CE211027 - CAN Communication Between FM4-S6E2Gx Series and PSoC® 4 M-Series Using CY8CKIT-026 CAN and LIN Shield Kit

Objective

This code example demonstrates CAN communication between FM4-S6E2Gx Series & PSoC® 4 M-Series using the CY8CKIT-026 CAN and LIN Shield Kit.

Overview

This code example demonstrates the configuration of FM4 S6E2GM Pioneer Kit and CY8CKIT-044 for CAN communication using CY8CKIT-026 as the CAN transceiver. FM4 S6E2GM acts as the CAN transmitter (Tx) and PSoC® 4 M as the CAN receiver (Rx). The CAN transmitter sends data for ON/OFF status, color, and brightness of an RGB LED. This data is transmitted over the CAN; at the CAN receiver, the RGB LED is configured to reflect the received data.

Requirements

Tool: PSoC Creator™ 3.3 SP1 (PSoC 4 M-Series) or later, Keil µVision® 5 (FM4 S6E2Gx series).

Programming Language: C (Armcc V5.05, ARM GCC 4.9.3 and MDK compilers)

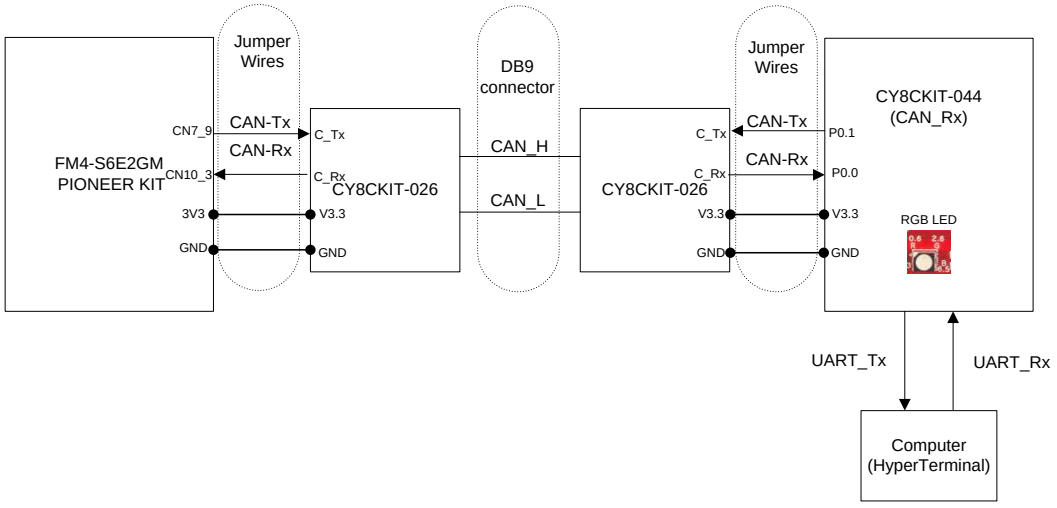
Associated Parts: 4200-M and 4100-M, FM4

Related Hardware: CY8CKIT-044 – PSoC 4 M-Series Pioneer Kit, FM4-176L-S6E2GM, two CY8CKIT-026 kits

Design

This code example consists of two projects; one for the CAN transmitter, and one for the CAN receiver. Figure 1. CAN Network Topologyshows the hardware connections between the kits.

Figure 1. CAN Network Topology



The transmitter project continuously transmits LED ON/OFF status, color, and brightness values. The receiver project controls its LEDs accordingly.

CAN_Tx_FM4 Project

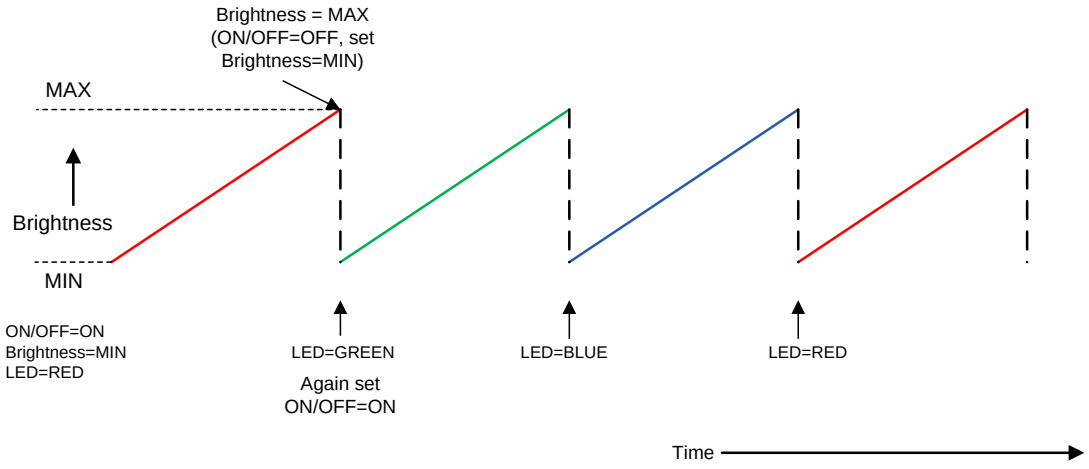
This project is configured to work as a CAN transmitter. The project continuously sends three bytes of data when it is powered, as shown in Figure 2. Data Sent by CAN Tx, to the CAN receiver to the control ON/OFF status, color, and brightness of the LED. The brightness data is incremented and updated in the data frame. Once the brightness reaches the maximum value, the LED is turned off and the color of the LED is changed.

