AN204833

F²MC-16FX Family, Infrared Object Detection and Communication

This application note will show a simple way to detect objects and communicate via infrared signals

Contents

1 Introduction....1
1.1 Features....1
1.2 Functional principle....1
2 Hardware....3
2.1 Infrared module....3
2.2 Module multiplexer....4
2.3 Microcontroller connection....4
3 Software....7
3.1 Initialization....7
3.2 Interrupt handling for PPG modulation....9
3.3 Multiplexer channel selection....9
3.4 Send and receive data....10
3.5 Autonomous object detection....10
3.6 PC terminal....11
4 Additional information....12
5 Document History....13
Worldwide Sales and Design Support....14
Products....14
PSoC® Solutions....14
Cypress Developer Community....14
Technical Support....14

1 Introduction

This application note will show a simple way to detect objects and communicate via infrared signals. Furthermore it will show how to multiplex multiple infrared channels to reduce microcontroller resource usage.

1.1 Features

- Infrared object detection
- Infrared communication
- Channel multiplexing
- Infrared transmit/receive via USART channel

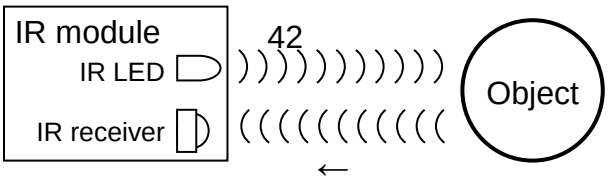
1.2 Functional principle

1.2.1 Object detection

Object detection is based on sending a known value. When a value is received it can be compared to the one that has been sent. If it is the same it can be assumed that an object was reflecting the signal.

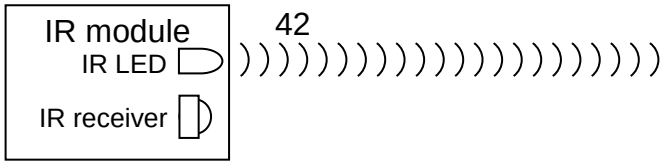
With reflecting object

A known value is sent (e.g. 42). The object in front of the LED will reflect the signal back to the infrared receiver. The received value is the same (42) as the sent one.



No reflecting object

The signal sent will not be reflected back to the receiver. Thus no value is received.

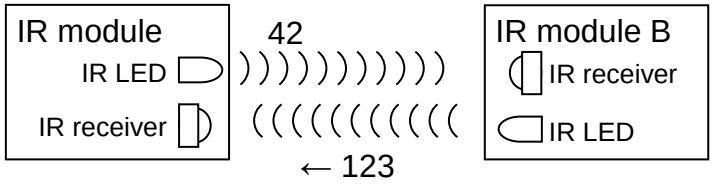


Other infrared module in range

If multiple modules or other infrared communications (e.g. remote controls) are used in a smaller area they can disturb each other.

Depending on the signal characteristics and value sent by such other module it can be detected that there is another module:

- If the value sent by the second module is the same as the one sent by the first module and the time of transmission is the same the first module will recognize the second one as an object.
- If the value sent by the second module is different or it is sent at a different time the first module will recognize that another infrared transmitter is close.

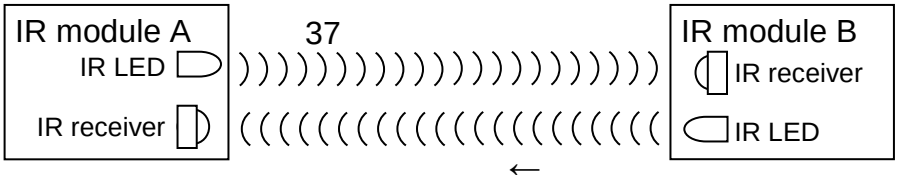


1.2.2 Communication

Communication is used to exchange data between different infrared modules. Any data can be transmitted but special care has to be taken to protect the data against corruption because an infrared transmission can easily be disturbed by other senders or reflections.

No disturbing objects

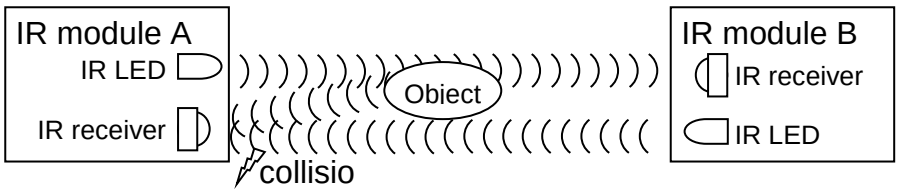
Infrared module A can send any data to infrared module B while B can send any data to A which means a full duplex operation is possible.



