CY8CKIT-042-BLE – F-RAM Data Logger ®

Features

- Opamp as non-inverting amplifier
- ADC used in single ended mode
- HyperTerminal displays ADC results sent via UART
- LED indicates whenADC input is outside defined voltage window
- Analog Muxes multiplex three inputs and their corresponding gains
- Debouncer detects valid switch press
- F-RAM Data Logger

General Description

This example project is based on a PSoC Creator starter design for the PSoC 4 device. It demonstrates how F-RAM can be used with the PSoC to capture and log the analog routing capability of PSoC 4 in the real time. In this project, every input has corresponding dedicated feedback networks and hence different gains. The user can change the active channel by pressing a switch. In addition the data is recorded to the on-board F-RAM device which implements a rolling buffer.

Development Kit Configuration

This example project is designed to run on the CY8CKIT-042 kit from Cypress Semiconductor. A description of the kit, along with more example programs and ordering information, can be found at http://www.cypress.com/go/cy8ckit-042.

The project requires configuration settings changes to run on other kits from Cypress Semiconductor. Table 1 is the list of the supported kits. To switch from CY8CKIT-042 to any other kit, change the project's device with the help of Device Selector called from the project's context menu.

Table 1. Development Kits vs Parts

Development Kit	Device
CY8CKIT-042	CY8C4245AXI-483
CY8CKIT-042-BLE	CY8C4247LQI-BL483
CY8CKIT-044	CY8C4247AZI-M485
CY8CKIT-046	CY8C4248BZI-L489

The pin assignments for the supported kits are in Table 2.

Pin Name	Development Kit			
	CY8CKIT-042	CY8CKIT-042 BLE	CY8CKIT-044	CY8CKIT-046
A_Out	P1[2]	P1[2]	P1[2]	P1[2]
\UART:tx\	P0[5]*	P1[5]	P7[1]	P3[1]
LED	P1[6]	P2[6]	P0[6]	P5[2]
Input_1	P2[0]	P2[0]	P2[0]	P2[0]
Input_2	P2[1]	P2[1]	P2[1]	P2[1]
Input_3	P2[2]	P2[2]	P2[2]	P2[2]
G1	P2[3]	P2[3]	P2[3]	P2[3]
G2	P2[4]	P2[4]	P2[4]	P2[4]
G3	P2[5]	P2[5]	P2[5]	P2[5]
In_Sel	P0[7]	P2[7]	P0[7]	P0[7]

* Connect P0[5] to pin P12[6] on header J8.

