

Avnet Visible Things GPIO Type 1 6 pin Example by

Michael C. Li

<https://www.miketechuniverse.com/>
4/4/2018

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

Avnet Visible Things Platform

User's Manual

Gateway PMOD	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

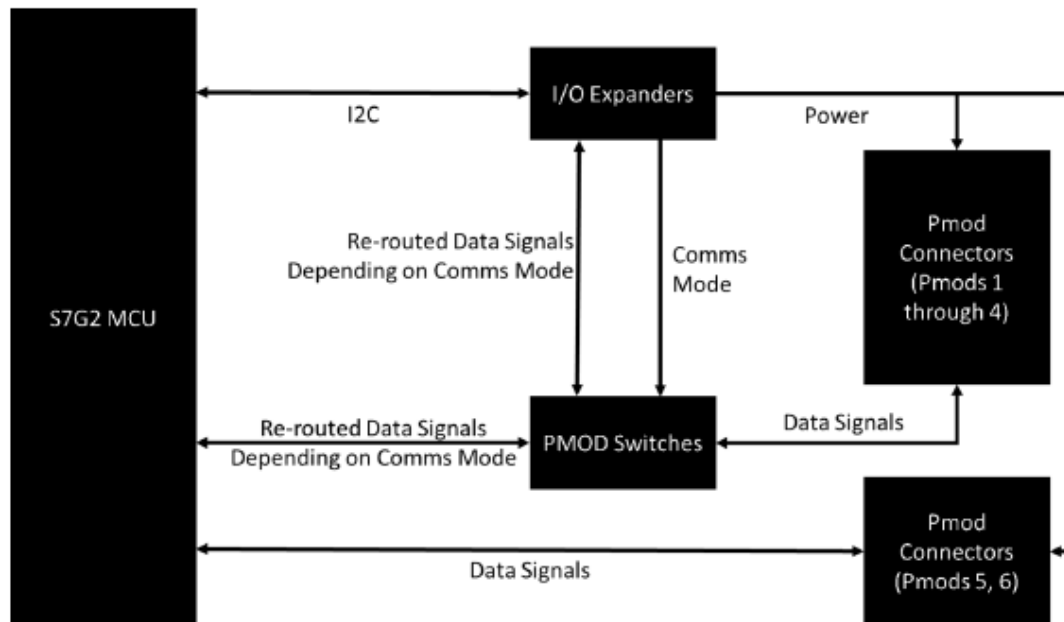
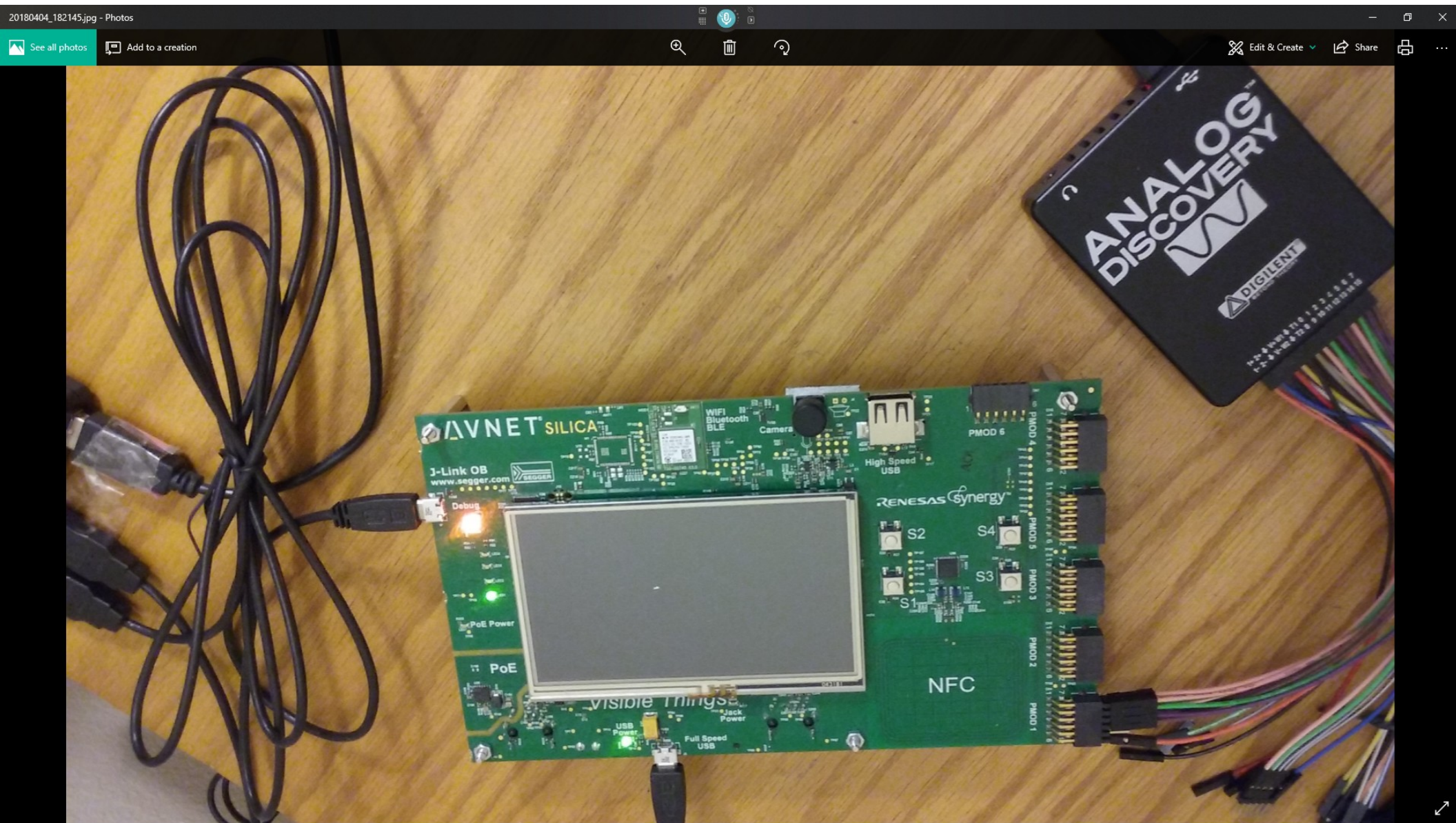