Note: The infrared modules themselves can reflect signals of the other module. In that case they act as disturbing objects (see Disturbing objects).

Disturbing objects

When a module sends data while an object is in range the object will reflect the signal back to the sending module. In this case no duplex communication can be established since the signal from the second module would collide with the reflected signal from the first module resulting in undetermined data.



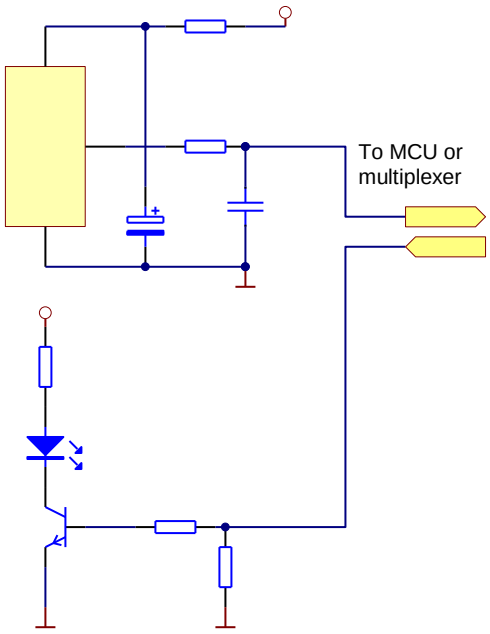
2 Hardware

The two operation modes object detection and communication can be realized by the same hardware and software. The only differences are that for communication you need at least a second infrared module connected to a second microcontroller.

2.1 Infrared module

The infrared module is based on a SFH5110 infrared receiver, an infrared LED, e.g. LD274, and a few passive components:

Figure 1. Basic Infrared Circuit



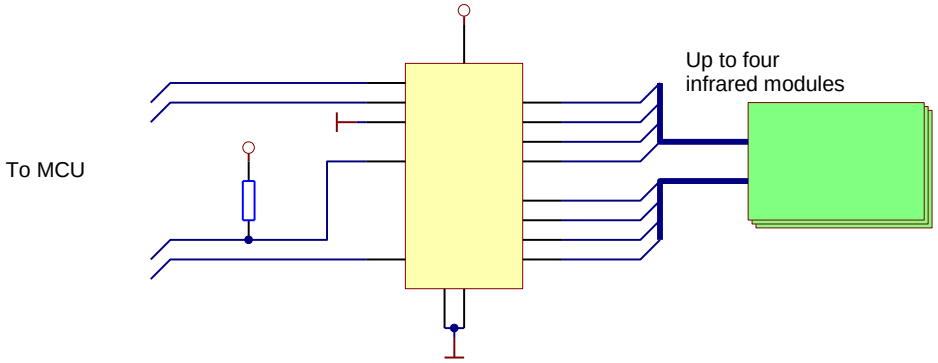
The SFH5110 has an integrated demodulator. Multiple versions for different carrier frequencies are available, this example is based on 36 kHz. R1 and C3 are used for stabilization of the power supply. R2 and C2 result in a low-pass filter with a cut-off frequency of 10 kHz to reduce noise on the OUT line of the receiver. The receiver only recognizes signals with a length of at least 6 periods thus the maximum data rate is 6000 baud. For better signal integrity at least 10 periods should be used resulting in 3600 baud.

The transmitter LED is driven by a NPN transistor for higher current output than provided by the MCU pins. R5 ensures that the LED is off when no signal is applied to the TX pin.

2.2 Module multiplexer

If multiple infrared channels should be used which do not need to be accessed in parallel one can multiplex multiple infrared modules, e.g. with a 74HC4052 analog multiplexer for bidirectional connection:

Figure 2. Infrared Channel Multiplexer



The A and B inputs of the multiplexer can then be used to select which infrared module will be connected to the microcontroller pins. R6 ensures that the received signal has a defined state if there is no connection to an infrared module and the RX pin of the multiplexer is open.