Figure 2. Data Sent by CAN Tx

1st BYTE ON-OFF Status	2nd BYTE Color	3rd BYTE Brightness

The project changes LED color and brightness as shown in Figure 3.

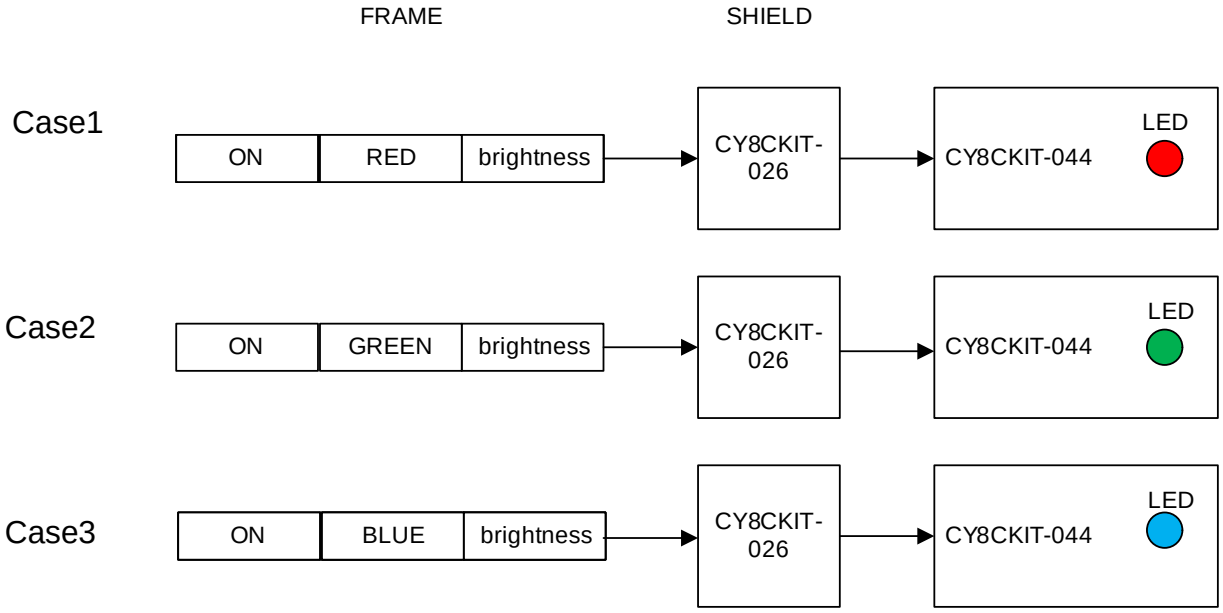
Figure 3. Waveform Representing RGB Pattern



CAN_Rx_PSoC4M Project

This project receives the CAN data sent from the CAN_Tx_FM4 project. The received data is a packet of three bytes consisting of the ON/OFF status, color, and brightness, as Figure 4 shows. The color and brightness of the RGB LED on the receiving CY8CKIT-044 is controlled based on the received data.

Figure 4 Controlling LED According to Received Data Frame.



Design Considerations

CAN_Tx_FM4 Project

FM4 S6E2GM controller supports two channels of CAN. This project uses CAN channel 0. The corresponding Rx and Tx pins are assigned to P16 and P17 respectively. These pins are on CN10_3 and CN7_9 respectively on the FM4 kit.

CAN_Rx_PSoC4M Project

CAN is a fixed-function Component. Dedicated pins must be used. In this project, P0.0 is used for CAN_Rx and P0.1 is used for CAN_Tx.

The UART Component displays the received CAN messages on a serial terminal. UART is used as Tx only, and the UART Tx pin is assigned to P7.1.