Figure 30 – Pmod, I/O Expander Interconnections Diagram

Test: Probe PMOD 1 ports.



Actual Hardware Setup (Connecting PMOD ports. Only 4 IO pins)



PMOD 1 Port

GPIO Type 1 Standard Pmods

GPIO Type 1 Pmod can be used when the connector is configured in UART/SPI Comms Mode. These types of Pmods can be inserted in either row of the connector.

Pmod		S7G2	
Pmod Pin	Pmod Pin Function	S7G2 Pin	S7G2 Pin Function
1	GPIO	PA05	GPIO
2	GPIO	PA04	GPIO
3	GPIO	PA03	GPIO
4	GPIO	PA02	GPIO
5	GND	GND	GND
6	VCC (3.3V)	VCC (3.3V)	VCC (3.3V)

Table 10 – Pmod1 GPIO Type 1 Pin Connections (I2C Mode)

Both upper and bottom rows are connected to the same S7G2 pins.

PMOD 2 Port

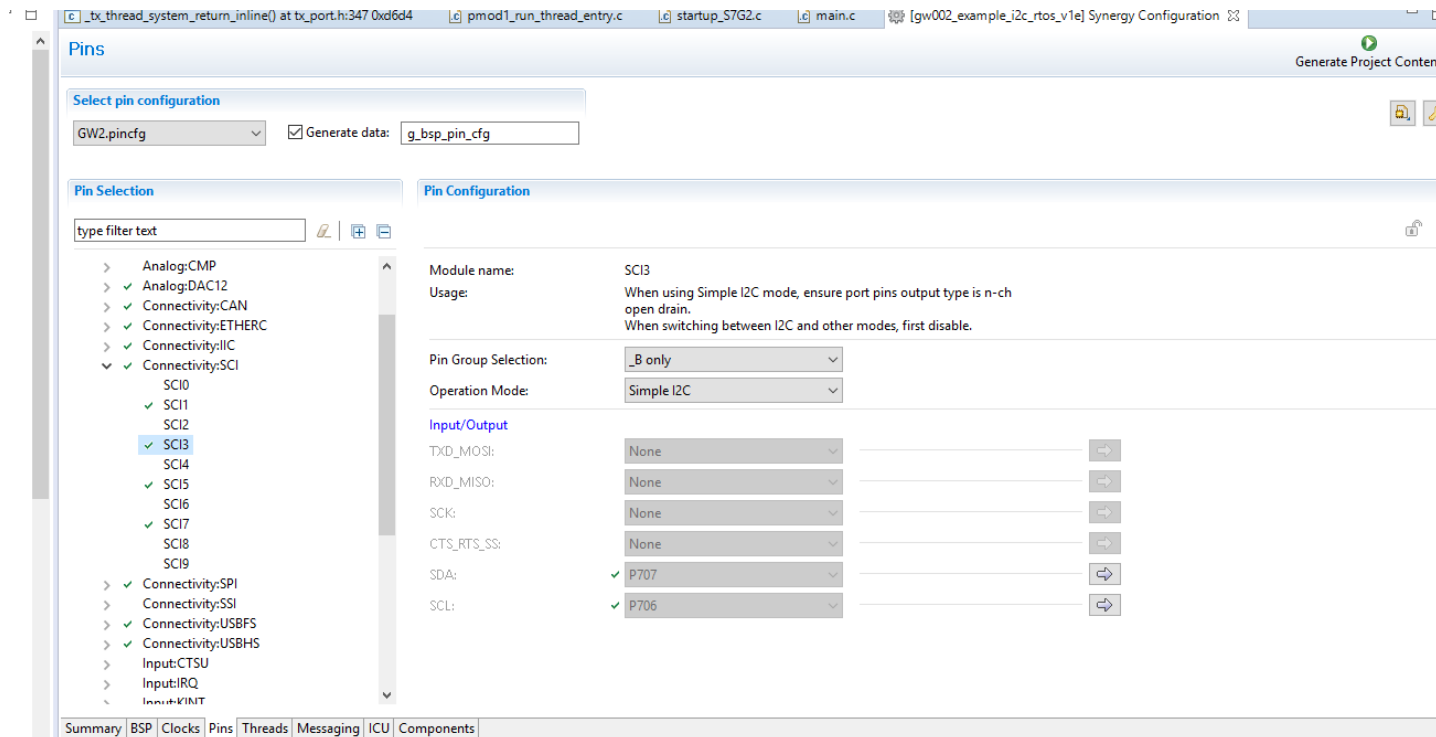
GPIO Type 1 Standard Pmods

GPIO Type 1 Pmod can be used when the connector is configured in UART/SPI Comms Mode. These types of Pmods can be inserted in either row of the connector.