2.3 Microcontroller connection

The microcontroller has the following two tasks:

- Generate the carrier
- Modulate the USART data on the carrier

The carrier is needed for the receiver to recognize the signal. The frequency of the carrier depends on the receiver used, this document is referring to the 36 kHz variant. Modulation of the data on the carrier is done by switching the carrier on or off according to the state of the USART output.

The power output to the LED can be varied by different duty cycles of the carrier. This can be used to reduce range of object detection or communication if required.

Two different solutions to achieve these tasks are given in this document:

- Programmable Pulse Generator (PPG)
- Reload Timer (RLT)

The PPG has the advantage that it is designed to output PWM waveforms, thus the duty cycle can be varied. On the other hand the PPG is not designed to be switched on or off according to another signal so the software has to take care of controlling the PPG output by checking the USART output which results in a slightly increased CPU load.

The Reload Timer output is directly controlled by a gate input so no additional software overhead is needed. The drawback is that the Reload Timer cannot be used to set a duty cycle other than 50 %, power output cannot be changed.

2.3.1 PPG

Figure 3. Schematic Outline Using PPG mode

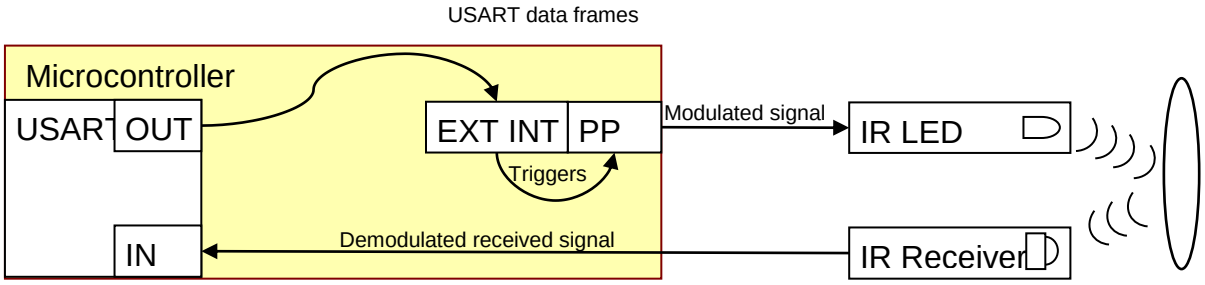
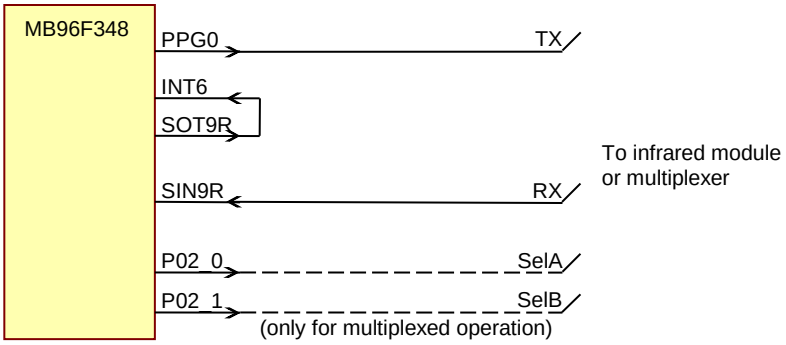


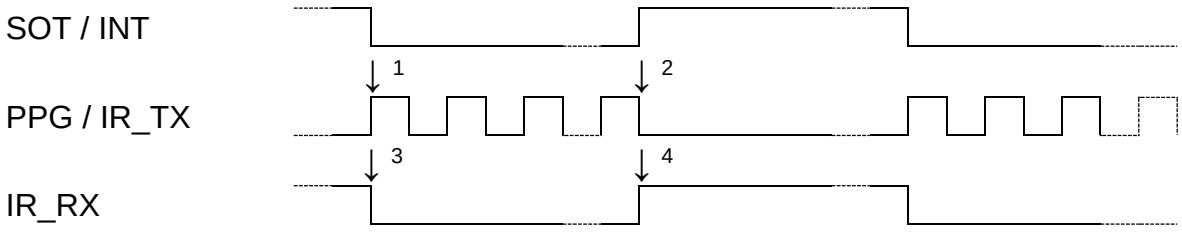
Figure 4. Microcontroller Interface Using PPG Mode



The TX line of the infrared modules is driven by a PPG output which is used to generate the 36 kHz carrier frequency. The USART output SOT is connected to an external interrupt to switch the PPG output on or off according to the state of the SOT output. This results in a modulated output on the PPG.

