AN214819

Infrared Transmission Support

Associated Part Family: CYW20730/CYW20733

This document provides the infrared (IR) support capability of the Cypress CYW20730 and CYW20733. The focus of the document is IR transmission support, but some aspects of IR reception support are also provided.

It is for hardware and software designers developing IR-capable applications, such as remote controls, 3D glasses, etc.

Contents

1 About the CYW20730 and CYW20733

The Cypress CYW20730 and CYW20733 are Bluetooth 3.0-compliant, stand-alone baseband processors with integrated 2.4 GHz transceivers. They are ideal for wireless input device applications including game controllers, keyboards, 3D glasses, remote controls, gestural input devices, and sensors.

Both baseband processors have built-in firmware that adheres to the Bluetooth Human Interface Device (HID) profile and Bluetooth Device ID profile specifications.

1.1 Cypress Part Numbering Scheme

Cypress is converting the acquired IoT part numbers from Broadcom to the Cypress part numbering scheme. Due to this conversion, there is no change in form, fit, or function as a result of offering the device with Cypress part number marking. The table provides Cypress ordering part number that matches an existing IoT part number.

Table 1. Mapping Table for Part Number between Broadcom and Cypress

Broadcom Part Number	Cypress Part Number
BCM20730	CYW20730
BCM20733	CYW20733

2 IoT Resources

Cypress provides a wealth of data at http://www.cypress.com/internet-things-iot to help you to select the right IoT device for your design, and quickly and effectively integrate the device into your design. Cypress provides customer access to a wide range of information, including technical documentation, schematic diagrams, product bill of materials, PCB layout information, and software updates. Customers can acquire technical documentation and software from the Cypress Support Community website (http://community.cypress.com/).

3 Infrared Background

3.1 Infrared Introduction

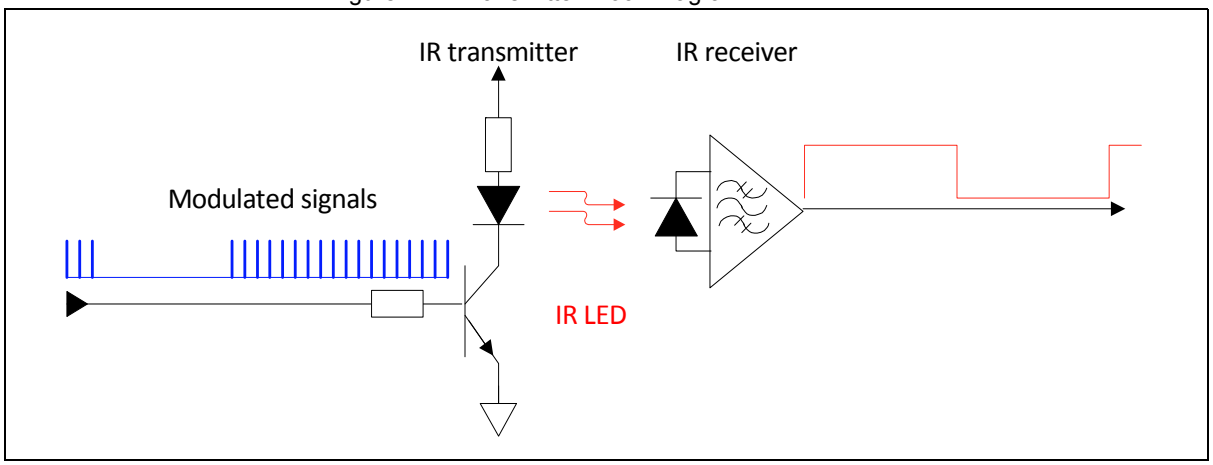
Infrared (IR) is light whose wavelengths are longer than the wavelengths of visible light; therefore, human eyes cannot see it. Infrared light is widely used in industrial, scientific, medical applications and short-range optical communications between computer peripherals.

TV remote controls are a common example where IR optical communication is used.