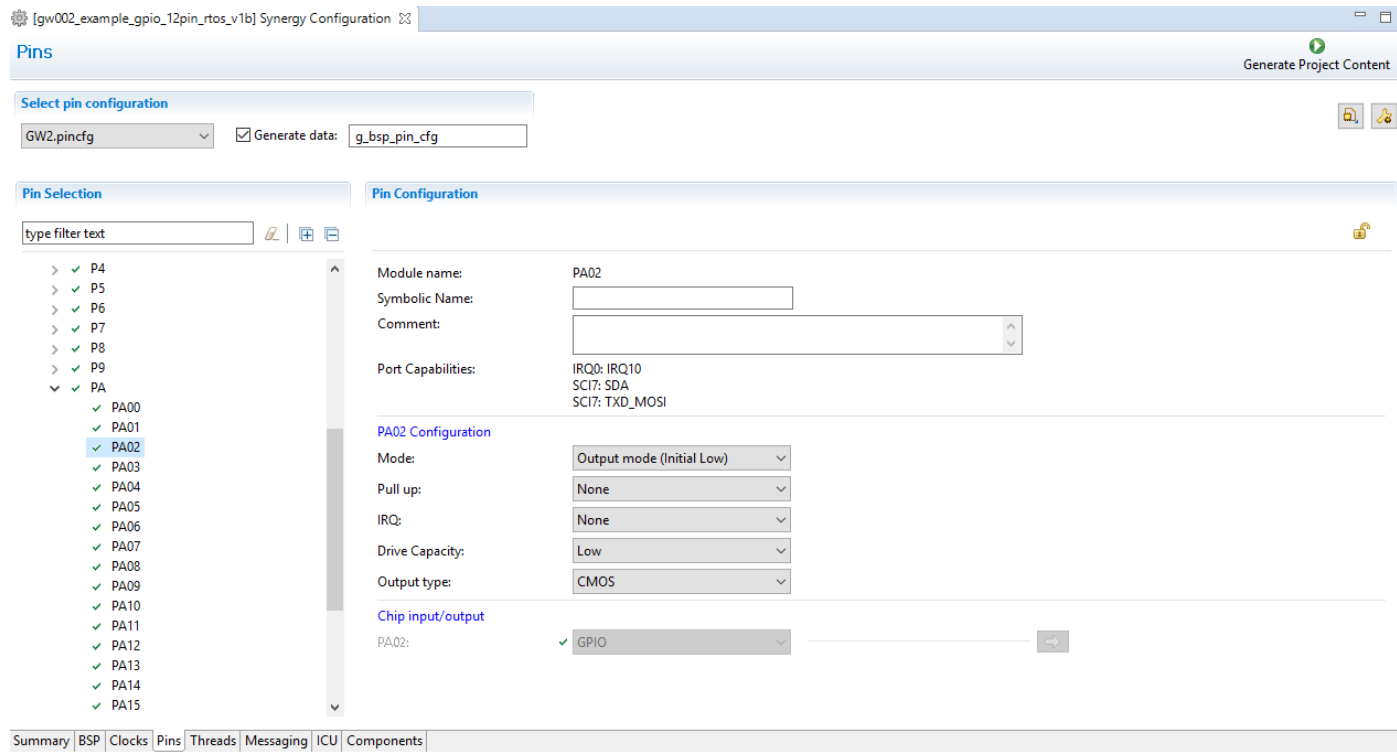
Pmod		S7G2	
Pmod Pin	Pmod Pin Function	S7G2 Pin	S7G2 Pin Function
1	GPIO	PB02	GPIO
2	GPIO	PB03	GPIO
3	GPIO	PB05	GPIO
4	GPIO	PB04	GPIO
5	GND	GND	GND
6	VCC (3.3V)	VCC (3.3V)	VCC (3.3V)

Table 17 – Pmod2 GPIO Type 1 Pin Connections (I2C Mode)