The RX line of the infrared module is directly connected to the SIN pin of the USART. If a multiplexed setup is used two IO pins have to be connected the multiplexer channel select pins.

Figure 5. Relation of SOT, PPG and SIN in infrared Communication



[1]: Interrupt caused by SOT falling edge starts PPG output

[2]: Interrupt caused by SOT rising edge disables PPG output

[3]: IR receiver detects 36 kHz carrier and sets its output low

[4]: No carrier detected by the IR receiver makes its output go high

Note: The signal output by the IR receiver has a time offset related to the PPG output because the receiver has to detect a few 36 kHz pulses before it recognizes the signal.

2.3.2 Reload Timer

Figure 6. Schematic outline using Reload Timer Mode

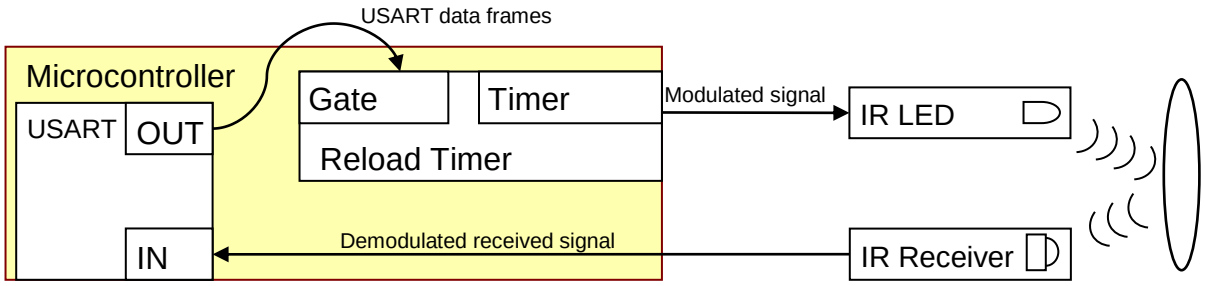
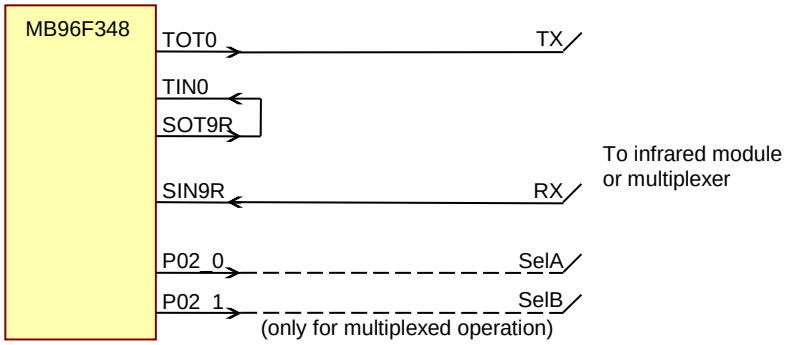


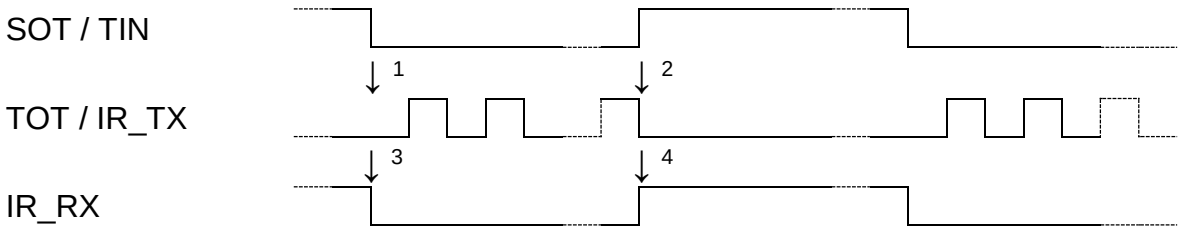
Figure 7. Microcontroller interface using Reload Timer Mode



If a Reload Timer is used to generate the carrier frequency for the infrared signal the USART output is fed into the gate input of the timer. The timer only counts when the USART output is low and thus only toggles its output during that time. The timer has to be set up so that the reload frequency is twice the carrier frequency because it only toggles its output once after each reload of the timer counter.

