routines, as well as a decoding routine for this particular protocol. Where calls are made to functions used to initialize the peripheral, read data, acknowledge interrupts, and so on, these functions can be found in the file `llpi_wb_irdec.c`. This file can be found in the `\System\Tasking\dsf\llpi\src` folder of the installation.

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 12.

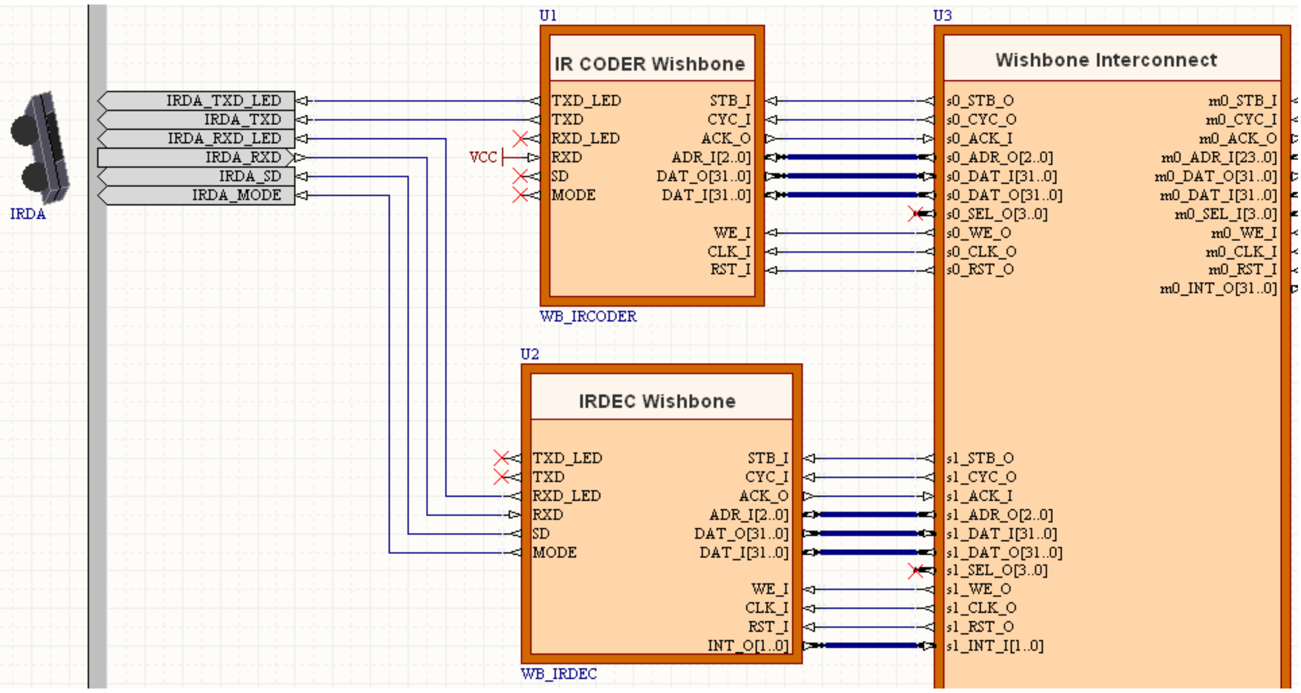

Figure 12. Transmitting and receiving IR remote control codes in the same FPGA design.

Looking at the IRDA port component in Figure 12, 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 12, 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
07-Mar-2008	2.0	Updated for Altium Designer Summer 08