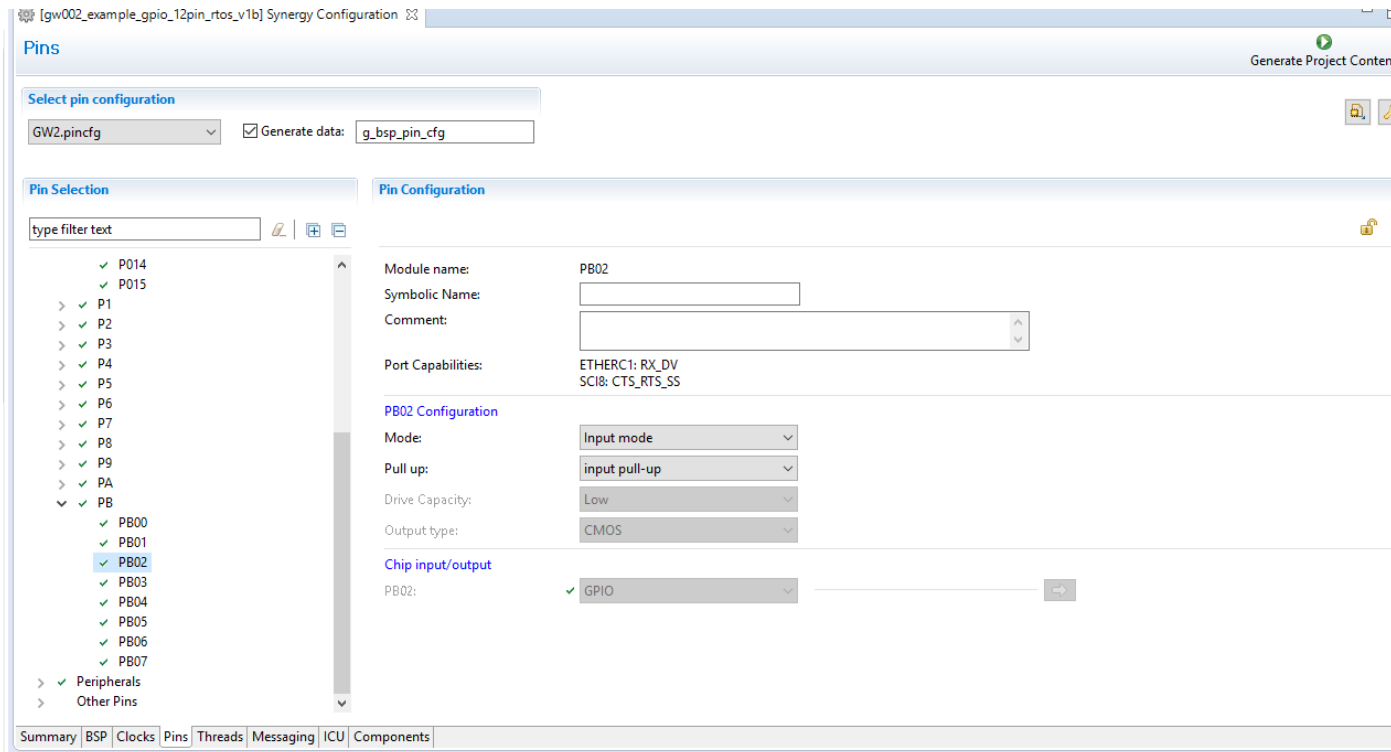
IO Expanders U18/U19 I2C port



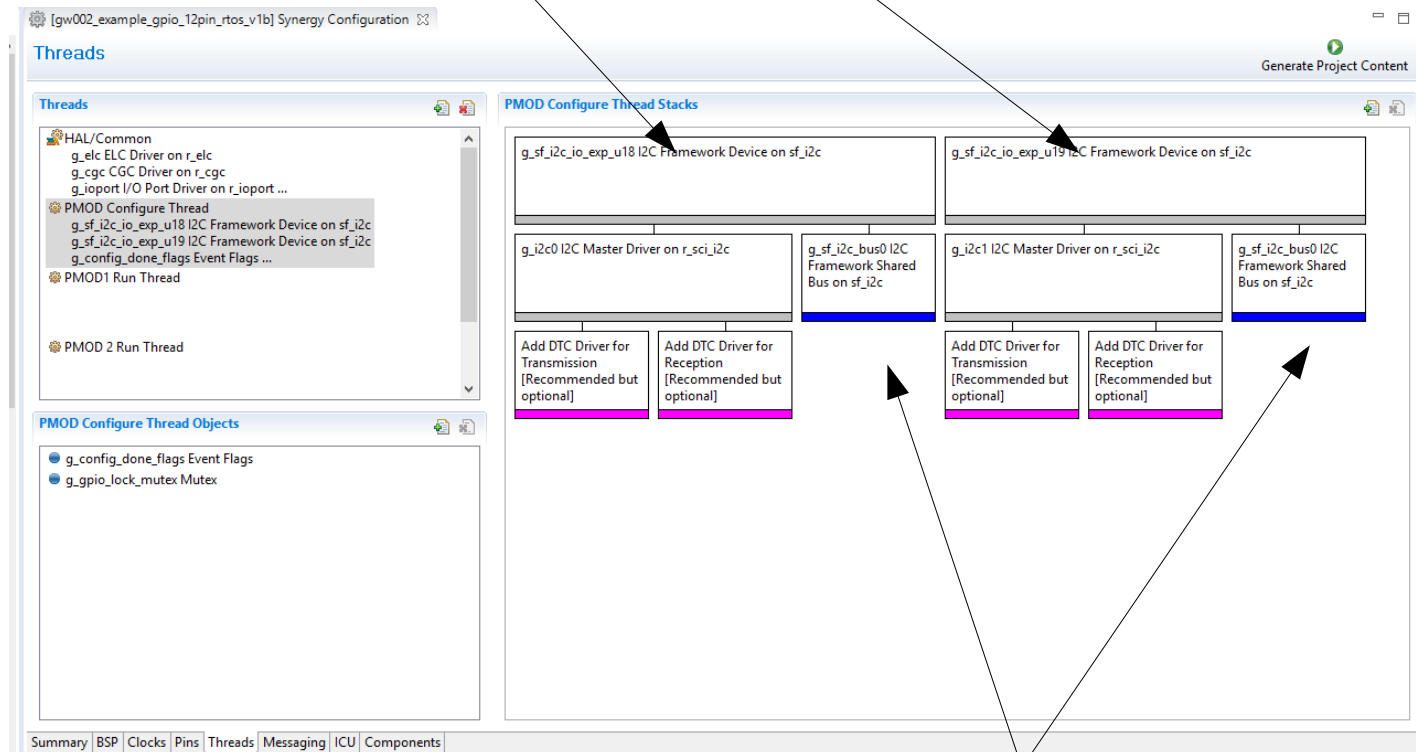
PMOD 1 : PA02/3/4/5 Output Mode (P400 disabled)



PMOD 2 : PB02/3/4/5 Input Mode (P001 disabled)



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



Both drivers share the same g_sf_i2c_bus0 bus.