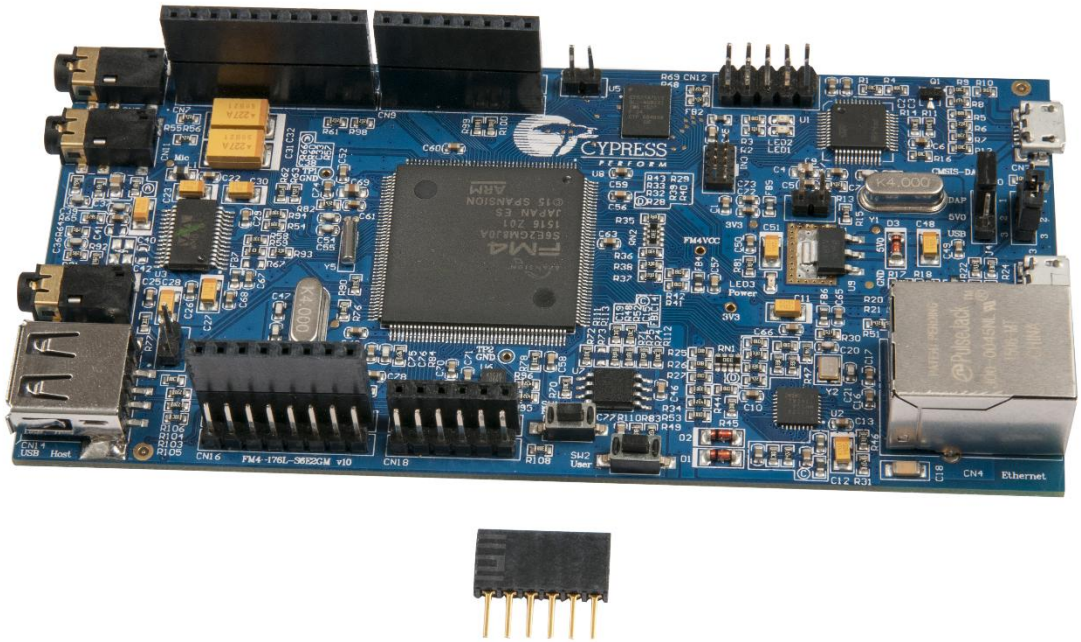
Hardware Setup

CAN Tx Hardware Connections

To use CY8CKIT-026 with an FM4 Kit (FM4 S6E2GM Pioneer Kit), make the following connections:

- To connect CY8CKIT-026 with the FM4 kit, connect extra headers to CY8CKIT-026 as shown in Figure 5.

Figure 5. FM4 Kit with Extra Headers



- Connect CY8CKIT-026 on top of the FM4 kit as shown in Figure 6. CY8CKIT-026 Connected to FM4 Kit
- Make connections for CAN Tx and Rx on CY8CKIT-026 per Table 1.Connections on CY8CKIT and Figure 7