The transmitter in the remote control sends out a stream of infrared light pulses when a user presses a button. The pattern of the infrared light pulses is unique to each button. The IR receiver in the TV detects the pattern and the TV responds accordingly.

Figure 1 shows a block diagram of an IR transmission system.

Figure 1. IR Transmitter Block Diagram



The demodulated waveform from the IR receiver is not the fully decoded digital signal. An IR decoding protocol must be applied to the demodulated waveform to decode the baseband signal. For information on some example encoding and decoding protocols see 3.2. IR Encoding and Decoding Protocols.

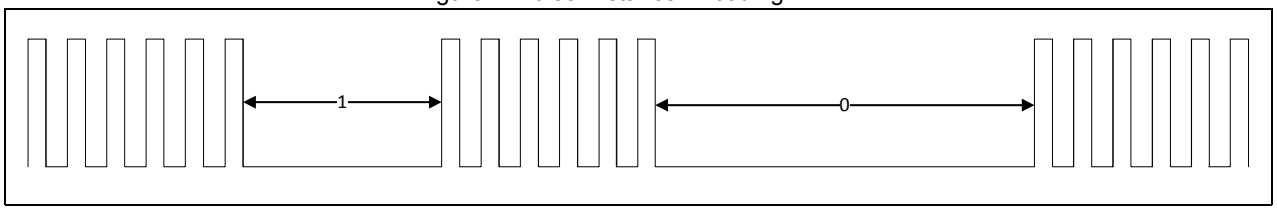
3.2 IR Encoding and Decoding Protocols

Three common IR encoding/decoding approaches are provided in this section: pulse distance encoding, pulse width encoding, and manchester biphase encoding. Other approaches not described here can also be used.

3.2.1 Pulse Distance Encoding

Figure 2 shows how the distances between bursts of pulses are used to encode (and decode) 0's and 1's. A 0 is encoded with a longer distance between bursts than a 1 in the example shown.

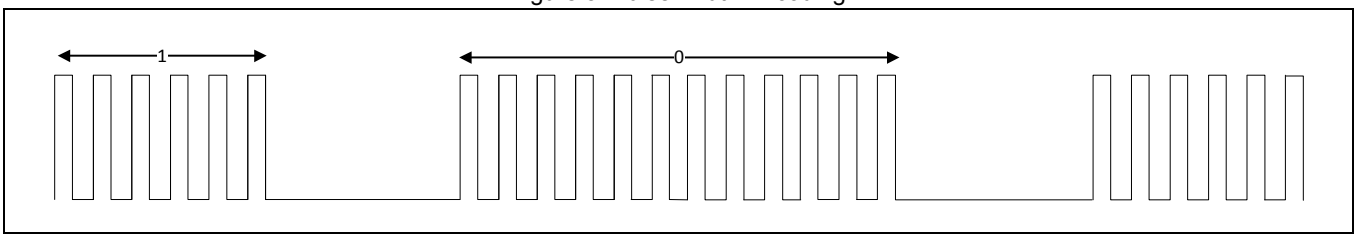
Figure 2. Pulse Distance Encoding



3.2.2 Pulse Width Encoding

Figure 3 shows how the envelope of the pulse bursts (otherwise known as the pulse width) is used to encode data. A 0 is encoded with a wider pulse width than a 1 in the example shown.

