

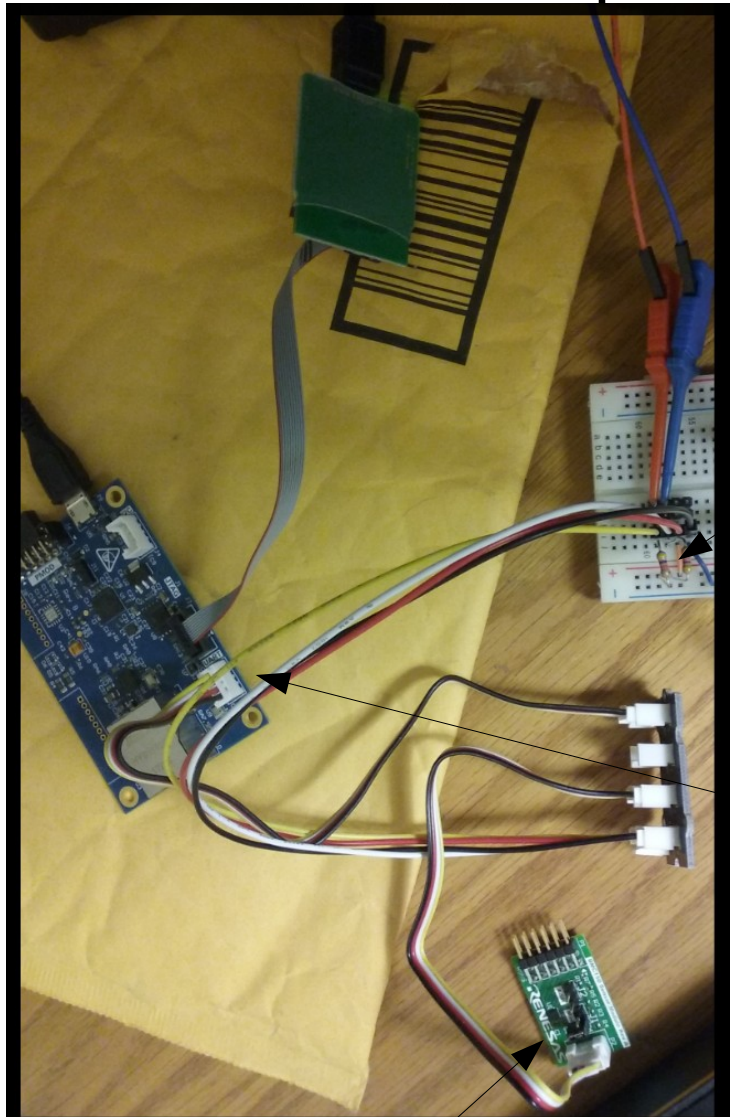
# S5D9 I2C Bus Example (SCI Driver Version)

By  
Michael Li  
(1/2/2008)

<https://www.miketechuniverse.com>

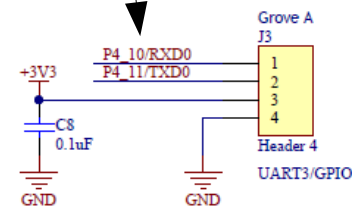
E2 Studio 5.4.0.023  
SSP 1.3.0

Connect to the UART grove A since it can be configured as an I2C port by changing the SCI0 driver.



Note: No pullup resistors for P4\_10/11. Need to add a pair of 4.7k ohm pull up resistors between SCL/SDA and VCC. (Extra components)