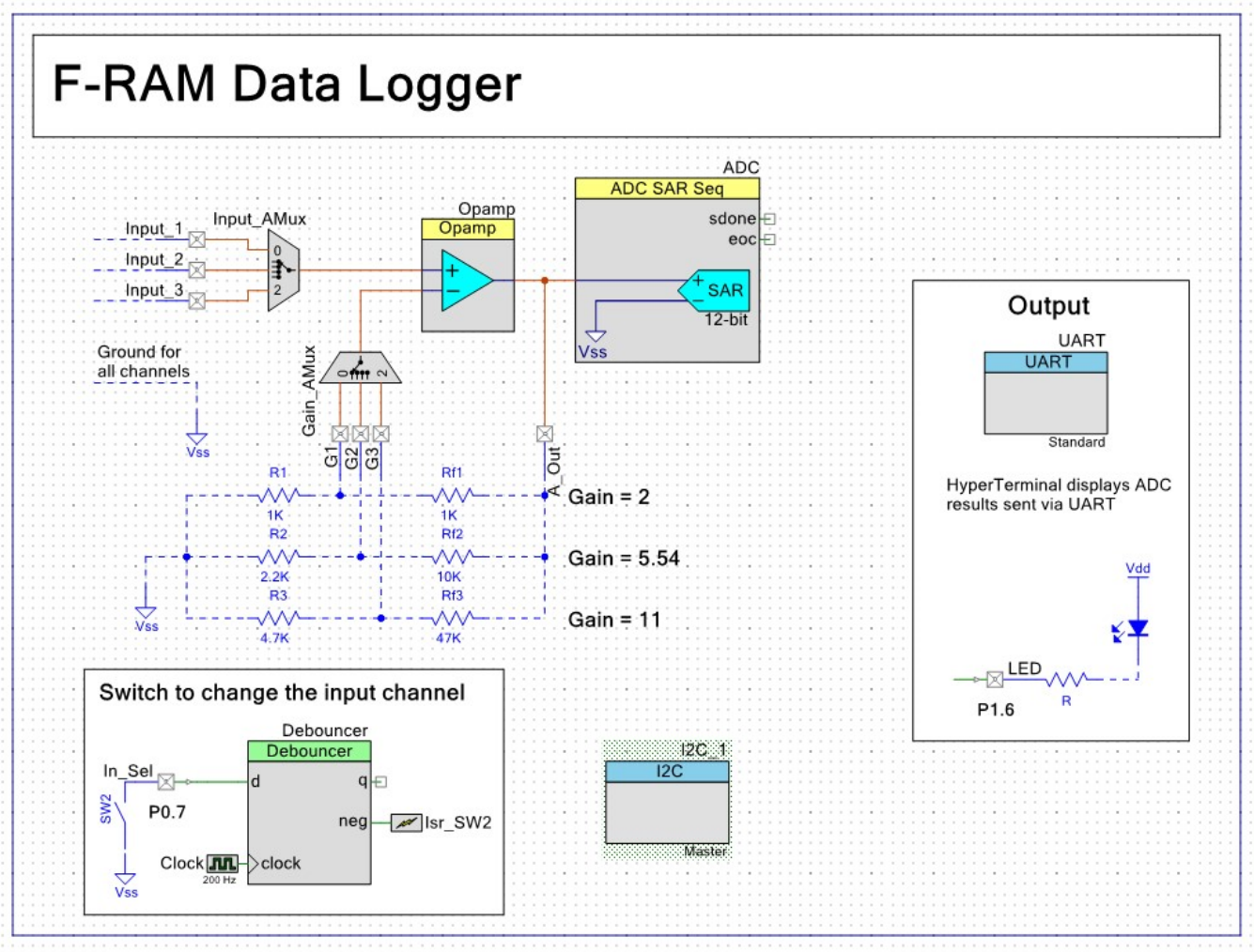
The following configuration instructions provide a guideline to test this design. For simplicity, the instructions describe the stepwise process to be followed when testing this design with the PSoC 4 Pioneer Kit (CY8CKIT-042).

1. Set jumper J9 (J16 for CY8CKIT-042-BLE) to 5.0V position.
2. Connect three input signals to Input_1, Input_2, Input_3.
3. Connect all the external resistors as shown in the top design schematic.
4. Connect a USB cable to the PSoC 4 Pioneer Kit DVK and PC with the HyperTerminal program.

Project Configuration

This example project consists of ADC SAR Seq, Opamp, AMuxSeq, UART, and Debouncer components. The top design schematic is shown in Figure 1. The Opamp is used to amplify the input signal; the input channel is selected using Input_AMux and the feedback network is selected using Gain_AMux. A UART is used to send ADC results to HyperTerminal. Debouncer is used to remove glitches from the input switch. The SAR ADC converts the analog output of the Opamp, into digital values. The ADC also generates an interrupt when its input is outside the defined voltage window (250mV – 750mV). The LED turns on when the ADC generates this interrupt. The UART component sends the ADC output of the active channel along with the channel number to HyperTerminal.

Figure 1. Top Design Schematic



The Opamp is configured in high stability, high power, 10mA output current mode. The ADC is configured in single ended mode. The ADC averages 256 consecutive samples to produce the final result. The ADC component configuration is shown in Figure 2. Switch SW2 and pin In_Sel is used to change the analog mux channels, Input_AMux selects the input channel and Gain_AMux selects the corresponding gain. The I2C Master is used to move data to and from the F-RAM device. The I2C component configuration is shown in Figure 3.