Figure 6. CY8CKIT-026 Connected to FM4 Kit

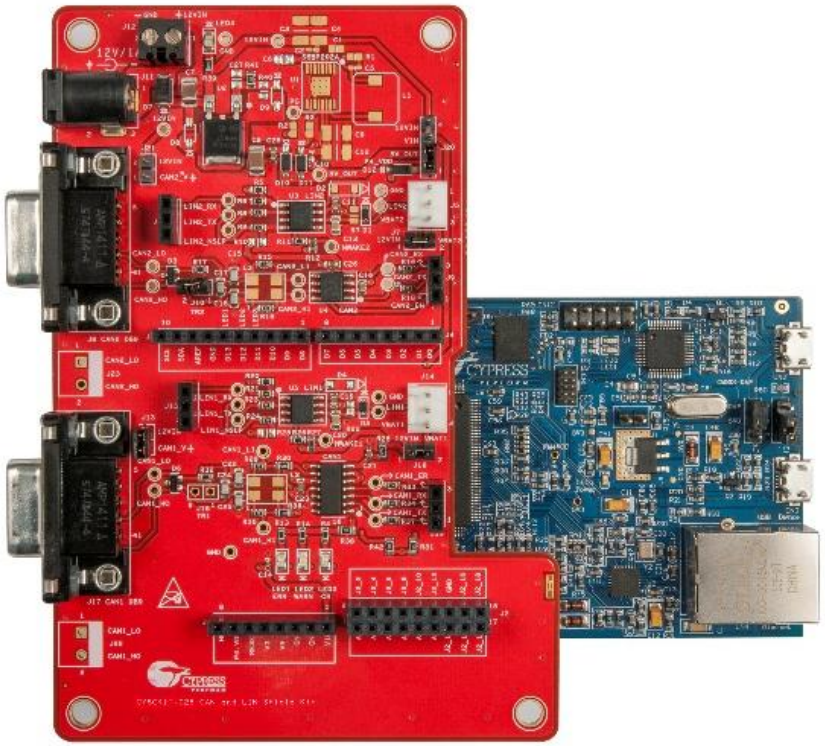


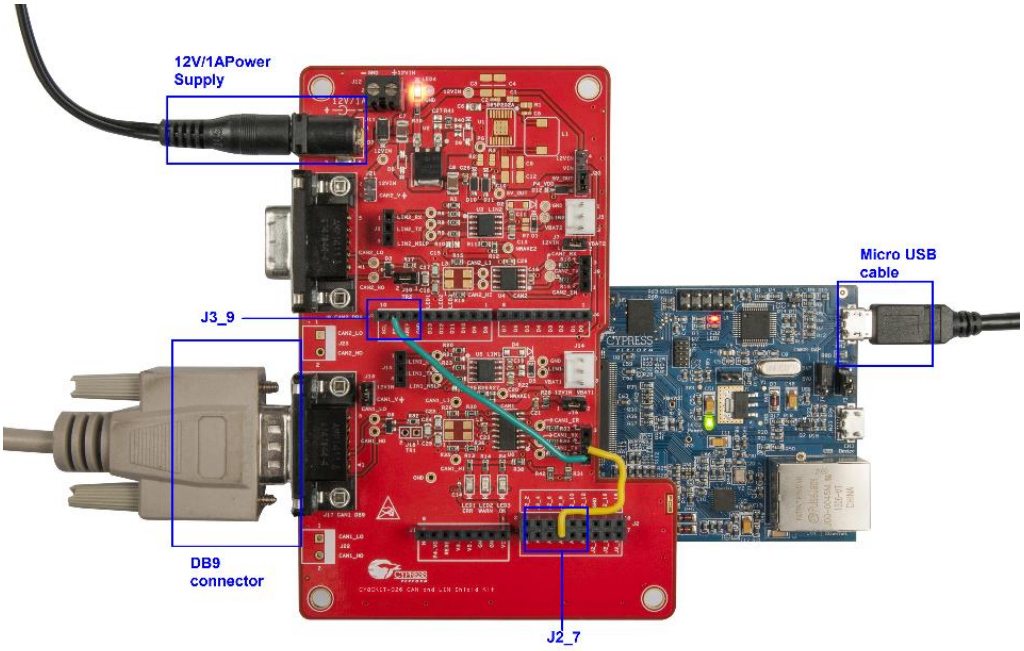
Table 1.Connections on CY8CKIT-026

J2_7 (A3)	J19_2(CAN1_Rx)
J3_9(SDA)	J19_1(CAN1_Tx)

- Connect a micro USB cable to the FM4 kit as shown in Figure 7.
- Place the jumper J13 on CY8CKIT-026.

- Power the Shield kit with a 12-V/1-A input and connect the DB9 connector to CY8CKIT-026 as shown in Figure 7.

Figure 7. CY8KIT-026 and FM4 Kit

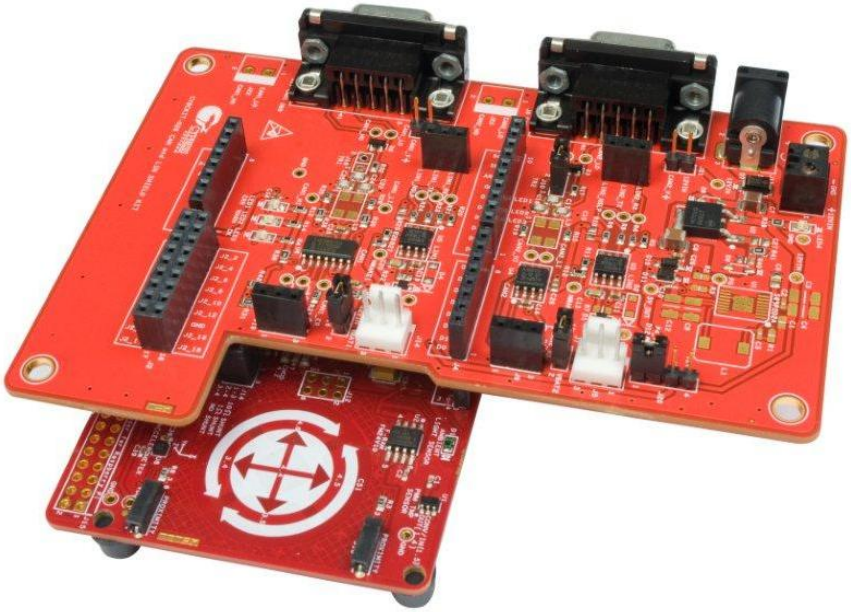


CAN Rx Hardware Connections

To use CY8CKIT-026 with CY8CKIT-044, make the following connections:

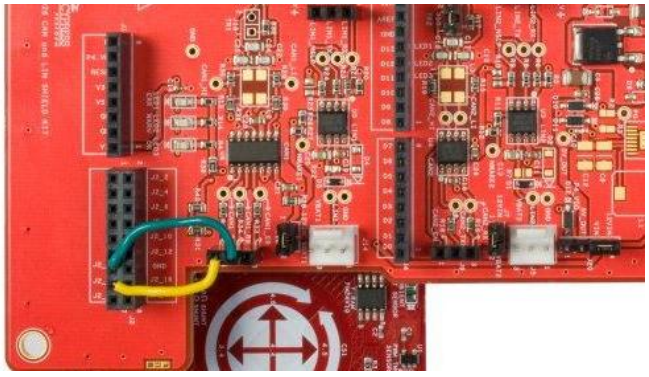
- Make sure that the jumper J9 on the baseboard (CY8CKIT-044) is in the 3.3V position.
- Connect CY8CKIT-026 on top of CY8CKIT-044 as shown in Figure 8.