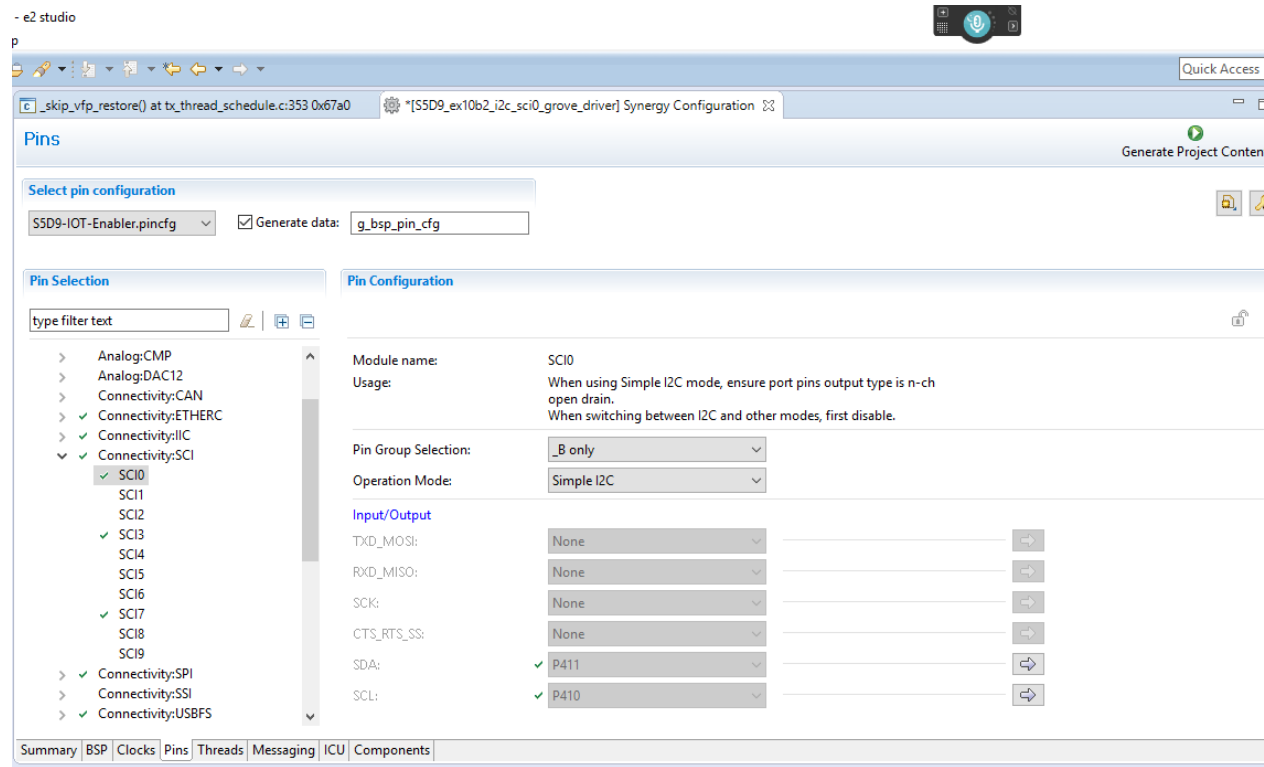
I2C hub (used to add the pull up resistors!)



