

I2C 개요

I2C란?

Inter-Integrated Circuit

- 발음 : I,I,C 또는 I-two-C 또는 I-Square-C

속도가 느린 직렬통신

- 손쉽게 여러 장치와 통신 가능
- Normal : 100K bps
- Fast : 400K bps
- Fast Plus : 1M bps
- Ultra Fast : 3.4M bps

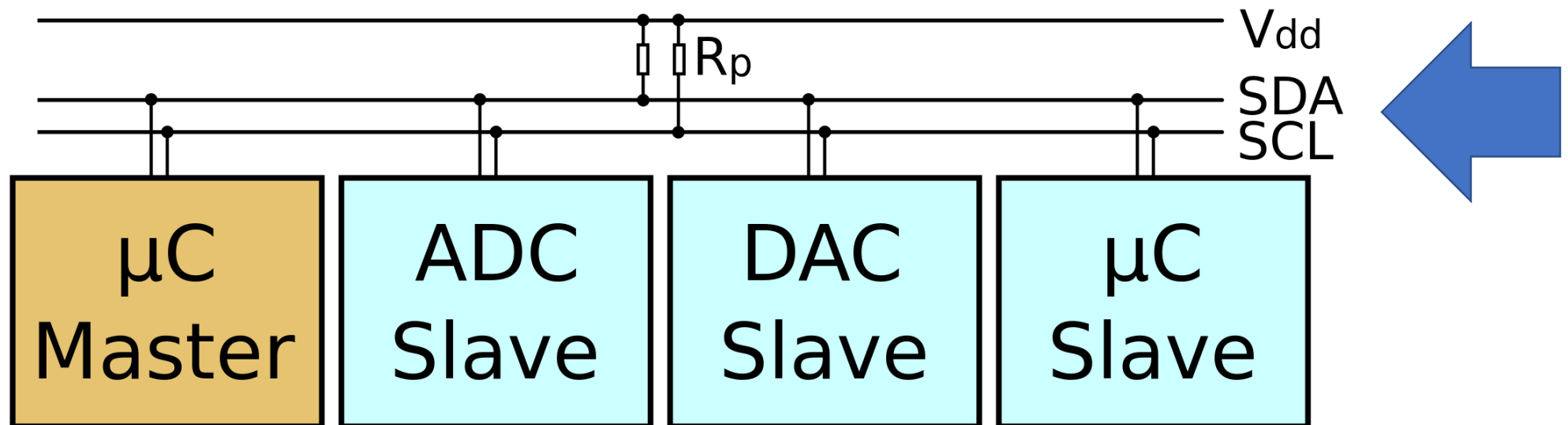
I2C 통신

UART와 다른점

- 비동기가 아닌, 동기 통신이기에 Clock 필요.
- Data 전송하는데 Tx / Rx 2개 Line 이 아닌,
Data 1개 Line 사용
- UART는 CMOS 회로 / I2C는 Open-Drain 회로 사용

2개의 선을 사용한 통신

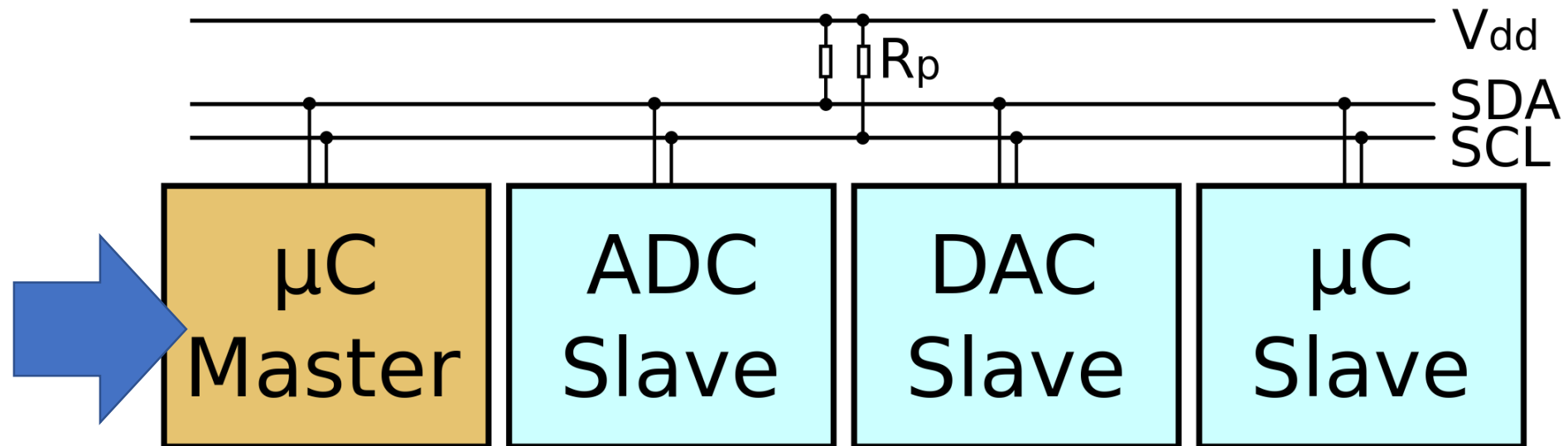
1. SDA (Serial Data와 Address)
2. SCL (Serial Click)



2개의 선을 사용한 통신

uC : 마이크로 컨트롤러 줄임말

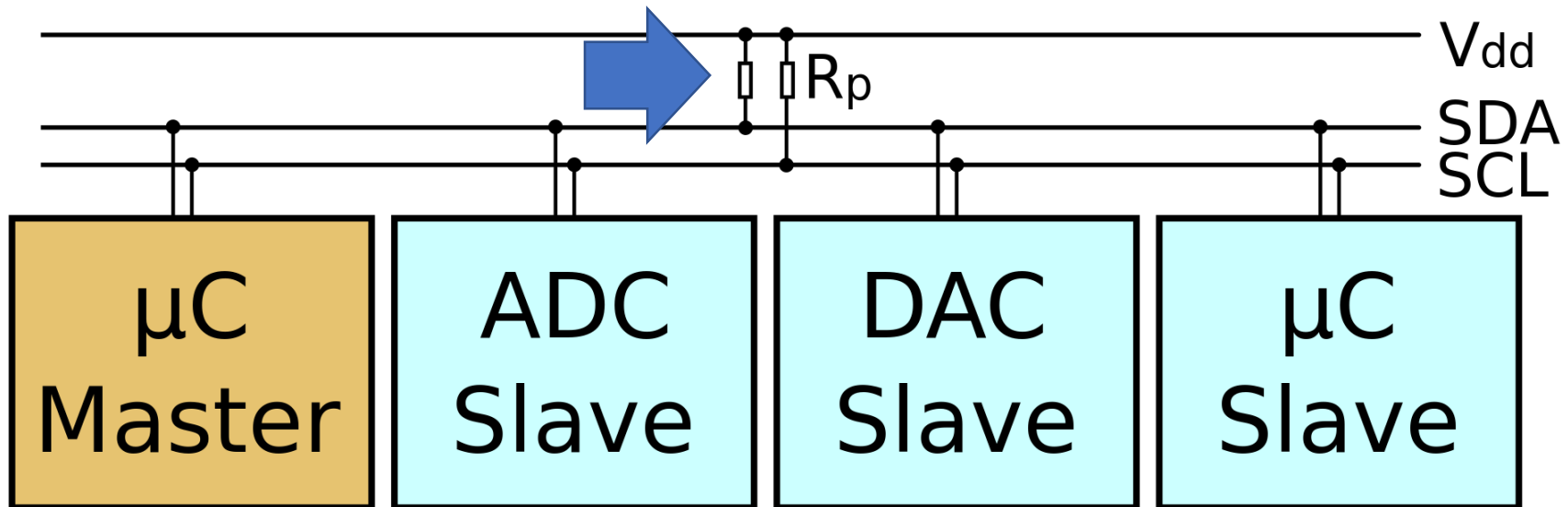
- Master - Slave 통신
 - Mater가 먼저 통신을 시작한다.
 - Master가 Slave를 지정하고, Read 할지, Write할지 결정한다.



2개의 선을 사용한 통신

풀업 저항 필요

- 풀업 저항이 까지 포함해서, Open-Drain 회로 완성
- 3.3k or 4.7k 를 일반적으로 사용하나,
10k옴 + Drive Currnet (High) 로 진행하여 실습

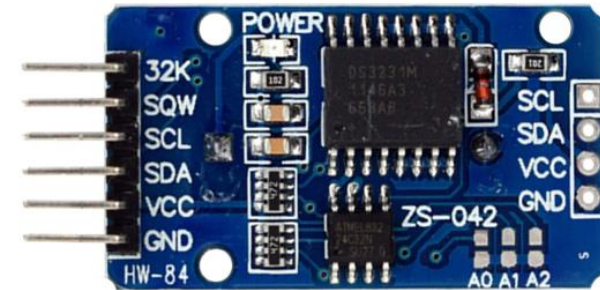


I2C 프로토콜

1. Stop / Stop Signal
2. Slave Address
3. Read / Write
4. Data
5. Acknowledged Bit
(응답을 위한 비트)



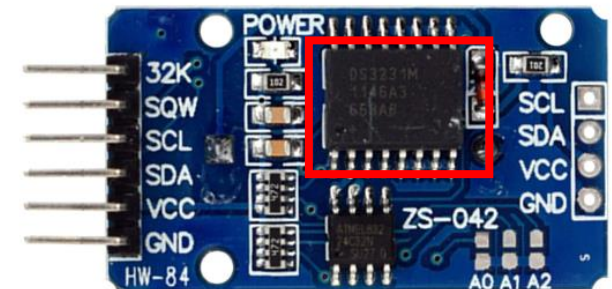
DS3231 제어 (I2C)



