Avnet Visible Things
I2C Example
by
Michael C. Li
https://www.miketechuniverse.com/
4/5/2018

e2 Studio Version: 5.3.1.002

Project Summary

Board:	gw002_rev1_3
Device:	R7FS7G27H2A01CBD
Toolchain:	GCC ARM Embedded
Toolchain Version:	4.9.3.20150529
SSP Version:	1.2.0

PMOD Configuration for this example

Gateway PMOD	I2C I2C	Type 1 GPIO	Type 2 SPI	Type 2A Expanded SPI	Type 3 UART	Type 4 UART	Type 4A Expanded UART	Type 5 H-Bridge	Type 6 Dual H-Bridge
1	Y	Y	Y	Y	Y	Y	Y	N	N
2	Y	Y	Y	Y	Y	Y	Y	N	N
3	Y	Y	Y	Y	Y	Y	Y	N	N
4	Y	Y	Y	Y	Y	Y	Y	N	N
5	N	Y	Y	Y	N	Y	Y	Y	Y
6	Y	Y	N	N	N	N	N	N	N

Table 3 – Pmod Compatibility Chart

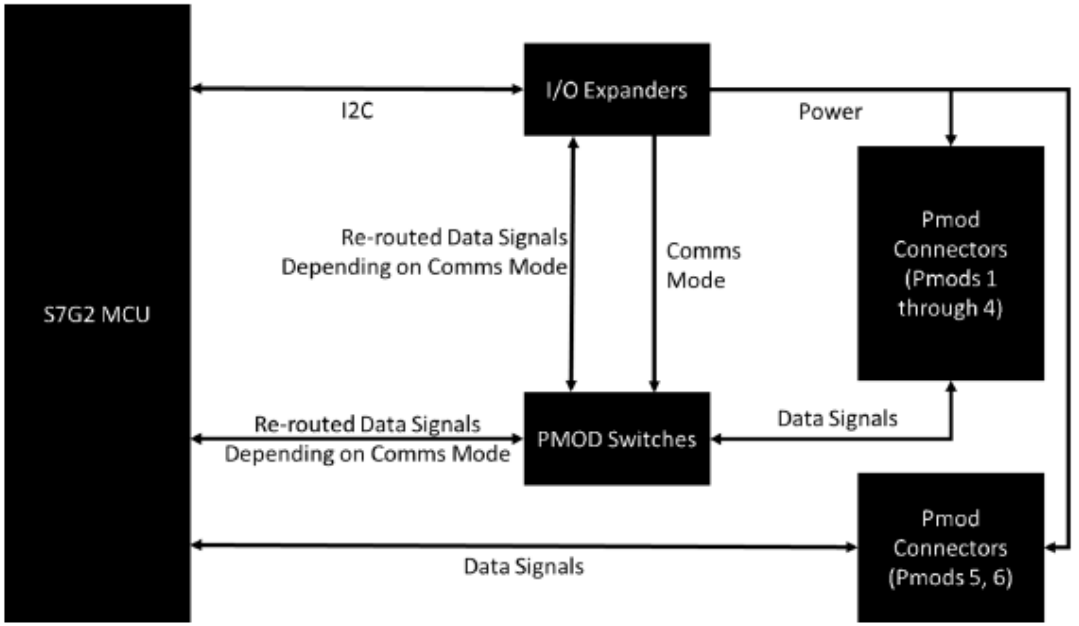
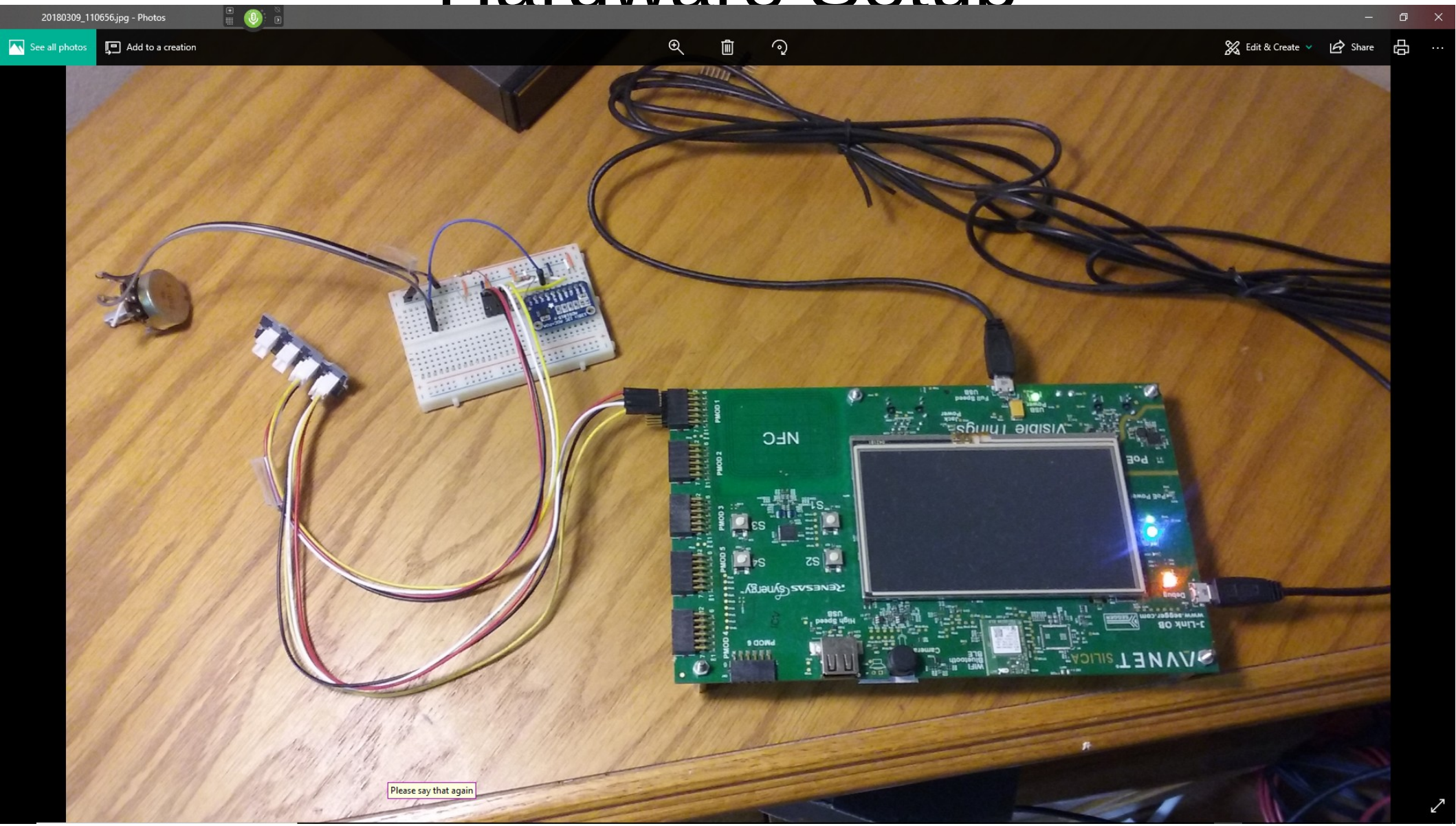