The carrier frequency also has to be an exact multiple of the USART baud rate. Otherwise the reload timer and USART signal will get out of synchronization. In that case the timer would stop to count in the wrong moment so that the timer output will not always be low when the USART output goes high.

Figure 8. Relation of SOT, RLT operation and SIN in Infrared Communication



[1]: Low level at TIN enables counting of RLT

[2]: High level at TIN disables counting of RLT

[3]: IR receiver detects 36 kHz carrier and sets its output low

[4]: No carrier detected by the IR receiver makes its output go high

Note: The signal output by the IR receiver has a time offset related to the PPG output because the receiver has to detect a few 36 kHz pulses before it recognizes the signal.

3 Software

The software is split into six parts, depending on the setup:

- Initialization
- Interrupt handling for PPG modulation (in PPG modulation mode only)
- Multiplexer channel selection
- Send and receive data
- Autonomous object detection
- Simple PC terminal

3.1 Initialization

Initialization of the infrared communication includes setting up the USART for data transfer, PPG for carrier generation, SOT pin interrupt for PPG modulation and optionally the pins for multiplexer control.

3.1.1 USART

USART initialization is done by setting up the SOT and SIN pins and configuring USART control registers. If interrupts should be used the TIE and/or RIE bits in the serial status register (SSR) have to be set accordingly.

```
// Relocate SOT and SIN pin if required:
PRRR8_SOT9_R = 1;
PRRR8_SIN9_R = 1;

DDR07_D6     = 1; // SOT9 pin output
DDR07_D7     = 0; // SIN9 pin input
PIER07_IE7   = 1; // SIN9 input enable

// USART settings
SCR9 = 0x17;    // TX/RX enable, Clear reception errors, 8N1
SMR9 = 0x01;    // Asynchronous mode, SOT enable
SSR9 = 0x00;    // LSB first, no interrupts
BGR9 = 6659;    // 16 MHz (CLKP1) / 6660 = 2400 Baud
```

3.1.2 Carrier generation and signal modulation

Based on PPG

The PPG is used in PWM mode with a period length set so that the generated frequency matches the receiver's frequency, which is 36 kHz in this setup. The output mask is used to disable PPG output when the SOT pin is in idle / high state. In this mode the duty cycle and with it the power output to the LED can be set by setting the PDUT register value to a fraction of the period register PCSR.

```
DDR06_D0  = 1;    // PPG0 pin output
PCN0_MDSE = 0;    // PWM mode
PCN0_RTRG = 1;    // Enable restart
PCN0_CKS  = 0;    // CKS = 00 -> CKSEL (CLKP1) / 1 = 16 MHz
PCN0_PGMS = 1;    // Enable output mask
PCN0_EGS  = 0;    // No trigger input
PCN0_IREN = 0;    // No interrupt requests
PCN0_IRS  = 0;    // Interrupt cause software/external trigger
PCN0_OE   = 1;    // Output enable
PCN0_OSEL = 0;    // Non-inverted Output
PCN0_CNTE = 1;    // Enable counting
PCSR0 = 443;      // Period: CLKP1 / (PCSR+1) = 16 MHz / 444 = 36 kHz
PDUT0 = 443 / 5;  // Duty cycle (lower duty for lower power output)
PCN0_STGR = 1;    // Start PPG0
```

The toggling of the PPG output is handled by an edge interrupt connected to the SOT output of the USART. Using a MB96F340 series microcontroller the USART output SOT9R and the external interrupt INT6 share the same pin so there is no need to provide an external connection between those two. The data direction of the pin has to be set to output for SOT, the interrupt still works if input enable is set for the pin.

```
PIER07_IE6 = 1;  // Enable input for INT6 (SOT9R)
ELVR0_LA6  = 1;  // LB=1 and LA=1: Interrupt on falling edge
ELVR0_LB6  = 1;
EIRR0_ER6  = 0;  // Clear interrupt request flag
ENIR0_EN6  = 1;  // Enable interrupt request for INT6
```