DS3231

RTC

달라스를 인수한 Maxim 제품
모듈이 켜진 후 시간을 유지할 수 있음

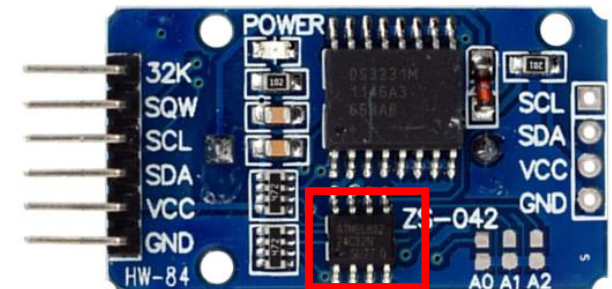


AT24C32

EERPROM

1 M 횟수 Write 가능한 32 Byte 저장공간 칩

I2C 주소 : 0x57



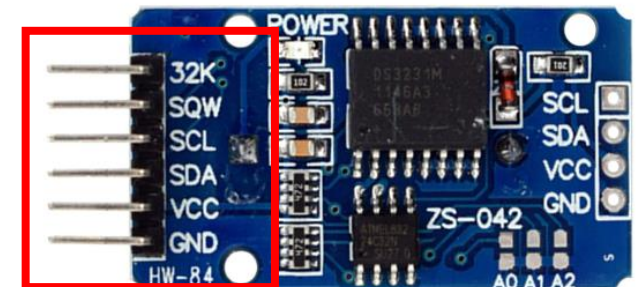
보드 이름 : HW-084A

사용 안할 핀

- 32K : 32KHz Clock 출력 (오픈드레인, 사용시 풀업 필요)
- SQW : 구형파 / Interrupt 출력 옵션

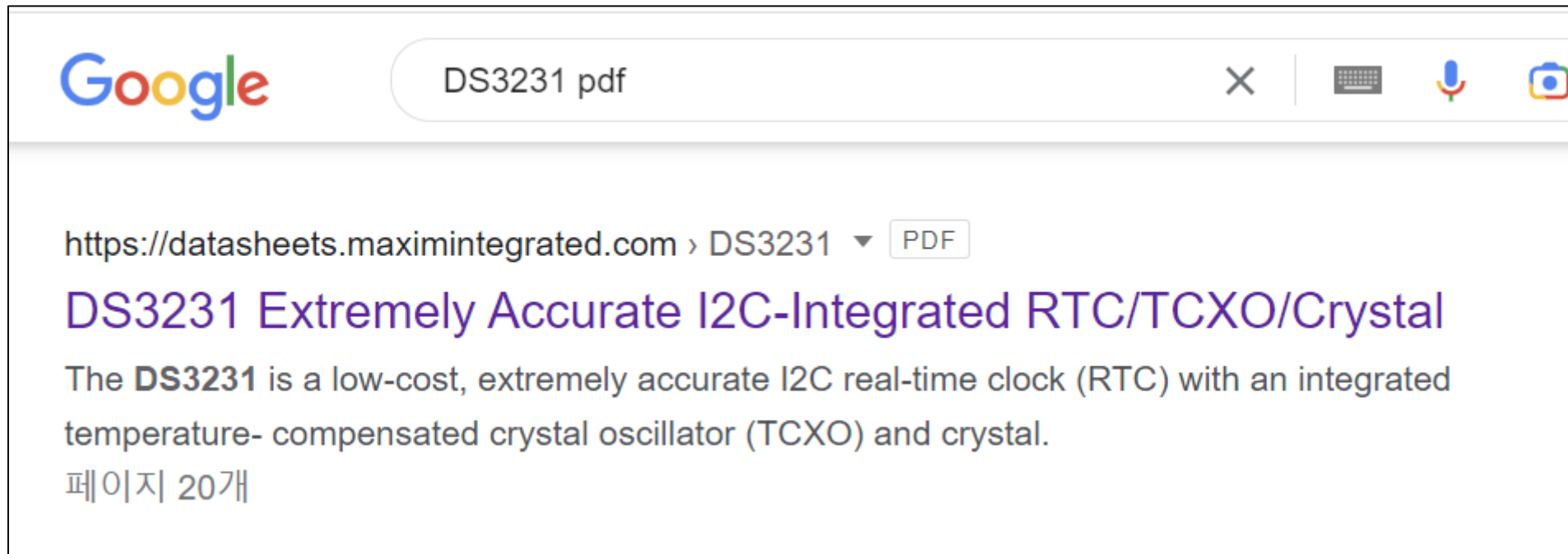
사용할 핀

- VCC : 3.3V ~ 5V
- GND
- SCL : I2C Clock Pin
- SDA : I2C Data Pin



데이터 시트

maxim 에서 제공하는 datasheet 검색



16 Page

Slave Addr

- 0x68 임을 알수있음

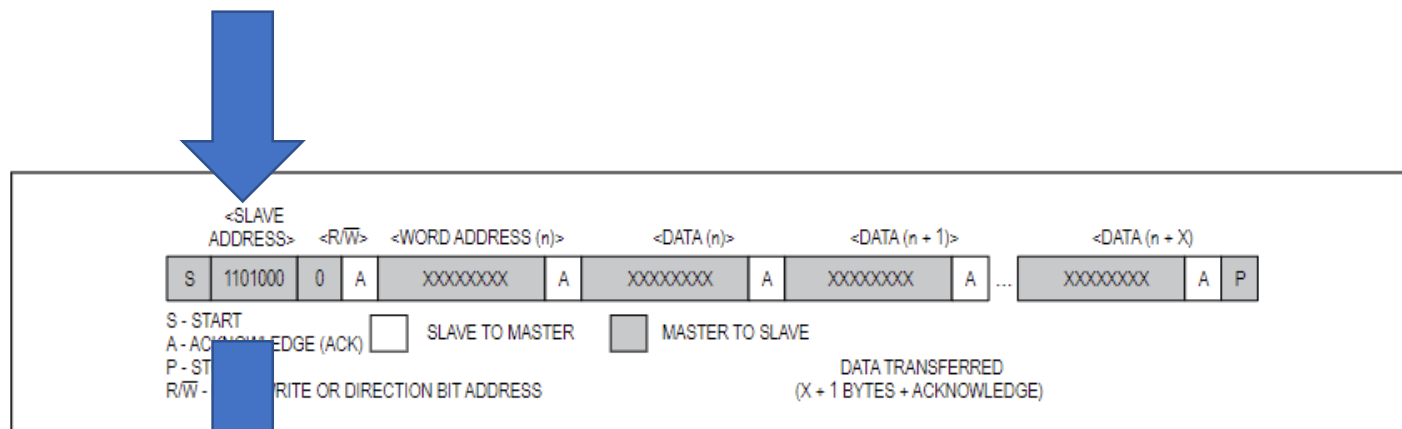


Figure 3. Data Write—Slave Receiver Mode

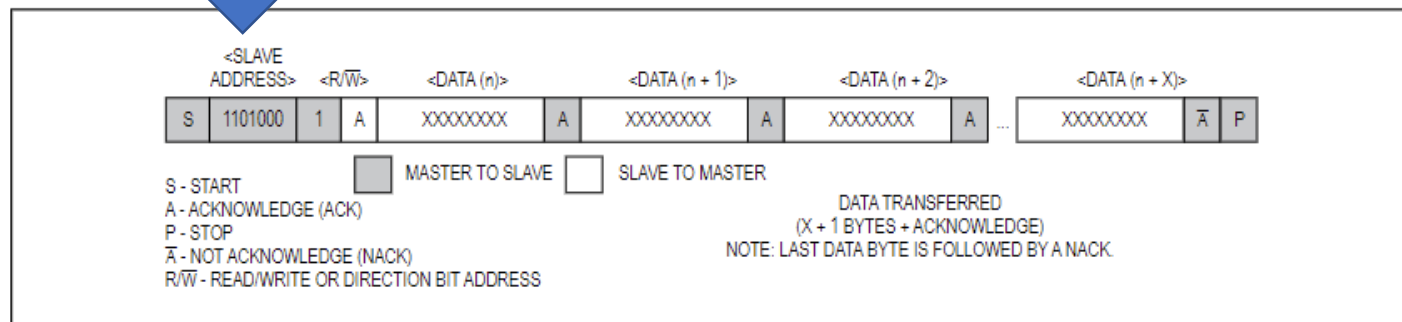


Figure 4. Data Read—Slave Transmitter Mode

11 Page, Address Map

Register Address

해당 Address 읽으면
나오는 값들이
명시되어 있음

ADDRESS	BIT 7 MSB	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0 LSB	FUNCTION	RANGE
00h	0	10 Seconds			Seconds				Seconds	00–59
01h	0	10 Minutes			Minutes				Minutes	00–59
02h	0	12/24	AM/PM 20 Hour	10 Hour	Hour				Hours	1–12 + AM/PM 00–23
03h	0	0	0	0	0	Day			Day	1–7
04h	0	0	10 Date		Date				Date	01–31
05h	Century	0	0	10 Month	Month				Month/ Century	01–12 + Century
06h	10 Year				Year				Year	00–99
07h	A1M1	10 Seconds			Seconds				Alarm 1 Seconds	00–59
08h	A1M2	10 Minutes			Minutes				Alarm 1 Minutes	00–59
09h	A1M3	12/24	AM/PM 20 Hour	10 Hour	Hour				Alarm 1 Hours	1–12 + AM/PM 00–23
0Ah	A1M4	DY/DT	10 Date		Day				Alarm 1 Day	1–7
					Date				Alarm 1 Date	1–31
0Bh	A2M2	10 Minutes			Minutes				Alarm 2 Minutes	00–59
0Ch	A2M3	12/24	AM/PM 20 Hour	10 Hour	Hour				Alarm 2 Hours	1–12 + AM/PM 00–23
0Dh	A2M4	DY/DT	10 Date		Day				Alarm 2 Day	1–7
					Date				Alarm 2 Date	1–31
0Eh	EOSC	BBSQW	CONV	RS2	RS1	INTCN	A2IE	A1IE	Control	—
0Fh	OSF	0	0	0	EN32kHz	BSY	A2F	A1F	Control/Status	—
10h	SIGN	DATA	DATA	DATA	DATA	DATA	DATA	DATA	Aging Offset	—
11h	SIGN	DATA	DATA	DATA	DATA	DATA	DATA	DATA	MSB of Temp	—
12h	DATA	DATA	0	0	0	0	0	0	LSB of Temp	—