Figure 30 – Pmod, I/O Expander Interconnections Diagram

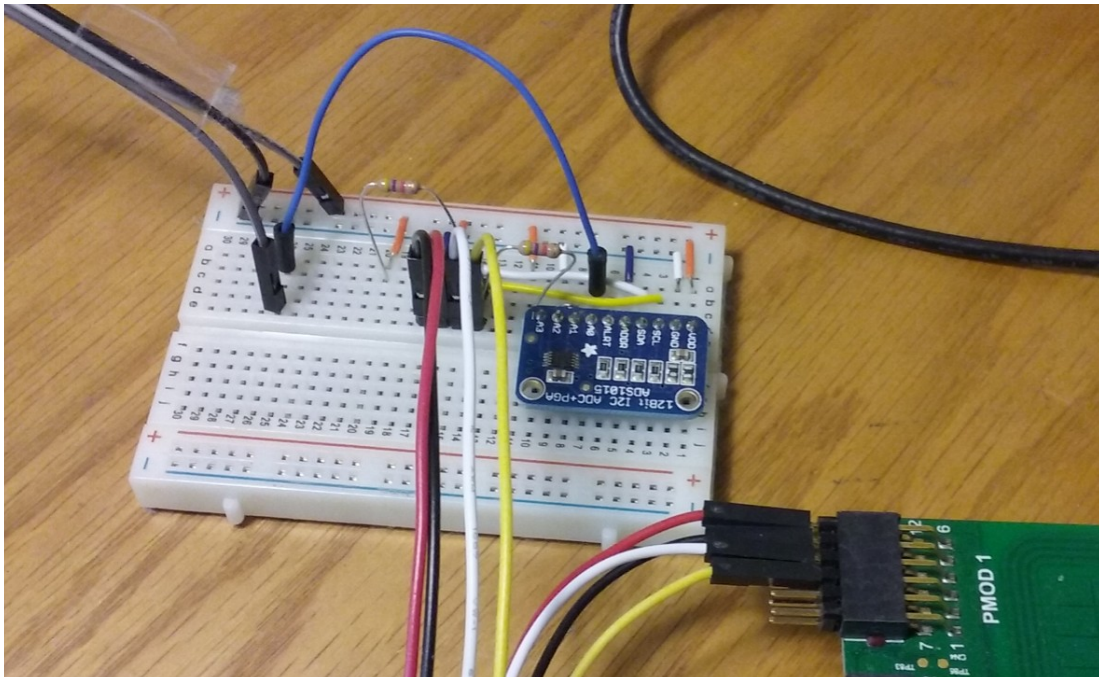
Hardware Setup



Add two 4.7k pullup on the daughter board.

PMOD1	ADS1015
Pin 1 NC	
Pin 2 NC	
Pin 3 SCL	SCL+4.7k pullup (Yellow)
Pin 4 SDA	SDA+4.7k Pullup (White)
Pin 5 GND	GND (Black)
Pin 6 VCC	VCC (Orange)

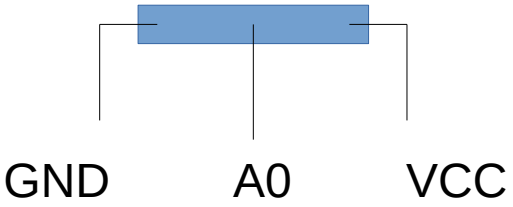
* NC = No Connect

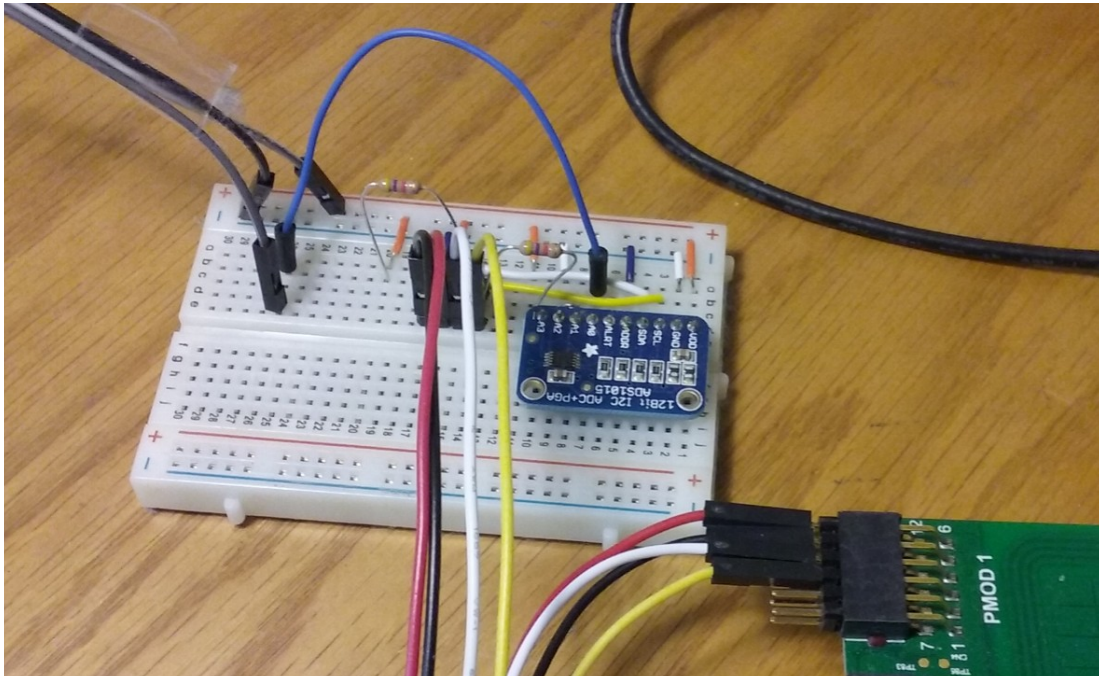


Connection Setup for ADC Module

ADS1015
ADDR = GND
Alert = float (not used)
A0 = Potentiometer
A1/2/3 = GND (not used)

Potentiometer





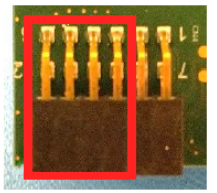
PMOD1 Pins

I2C Comms Mode

The I2C Comms Mode can be used for both I2C standard and UART Type 3 standard Pmods.

I2C Pmods

I2C Pmods only have 8 pins instead of the 12 provided by the Gateway's connector. Insert these Pmods into the left-most side of the connector.