Based on Reload Timer

The Reload Timer is used in reload mode with gate input so that it only counts when the SOT state is low. The output of the Reload Timer is toggled on counter reload so the reload frequency has to be two times the infrared carrier frequency. The frequency also has to be an exact multiple of the USART baud rate because otherwise the Reload Timer output gets out of synchronisation with the USART output.

```
DDR08_D1    = 1;    // TOT0 output
TMCSR0_FSEL = 1;    // No additional clock division by 2
TMCSR0_CSL  = 0;    // Clock CLKP1 / 2^1 = 8 MHz
TMCSR0_MOD2 = 1;    // Gate input mode
TMCSR0_MOD1 = 0;    // Don't care
TMCSR0_MOD0 = 0;    // Low level gate
TMCSR0_OUTE = 1;    // Enable timer output
TMCSR0_OUTL = 0;    // Start with low level output
TMCSR0_RELD = 1;    // Reload mode
TMCSR0_CNTE = 1;    // Enable counting
TMCSR0_TRG  = 1;    // Trigger timer to start
TMRLR0      = 110;  // Reload value 111: 8 MHz / 111 = 72 kHz
```

3.1.3 Multiplexer control

If the optional multiplexer circuit is used to connect multiple infrared modules to one USART/PPG then the pins used for control of the multiplexer channel have to be initialized to output.

```
DDR02 |=  0x03;  // Set P02_0 and P02_1 to output
PDR02 &= ~0x03;  // Set both pins to low state (channel 0 selected)
```

3.2 Interrupt handling for PPG modulation

If the PPG is used as carrier generator its output has to be toggled on and off according to the USART output, which results in a modulated signal on a 36 kHz carrier. This is done in the interrupt service routine of the external interrupt.

```
__interrupt void irq_irTrig(void)
{
    // Check whether the external state of the pin is low
    if (EPSR07_PS6 == 0)
    {
        ELVR0_LA6 = 0;  // Set interrupt to rising edge
        PCN0_STGR = 1;  // Restart PPG to synchronize output
                        //   to the start of the USART frame
        PCN0_PGMS = 0;  // Disable PPG output mask
    } else
    {
        ELVR0_LA6 = 1;  // Set interrupt to falling edge
        PCN0_PGMS = 1;  // Enable PPG output mask
    }
    EIRR0_ER6 = 0;      // Clear interrupt request
}
```

3.3 Multiplexer channel selection

If a multiplexer is used the channel can be selected by setting the IO bits to the value of the desired channel.

```
en_result_t Ir_SelectChannel(uint8_t irChannel)
{
    if (irChannel < 4)      // irChannel in range?
    {
        PDR02 &= ~0x03;     // Clear bits
        PDR02 |= irChannel; // Write channel on bits 0 and 1
        return Ok;
    }
    return ErrorInvalidParameter;
}
```

3.4 Send and receive data

Sending and receiving data can be handled just as with any other USART interface. The basic routines for single byte transfers are given below.

3.4.1 Send single byte

```
void Ir_SendByte(uint8_t data)
{
    while (SSR9_TDRE == 0)  // Wait until transmit buffer is free
        __wait_nop();
    TDR9 = data;            // Put byte in transmit buffer
}
```

3.4.2 Receive single byte

```
en_result_t Ir_GetByte(uint8_t* data)
{
    if (SSR9_RDRF == 0)     // No byte in receive buffer?
    {
        return Error;
    }

    *data = RDR9;           // Store received byte
    if ((SSR9 & 0xE0) != 0) // Any error occurred?
    {
        SCR9_CRE = 1;       // Clear receive errors
        return Error;
    } else
        return Ok;          // Successfully return
}
```

3.5 Autonomous object detection

For autonomous object detection an additional timer can be used which regularly sends data via infrared and checks whether it is received afterwards.

Initialization of timer is given below:

```
TMCSR1_FSEL = 1;     // No additional clock division by 2
TMCSR1_CSL  = 0;     // Clock CLKP1 / 2^1 = 8 MHz
TMCSR1_RELD = 1;     // Reload mode
TMCSR1_INTE = 1;     // Enable interrupt requests
TMCSR1_CNTE = 1;     // Enable counting
TMCSR1_TRG  = 1;     // Trigger timer to start
TMCSR1_INTE = 1;     // Enable interrupt
TMRLR1      = 39999; // Reload value: 8 MHz / (39999 + 1) = 200 Hz
```

The period of the timer is a bit longer than the time for one USART frame to be completely sent so that a byte that is put in the send-register of the USART can be transmitted and the reflected signal is detected by the receiver before the next interrupt of the timer occurs.