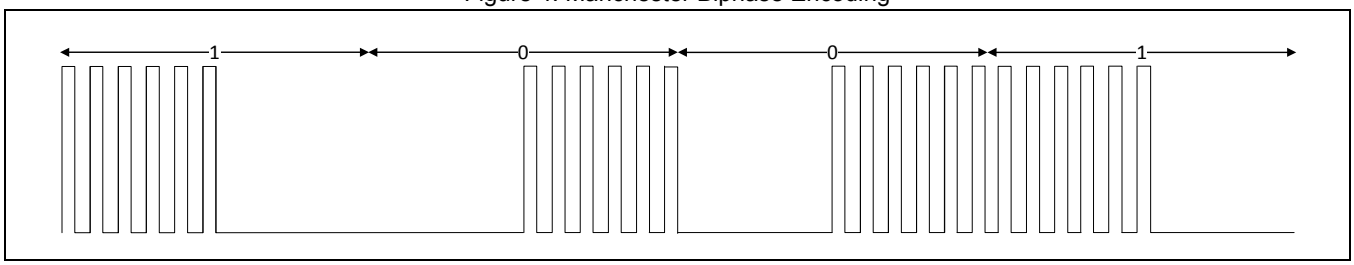
Figure 3. Pulse Width Encoding



3.2.3 Manchester Biphase Encoding

Figure 4 illustrates manchester biphase encoding. In this example, a 1 is encoded such that the first half of a bit period has pulses and the second half has none. A 0 is encoded such that the first half of a bit period has no pulses and the second half has pulses.

Figure 4. Manchester Biphase Encoding



4 CYW20730 and CYW20733 Infrared Hardware Support

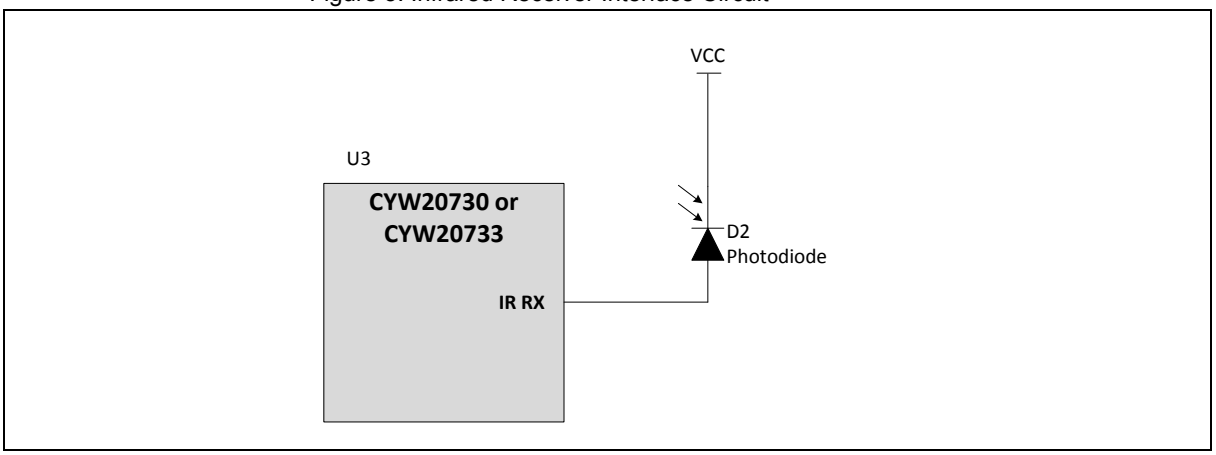
4.1 Infrared Receiver

4.1.1 Infrared Receiver Learning

The CYW20730 and CYW20733 both include hardware support for infrared learning. The hardware can detect both modulated and unmodulated signals. For modulated signals, the CYW20730 and CYW20733 can detect carrier frequencies, which range from 10–500 kHz, and the duration that the signal is present or absent. The CYW20730 and CYW20733 firmware drivers support further analysis and compression of a learned signal. A learned signal can then be played back through the IR transmitter subsystem (see 4.2. Infrared Transmitter).

Figure 5 shows the IR receiver interface circuit, which is simply a photodiode.

Figure 5. Infrared Receiver Interface Circuit



4.1.2 Infrared Receiver Input Pin Selection

One of four or one of three GPIO pins can be selected as the IR receiver input pin for the CYW20730 and CYW20733, respectively.

Table 3 shows the input pins that can be selected as the receiver input pin for each of the CYW20730 and CYW20733 devices.