Properties

The PMOD Configure Thread Stacks window displays the following components:

- g_sf_i2c_exp_u18 I2C Framework Device on sf_i2c
- g_sf_i2c_exp_u19 I2C Framework Device on sf_i2c
- g_sf_i2c0 I2C Master Driver on r_sci_i2c
- g_sf_i2c0 I2C Framework Shared Bus on sf_i2c
- g_sf_i2c1 I2C Master Driver on r_sci_i2c
- g_sf_i2c1 I2C Framework Shared Bus on sf_i2c

Below the main configuration area, there are two detailed property tables for the master drivers.

g_sf_i2c0 I2C Master Driver on r_sci_i2c

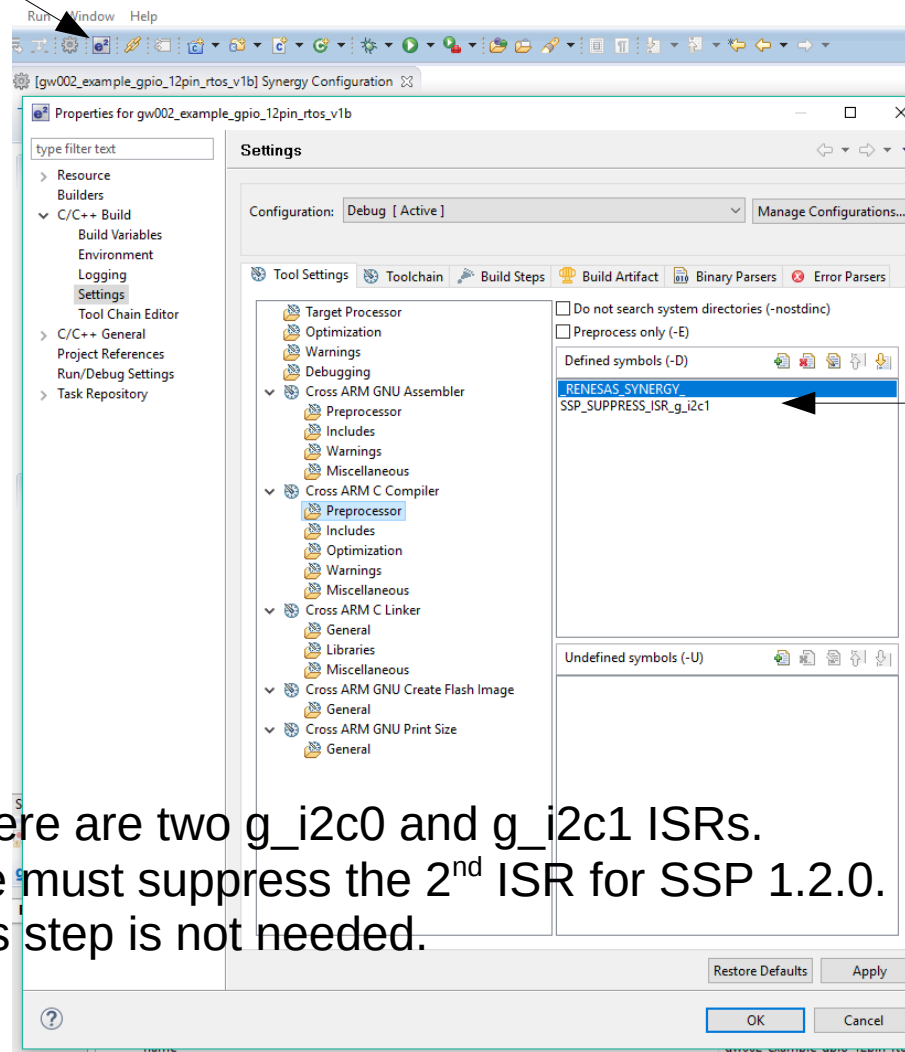
Property	Value
Module g_sf_i2c0 I2C Master Driver on r_sci_i2c	
Name	g_sf_i2c0
I2C Implementation	SCI I2C
Channel	3
Parameter Checking	Default (BSP)
Rate	Standard
Slave Address	0x25
Address Mode	7-Bit
SDA Output Delay (nano seconds)	300
Bit Rate Modulation Enable	Enable
Callback	NULL
Receive Interrupt Priority	Priority 2
Transmit Interrupt Priority	Priority 2
Transmit End Interrupt Priority	Priority 2

g_sf_i2c1 I2C Master Driver on r_sci_i2c

Property	Value
Module g_sf_i2c1 I2C Master Driver on r_sci_i2c	
Name	g_sf_i2c1
Channel	3
Rate	Standard
Slave Address	0x27
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