Figure 2. ADC Configuration Window

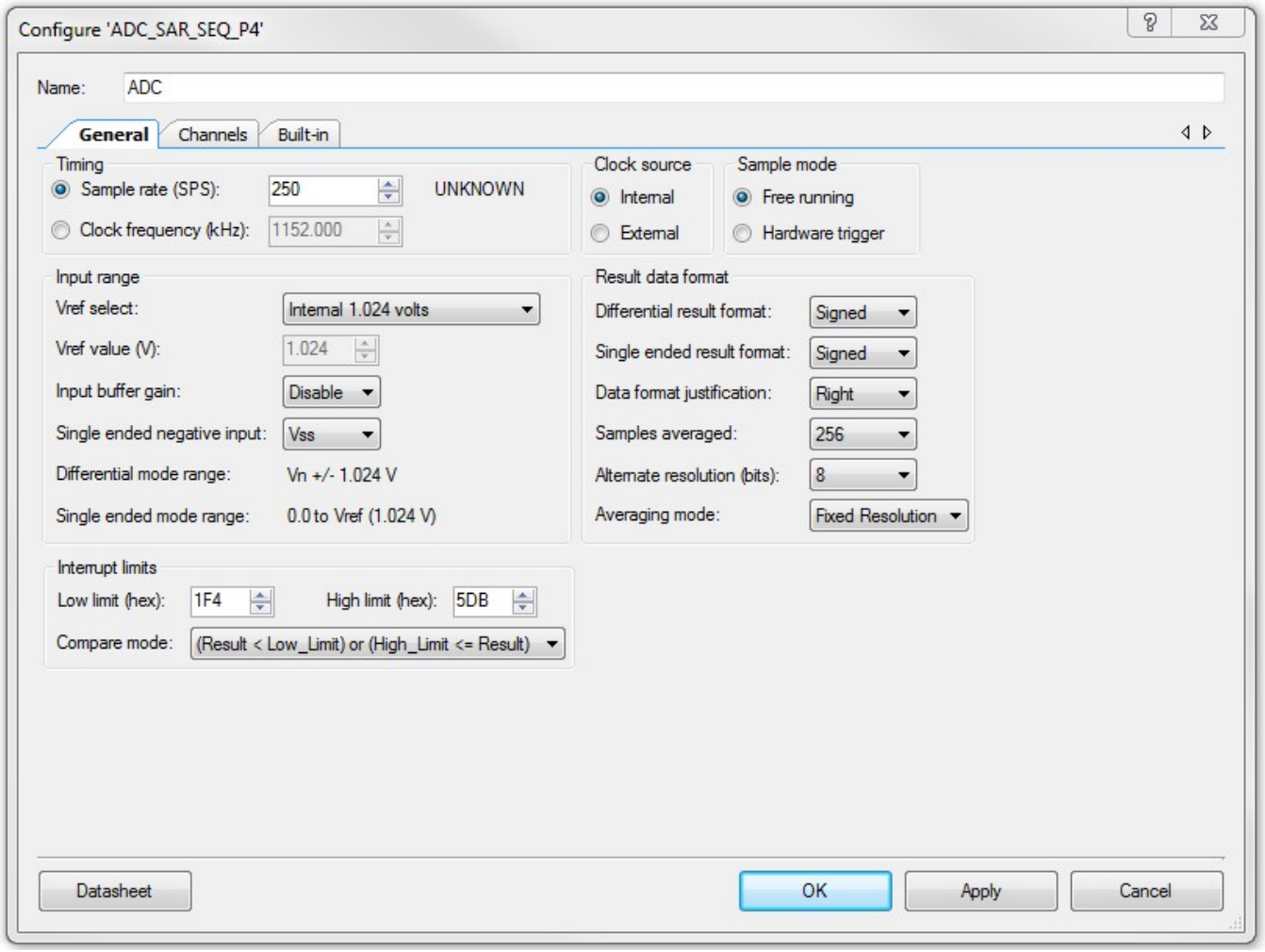