Table 2. IR Receiver Input Pin Choices

Pin	Selectable as CYW20730 IR Input?	Selectable as CYW20733 IR Input?
P0	Yes	Yes
P15	Yes	Yes
P29	Yes	No
P39	Yes	Yes

4.2 Infrared Transmitter

4.2.1 Infrared Transmitter Overview

The CYW20730 and CYW20733 include hardware support for infrared transmission. The hardware can transmit both modulated and unmodulated waveforms. For modulated waveforms, hardware injects the desired carrier frequency into all IR transmissions. The IR transmitter can be sourced from:

- A programmable bit (referred to as ir_raw_data later in the document).
- The peripheral UART transmitter (puart_tx).

If descriptors are used, they include the IR on/off state and the duration, which has a range of 1–32767 µsec. The CYW20730 and CYW20733 IR transmitter firmware drivers insert this information in a hardware FIFO and ensure that all descriptors are played back without a glitch.

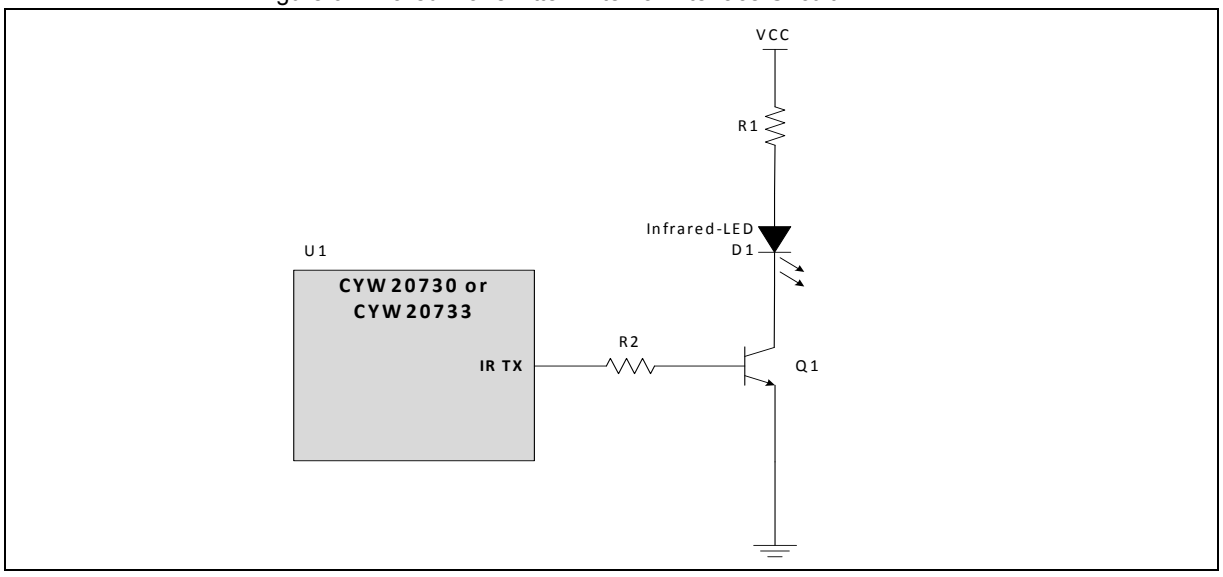
The modulator includes the following main features:

- Eight hardware FIFO entries (cmd0 to cmd7)
- A configurable IR output pin (either pin 1, 4, or 38)
- Configurable IR carrier modulation as either aclk0 or aclk1
- Individual interrupts for every IR FIFO transmitted

4.2.2 Infrared Transmitter External Interface Circuit

Figure 6 shows an IR transmitter output interface circuit.