Compiler Pre-processor Requirement

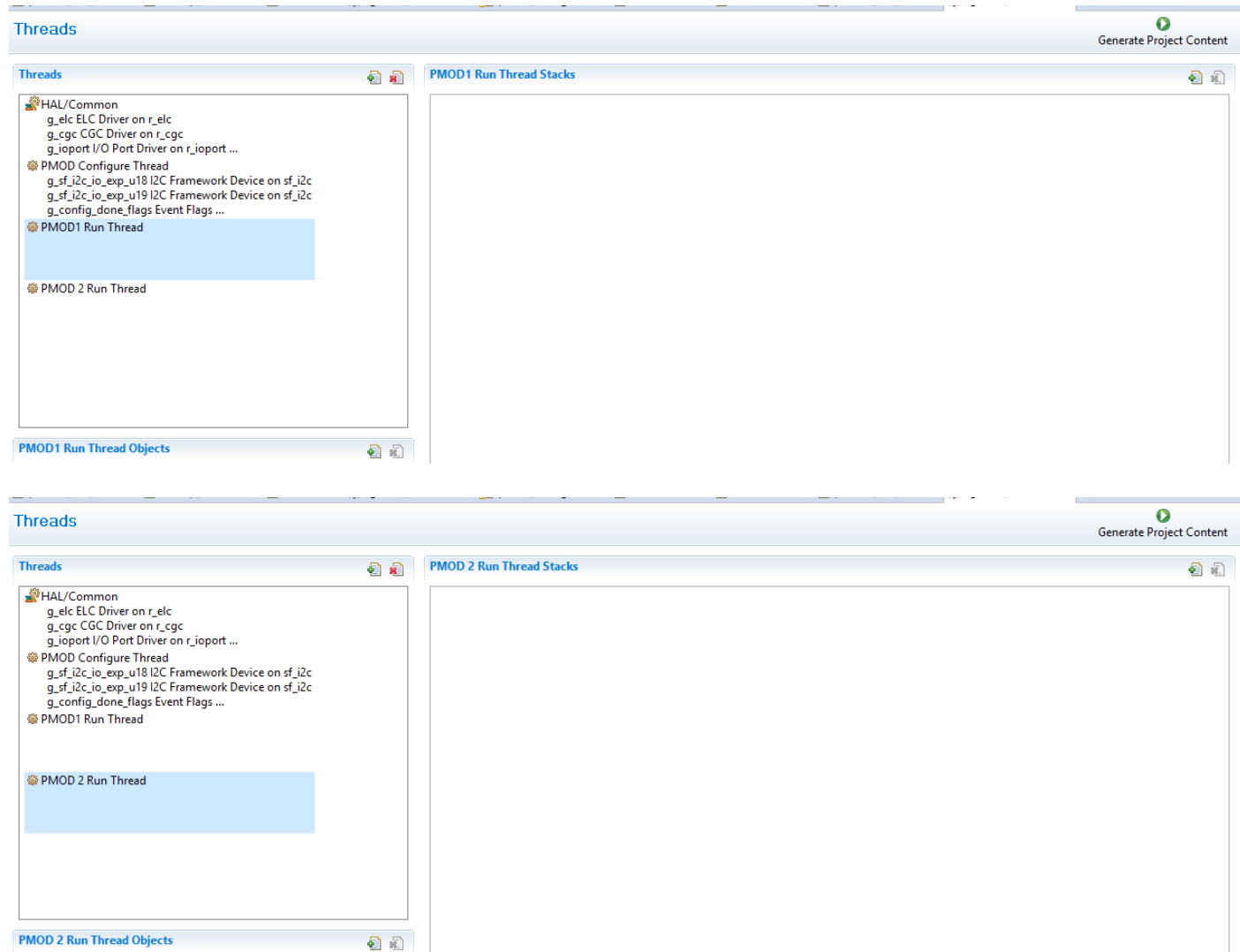
Click this icon!



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.

Both PMOD 1 and 2 Threads have no drivers needed because the ports are configured as GPIO ports.



Configure PMOD 1 and PMOD 2

```
pmod_configure_thread_entry.c [gw002_example_gpio_6pin_rtos_v1a] Synergy Configuration
190 00008130 ul8port0outreg.bit.pmod1_power = 0; // set power off
191 00008fa2
192 00008fac ul9port1outreg.bit.secured_element_en = 0; // Secured Element disabled.
193 00008fb6 }
194 00008fc0
195 /*
196 00008fca * Assign user setup value to IO Expanders' registers
197 00008fd4 *
198 * input: None
199 * Return : None
200 *
201 */
202 void setup_ioexp_registers(void) {
203
204
205 ///////////////////////////////////////////////////
206 00008ffc /// U18 setup for PMOD1 : GPIO Type 1 6 pin configuration
207 // Set IO1-4 output pins (S7G2 MCU pins)
208 ///////////////////////////////////////////////////
209 pmod_bus_type_cfg[PMOD1_PORT] = GPIO_TYPE1_6PINS_COMH;
210
211 ul8port0outreg.bit.pmod1_comms = set_pmod_com_bit(pmod_bus_type_cfg[PMOD1_PORT]);
212
213 ///////////////////////////////////////////////////
214 /// U19 setup for PMOD1 : Power enabled
215 00009000 ///////////////////////////////////////////////////
216 ul9port0outreg.bit.pmod1_power = 1; // pmod1 power enabled.
217 00009006
218 ///////////////////////////////////////////////////
219 00009022 /// U18 setup for PMOD2 : GPIO Type 1 6 pin configuration
220 0000902c // Set IO1-4 input pins (S7G2 MCU pins)
221 00009036 /// Note IO5 (This S7G2 pin is input mode only)
222 ///////////////////////////////////////////////////
223 00009040 pmod_bus_type_cfg[PMOD2_PORT] = GPIO_TYPE1_6PINS_COMH;
224 0000904a
225 00009054 ul8port0outreg.bit.pmod2_comms = set_pmod_com_bit(pmod_bus_type_cfg[PMOD2_PORT]);
226
227 ///////////////////////////////////////////////////
228 /// U19 setup for PMOD2 : Power enabled
229 ///////////////////////////////////////////////////
230 0000905e ul9port0outreg.bit.pmod2_power = 1; // pmod2 power enabled.
231
```