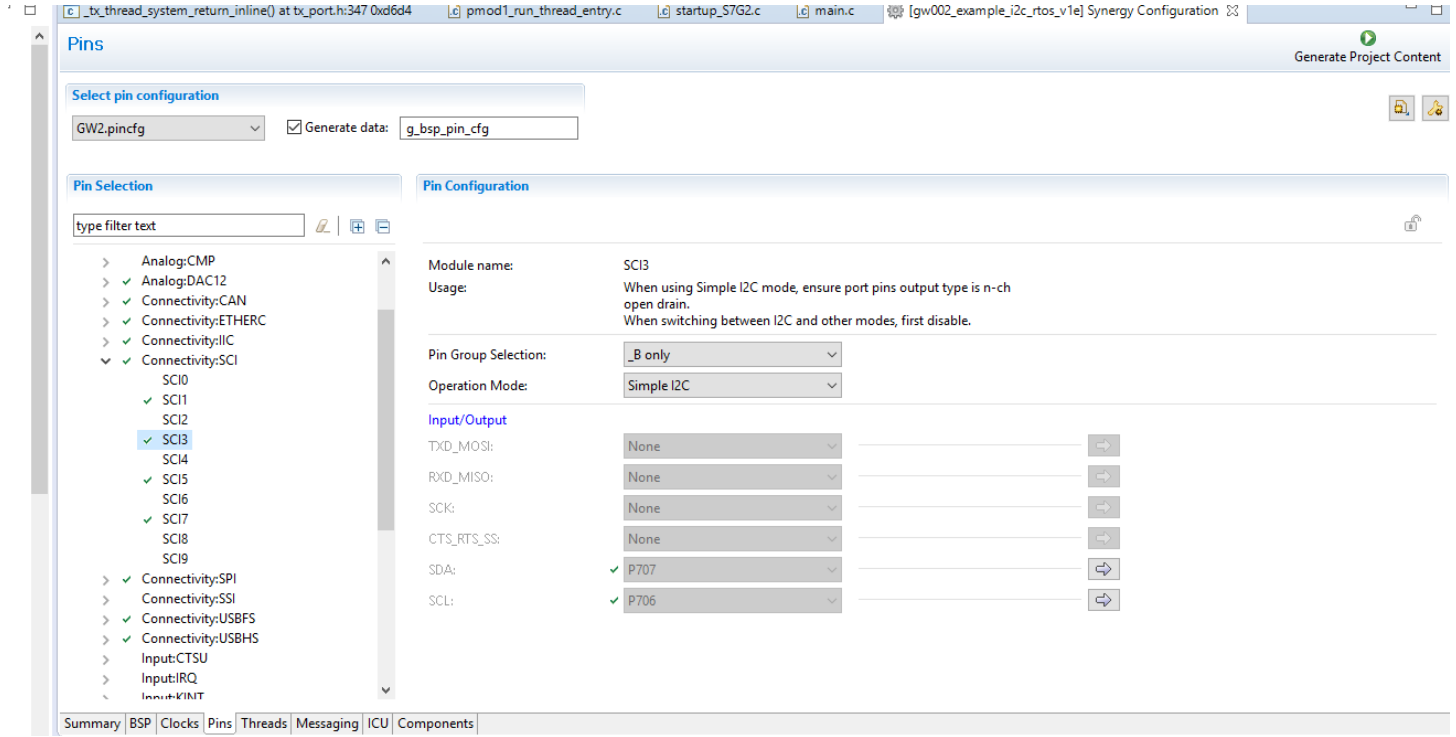


Pmod		S7G2	
Pin	Pin Function	Port Pin	Pin Function
1	SCL	PA02	SDA
2	SCL	PA02	SDA
3	SDA	PA03	SCL
4	SDA	PA03	SCL
5	GND	GND	GND
6	GND	GND	GND
7	VCC (3.3V)	VCC (3.3V)	VCC (3.3V)
8	VCC (3.3V)	VCC (3.3V)	VCC (3.3V)