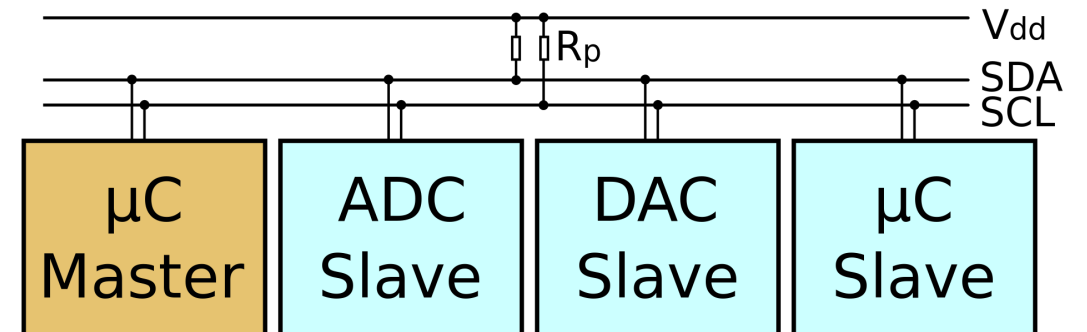
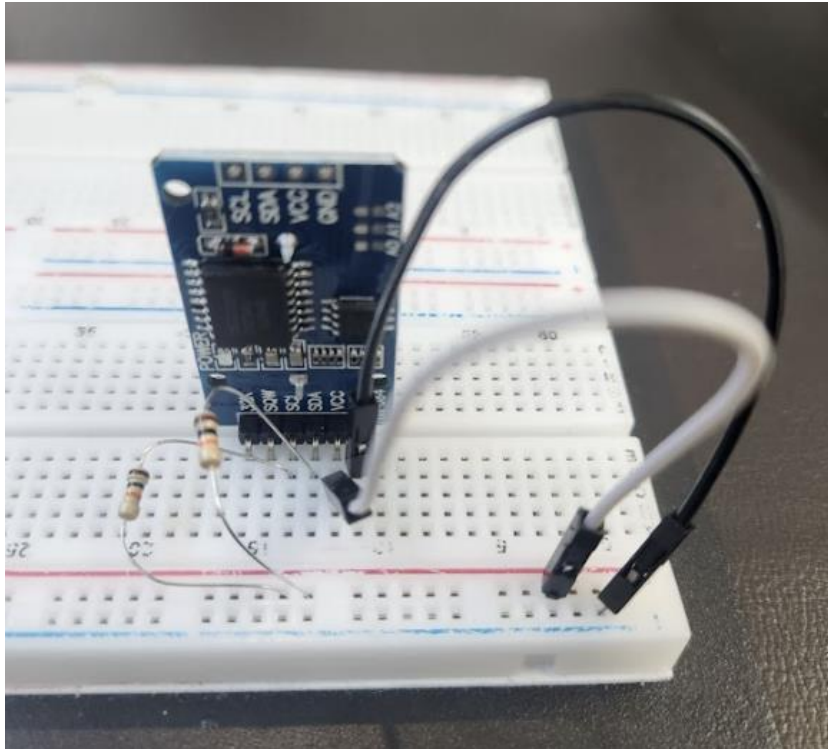
Figure 1. Timekeeping Registers

Note: Unless otherwise specified, the registers' state is not defined when power is first applied.

회로 연결하기



10K : 갈,검,주,골드



MCU 어디에 연결을 할까?

RA6E1 Datasheet를 살펴보자.

I2C 제어 방법

1. SCI로 Simple IIC 사용

- 단순 Master 개발용

2. IIC Interface

Simple IIC 를 먼저 실습해보고,
IIC Interface도 사용해보자.

29. Serial Communications Interface (SCI)

29.1 Overview

The Serial Communications Interface (SCI) × 6 channels have asynchronous and synch

- Asynchronous interfaces (UART and Asynchronous Communications Interface Ad
- 8-bit clock synchronous interface
- Simple IIC (master-only)
- Simple SPI
- Smart card interface
- Manchester interface
- Extended Serial interface

30. I²C Bus Interface (IIC)

30.1 Overview

The I²C bus interface (IIC) has 2 channels. The IIC (Integrated Circuit) bus interface functions.

Table 30.1 lists the IIC specifications, Figure 30.1 shows connections to external circuits, with an I²C bus con

[참고] Simple IIC vs IIC Interface

- <https://www.renesas.com/document/apn/rx-family-specification-differences-between-riic-and-scisimple-i2c-mode-and-selection-guide-rev100?language=en>

Simple IIC 사용하기

SCI 채널 0, 1, 2, 3, 9 모두 사용할 수 있음

하지만 아두이노 헤더만 현재 사용할 수 있으므로 (납땀..)

SCI 9번 채널을 사용하여 실습하자.

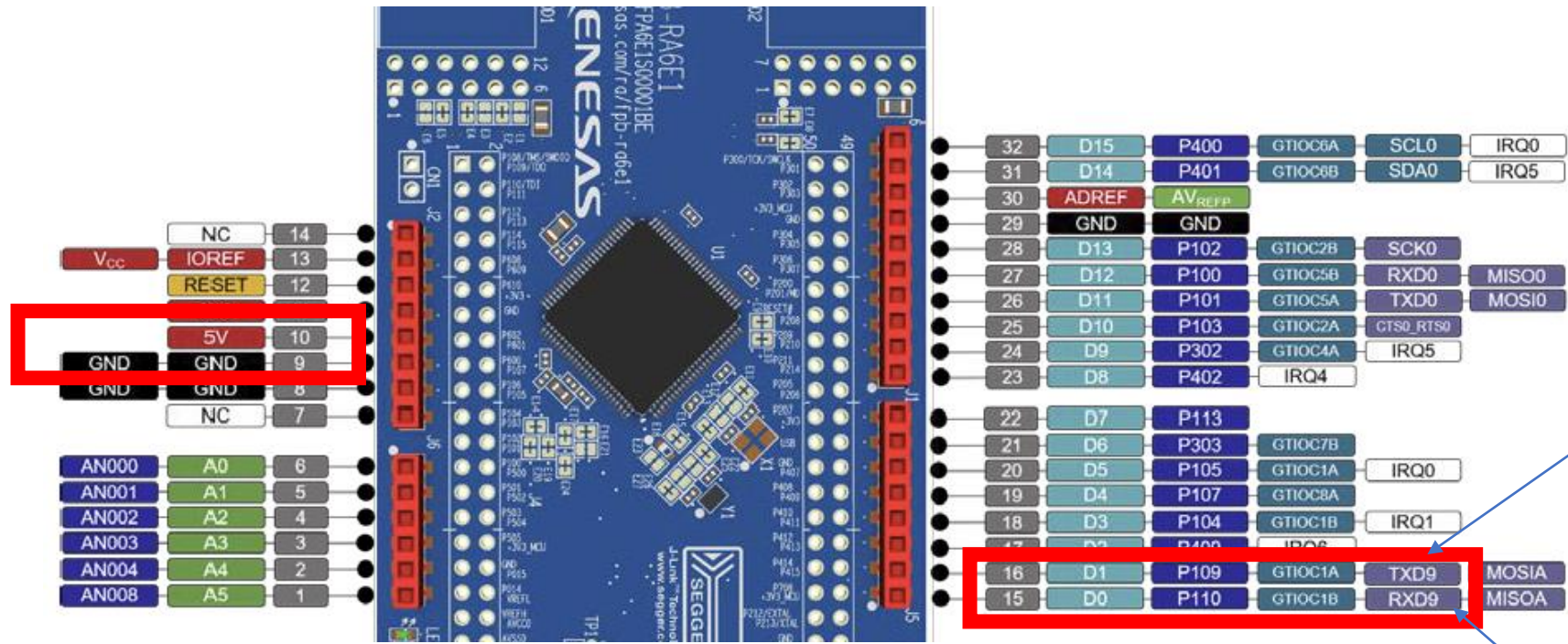
Table 29.2 Functions of SCI Channel (1 of 2)

Item	SCI0, SCI9	SCI3, SCI4	SCI1	SCI2
Asynchronous mode	Available	Available	Available	Available
Clock synchronous mode	Available	Available	Available	Available
Smart card interface mode	Available	Available	Available	Available
Simple I2C mode	Available	Available	Available	Available
Simple SPI mode	Available	Available	Available	Available
FIFO mode	Available	Available	Not Available	Not Available
Address match	Available	Available	Not Available	Not Available
Manchester mode	Not Available	Available	Not Available	Not Available