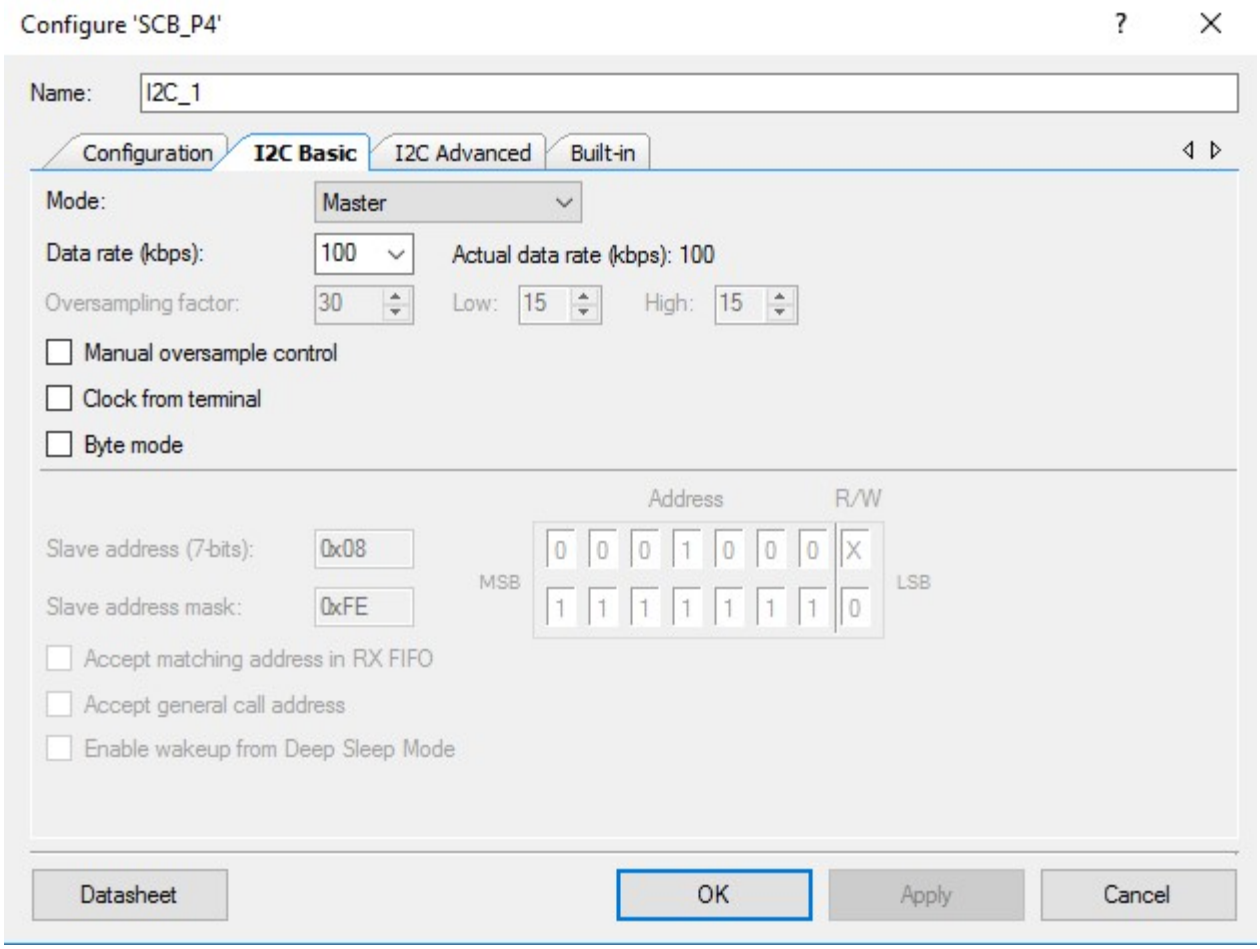