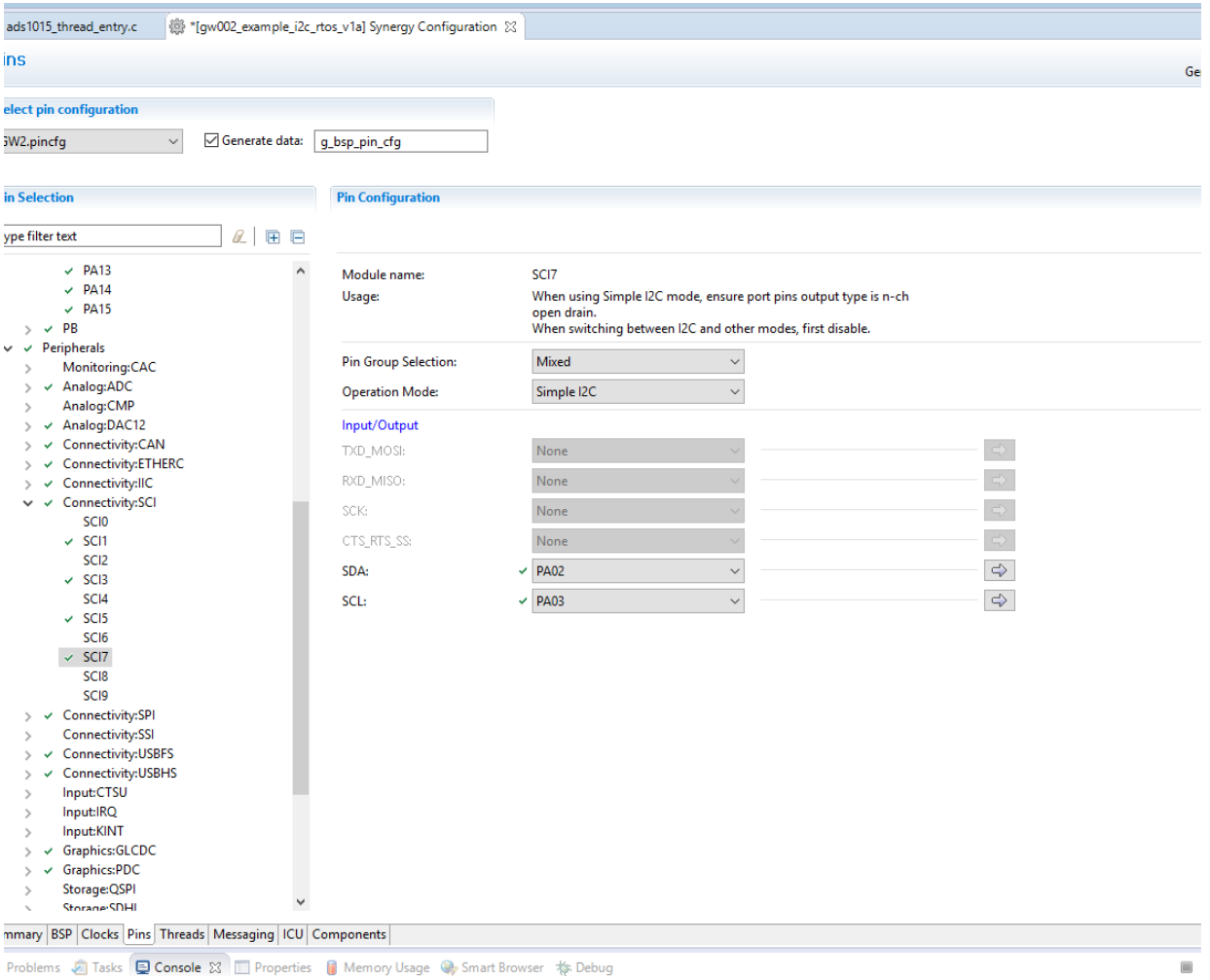
Table 8 – Pmod1 I2C Pin Connections

SCI7 Channel

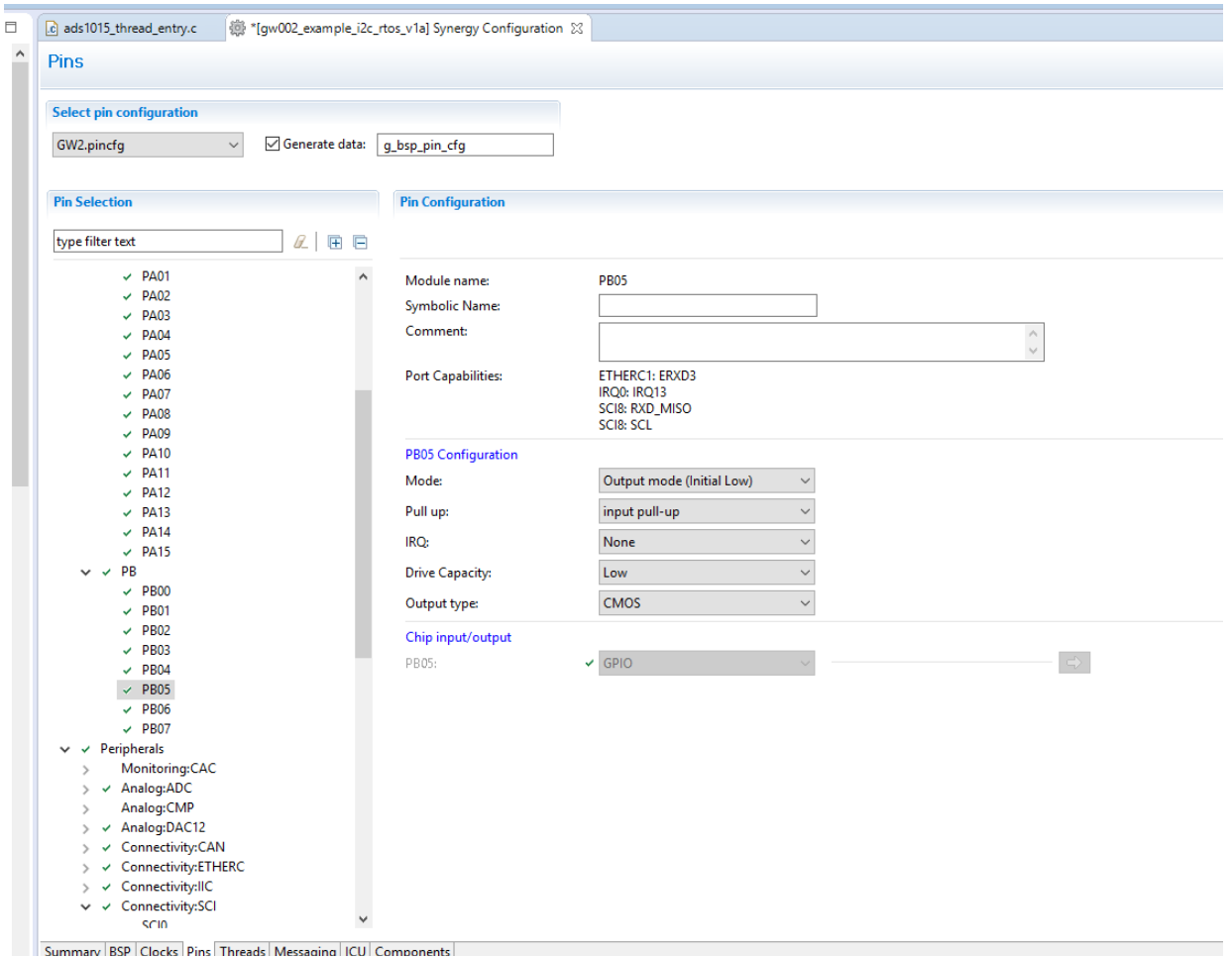
IO Expanders U18/U19 I2C port



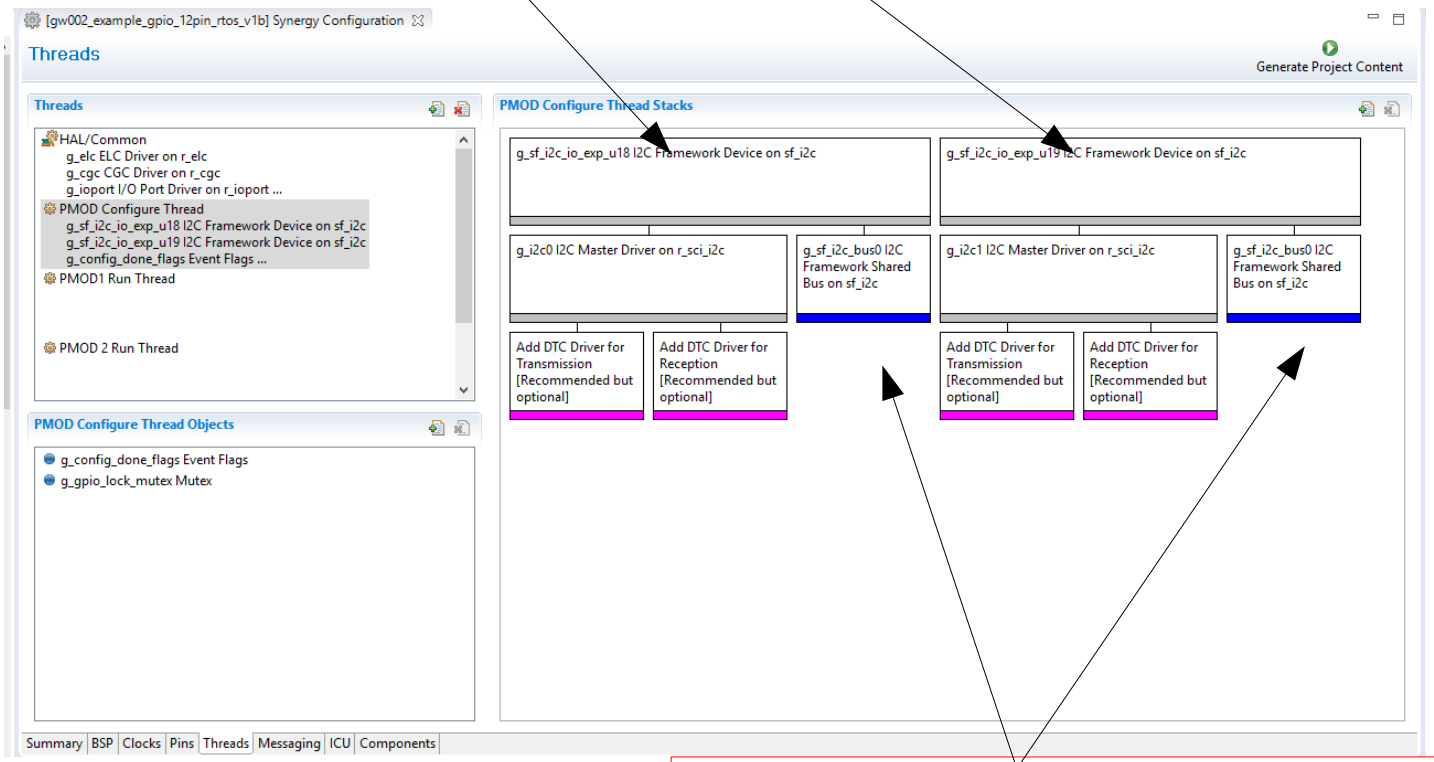
Configure PMOD 1 (Simple I2C)



PMOD 2 : PB02/3/4/5 Output Mode (P001 input only)