SCI 9번 채널

SDA : P109

SCL : P110



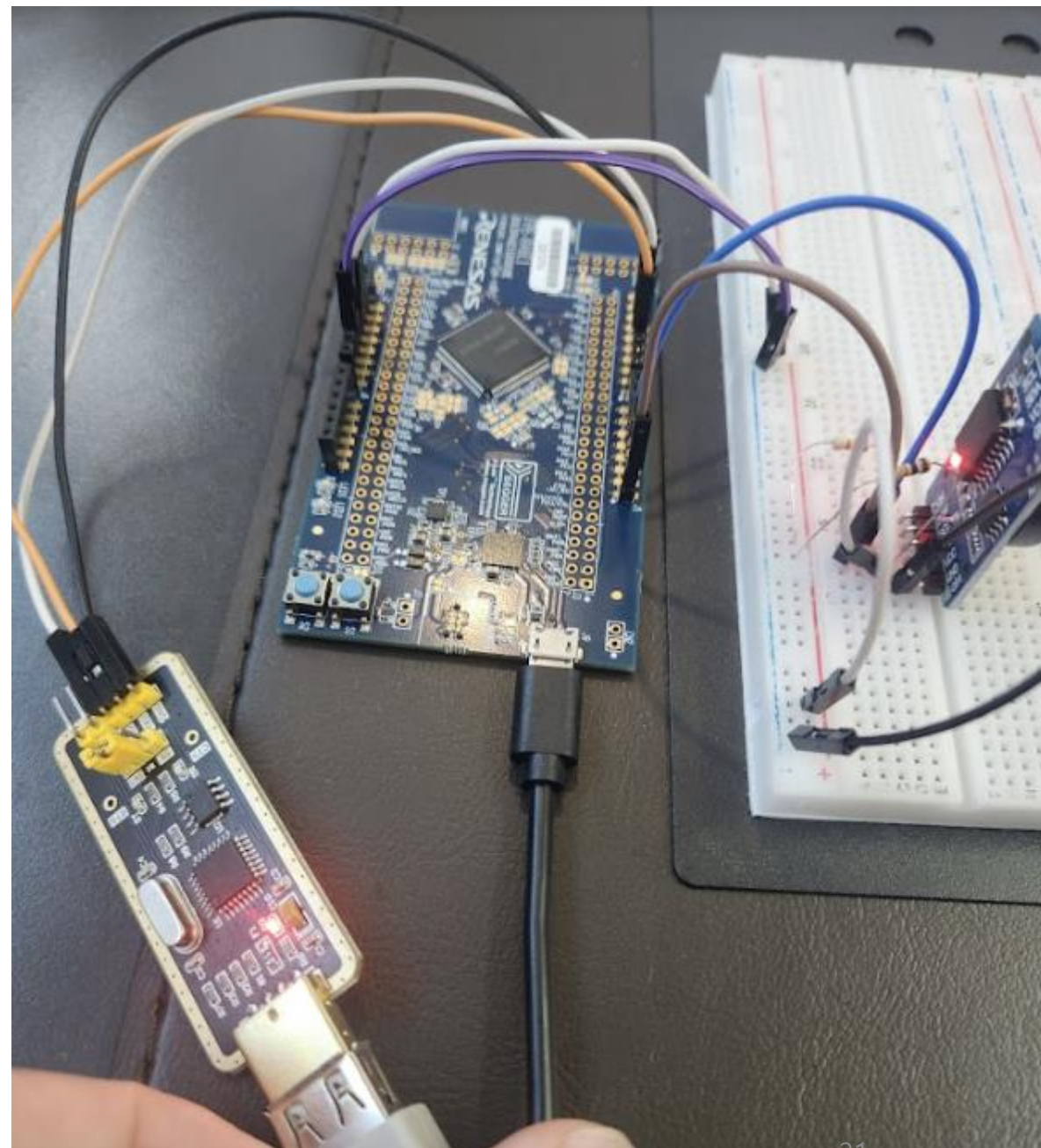
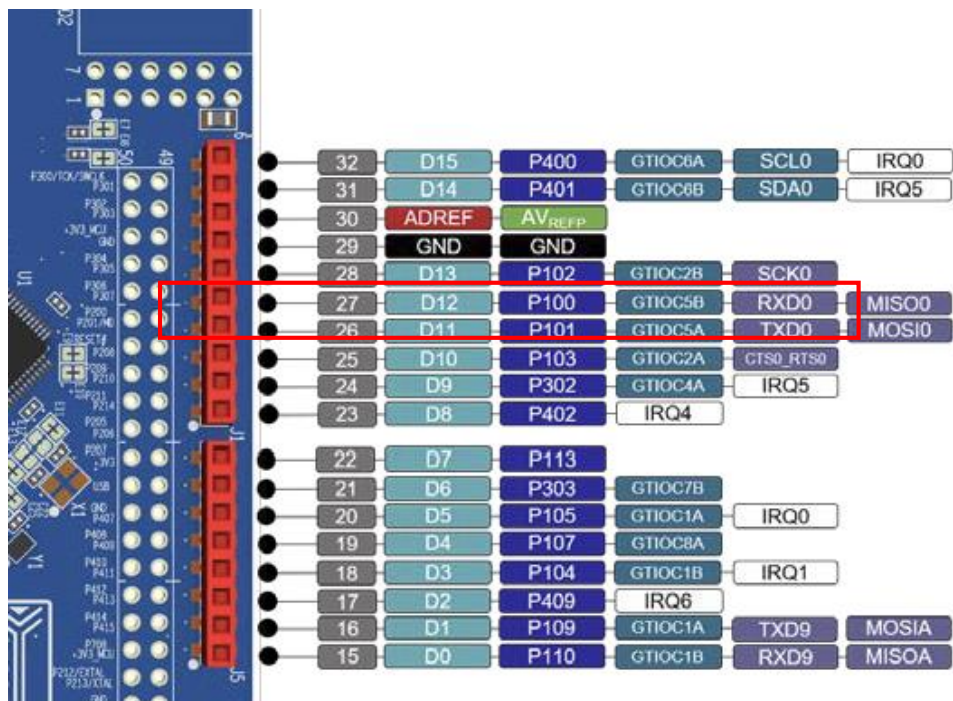
SDA

SCL

연결완료

UART

I2C 연결



SCI 핀 설정

SCI0

- UART 설정

SCI9

- Simple I2C로 설정

The screenshot shows the 'Pin Configuration' window for the 'FPB_RA6E1.pincfg' file. The 'Pin Selection' tree on the left has 'Connectivity:SCI' expanded, with 'SCI0' selected. The 'Pin Configuration' table on the right shows the 'Operation Mode' set to 'Asynchronous UART' and the 'Input/Output' pins configured as follows:

Name	Value
Pin Group Selection	Mixed
Operation Mode	Asynchronous UART
Input/Output	
CTS0	None
CTS_RTS0	None
RXD0	✓ P100
-	None
TXD0	✓ P101

The screenshot shows the 'Pin Configuration' window for the 'FPB_RA6E1.pincfg' file. The 'Pin Selection' tree on the left has 'Connectivity:SCI' expanded, with 'SCI9' selected. The 'Pin Configuration' table on the right shows the 'Operation Mode' set to 'Simple I2C' and the 'Input/Output' pins configured as follows:

Name	Value
Pin Group Selection	Mixed
Operation Mode	Simple I2C
Input/Output	
-	None
-	None
SCL9	✓ P110
-	None
SDA9	✓ P109

친절한 가이드

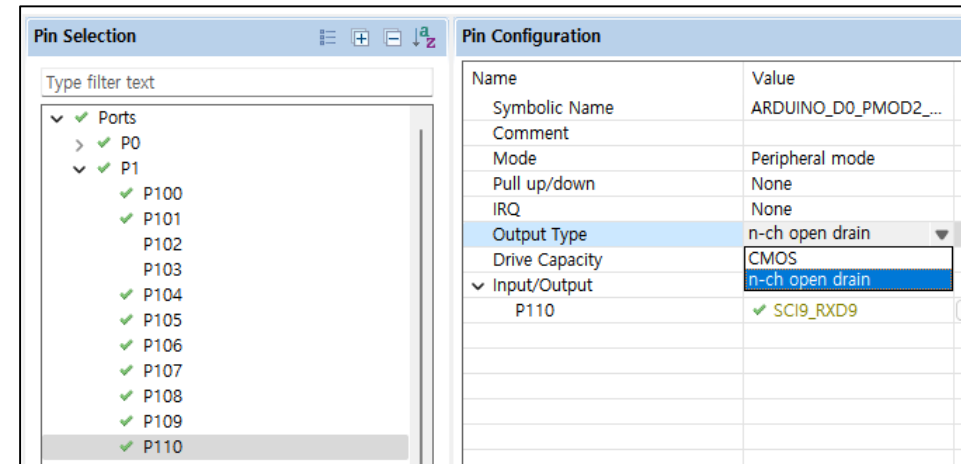
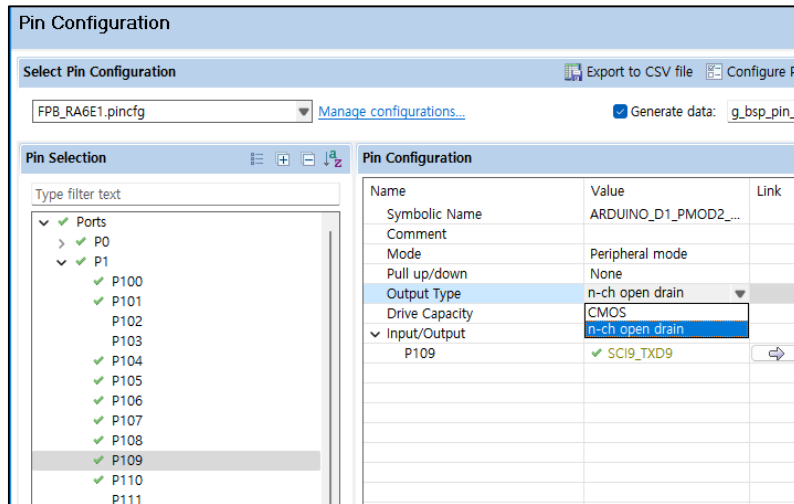
1. Simple I2C를 선택하면
오픈드레인 설정할 것
2. 동작모드를 Disable 부터 누르고
Simple I2C를 선택할 것
3. PIN 이름은
TXD는 SDA 이고,
RXD는 SCL 이다.