Figure 8. Connection of CY8CKIT-044 with CY8CKIT-026



- Place the jumper on pin 2 and 3 of the J20 connector.
- Connect J2_13 to J19_2 (CAN1_RX) and J2_15 to J19_1 (CAN1_TX) using connecting wires as shown in Figure 7.

Figure 9. CAN TX and RX Pin Connections



- Connect a USB cable to CY8CKIT-044 for observing the LED data on Terminal.

- Connect a DB9 connector to CY8CKIT-026 as in Figure 10.

Figure 10.DB9 Connector



Components

Table 2. Peripherals Used – CAN_Tx_FM4lists the peripherals used in the CAN_Tx_FM4 project, as well as the hardware resources used by each.

Table 2. Peripherals Used – CAN_Tx_FM4

Peripherals	Hardware Resources
CAN	1 CAN channel
I/Os	2 CAN Rx and Tx

Table 3 lists the PSoC Creator Components used in the CAN_Rx_PSoC4M project, as well as the hardware resources used by each.

Table 3. PSoC Creator Components – CAN_Rx_PSoC4M

Component	Hardware Resources
CAN	1 CAN block
PWMs	3 TCPWMs
UART	1 SCB UART
Pins	2 CANs: Rx and Tx 3 LEDs 2 UARTs: Rx and Tx
Clock	1 Clock Component

Parameter Settings

CAN_Tx_FM4 project

The CAN peripheral block is configured using the associated peripheral registers. See the 32-BIT MICROCONTROLLER FM4 Family Communication Macro Part PERIPHERAL MANUAL for more information on the CAN peripherals. This project uses CAN peripheral and I/O pins. In this project, CAN channel 0 is used for communication. P16 and P17 pins are configured as Rx and Tx respectively.

The following are the steps for configuring CAN and pins:

- Select P16 and P17 using the PFR3 register.
- Configure the selected pins to CAN Rx0_0 and Tx0_0 by using the EPFR09 register. Set bits 26 and 24 of this register to configure the I/O pins as CAN pins.
- Set the clock gating for CAN channel 0 using the CKEN2 register.
- To configure CAN channel 0, first set CAN in initialization mode using the control register CTRLR, and allow access to the Bit Timing Register (BTR) using CTCLR.
- Set the CAN prescalar using the CANPRE register. Writing 0x02u sets CAN Prescaler to 1/4 and this CAN System Clock to 40 MHz based on PLL=160 MHz.
- Write 0x4993u to the Bit Timing Register (BTR) .This sets the baud rate to 125 kbps with the following CAN controller parameters for bit timing:

  Tseg1=10

  Tseg2=3

  SJW=3

  BRP=16

- Use the CTCLR register to change CAN to run mode and disable the access to the BTR register.

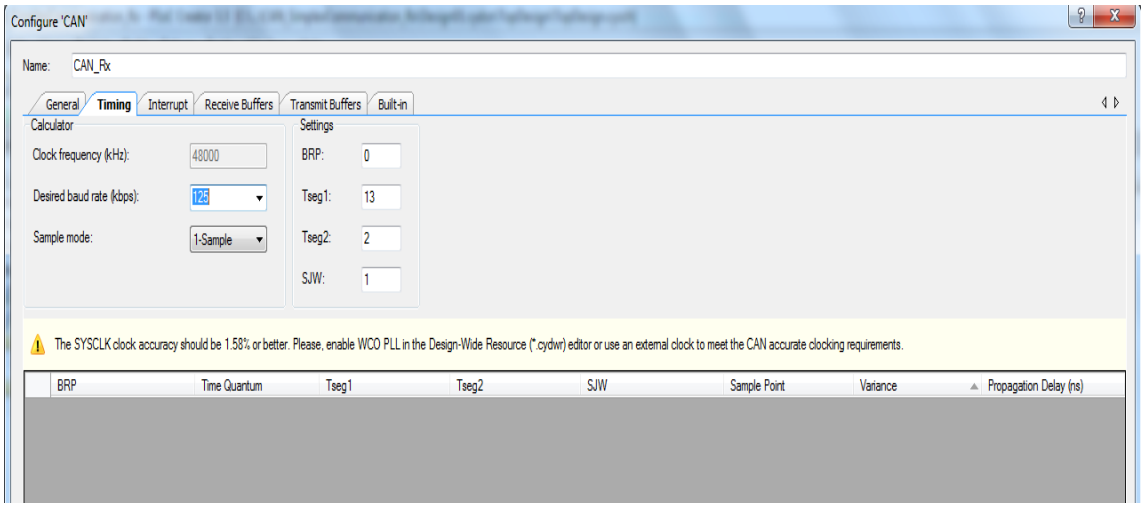
CAN_Rx_PSoC4M Project

The following are the steps for configuring the CAN Component in PSoC Creator for PSoC 4M:

- Configure the clocks to use Trim with WCO option.

