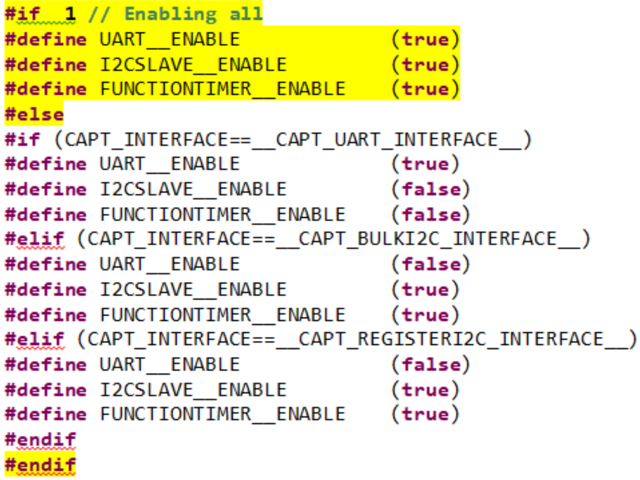
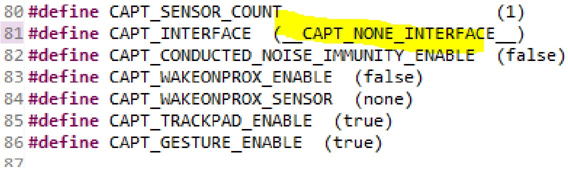
How to configure Captivate Library to use UART with host MCU.

Modify the CAPT_CommConfig.h as show in the highlighted text. This is the trick that allows you to re-use the I2C and UART drivers that are normally dedicated to communications with the GUI.



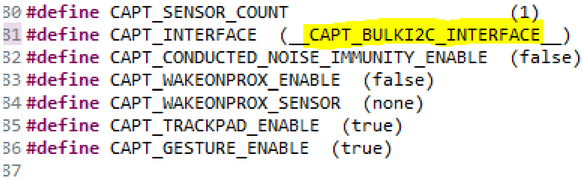
```
#if  1 // Enabling all
#define UART__ENABLE                (true)
#define I2CSLAVE__ENABLE            (true)
#define FUNCTIONTIMER__ENABLE       (true)
#else
#if (CAPT_INTERFACE==__CAPT_UART_INTERFACE__)
#define UART__ENABLE                (true)
#define I2CSLAVE__ENABLE            (false)
#define FUNCTIONTIMER__ENABLE       (false)
#elif (CAPT_INTERFACE==__CAPT_BULKI2C_INTERFACE__)
#define UART__ENABLE                (false)
#define I2CSLAVE__ENABLE            (true)
#define FUNCTIONTIMER__ENABLE       (true)
#elif (CAPT_INTERFACE==__CAPT_REGISTERI2C_INTERFACE__)
#define UART__ENABLE                (false)
#define I2CSLAVE__ENABLE            (true)
#define FUNCTIONTIMER__ENABLE       (true)
#endif
#endif
```

Next, modify CAPT_UserConfig.h as shown, if no longer using CapTIvate Design Center and only using the UART to communicate with host MCU.



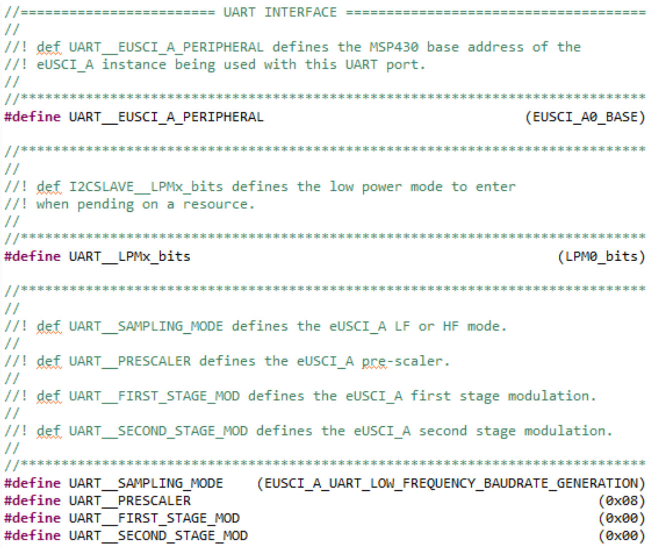
```
#define CAPT_SENSOR_COUNT                        (1)
#define CAPT_INTERFACE   (__CAPT_NONE_INTERFACE__)
#define CAPT_CONDUCTED_NOISE_IMMUNITY_ENABLE   (false)
#define CAPT_WAKEONPROX_ENABLE  (false)
#define CAPT_WAKEONPROX_SENSOR  (none)
#define CAPT_TRACKPAD_ENABLE  (true)
#define CAPT_GESTURE_ENABLE  (true)
```

Else, modify CAPT_UserConfig.h as shown using I2C with the CapTIvate design center to continue to tune sensors while using UART to communicate with host MCU.