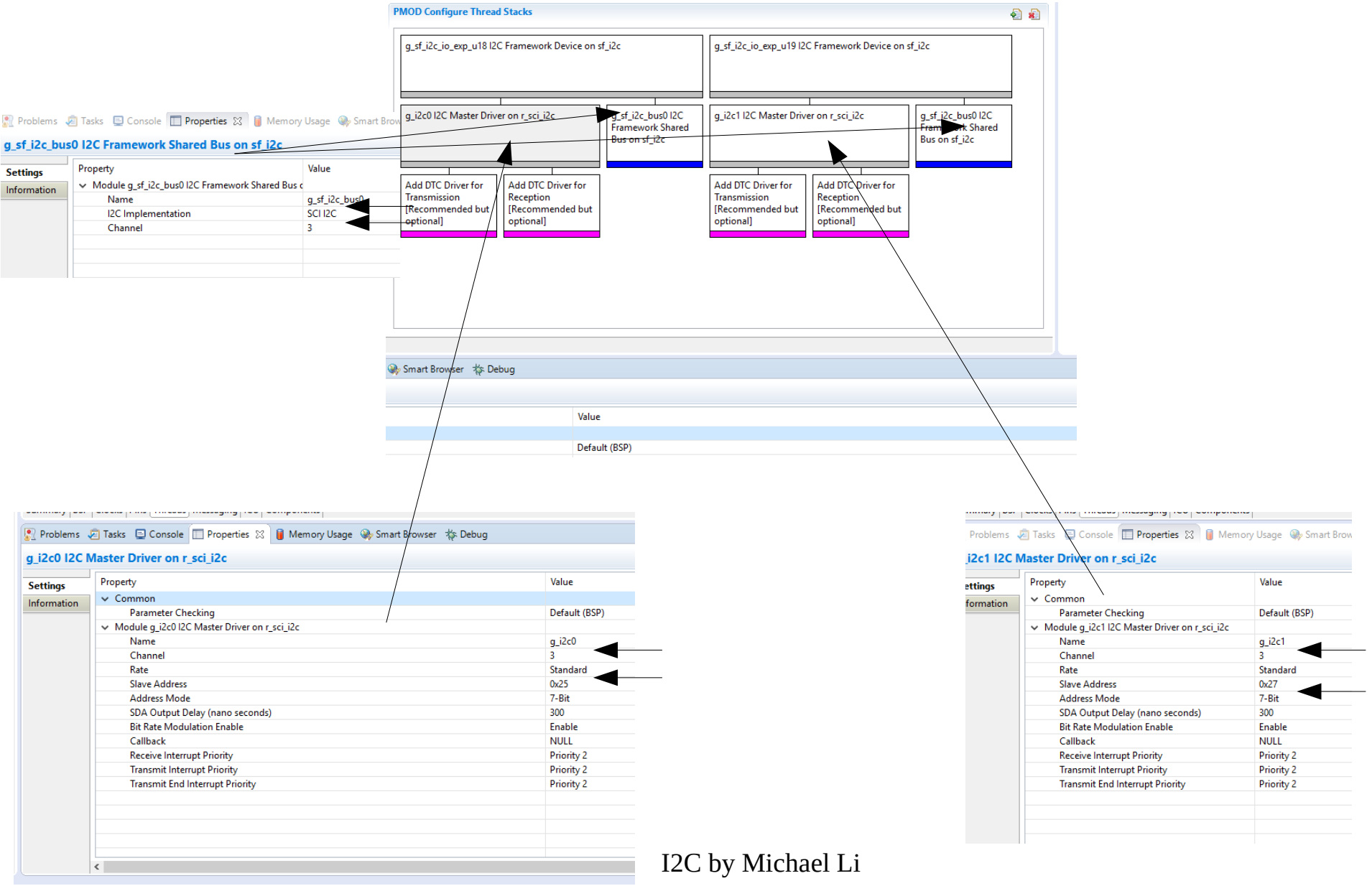


PMOD Configure Thread
(Create U18/U19 I2C Framework Drivers)



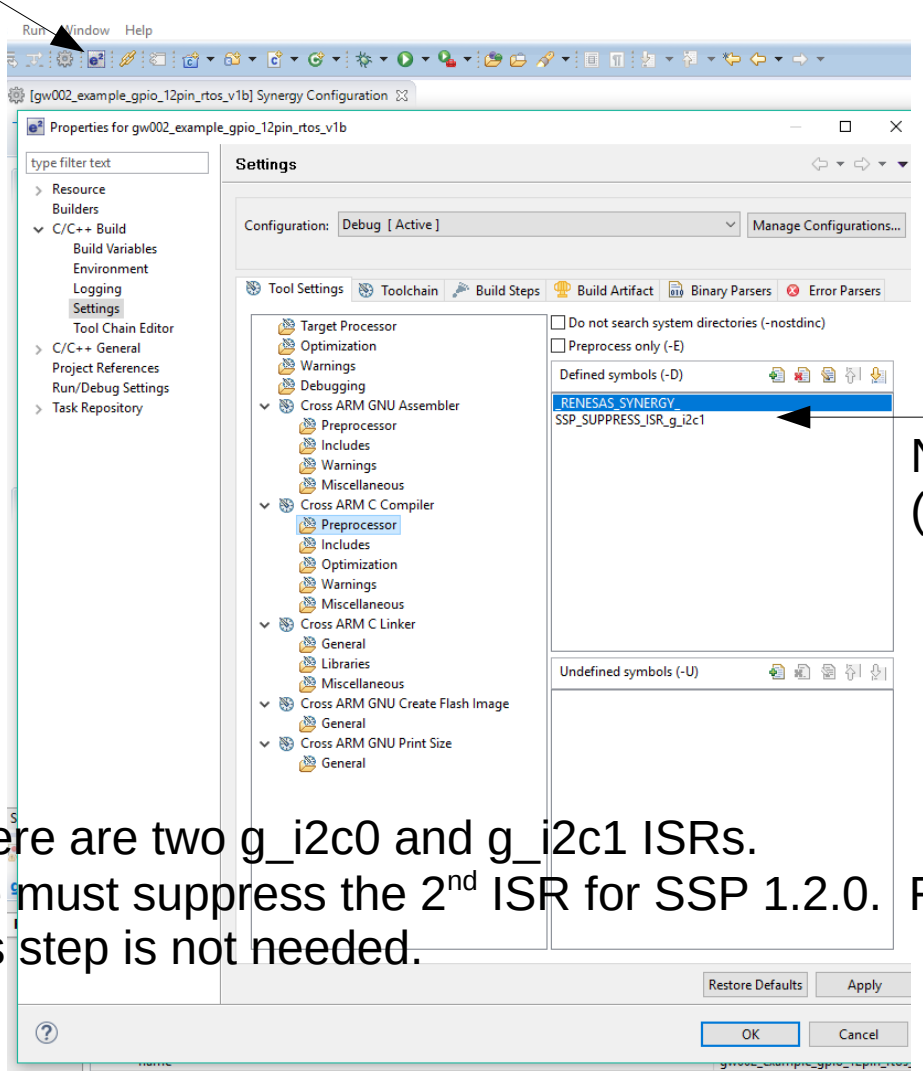
Both drivers share the same g_sf_i2c_bus0 bus.