Figure 3. ADC Configuration Window



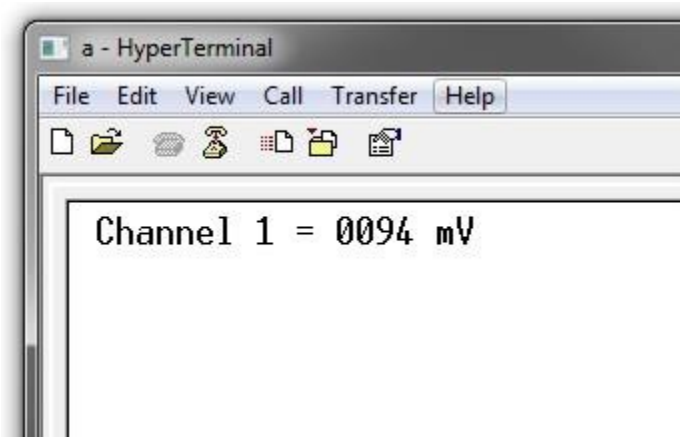
Project Description

In the main function all components are started, both the analog muxes are initialized to select the channel zero and ADC conversion is started. The “for” loop in the main.c waits for the ADC to finish conversion. When the ADC result is available, it is sent through a UART to HyperTerminal. The ADC continuously generates an interrupt when its input is outside the defined voltage window (250mV –750mV). This interrupt is used to control an LED. This LED is turned ON when the ADC input is outside the window.

Expected Results

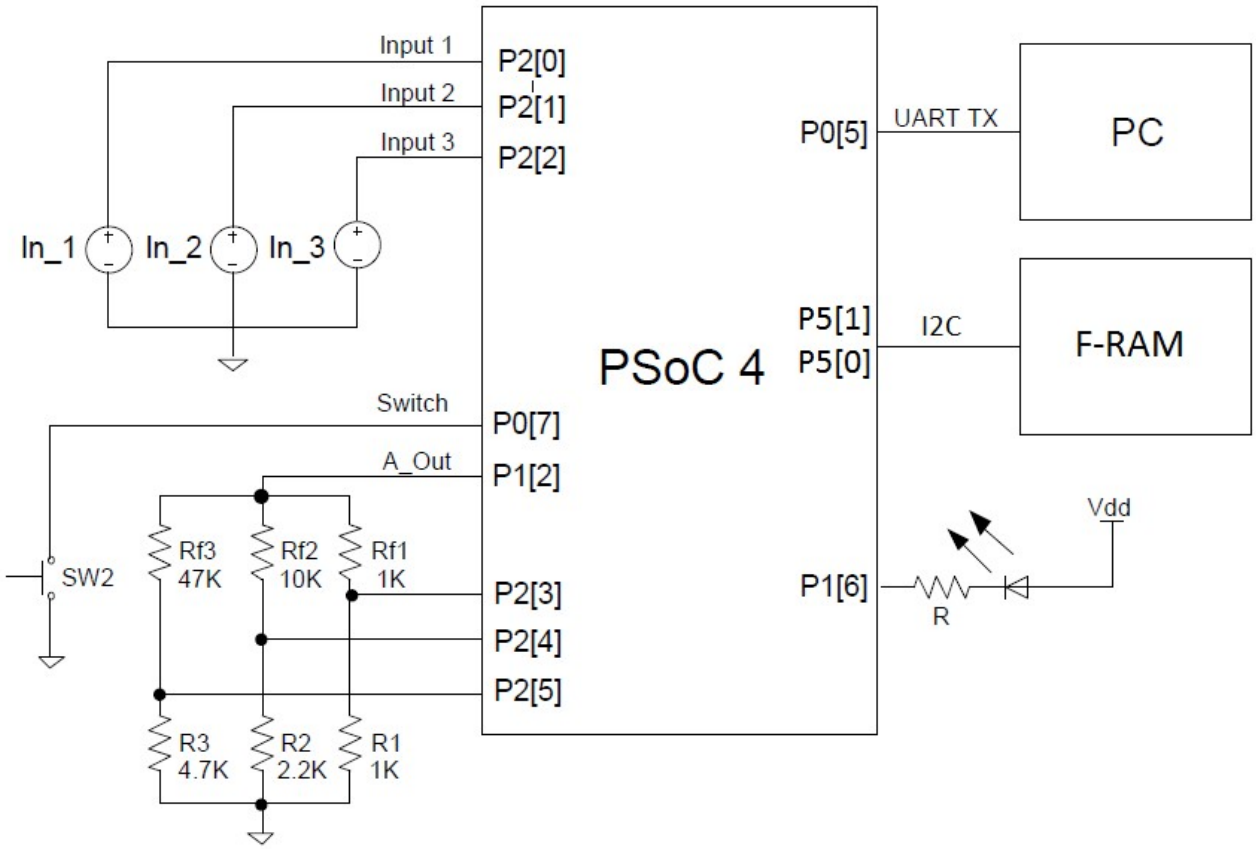
The analog input channel should change when switch SW2 is pressed. HyperTerminal displays the active channel and its input voltage. The LED will turn on when the ADC result is outside the voltage window (250mV – 750mV).