[illegible]

핀 설정 설정

P110, P109 핀 설정 (2개 핀 모두 설정해주세요)

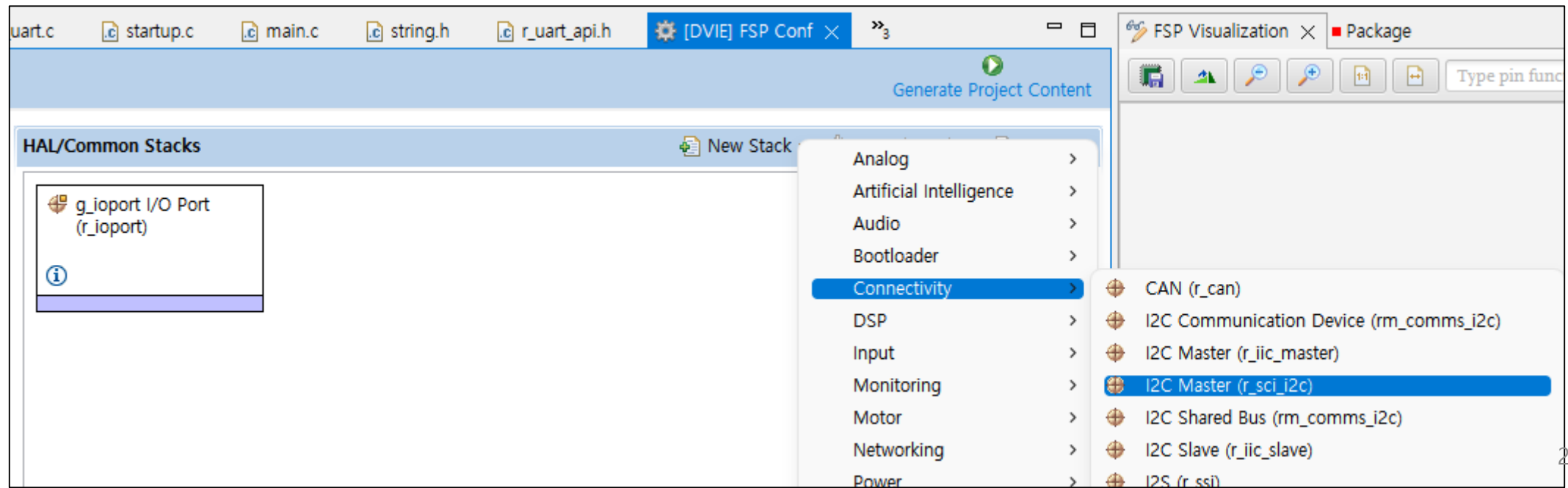
1. Open-Drain Output 회로
2. Drive Capacity : High



HAL Stack 추가하기

r_sci_i2c : SCI Interface의 I2C (Simple IIC)

r_iic_master : I2C Interface



Stack 설정

Channel : 9 (SCI 9번)

Slave Addr : 0x68

Speed : Standard

자동으로

Callback 함수명이 지정되어있음

Properties Problems Smart Browser

g_i2c0 I2C Master (r_sci_i2c)

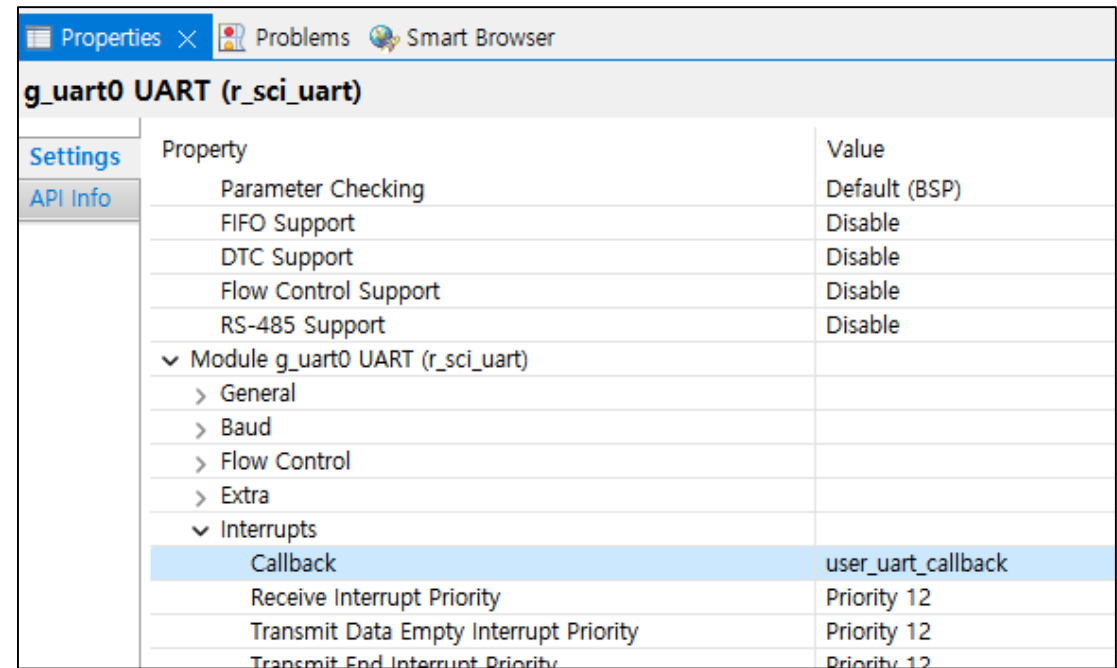
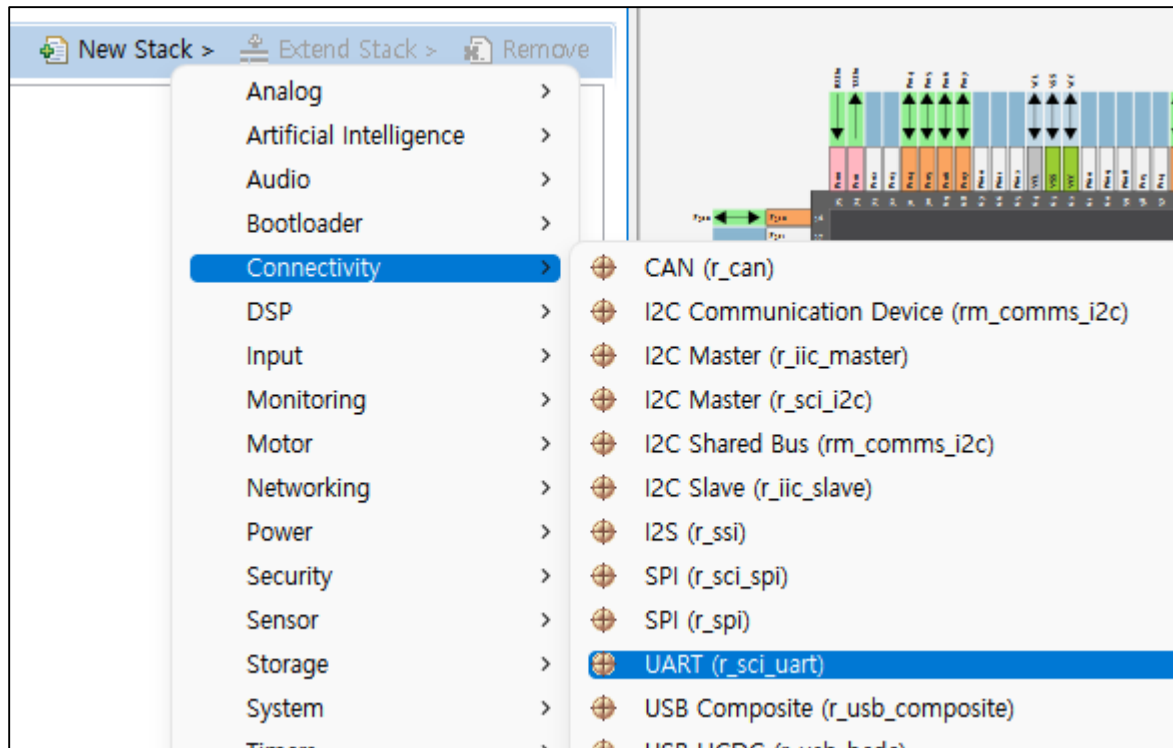
Settings

API Info

Property	Value
Common	
Parameter Checking	Default (BSP)
DTC on Transmission and Reception	Disabled
10-bit slave addressing	Disabled
Module g_i2c0 I2C Master (r_sci_i2c)	
Name	g_i2c0
Channel	9
Slave Address	0x68
Address Mode	7-Bit
Rate	Standard
SDA Output Delay (nano seconds)	300
Noise filter setting	Use clock signal divided by 1 with no
Bit Rate Modulation	Enable
Callback	sci_i2c_master_callback
Interrupt Priority Level	Priority 12
RX Interrupt Priority Level [Only used when DTC is enable	Disabled