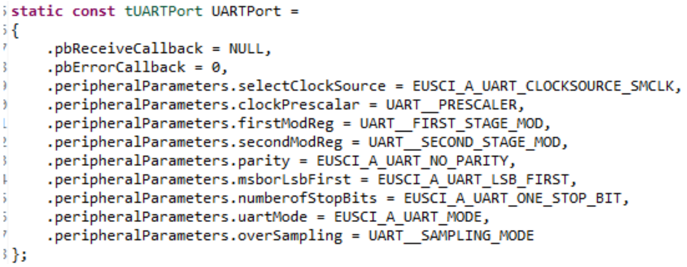


```
#define CAPT_SENSOR_COUNT                        (1)
#define CAPT_INTERFACE   (__CAPT_BULKI2C_INTERFACE__)
#define CAPT_CONDUCTED_NOISE_IMMUNITY_ENABLE   (false)
#define CAPT_WAKEONPROX_ENABLE  (false)
#define CAPT_WAKEONPROX_SENSOR  (none)
#define CAPT_TRACKPAD_ENABLE  (true)
#define CAPT_GESTURE_ENABLE  (true)
```

Because the UART is not selected as the CAPT_INTERFACE, you will need to copy the tUartPort and #defines that are located in CAPT_Interface_definitions.h (shown here), and paste somewhere in your project (maybe same file as your sensor callback function).



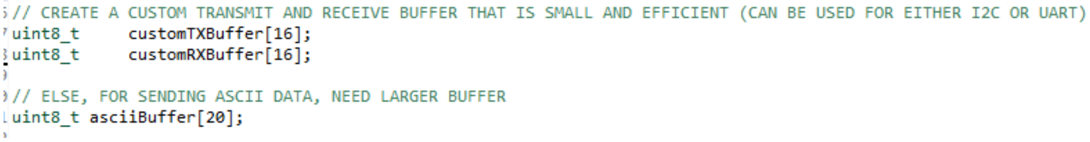
```
//======================= UART INTERFACE ====================================
//
//! def UART__EUSCI_A_PERIPHERAL defines the MSP430 base address of the
//! eUSCI_A instance being used with this UART port.
//
//*****************************************************************************
#define UART__EUSCI_A_PERIPHERAL                            (EUSCI_A0_BASE)

//*****************************************************************************
//
//! def I2CSLAVE__LPMx_bits defines the low power mode to enter
//! when pending on a resource.
//
//*****************************************************************************
#define UART__LPMx_bits                                         (LPM0_bits)

//*****************************************************************************
//
//! def UART__SAMPLING_MODE defines the eUSCI_A LF or HF mode.
//
//! def UART__PRESCALER defines the eUSCI_A pre-scaler.
//
//! def UART__FIRST_STAGE_MOD defines the eUSCI_A first stage modulation.
//
//! def UART__SECOND_STAGE_MOD defines the eUSCI_A second stage modulation.
//
//*****************************************************************************
#define UART__SAMPLING_MODE    (EUSCI_A_UART_LOW_FREQUENCY_BAUDRATE_GENERATION)
#define UART__PRESCALER                                                (0x08)
#define UART__FIRST_STAGE_MOD                                          (0x00)
#define UART__SECOND_STAGE_MOD                                         (0x00)
```



```
static const tUARTPort UARTPort =
{
    .pbReceiveCallback = NULL,
    .pbErrorCallback = 0,
    .peripheralParameters.selectClockSource = EUSCI_A_UART_CLOCKSOURCE_SMCLK,
    .peripheralParameters.clockPrescalar = UART__PRESCALER,
    .peripheralParameters.firstModReg = UART__FIRST_STAGE_MOD,
    .peripheralParameters.secondModReg = UART__SECOND_STAGE_MOD,
    .peripheralParameters.parity = EUSCI_A_UART_NO_PARITY,
    .peripheralParameters.msborLsbFirst = EUSCI_A_UART_LSB_FIRST,
    .peripheralParameters.numberofStopBits = EUSCI_A_UART_ONE_STOP_BIT,
    .peripheralParameters.uartMode = EUSCI_A_UART_MODE,
    .peripheralParameters.overSampling = UART__SAMPLING_MODE
};
```

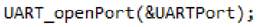
You will need to determine the appropriate settings for the UART (the 4 #defines above) with your clock system and desired baud rate. Refer to the MSP430FR2xx_4xx family users guide, chapter 22 (refer to table 22-5).
http://www.ti.com/lit/ug/slau445h/slau445h.pdf

You will also need to create a communications buffer something like:



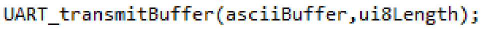
```
// CREATE A CUSTOM TRANSMIT AND RECEIVE BUFFER THAT IS SMALL AND EFFICIENT (CAN BE USED FOR EITHER I2C OR UART)
uint8_t     customTXBuffer[16];
uint8_t     customRXBuffer[16];

// ELSE, FOR SENDING ASCII DATA, NEED LARGER BUFFER
uint8_t asciiBuffer[20];
```

Somewhere early in your code you will need to initialize the UART, passing the address of the UARTPort structure you copied:



```
UART_openPort(&UARTPort);
```

Then lastly, populate your buffer and transmit the data in your sensor callback function:



```
UART_transmitBuffer(asciiBuffer,ui8Length);
```