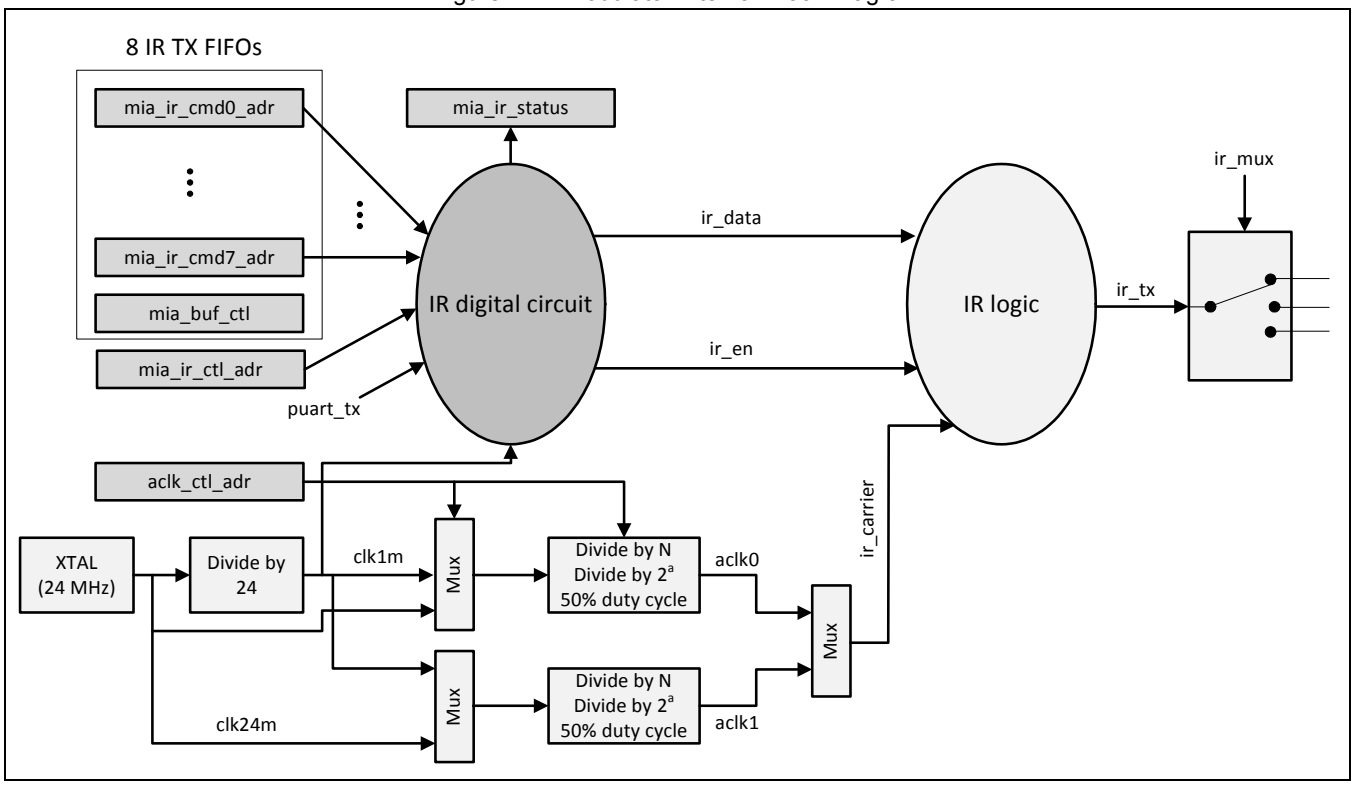
Figure 6. Infrared Transmitter External Interface Circuit



4.2.3 Infrared Modulator Internal Block Diagram

Figure 7 shows a block diagram of the IR modulator.

Figure 7. IR Modulator Internal Block Diagram



4.2.4 Infrared Transmitter Output Pin Selection

One of four or one of three GPIO outputs can be selected as the modulator output pin for the CYW20730 and CYW20733, respectively.

Table 3 shows the output pins that can be selected as the IR modulator output for each of the CYW20730 and CYW20733 devices.

Table 3. IR Transmitter Output Pin Choices

Pin	Selectable in CYW20730?	Selectable in CYW20733?
P1	Yes	Yes
P4	Yes	Yes
P28	Yes	No
P38	Yes	Yes

See 5.2. Configuring a Device Output Pin for IR Transmission to understand how to select the output pin via software.

5 IR Firmware Drivers

The CYW20730 and CYW20733 include hardware IR transmitter blocks (see 4.2. Infrared Transmitter).