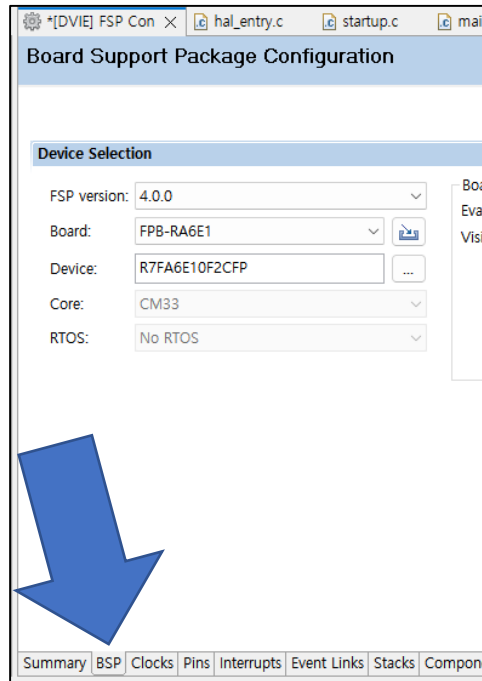
HAL UART Stack 추가하기

Callback : **user_uart_callback** 으로 이름 지정



printf를 쓰기위한 Heap 설정

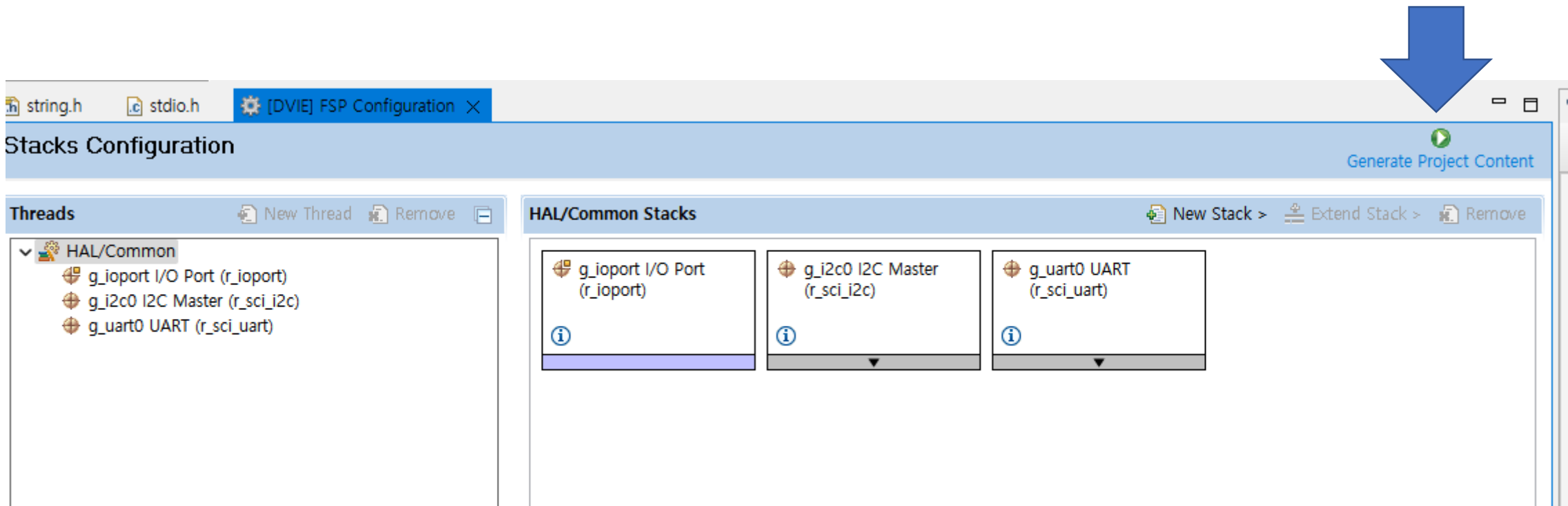
힙 사이즈 0x1000 으로 세팅



Property	Value
> Permanent Block Protection Settings (PBPS)	
> Clocks	
Startup C-Cache Line Size	32 Bytes
Dual Bank Mode	Disabled
Main Oscillator Wait Time	8763 cycles
▼ RA Common	
Main stack size (bytes)	0x400
Heap size (bytes)	0x1000
MCU Vcc (mV)	3300

HAL 추가 완료

Generate Project



WarmStart

Open 코드 추가

```
void R_BSP_WarmStart(bsp_warm_start_event_t event)
{
    if (BSP_WARM_START_RESET == event)
    {
    }

    if (BSP_WARM_START_POST_C == event)
    {
        /* C runtime environment and system clocks are setup. */

        /* Configure pins. */
        R_IOPORT_Open(&g_ioport_ctrl, g_ioport.p_cfg);
        R_SCI_UART_Open(&g_uart0_ctrl, &g_uart0_cfg);
        R_SCI_I2C_Open(&g_i2c0_ctrl, &g_i2c0_cfg);
    }
}
```

scanf / printf 코드 가져오기

<https://gist.github.com/mincoding1/8226e90f554eebafdf00ef7d51a73605>

(첨부파일 : Source > uart_io.h)

src>uart_io.h 에 추가한다.

UART에서 printf, scanf를 사용하기 위한 코드

uart_io.h

```
1  #ifndef UART_IO_H_
2  #define UART_IO_H_
3
4  #if 0
5  //사용 가이드 (by mincoding)
6
7  //1. 힙 사이즈 (FSP Configuration > BSP > HeapSize) 0x10000
8  //2. 인터럽트 콜백함수 이름을 "user_uart_callback"로 변경
9  //3. 출력버퍼를 사용 안함으로 설정하기 위해 setbuf(stdout, NULL); 코드 삽입
10 // (출력 버퍼를 사용하면 \n 을 수행하기 전 까지, 화면출력이 안됨)
11
12 //예시 소스코드
13 void hal_entry(void)
14 {
15     /* TODO: add your own code here */
16     setbuf(stdout, NULL);
17
18     while(1) {
```

소스코드 추가

1. uart_io.h 파일 #include
2. Callback 함수 등록
 - 마우스 드래그로 등록
3. 무한루프, printf문 추가

```
#include "hal_data.h"
#include "uart_io.h"

FSP_CPP_HEADER
void R_BSP_WarmStart(bsp_warm_start_event_t event);
FSP_CPP_FOOTER

/* Callback function */
void sci_i2c_master_callback(i2c_master_callback_args_t *p_args)
{
    /* TODO: add your own code here */
}

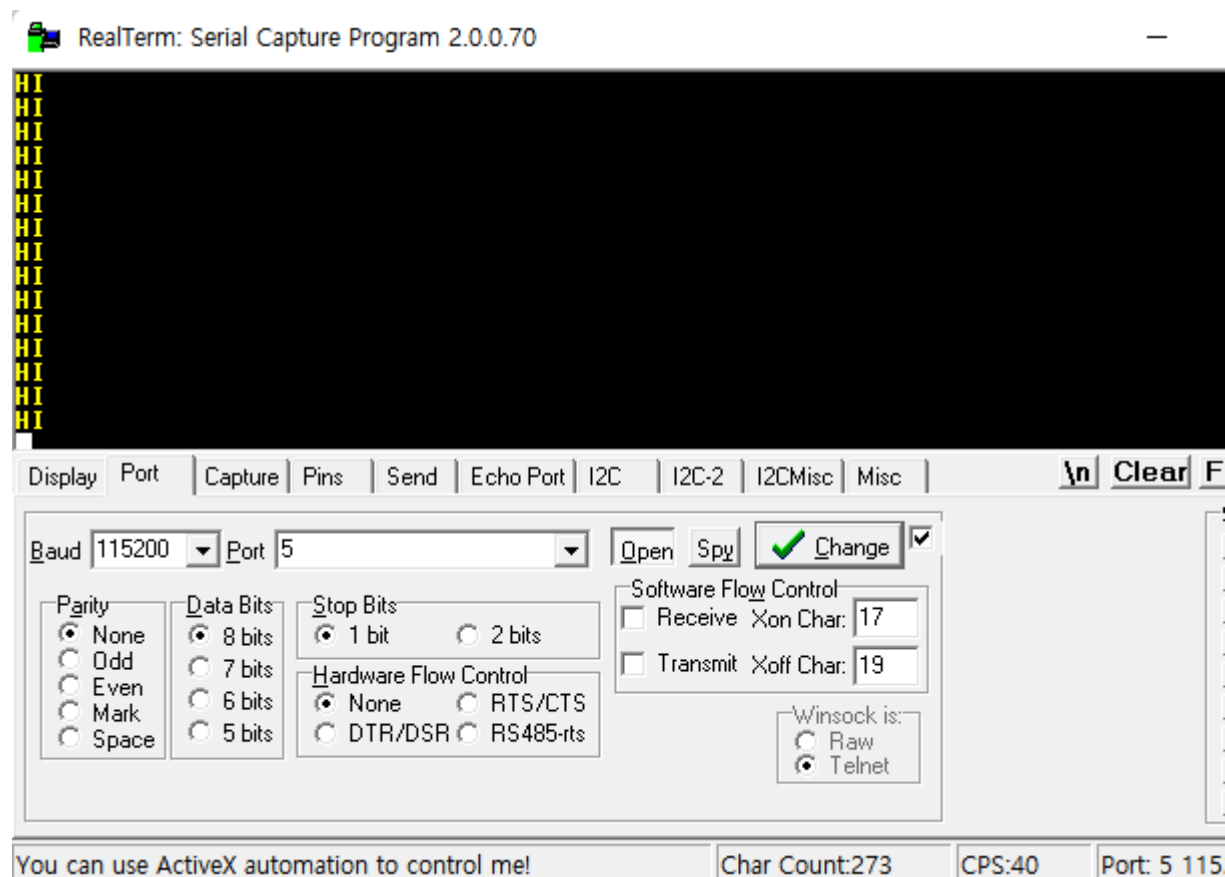
void hal_entry(void)
{
    setbuf(stdout, NULL);

    while(1) {
        printf("HI\r\n");
        R_BSP_SoftwareDelay(100, BSP_DELAY_UNITS_MILLISECONDS);
    }
}
```


UART Test 완료

출력이 잘 된다.

이제 I2C 할 차례



[참고] I2C 르네사스 공식 예제코드

- https://renesas.github.io/fsp/group_s_c_i_i2_c.html

Examples

Basic Example

This is a basic example of minimal use of the r_sci_i2c in an application. This example shows how this driver can be used for basic read and write operations.

```
void basic_example(void)
{
    fsp_err_t err;
    uint32_t i;
    uint32_t timeout_ms = I2C_TRANSACTION_BUSY_DELAY;

    /* Initialize the I2C module */
    err = R_SCI_I2C_Open(&g_i2c_device_ctrl_1, &g_i2c_device_cfg_1);

    /* Handle any errors. This function should be defined by the user. */
    assert(FSP_SUCCESS == err);

    /* Write some data to the transmit buffer */
    for (i = 0; i < I2C_BUFFER_SIZE_BYTES; i++)
    {
        g_i2c_tx_buffer[i] = (uint8_t) i;
    }

    /* Send data to I2C slave */
    g_i2c_callback_event = I2C_MASTER_EVENT_ABORTED;

    err = R_SCI_I2C_Write(&g_i2c_device_ctrl_1, &g_i2c_tx_buffer[0], I2C_BUFFER_SIZE_BYTES, false);
    assert(FSP_SUCCESS == err);

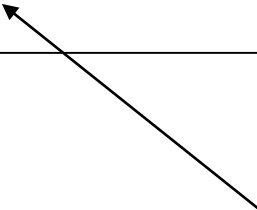
    /* Since there is nothing else to do, block until Callback triggers */
    while ((I2C_MASTER_EVENT_TX_COMPLETE != g_i2c_callback_event) && timeout_ms)
    {
        R_BSP_SoftwareDelay(1U, BSP_DELAY_UNITS_MILLISECONDS);
        timeout_ms--;
    }
}
```