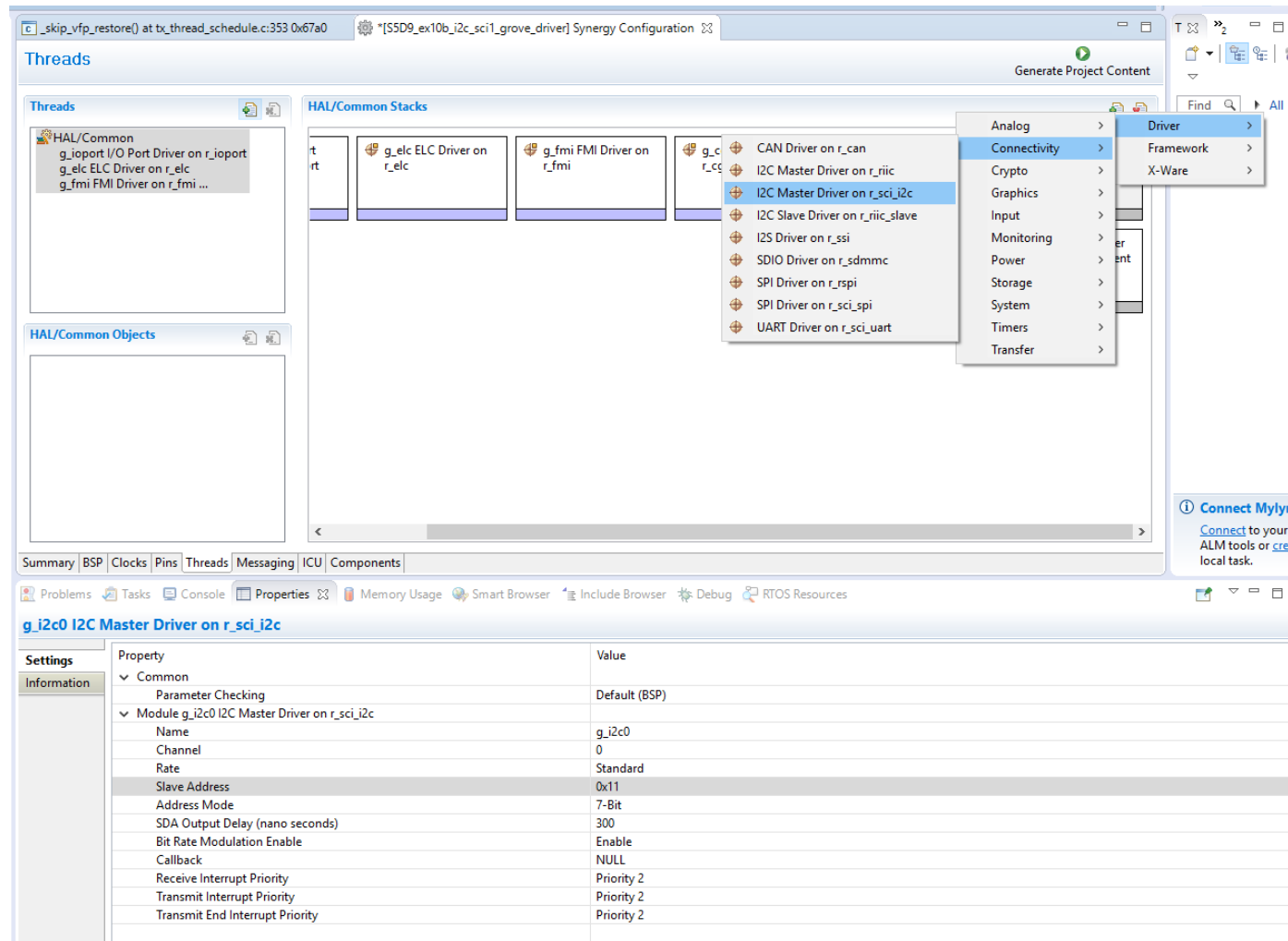
Open for the I2C mode

I2C Example by Michael C. Li

# Change the SCI0 module from UART to I2C



# Add the SCI driver



# Change Properties

The screenshot shows the Synergy Configuration tool interface. The top bar includes 'Window' and 'Help' menus. The main workspace is divided into several panes. On the left, there's a 'Threads' pane showing a list of threads under 'HAL/Common'. Below it is the 'HAL/Common Objects' pane. The central area displays 'HAL/Common Stacks' with a hierarchical diagram of components. On the right, there's a 'Generate Project Content' button. At the bottom, there's a tabbed interface with 'Summary', 'BSP', 'Clocks', 'Pins', 'Threads', 'Messaging', 'ICU', and 'Components'. The 'Properties' tab is active, showing the properties for the 'g\_i2c0 I2C Master Driver on r\_sci\_i2c'. The properties are organized into 'Settings' and 'Information' sections. The 'Settings' section includes a table of properties and their values.

Property	Value
Common	
Parameter Checking	Default (BSP)
Module g_i2c0 I2C Master Driver on r_sci_i2c	
Name	g_i2c0
Channel	0
Rate	Standard
Slave Address	0x11
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

Use channel 0 since SCI0 is used.

# Main Code

```
[S5D9_ex10a_i2c_iic1_grove_driver] Synergy Configuration  hal_entry.c  startup_S5D9.c  main.c
32 00004a50 g_ioport.p_api->pinWrite(IOPORT_PORT_01_PIN_13, false);
33
34
35 //read acceleration
36 00004a60 err = g_i2c0.p_api->open(g_i2c0.p_ctrl, g_i2c0.p_cfg);
37 00004a6e if (err)
38 00004a70 g_ioport.p_api->pinWrite(IOPORT_PORT_01_PIN_13, true);
39
40
41
42 while (1)
43 {
44     // read xyz value
45 00004a7a buf[0] = 0x02;
46 00004a7e err = g_i2c0.p_api->write(g_i2c0.p_ctrl, buf, 1, false);
47 00004a8e if (err)
48 00004af2 g_ioport.p_api->pinWrite(IOPORT_PORT_01_PIN_13, true);
49 00004a84 err = g_i2c0.p_api->read(g_i2c0.p_ctrl, &buf[7], 6, false);
50 00004a92 if (err)
51 00004b22 g_ioport.p_api->pinWrite(IOPORT_PORT_01_PIN_13, true);
52
53 //read chip id
54 00004a96 buf[0] = 0x00;
55 00004a9a err = g_i2c0.p_api->write(g_i2c0.p_ctrl, buf, 1, false);
56 00004aa6 if (err)
57 00004b16 g_ioport.p_api->pinWrite(IOPORT_PORT_01_PIN_13, true);
58
59 00004aaa err = g_i2c0.p_api->read(g_i2c0.p_ctrl, &buf[7], 1, false);
60 00004ab8 if (err)
61 00004b0a g_ioport.p_api->pinWrite(IOPORT_PORT_01_PIN_13, true);
62
63 //read temperature
64 00004aba buf[0] = 0x08;
65 00004ac0 err = g_i2c0.p_api->write(g_i2c0.p_ctrl, buf, 1, false);
66 00004acc if (err)
67 00004afe g_ioport.p_api->pinWrite(IOPORT_PORT_01_PIN_13, true);
68
69 00004ace err = g_i2c0.p_api->read(g_i2c0.p_ctrl, &buf[7], 1, false);
70 00004adc if (err)
71 00004b2e g_ioport.p_api->pinWrite(IOPORT_PORT_01_PIN_13, true);
72
73 }
74
75 }
```

With the pullup resistors, SDA/SCL signals are correct. Read operation is successful.