Figure 3. Result



Schematic

Figure 4. Connection Schematic



Note: If the LED is active HIGH, replace LOW with HIGH to turn the LED ON and vice-versa. The Section of the code that needs to be changed in “main.c” is mentioned below:

```
/* Turn ON the LED when the input is outside the window (250mV - 750mV) */
LED_Write(LOW);

/* Turn OFF the LED when the input is within the defined window (250mV - 750mV) */
LED_Write(HIGH)
```

Using UART to communicate with PC Host

This example project communicates with the PC host using a UART. The HyperTerminal program is required in the PC to communicate with PSoC 4. If you don’t have the HyperTerminal program installed, download, and install any serial port communication program. Free wares such as HyperTerminal, Bray’s Terminal etc. are available on the Web.

Follow these steps to communicate with the PC host.

1. Connect the USB cable between the PC and PSoC 4 Pioneer Kit.

2. Open the device manager program in your PC, find the COM port in which the PSoC 4 is connected, and note the port number.

3. Open the HyperTerminal program and select the COM port in which the PSoC 4 is connected.

4. Configure the Baud rate, Parity, Stop bits, and Flow control information in the HyperTerminal configuration window. These settings should match the configuration of the PSoC Creator UART component in the project

5. Start communicating with the device as explained in the project description.

Using I2C to communicate with the F-RAM

This example project communicates with the F-RAM using I2C. I2C is a simple Clock/Data Bus using 8 bit serial words. Figures 5 and 6 so the Write and Read operation for the I2C master to access the F-RAM device.

Figure 5. I2C F-RAM Write

```
/*******************************************************************************
```

```
* Function Name: FRAM_Write
********************************************************************************
*
* Summary:
*  F-RAM Byte Write Command
* uint16 address - address in the F-RAM array
* uint8 data - data to be written
*
*******************************************************************************/
void FRAM_Write(uint16 address, uint8 data)
{
    uint8 txBuffer[BUFFER_SIZE];

    txBuffer[0] = address >> 8;
    txBuffer[1] = address;
    txBuffer[2] = data;


    /* Clear any previous status */
    I2C_1_I2CMasterClearStatus();

    /* I2C Wrtie Command */
    status = I2C_1_I2CMasterWriteBuf(SLAVE_ADDRESS, (uint8 *) txBuffer, BUFFER_SIZE,
I2C_1_I2C_MODE_COMPLETE_XFER);

    for(;;)
    {
        if(0u != (I2C_1_I2CMasterStatus() & I2C_1_I2C_MSTAT_WR_CMPLT))
        {
            /* Transfer complete. Check Master status to make sure that transfer
            completed without errors. */
            break;
        }
    }
}
```