Properties


PMOD Configure Thread Stacks
g_sf_i2c_io_exp_u18 I2C Framework Device on sf_i2c
g_sf_i2c_io_exp_u19 I2C Framework Device on sf_i2c
g_i2c0 I2C Master Driver on r_sci_i2c
g_sf_i2c_bus0 I2C Framework Shared Bus on sf_i2c
g_i2c1 I2C Master Driver on r_sci_i2c
Add DTC Driver for Transmission [Recommended but optional]
Add DTC Driver for Reception [Recommended but optional]
Problems Tasks Console Properties Memory Usage Smart Browser Debug
g_sf_i2c_bus0 I2C Framework Shared Bus on sf_i2c
Settings
Information
Property
Value
Module g_sf_i2c_bus0 I2C Framework Shared Bus
Name
g_sf_i2c_bus0
I2C Implementation
SCI I2C
Channel
3
Default (BSP)
g_i2c0 I2C Master Driver on r_sci_i2c
Common
Parameter Checking
Default (BSP)
Module g_i2c0 I2C Master Driver on r_sci_i2c
Name
g_i2c0
Channel
3
Rate
Standard
Slave Address
0x25
Address Mode
7-Bit
SDA Output Delay (nano seconds)
300
Bit Rate Modulation Enable
Enable
Callback
NULL
Receive Interrupt Priority
Priority 2
Transmit Interrupt Priority
Priority 2
Transmit End Interrupt Priority
Priority 2
i2c1 I2C Master Driver on r_sci_i2c
Module g_i2c1 I2C Master Driver on r_sci_i2c
Name
g_i2c1
Slave Address
0x27

Compiler Pre-processor Requirement

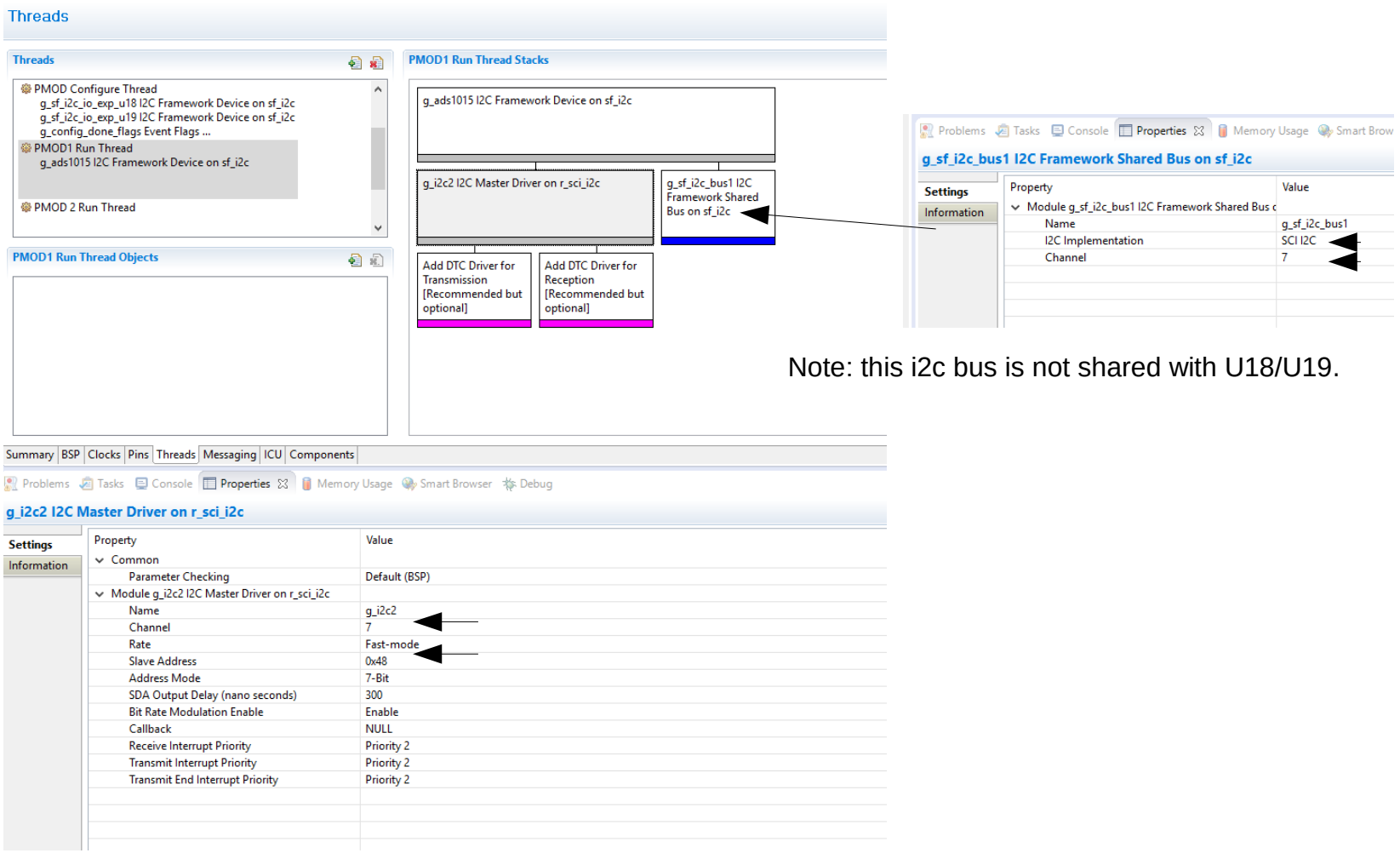
Click this icon!



Need to Suppress 2nd i2c ISR (needed for SSP 1.2.0 only)

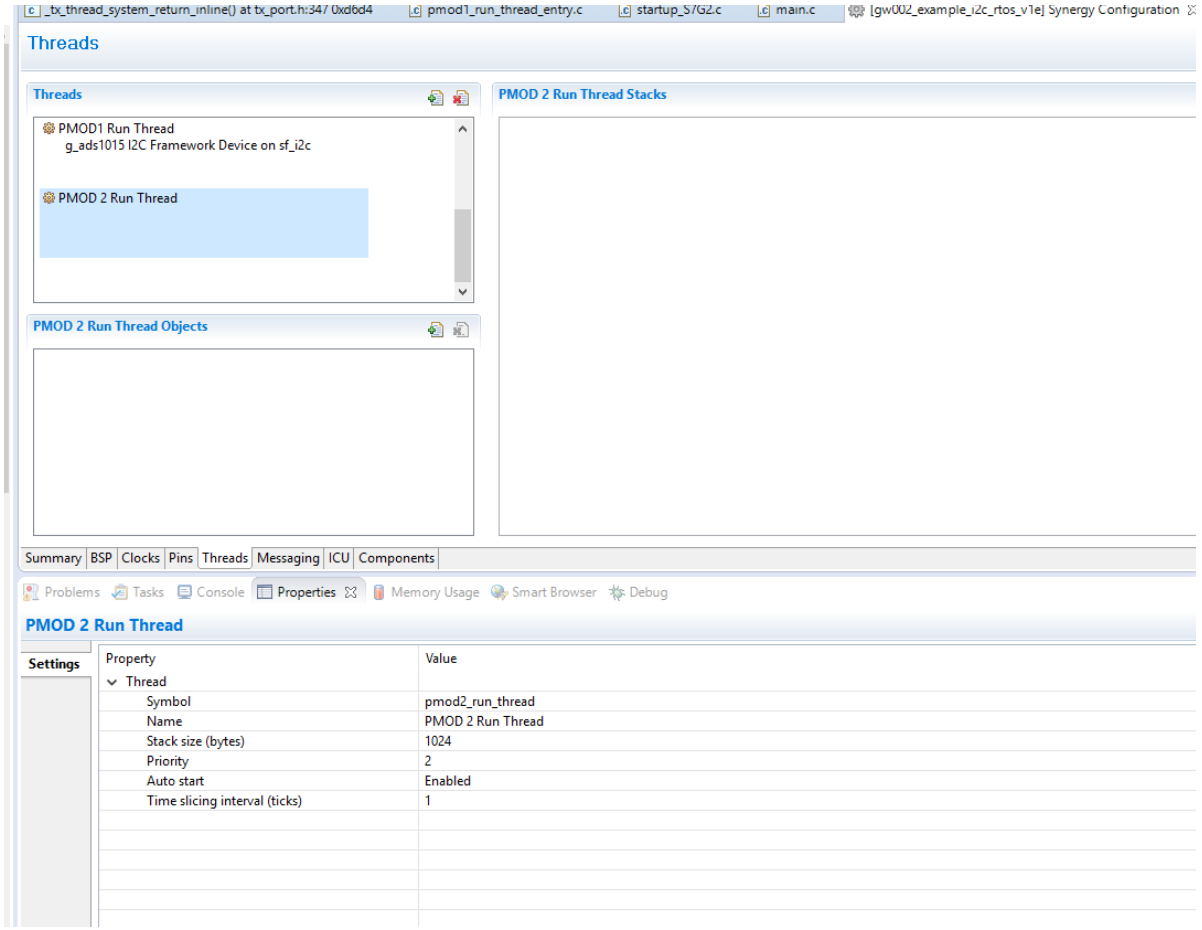
There are two g_i2c0 and g_i2c1 ISRs.
We must suppress the 2nd ISR for SSP 1.2.0. For later SSP versions, this step is not needed.