The interrupt routine for the timer will first check if a byte was received since last execution. If so the received value is compared to the byte that was sent and if it is the same it is assumed that an object is in range. If not or if no byte was received no object is in range. Afterwards the next multiplexer channel is selected and a new byte is sent.

```
__interrupt void irq_objDetection(void)
{
    uint8_t data;
    TMCSR1_UF = 0;      // Clear interrupt cause

    // Check if a byte was received
    if (Ir_GetByte(&data) == Ok)
    {
        // Is byte the same as the one sent?
        if (data == OBJECT_DETECTION_ID)
        {
            // Set bit in states variable
            Ir_Object_States |= (1 << currentChannel);
        } else
        {
            // Clear bit in states variable
            Ir_Object_States &= ~(1 << currentChannel);
        }
    } else
        // Clear bit in states variable
        Ir_Object_States &= ~(1 << currentChannel);

    // Switch to next multiplexer channel
    Ir_SelectChannel((currentChannel + 1) % 4);
    // Send a byte
    Ir_SendByte(OBJECT_DETECTION_ID);
}
```

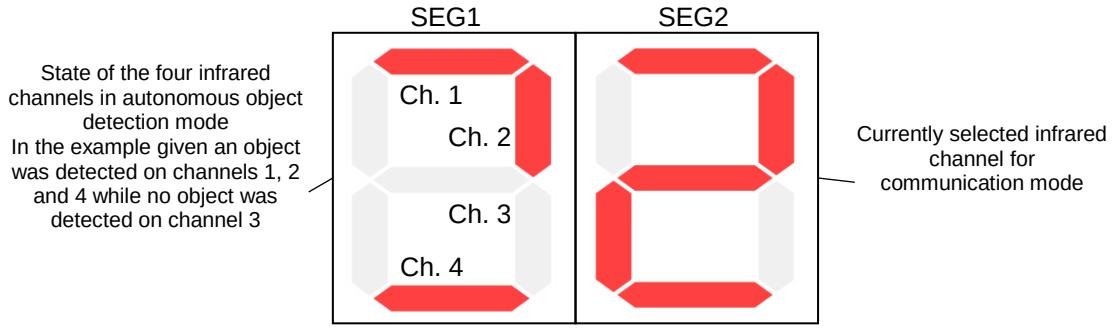
3.6 PC terminal

The terminal software is used to:

- Select the multiplexer channel
- Toggle between autonomous object detection and manual transmission
- Send a byte via infrared (only in manual transmission mode)

In autonomous object detection mode the current state of object detection is displayed on the left seven segment display by the segments A through D representing the states of channels 1 through 4.

In communication mode the current multiplexer channel is displayed on the right seven segment display.



Keys used in the terminal software to control the operation of the software:

Character	Action
“1” .. “4”	Select channel 1 through 4
“O” or “o”	Toggle autonomous object detection on/off
Other characters	Send character via infrared if not in autonomous object detection mode

The USART for the interface to the PC is configured as following:

Parameter	Value
Baud rate	115200 bit/s
Frame length	8 bit
Parity	None
Stop bits	1

4 Additional information

Information about Cypress microcontrollers can be found on the following internet page:

http://www.cypress.com/cypress-microcontrollers

The software examples are:

- 96340_uart_infrared_ppg
- 96340_uart_infrared_rlt

They can be found on the following internet page:

http://www.cypress.com/products/16FX

5 Document History

Document Title: AN204833 - F²MC-16FX Family, Infrared Object Detection and Communication

Document Number: 002-04833

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	-	MKEA	02/25/2010	Initial Release
*A	5062201	MKEA	12/24/2015	Migrated Spansion Application Note from MCU-AN-300252-E-V10 to Cypress format
*B	5836291	AESATMP8	07/28/2017	Updated logo and Copyright.
*C	6038357	NOFL	01/19/2018	Update logo and links. Updated Sales page and Copyright year.

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's 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 MCU

Cypress Developer Community

Community | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

All other trademarks or registered trademarks referenced herein are the property of their respective owners.

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

© Cypress Semiconductor Corporation, 2010-2018. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC ("Cypress"). This document, including any software or firmware included or referenced in this document ("Software"), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress's patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. No computing device can be absolutely secure. Therefore, despite security measures implemented in Cypress hardware or software products, Cypress does not assume any liability arising out of any security breach, such as unauthorized access to or use of a Cypress product. In addition, the products described in these materials may contain design defects or errors known as errata which may cause the product to deviate from published specifications. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage ("Unintended Uses"). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.