코드 추가하기

Callback의 이벤트를 전역변수에 저장

```
volatile i2c_master_event_t g_i2c_callback_event;  
void sci_i2c_master_callback(i2c_master_callback_args_t *p_args)  
{  
    g_i2c_callback_event = p_args->event;  
}
```

```
/** Callback events */  
typedef enum e_i2c_master_event  
{  
    I2C_MASTER_EVENT_ABORTED      = 1,  
    I2C_MASTER_EVENT_RX_COMPLETE = 2,  
    I2C_MASTER_EVENT_TX_COMPLETE = 3  
} i2c_master_event_t;
```



1, 2, 3 중에 하나의 값이 대입 됨

초 읽을 예정

Addr : 0x00

Size : 1 Byte

ADDRESS	BIT 7 MSB	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0 LSB	FUNCTION	RANGE
00h	0	10 Seconds			Seconds				Seconds	00–59
01h	0	10 Minutes			Minutes				Minutes	00–59
02h	0	12/24	AM/PM 20 Hour	10 Hour	Hour				Hours	1–12 + AM/PM 00–23
03h	0	0	0	0	0	Day			Day	1–7
04h	0	0	10 Date		Date				Date	01–31
05h	Century	0	0	10 Month	Month				Month/ Century	01–12 + Century
06h	10 Year				Year				Year	00–99
07h	A1M1	10 Seconds			Seconds				Alarm 1 Seconds	00–59
08h	A1M2	10 Minutes			Minutes				Alarm 1 Minutes	00–59
09h	A1M3	12/24	AM/PM 20 Hour	10 Hour	Hour				Alarm 1 Hours	1–12 + AM/PM 00–23
0Ah	A1M4	DY/DT	10 Date		Day				Alarm 1 Day	1–7
					Date				Alarm 1 Date	1–31
0Bh	A2M2	10 Minutes			Minutes				Alarm 2 Minutes	00–59
0Ch	A2M3	12/24	AM/PM 20 Hour	10 Hour	Hour				Alarm 2 Hours	1–12 + AM/PM 00–23
0Dh	A2M4	DY/DT	10 Date		Day				Alarm 2 Day	1–7
					Date				Alarm 2 Date	1–31
0Eh	EOSC	BBSQW	CONV	RS2	RS1	INTCN	A2IE	A1IE	Control	—
0Fh	OSF	0	0	0	EN32kHz	BSY	A2F	A1F	Control/Status	—
10h	SIGN	DATA	DATA	DATA	DATA	DATA	DATA	DATA	Aging Offset	—
11h	SIGN	DATA	DATA	DATA	DATA	DATA	DATA	DATA	MSB of Temp	—
12h	DATA	DATA	0	0	0	0	0	0	LSB of Temp	—

Figure 1. Timekeeping Registers

Note: Unless otherwise specified, the registers' state is not defined when power is first applied.

Delay 함수 하나 추가하기

가독성을 위한 delay 함수 하나 추가.

```
void delay(int ms) {  
    R_BSP_SoftwareDelay(ms, BSP_DELAY_UNITS_MILLISECONDS);  
}
```

소스코드 추가

1. 읽을 곳 Addr Write로 지정

- restart = true로 해둔다.

2. 읽을 곳을 Read한다.

```
void hal_entry(void)
{
    setbuf(stdout, NULL);

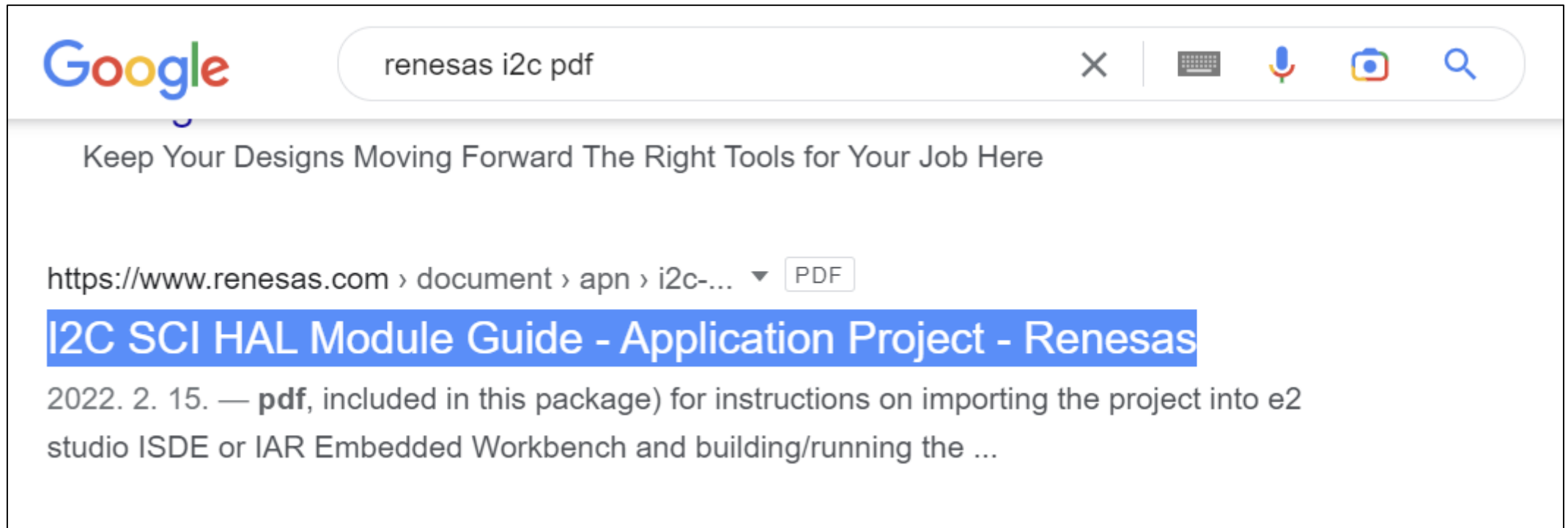
    uint8_t addr = 0x0;
    uint8_t buf[100] = {0};
    while(1) {
        //select addr (by I2C_WRITE)
        addr = 0x0;
        R_SCI_I2C_Write(&g_i2c0_ctrl, &addr, 1, true);
        while(g_i2c_callback_event != I2C_MASTER_EVENT_TX_COMPLETE);

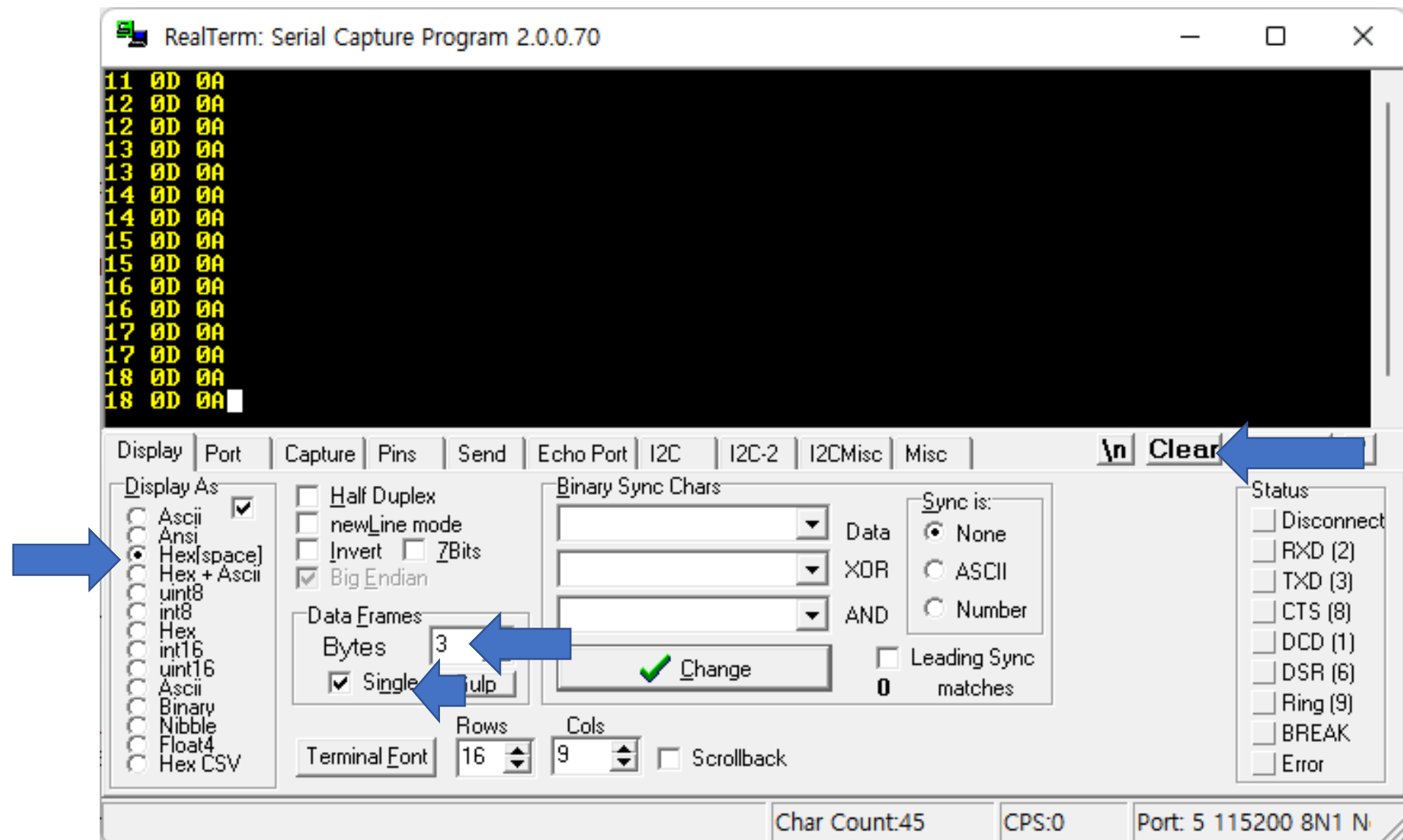
        //read data
        memset(buf, 0, sizeof(buf));
        R_SCI_I2C_Read(&g_i2c0_ctrl, buf, 1, false);
        while(g_i2c_callback_event != I2C_MASTER_EVENT_RX_COMPLETE);

        //print
        printf("%s\r\n", buf);
        delay(500);
    }
}
```