The CAN Component needs a clock with an accuracy of ±1.58 percent or better. However, the internal main oscillator (IMO) in PSoC 4 M is 2 percent accurate; therefore, you must either trim the IMO or use an external clock. These settings must be done before the CAN Component is configured because the timing section in the Component configuration will be updated based on this clock. If the clock is not selected before the Component is configured, the Component displays a warning to do so, as Figure 11 shows.

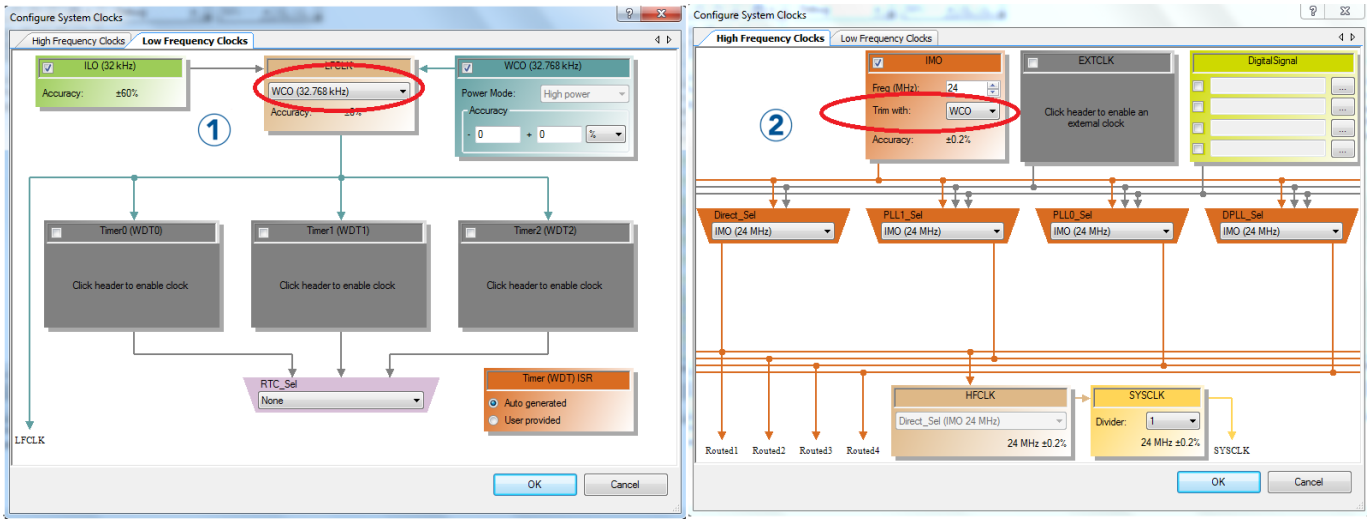
Figure 11. CAN Component Clock Accuracy Warning



In the clock configuration wizard of the *CAN_Rx_PSoC4M.cydwr* file, the IMO can be trimmed to an accuracy of 0.5 percent using the watch crystal oscillator (WCO) in PSoC 4 M. To trim the IMO, first select LFCLK as the WCO and select **Trim with WCO** in the **IMO** section, as

Figure 12 shows.