The IrTx class defines a driver that can be used by applications, such as a remote controller, to transmit IR commands to a host.

5.1 Using the IrTx Class

To use the IrTx class, first get the IrTx class driver instance. To avoid multiple instances, access the same IR block at the same time. Most of the CYW20730 and CYW20733 hardware drivers are singleton. To use the IrTx driver, it is necessary to get the driver instance first.

```
#include “irtxdriver.h”
IrTx *irTxDriver = IrTx::getInstance();
```

The irTxDriver class pointer is used to access all variables and function calls supported by the IrTx driver.

5.2 Configuring a Device Output Pin for IR Transmission

Both the CYW20730 and CYW20733 allow one of several GPIO outputs to be selected as the IR transmitter pin (IR_TX). See Table 3 for the selection possibilities.

Use the following function call to properly configure the GPIO IR_TX output.

```
void    setIrTxPortPin(BYTE port, BYTE pin);
```

To configure the IR_TX output to P4:

```
#define IR_OUT_GPIO_PORT 4
irTxDriver ->setIrTxPortPin((IR_OUT_GPIO_PORT>>4), IR_OUT_GPIO_PORT&0xf);
```

5.3 Configuring the Carrier Frequency

See the following code for help configuring the IR carrier frequency.

```
BOOL8   sendData( const UINT16* sendBuff, UINT32 pktLen, IR_TX_CLOCK_SETTING
irtxClockSet);
typedef struct

{
        UINT32  clockSrc;
        UINT32  clockSrcFreq;
        BOOL8   invertOutput;
        UINT32  modulateFreq;
}IR_TX_CLOCK_SETTING;
```

The `clockSrc` variable can be Aclk::ACLK0 or Aclk::ACLK1. The `clockSrcFreq` variable can be Aclk::FREQ_24_MHZ or Aclk::FREQ_1_MHZ.

Note: Please refer to the Aclk class, which can be found in the Application Development Kit (ADK). Contact your Cypress sales representative if you do not have the ADK. In the ADK, navigate to \hidd\docs\html\index.html to search for Aclk class information.

5.4 Transmitting IR Data Example

See the following code for help transmitting IR data.

Note: This example applies to one pattern and duration. Patterns and durations can vary per customer requirements.

```
unsigned short irDataStream[] =
{
    (IR_LOW | 100),
    // Guide pulse
    (IR_HIGH | 2400),

    (IR_LOW | 600),
    (IR_HIGH | 600),

    (IR_LOW | 600),
    (IR_HIGH | 1200),
    (IR_LOW | 600),
    (IR_HIGH | 600),
    (IR_LOW | 600),
    (IR_HIGH | 600), // MSB
};

// 38K carrier frequency
#define APPIR_CARRIER_FREQ 38000

irTxClkSetting.clockSrc     = Aclk::ACLK0;
irTxClkSetting.clockSrcFreq = Aclk::FREQ_24_MHZ;
irTxClkSetting.invertOutput = FALSE;
irTxClkSetting.modulateFreq = APPIR_CARRIER_FREQ;

// tx IR data,
irTxDriver ->sendData(irDataStream,
sizeof(irDataStream)/sizeof(unsigned short),
irTxClkSetting);
```