PMOD 1 : I2C Framework Driver



Note: this i2c bus is not shared with U18/U19.

PMOD2 : No drivers needed for GPIO pins.



Configure PMOD 1 and PMOD 2 (pmod_configu_thread_entry.c)

```
 */
void setup_ioexp_registers(void) {


    ////////////////////////////////////////////////
    ///  U18 setup for PMOD1 : I2C 4PIN configuration
    ////////////////////////////////////////////////
    pmod_bus_type_cfg[PMOD1_PORT] = I2C_TYPE_4PINS_COMH;

    u18port0outreg.bit.pmod1_comms = set_pmod_com_bit(pmod_bus_type_cfg[PMOD1_PORT]);

    ////////////////////////////////////////////////
    ///  U19 setup for PMOD1 : Power enabled
    ////////////////////////////////////////////////
    u19port0outreg.bit.pmod1_power = 1;  // pmod1 power enabled.

    ////////////////////////////////////////////////
    ///  U18 setup for PMOD2 : GPIO Type 1 12 pin configuration
    ///                        Set IO1-5 output pins (S7G2 MCU pins)
    ///                        Set IO6/7/8 output pins
    ///                        Set IO6/7/8 = 010
    ///
    /// Note IO5 (This S7G2 pin is input mode only)
    ////////////////////////////////////////////////
    pmod_bus_type_cfg[PMOD2_PORT] = GPIO_TYPE1_12PINS_COML;

    u18port0outreg.bit.pmod2_comms = set_pmod_com_bit(pmod_bus_type_cfg[PMOD2_PORT]);

    u18port0cfg.bit.pmod2_reset_io6 = SET_CFG_PIN_OUTPUT;
    u18port1cfg.bit.pmod2_io7       = SET_CFG_PIN_OUTPUT;
    u18port1cfg.bit.pmod2_io8       = SET_CFG_PIN_OUTPUT;

    u18port0outreg.bit.pmod2_reset_io6  = 0;  // make 0100 0000 initially
    u18port1outreg.bit.pmod2_io7        = 0;
    u18port1outreg.bit.pmod2_io8        = 0;

    ////////////////////////////////////////////////
    ///  U19 setup for PMOD2 : Power enabled
    ////////////////////////////////////////////////
    u19port0outreg.bit.pmod2_power = 1;  // pmod2 power enabled.

}


/*-----------------------------------------------------------------------------*
 * End of File:  pmod_configure_thread_entry.c
 *-----------------------------------------------------------------------------*/
```

PMOD 1 and 2 Threads' entry code

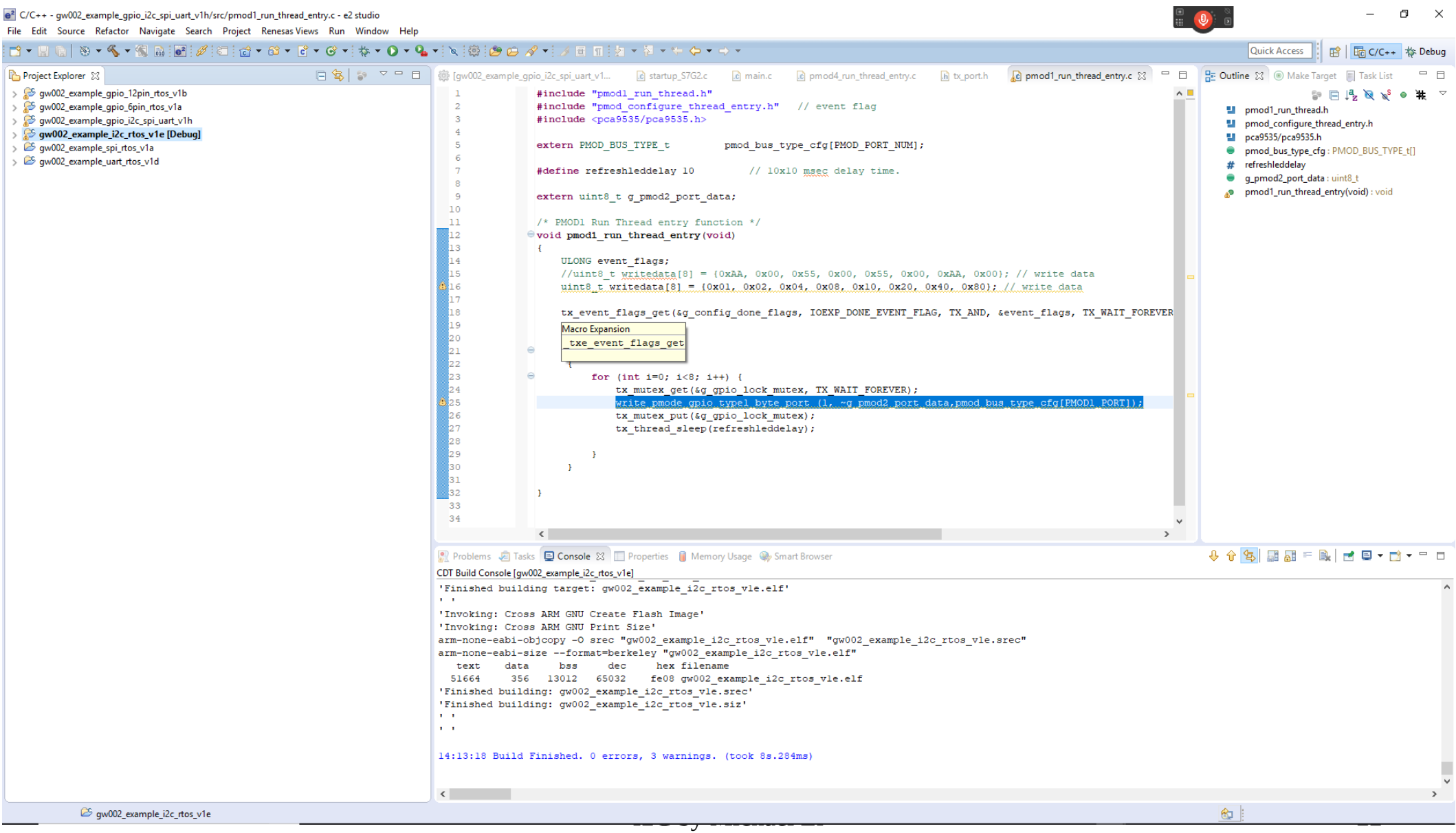
```
    while (1) {
        // read configuration register for conversion status check
        ssp_err = ads1015_register_read (ADS1015_CONFIGREG_ADDR, &devicestatus);
        if (SSP_SUCCESS != ssp_err) {
             g_ioport.p_api->pinWrite(LEDREDPIN, LED_ON);  // turn on red led for error
             while (1);
         }

        if (devicestatus & 0x8000) {    // Conversion is ready
            g_ioport.p_api->pinWrite(LEDBLUPIN, LED_ON);  // turn on blue LED
            // read the conversion data
            ssp_err = ads1015_register_read (ADS1015_CONVERSION_ADDR, &convdata);
            if (SSP_SUCCESS != ssp_err) {
                 g_ioport.p_api->pinWrite(LEDREDPIN, LED_ON);  // turn on red led for error
            } else {
                convdata = convdata >> 4;                       // shift data down by 4 bits
                ain_level = (float)( convdata * 4.096 / 2048);   // FSR = +/-4.096.  For postive voltage range, only 2048 steps.
                // write configuration register and start the conversion!  (Set for the continuous sampling mode)
                ssp_err = ads1015_register_write (ADS1015_CONFIGREG_ADDR,ADS1015_CONFIGREG_VALUE);
                if (SSP_SUCCESS != ssp_err) {
                     g_ioport.p_api->pinWrite(LEDREDPIN, LED_ON);  // turn on red led for error
                     while (1);
                 }
```