Figure 12. Clock Configuration



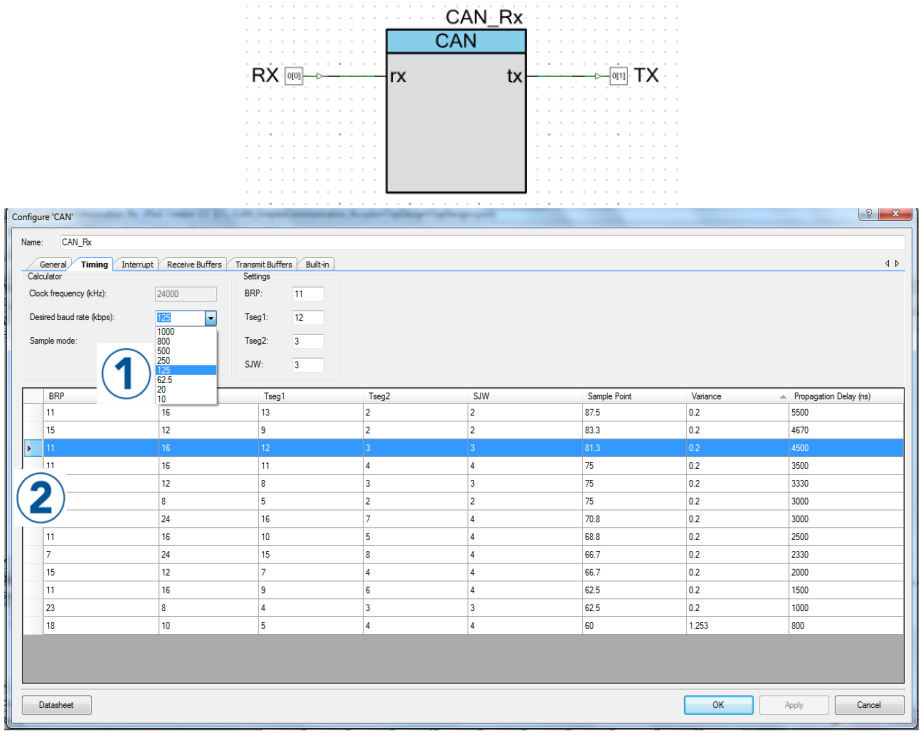
- Configure the CAN Component

Drag and drop a CAN Component from the Component Catalog to TopDesign. Double-click the Component to configure it.

The settings in the **General** tab are left with the default selections.

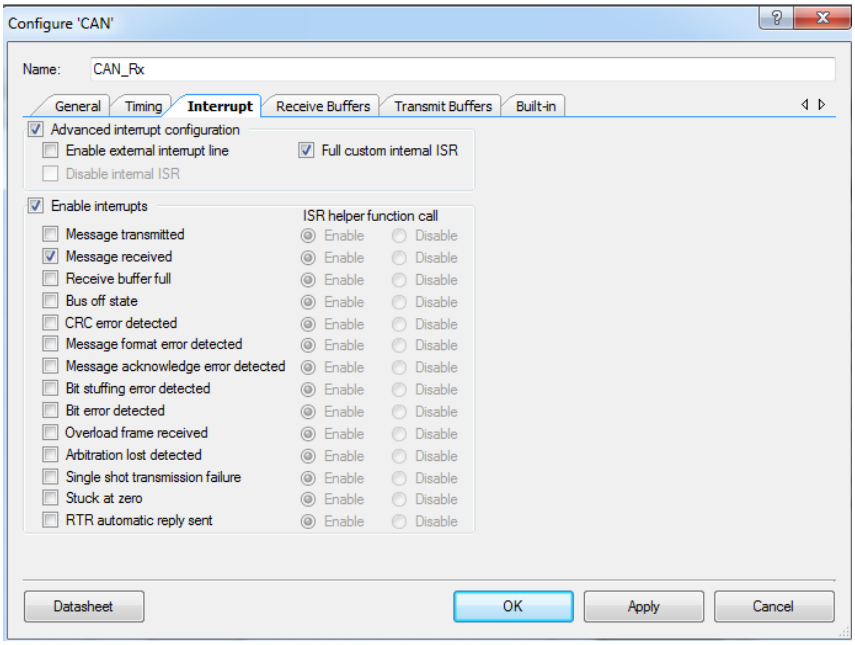
In the **Timing** tab, select the **Desired baud rate**. The table with different timing parameters will be updated based on the selected baud rate. Then, select one of the rows with the criteria that **Variance** is minimal and **Sample point** is around 80 percent, as shown in Figure 13. Double-click on that row to make sure that the values are updated in the settings section.

Figure 13. CAN_Rx Configuration – Timing Tab



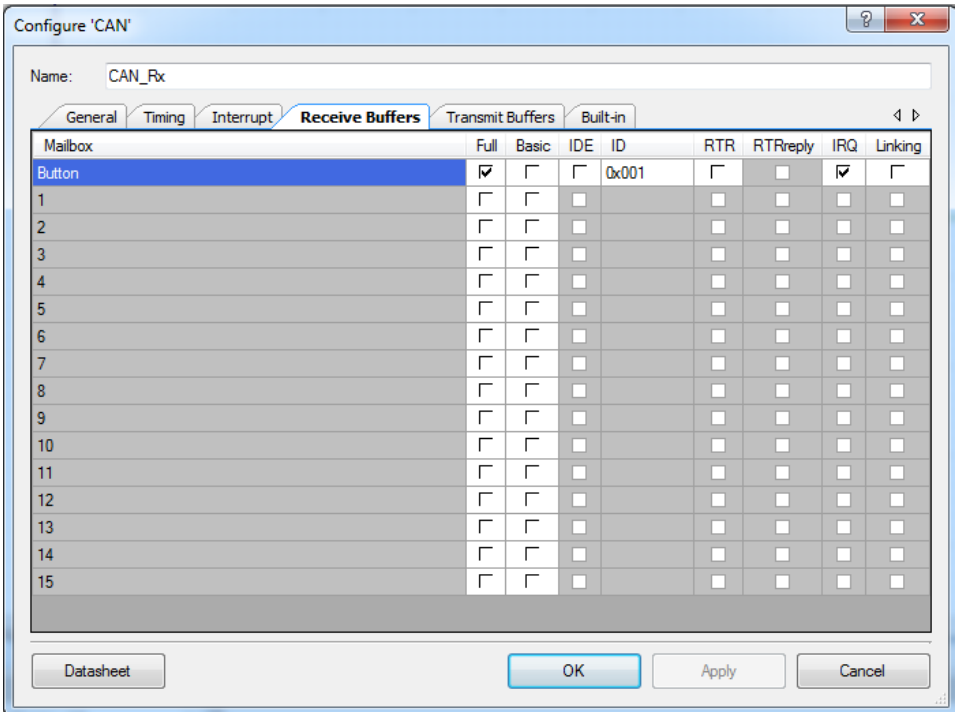
In the Interrupt tab, enable **Advanced interrupt configuration** and select **Full custom internal ISR,** as shown in Figure 14