5.5 Determining Whether IR Hardware Is Available

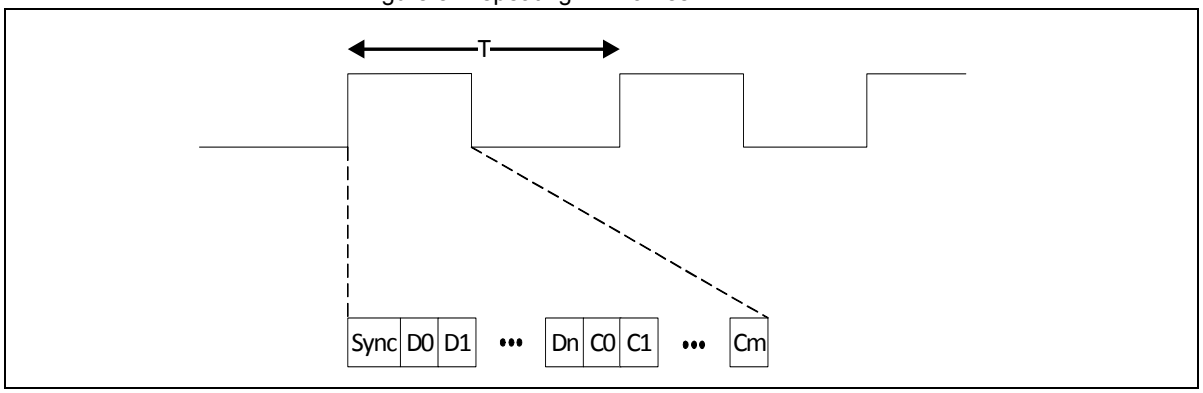
Use this function to see if the IR transmission hardware is available to support IR transmissions.

```
BOOL8 IrTx::isAvailable(void)
```

5.6 Repeating IR Frames

Some IR protocols require repeating the IR frame code at a defined interval (see Figure 8).

Figure 8. Repeating IR Frames



Application firmware is responsible for repeating IR frames. More specifically, the application is responsible for repeating the code and the code-repeat interval.

ROM code in the CYW20730 and CYW20733 does not support IR frame repeating directly, but it does provide the following timer callback functions.

```
void bcs_appPollEnable(void (*clientCallback)(BCS_TASK*, UINT32),
                        UINT32 clientContext,
                        UINT16 defaultPeriod);

void bcs_appPollDisable(void);
```

Sample application code to support IR frame repeating is shown here:

```
void AppIRtx::SendIR(..)
{

    if (irTxDriver ->isAvailable())
    {
        // IR TX
irTxDriver ->sendData(irDataStream,
sizeof(irDataStream)/sizeof(unsigned short),
irTxClkSetting);

        // set up a BCS task timer callback
        bcs_appPollEnable( (void (*)(BCS_TASK*,
           UINT32)) AppIRtx::irTxTimerCallback,
           (UINT32) this,
           IR_CYCLE/BT_SLOT_IN_US);
    }
}
```

The function call AppIRtx::irTxTimerCallback() will be called when IR_CYCLE expires.

6 References

The references in this section may be used in conjunction with this document.

Note: Cypress provides customer access to technical documentation and software through its Customer Support Portal (CSP) and Downloads and Support site (see IoT Resources).

For Cypress documents, replace the “xx” in the document number with the largest number available in the repository to ensure that you have the most current version of the document.

Document (or Item) Name	Broadcom Document Number	Cypress Document Number	Source
Single-Chip Bluetooth Transceiver for Wireless Input Devices, Datasheet	20730-DS1xx-R	002-14824, 002-15291	Cypress Developer Community
Single-Chip Bluetooth Transceiver for Wireless Input Devices, Datasheet	20733-DS0xx-R	002-14859, 002-14860	Cypress Developer Community
Application Development Kit (ADK)	–	–	Cypress Developer Community

6.1 Acronyms and Abbreviations

In most cases, acronyms and abbreviations are defined on first use.

For a comprehensive list of acronyms and other terms used in Cypress documents, go to: http://www.cypress.com/glossary

Document History Page

Document Title: AN214819 - Infrared Transmission Support Document Number: 002-14819				
Rev.	**ECN No.**	**Orig. of Change**	**Submission Date**	**Description of Change**
**	–	–	03/26/2013	20730_20733_AN200-R Initial release
*A	5466387	UTSV	10/07/2016	Updated to Cypress template. Added Cypress Part Numbering Scheme.
*B	5881436	AESATMP9	09/12/2017	Updated logo and copyright.

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturers' representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

ARM® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6

Cypress Developer Community

Forums | WICED IOT Forums | Projects | Video | Blogs | Training | Components

Technical Support

cypress.com/support

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709