PMOD1: Start and Capture the ADC data.

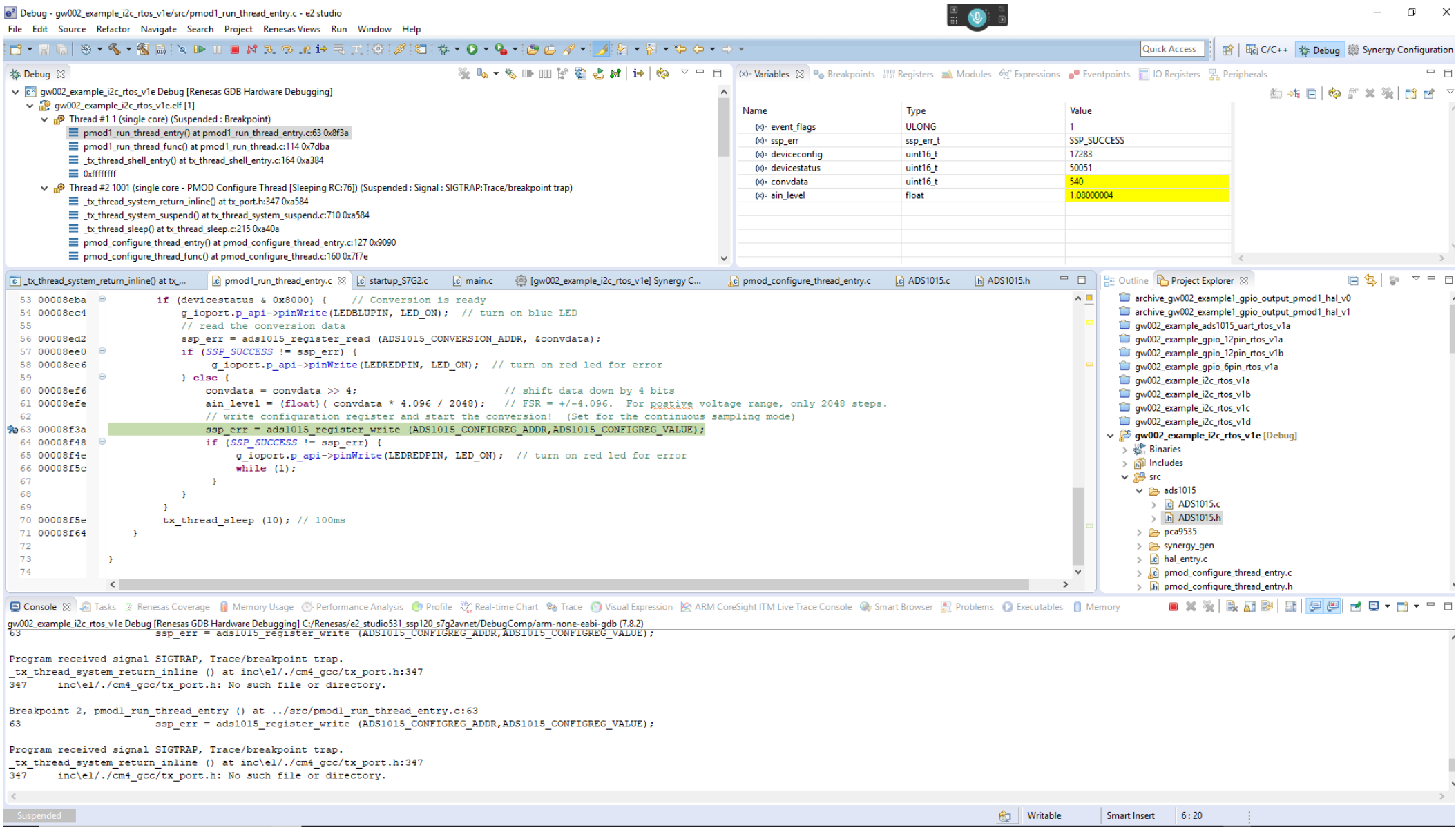
```
pmod1_run_thread...  startup_S7G2.c  main.c  [gw002_example_t2...  pmod_configure_t...  ADS1015.c  ADS10
#include "pmod2_run_thread.h"
#include "pmod_configure_thread_entry.h"   // event flag
#include <pca9535/pca9535.h>

extern PMOD_BUS_TYPE_t         pmod_bus_type_cfg[PMOD_PORT_NUM];


/* PMOD2 Run Thread entry function */
void pmod2_run_thread_entry(void)
{
    ULONG event_flags;
    uint8_t writedata[8] = {0x33, 0x4A, 0x33, 0xF1, 0x8F, 0x41, 0xCB, 0x99}; // write data

    tx_event_flags_get(&g_config_done_flags, IOEXP_DONE_EVENT_FLAG, TX_AND, &event_flags, TX_WAIT_FOREVER);


     while (true)
     {
         for (int i=0; i<8; i++) {
             tx_mutex_get(&g_gpio_lock_mutex, TX_WAIT_FOREVER);
             write_pmode_gpio_type1_byte_port (2, writedata[i],pmod_bus_type_cfg[PMOD2_PORT]);
             tx_mutex_put(&g_gpio_lock_mutex);
             tx_thread_sleep(10);
         }
     }

}
```

PMOD2 : Write data to the port.

Successful Build



Run in Debug Mode: Place a break point to see the ADC result.



Analog Scope Result (A slight difference)