Figure 14. CAN_Rx Configuration – Interrupt Tab



In the Receive Buffers tab, add a **Full** Rx mailbox with the mailbox **ID** as 0x01. Rename the mailbox to “Button,” as shown in Figure 15. **IRQ** is selected by default because the **Message received** interrupt is selected in the Interrupt tab.

Figure 15. CAN_Rx Configuration – Receive Buffers Tab



The **Transmit Buffers** tab is left with the default settings.

Design-Wide Resources

Figure 16 shows the CAN_Rx_PSoC4M project pin assignment.

Figure 16. CAN_Rx_PSoC4M – Pin Assignment

Name	Port	Pin	Lock
\UART_Tx:tx\	P7[1]	38	☑
LED_Blue	P6[5]	16	☑
LED_Green	P2[6]	8	☑
LED_Red	P0[6]	45	☑
RX	P0[0]	39	☑
TX	P0[1]	40	☑

Operation

CAN_Tx_FM4

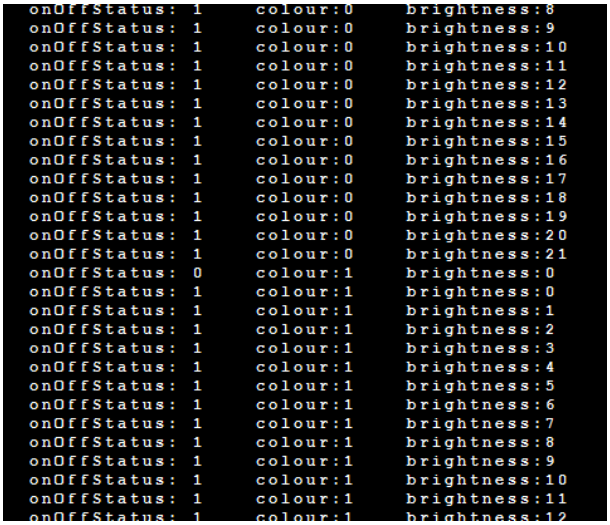
Program the CAN_Tx_FM4 project into the FM4 pioneer kit that is used as CAN Tx. Press reset, using SW1, on the kit.

CAN_Rx_PSOC4M

Program the CAN_Rx_PSoC4M project into the CY8CKIT-044 that is used as CAN Rx.

- Confirm that the LED color and brightness change as Figure 2 shows.
- Confirm on your PC that the UART outputs data similar to that shown in Figure 17.

Figure 17. UART Data on HyperTerminal



Related Documents

Table 3 lists the relevant application notes, code examples, knowledge base articles, device datasheets, and Component datasheets.

Table 3. Related Documents

Application Notes		
AN79953	Getting Started with PSoC 4	Introduces you to PSoC 4, an ARM® Cortex™-M0 MCU based programmable system-on-chip. It helps you explore the architecture and PSoC Creator development tools.
Code Examples		
CE95351 – Fixed Function PWM with PSoC 4		
CE97311 – PSoC® 4 M: CAN Simplex Communication with CapSense®		
PSoC Creator Component Datasheets		
Controller Area Network (CAN)		
Device Documentation		
PSoC 4 Datasheets		PSoC 4 Technical Reference Manuals
S6E2GM Series 32-BIT ARM CORTEX-M4F FM4 MICROCONTROLLER Datasheet		32-BIT MICROCONTROLLER FM4 Family PERIPHERAL MANUAL
32-BIT MICROCONTROLLER FM4 Family Communication Macro Part PERIPHERAL MANUAL		
Development Kit (DVK) Documentation		
CY8CKIT-044 –PSoC 4 M-Series Pioneer Kit		
FM4-176L-S6E2GM - ARM® Cortex®-M4 MCU Pioneer Kit with Ethernet and USB Host		
CY8CKIT-026		

Document History

Document Title: CE211027 – CAN Communication Between FM4-S6E2Gx Series and PSoC® 4 M-Series Using CY8CKIT-026 CAN and LIN Shield Kit

Document Number: 002-11027

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	5162311	HJPN	03/04/2016	New code example
*A	5718713	AESATMP9	05/11/2017	Updated logo and copyright.

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

Cypress Developer Community

Forums | WICED IOT Forums | 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, 2016-2017. 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. 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.