[참고] i2c Guide

SCI i2c 사용 가이드





[도전] 시, 분, 초 읽어보기

ADDRESS	BIT 7 MSB	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0 LSB	FUNCTION	RANGE
00h	0	10 Seconds			Seconds				Seconds	00–59
01h	0	10 Minutes			Minutes				Minutes	00–59
02h	0	12/24	AM/PM 20 Hour	10 Hour	Hour				Hours	1–12 + AM/PM 00–23
03h	0	0	0	0	0	Day			Day	1–7
04h	0	0	10 Date			Date			Da	
05h	Century	0	0	10 Month	Month				Mor Cent	
06h	10 Year			Year			Ye			
07h	A1M1	10 Seconds			Seconds				Alarm 1 S	
08h	A1M2	10 Minutes			Minutes				Alarm 1	
09h	A1M3	12/24	AM/PM 20 Hour	10 Hour	Hour				Alarm 1	
0Ah	A1M4	DY/DT	10 Date			Day			Alarm	
0Bh	A2M2	10 Minutes			Minutes				Alarm 2	
0Ch	A2M3	12/24	AM/PM 20 Hour	10 Hour	Hour				Alarm 2	
0Dh	A2M4	DY/DT	10 Date			Day			Alarm	
0Eh	EOSC	BBSQW	CONV	RS2	RS1	INTCN	A2IE	A1IE	Con	
0Fh	OSF	0	0	0	EN32kHz	BSY	A2F	A1F	Control	
10h	SIGN	DATA	DATA	DATA	DATA	DATA	DATA	DATA	Aging	
11h	SIGN	DATA	DATA	DATA	DATA	DATA	DATA	DATA	MSB of	
12h	DATA	DATA	0	0	0	0	0	0	LSB of	

Figure 1. Timekeeping Registers

Note: Unless otherwise specified, the registers' state is not defined when power is first applied.

RealTerm: Serial Capture Program 2.0.0.70

```

55 sec 59 min 0 hours
56 sec 59 min 0 hours
56 sec 59 min 0 hours
57 sec 59 min 0 hours
57 sec 59 min 0 hours
58 sec 59 min 0 hours
58 sec 59 min 0 hours
59 sec 59 min 0 hours
59 sec 59 min 0 hours
0 sec 0 min 1 hours
0 sec 0 min 1 hours
1 sec 0 min 1 hours
1 sec 0 min 1 hours
2 sec 0 min 1 hours
2 sec 0 min 1 hours

```

Solution

```
void hal_entry(void)
{
    uint8_t addr = 0x0;
    uint8_t buf[100] = {0};
    uint8_t str[100] = {0};
    while(1) {
        //select addr (by I2C_WRITE)
        addr = 0x0;
        R_SCI_I2C_Write(&g_i2c0_ctrl, &addr, 1, true);
        while(g_i2c_callback_event != I2C_MASTER_EVENT_TX_COMPLETE);

        //read data
        memset(buf, 0, sizeof(buf));
        R_SCI_I2C_Read(&g_i2c0_ctrl, buf, 3, false);
        while(g_i2c_callback_event != I2C_MASTER_EVENT_RX_COMPLETE);

        //print
        sprintf(str, "%X sec %X min %X hours\r\n", buf[0], buf[1], (buf[2] & 0xF));
        printf("%s", str);
        delay(500);
    }
}
```

온도 레지스터

DS3231은

온도 센서도 포함됨

오실레이터가 온도 영향을 받기에, 보정을 위한 센서

ADDRESS	BIT 7 MSB	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0 LSB	FUNCTION	RANGE
00h	0	10 Seconds			Seconds				Seconds	00–59
01h	0	10 Minutes			Minutes				Minutes	00–59
02h	0	12/24	AM/PM 20 Hour	10 Hour	Hour				Hours	1–12 + AM/PM 00–23
03h	0	0	0	0	0	Day			Day	1–7
04h	0	0	10 Date		Date				Date	01–31
05h	Century	0	0	10 Month	Month				Month/ Century	01–12 + Century
06h	10 Year				Year				Year	00–99
07h	A1M1	10 Seconds			Seconds				Alarm 1 Seconds	00–59
08h	A1M2	10 Minutes			Minutes				Alarm 1 Minutes	00–59
09h	A1M3	12/24	AM/PM 20 Hour	10 Hour	Hour				Alarm 1 Hours	1–12 + AM/PM 00–23
0Ah	A1M4	DY/DT	10 Date		Day				Alarm 1 Day	1–7
					Date				Alarm 1 Date	1–31
0Bh	A2M2	10 Minutes			Minutes				Alarm 2 Minutes	00–59
0Ch	A2M3	12/24	AM/PM 20 Hour	10 Hour	Hour				Alarm 2 Hours	1–12 + AM/PM 00–23
0Dh	A2M4	DY/DT	10 Date		Day				Alarm 2 Day	1–7
					Date				Alarm 2 Date	1–31
0Eh	EOSC	BBSQW	CONV	RS2	RS1	INTCN	A2IE	A1IE	Control	—
0Fh	OSF	0	0	0	EN32kHz	BSY	A2F	A1F	Control/Status	—
10h	SIGN	DATA	DATA	DATA	DATA	DATA	DATA	DATA	Aging Offset	—
11h	SIGN	DATA	DATA	DATA	DATA	DATA	DATA	DATA	MSB of Temp	—
12h	DATA	DATA	0	0	0	0	0	0	LSB of Temp	—

Figure 1. Timekeeping Registers

Note: Unless otherwise specified, the registers' state is not defined when power is first applied.

설명

16진수 BCD 코드

- MSB는 2의 보수, 정수부
- LSB는 0.25의 배수, 소수부

ex)

MSB : 0001 1001

LSB : 01 (6~7번 비트만 유효)

인경우

MSB : 0x19 → (10진수 : 25)

LSB : 0x1 → (0.25 x 1 = 0.25)

따라서 +25.25 도

DS3231

Extremely Accurate I²C-Integrated
RTC/TCXO/Crystal

Temperature Register (Upper Byte) (11h)

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
NAME:	Sign	Data	Data	Data	Data	Data	Data	Data
POR:	0	0	0	0	0	0	0	0

Temperature Register (Lower Byte) (12h)

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
NAME:	Data	Data	0	0	0	0	0	0
POR:	0	0	0	0	0	0	0	0

Temperature Registers (11h–12h)

Temperature is represented as a 10-bit code with a resolution of 0.25°C and is accessible at location 11h and 12h. The temperature is encoded in two's complement format. The upper 8 bits, the integer portion, are at location 11h and the lower 2 bits, the fractional portion, are in the upper nibble at location 12h. For example, 00011001 01b = +25.25°C. Upon power reset, the registers are set to a default temperature of 0°C and the controller starts a temperature conversion. The temperature is read on initial application of V_{CC} or I²C access on V_{BAT} and once every 64 seconds afterwards. The temperature registers are updated after each user-initiated conversion and on every 64-second conversion. The temperature registers are read-only.

line while the clock line is high are interpreted as control signals.

Accordingly, the following bus conditions have been defined:

Bus not busy: Both data and clock lines remain high.

START data transfer: A change in the state of the data line from high to low, while the clock line is high, defines a START condition.

STOP data transfer: A change in the state of the data line from low to high, while the clock line is high, defines a STOP condition.

Data valid: The state of the data line represents valid data when, after a START condition, the data line is stable for the duration of the high period of the clock

[도전] 온도 읽어보기

ADDRESS	BIT 7 MSB	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0 LSB	FUNCTION	RANGE
00h	0	10 Seconds			Seconds				Seconds	00–5
01h	0	10 Minutes			Minutes				Minutes	00–5
02h	0	12/24	AM/PM 20 Hour	10 Hour	Hour				Hours	1–12 + A 00–2
03h	0	0	0	0	0	Day			Day	1–7
04h	0	0	10 Date		Date				Date	01–3
05h	Century	0	0	10 Month	Month				Month/ Century	01–12 + C
06h	10 Year				Year				Year	00–9
07h	A1M1	10 Seconds			Seconds				Alarm 1 Seconds	00–5
08h	A1M2	10 Minutes			Minutes				Alarm 1 Minutes	00–5
09h	A1M3	12/24	AM/PM 20 Hour	10 Hour	Hour				Alarm 1 Hours	1–12 + A 00–2
0Ah	A1M4	DY/DT	10 Date		Day				Alarm 1 Day	1–7
					Date				Alarm 1 Date	1–3
0Bh	A2M2	10 Minutes			Minutes				Alarm 2 Minutes	00–5
0Ch	A2M3	12/24	AM/PM 20 Hour	10 Hour	Hour				Alarm 2 Hours	1–12 + A 00–2
0Dh	A2M4	DY/DT	10 Date		Day				Alarm 2 Day	1–7
					Date				Alarm 2 Date	1–3
0Eh	EOSC	BBSQW	CONV	RS2	RS1	INTCN	A2IE	A1IE	Control	—
0Fh	OSF	0	0	0	EN32kHz	BSY	A2F	A1F	Control/Status	—
10h	SIGN	DATA	DATA	DATA	DATA	DATA	DATA	DATA	Aging Offset	—
11h	SIGN	DATA	DATA	DATA	DATA	DATA	DATA	DATA	MSB of Temp	—
12h	DATA	DATA	0	0	0	0	0	0	LSB of Temp	—

RealTerm: Serial Captu

```
Temp : 23.50
Temp : 23.50
Temp : 23.50
Temp : 23.50
Temp : 23.50
Temp : 23.50
Temp : 23.50
Temp : 23.50
Temp : 26.25
Temp : 26.25
Temp : 26.25
Temp : 26.25
Temp : 26.25
Temp : 26.25
Temp : 26.25
```

Figure 1. Timekeeping Registers

Note: Unless otherwise specified, the registers' state is not defined when power is first applied.