PMOD 1 and 2 Threads' entry code

```
ire_thread_entry.c [gw002_example_gpio_6pin_rtos_v1a] Synergy Configuration pmod1_run_thread_entry.c pmod2_run_thread_entry.c startup_S7G2.c main.c

#include "pmod1_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];

/* PMOD1 Run Thread entry function */
void pmod1_run_thread_entry(void)
{
    ULONG event_flags;
    bool error_flag; // error flag
    uint8_t writedata[4] = {0x03, 0x0A, 0x02, 0x0F}; // write data

    tx_event_flags_get(&g_config_done_flags, IOEXP_DONE_EVENT_FLAG, TX_AND, &event_flags, TX_WAIT_FOREVER); // Don't clear it. Leave it enabled

    while (true)
    {
        for (int i=0; i<4; i++) {
            tx_mutex_get(&g_gpio_lock_mutex, TX_WAIT_FOREVER);
            error_flag = write_pmode_gpio_type1_nibble_port (UPPERROW,1,writedata[i],pmod_bus_type_cfg[PMOD1_PORT]); // write IO1-4
            tx_mutex_put(&g_gpio_lock_mutex);
            tx_thread_sleep(10);
        }
    }
}
```

Only write 4 bit data

```
thread_entry.c [gw002_example_gpio_6pin_rtos_v1a] Synergy Configuration pmod1_run_thread_entry.c pmod2_run_thread_entry.c startup_S7G2.c main.c

#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;
    bool error_flag; // error flag
    uint8_t readdata[4]; // storing the read data

    tx_event_flags_get(&g_config_done_flags, IOEXP_DONE_EVENT_FLAG, TX_AND, &event_flags, TX_WAIT_FOREVER); // Don't clear it. Leave it enabled

    while (true)
    {
        for (int i=0; i<4; i++) {
            tx_mutex_get(&g_gpio_lock_mutex, TX_WAIT_FOREVER);
            error_flag = read_pmode_gpio_type1_nibble_port (UPPERROW,2,readdata+i,pmod_bus_type_cfg[PMOD2_PORT]); // read IO1-4
            tx_mutex_put(&g_gpio_lock_mutex);
            tx_thread_sleep(10);
        }
    }
}
```

Only read 4 bit data

Successful Build

The screenshot displays an IDE interface with the following components:

- Project Explorer:** Lists project files including `gw002_example_gpio_12pin_rtos_v1b`, `gw002_example_gpio_6pin_rtos_v1a [Debug]`, `gw002_example_gpio_i2c_spi_uart_v1h`, `gw002_example_i2c_rtos_v1e`, `gw002_example_spi_rtos_v1a`, and `gw002_example_uart_rtos_v1d`.
- Source Editor:** Displays the code for `pm0d1_run_thread_entry.c`. The code includes headers for `pm0d1_run_thread.h`, `pm0d_configure_thread_entry.h`, and `pca9535/pca9535.h`. It defines `PMOD_BUS_TYPE_t` and `refreshleddelay`. The `pm0d1_run_thread_entry` function is shown, which includes a loop that writes data to a port and sleeps for a delay.
- Outline:** Shows the structure of the project, including `pm0d1_run_thread.h`, `pm0d_configure_thread_entry.h`, `pca9535/pca9535.h`, `pm0d_bus_type_cfg: PMOD_BUS_TYPE_t`, `refreshleddelay`, `g_pm0d2_port_data: uint8_t`, and `pm0d1_run_thread_entry(void): void`.
- Console:** Displays the build output for `gw002_example_gpio_6pin_rtos_v1a`. The output shows the build process, including the invocation of `arm-none-eabi-objcopy` and `arm-none-eabi-size`, and the final build status: `14:09:22 Build Finished. 0 errors, 4 warnings. (took 8s.621ms)`.

```
#include "pm0d1_run_thread.h"
#include "pm0d_configure_thread_entry.h" // event flag
#include <pca9535/pca9535.h>

extern PMOD_BUS_TYPE_t      pm0d_bus_type_cfg[PMOD_PORT_NUM];

#define refreshleddelay 10    // 10x10 msec delay time.

extern uint8_t g_pm0d2_port_data;

/* PMOD1 Run Thread entry function */
void pm0d1_run_thread_entry(void)
{
    ULONG event_flags;
    //uint8_t writedata[8] = {0xAA, 0x00, 0x55, 0x00, 0x55, 0x00, 0xAA, 0x00}; // write data
    uint8_t writedata[8] = {0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80}; // write data

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

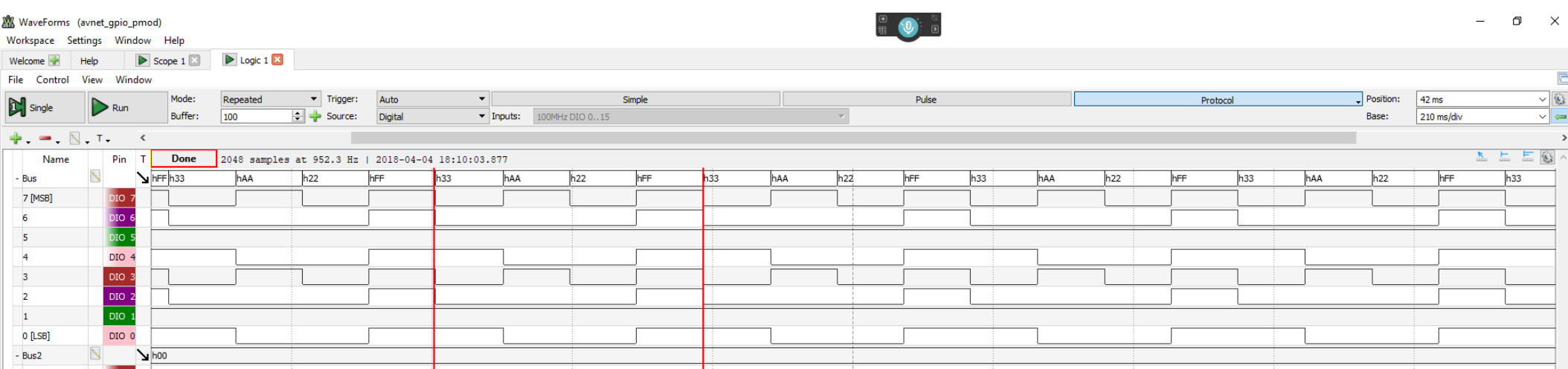
    while (true)
    {
        for (int i=0; i<8; i++) {
            tx_mutex_get(&g_gpio_lock_mutex, TX_WAIT_FOREVER);
            write_pm0d_gpio_type_byte_port(1, ~g_pm0d2_port_data, pm0d_bus_type_cfg[PMOD1_PORT]);
            tx_mutex_put(&g_gpio_lock_mutex);
            tx_thread_sleep(refreshleddelay);
        }
    }
}
```