Figure 6. I2C F-RAM Read

```
/*******************************************************************************
* Function Name: FRAM_Read
********************************************************************************
*
* Summary:
*  F-RAM Byte Read Command
* uint16 address - address in the F-RAM array
* uint8 retrun - data to be read
*
*******************************************************************************/
uint8 FRAM_Read(uint16 address)
{
    uint8 rxBuffer[BUFFER_SIZE-2];
    uint8 txBuffer[2];

    txBuffer[0] = address >> 8;
    txBuffer[1] = address;

    /* Clear any previous status */
    I2C_1_I2CMasterClearStatus();

    /* I2C Wrtie Command to Set the Address*/
    status = I2C_1_I2CMasterWriteBuf(SLAVE_ADDRESS, (uint8 *) txBuffer, 0x2,
I2C_1_I2C_MODE_COMPLETE_XFER);
    for(;;)
    {
        if(0u != (I2C_1_I2CMasterStatus() & I2C_1_I2C_MSTAT_WR_CMPLT))
        {
```

```
            /* Transfer complete. Check Master status to make sure that transfer
            completed without errors. */
            break;
        }
    }

    /* Clear any previous status */
    I2C_1_I2CMasterClearStatus();


    /* I2C Read Command */
    status = I2C_1_I2CMasterReadBuf(SLAVE_ADDRESS, (uint8 *) rxBuffer, 1, I2C_1_I2C_MODE_COMPLETE_XFER);
    for(;;)
    {
        if(0u != (I2C_1_I2CMasterStatus() & I2C_1_I2C_MSTAT_RD_CMPLT))
        {
            /* Transfer complete. Check Master status to make sure that transfer
            completed without errors. */
            break;
        }
    }
    return rxBuffer[0];
}
```

Document History

Document Title: CY8CKIT-042-BLE – F-RAM Data Logger ®

Document Number: 002-10988

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	5112947	JLTO	01/28/16	New spec

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

Automotive	cypress.com/go/automotive
Clocks & Buffers	cypress.com/go/clocks
Interface	cypress.com/go/interface
Lighting & Power Control	cypress.com/go/powerpsoc cypress.com/go/plc
Memory	cypress.com/go/memory
PSoC	cypress.com/go/psoc
Touch Sensing	cypress.com/go/touch
USB Controllers	cypress.com/go/usb
Wireless/RF	cypress.com/go/wireless

PSoC® Solutions

psoc.cypress.com/solutions

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

Cypress Developer Community

Community | Forums |Blogs | Video |Training

Technical Support

cypress.com/go/support

PSoC is a registered trademark and PSoC Creator is a trademark of Cypress Semiconductor Corp. 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

Phone : 408-943-2600
Fax : 408-943-4730
Website : www.cypress.com

© Cypress Semiconductor Corporation, 2016. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, life saving, critical control or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.
This Source Code (software and/or firmware) is owned by Cypress Semiconductor Corporation (Cypress) and is protected by and subject to worldwide patent protection (United States and foreign), United States copyright laws and international treaty provisions. Cypress hereby grants to licensee a personal, non-exclusive, non-transferable license to copy, use, modify, create derivative works of, and compile the Cypress Source Code and derivative works for the sole purpose of creating custom software and or firmware in support of licensee product to be used only in conjunction with a Cypress integrated circuit as specified in the applicable agreement. Any reproduction, modification, translation, compilation, or representation of this Source Code except as specified above is prohibited without the express written permission of Cypress.
Disclaimer: CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes without further notice to the materials described herein. Cypress does not assume any liability arising out of the application or use of any product or circuit described herein. Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress' product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.
Use may be limited by and subject to the applicable Cypress software license agreement.