CDT Build Console [gw002_example_gpio_6pin_rtos_v1a]

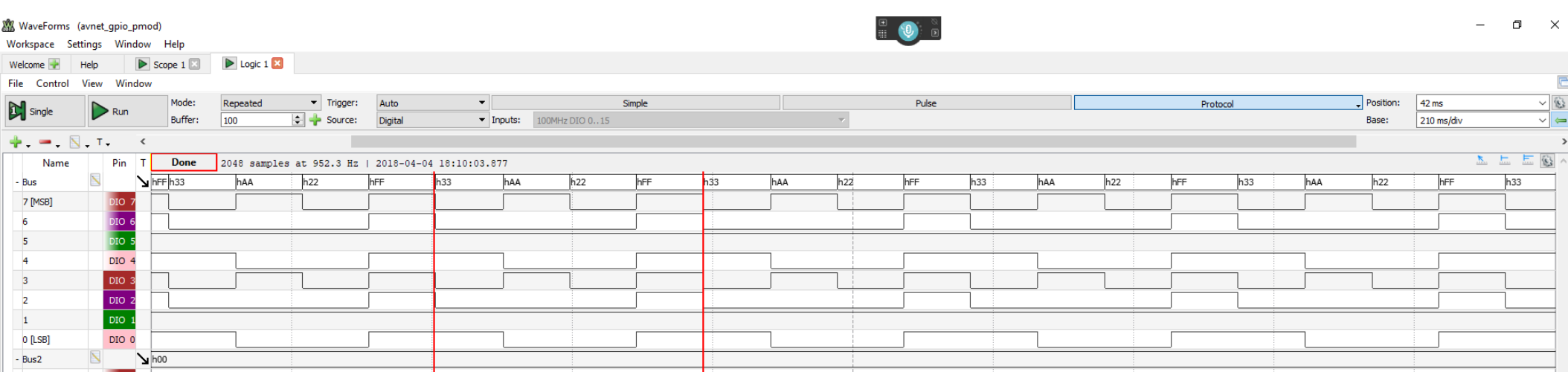
```
'Finished building target: gw002_example_gpio_6pin_rtos_v1a.elf'
'Invoking: Cross ARM GNU Create Flash Image'
arm-none-eabi-objcopy -O srec "gw002_example_gpio_6pin_rtos_v1a.elf" "gw002_example_gpio_6pin_rtos_v1a.srec"
'Invoking: Cross ARM GNU Print Size'
arm-none-eabi-size --format=berkeley "gw002_example_gpio_6pin_rtos_v1a.elf"
'Finished building: gw002_example_gpio_6pin_rtos_v1a.srec'
'
  text    data    bss     dec     hex filename
48924    268    12812   62004   f234 gw002_example_gpio_6pin_rtos_v1a.elf
'Finished building: gw002_example_gpio_6pin_rtos_v1a.siz'
'

14:09:22 Build Finished. 0 errors, 4 warnings. (took 8s.621ms)
```


PMOD 1: Same data appear on both upper and bottom rows.



PMOD 1 port scope output data = readdata from PMOD2 port read



Name	Type	Value
(x) event_flags	ULONG	1
(x) error_flag	_Bool	false
(x) readdata	uint8_t [4]	0x1ffe2ec8 <pmod2_run_thread_stack.
(x) readdata[0]	uint8_t	3 '\003'
(x) readdata[1]	uint8_t	10 '\n'
(x) readdata[2]	uint8_t	2 '\002'
(x) readdata[3]	uint8_t	15 '\017'

H03
HAA
H22
HFF

Decimal
values