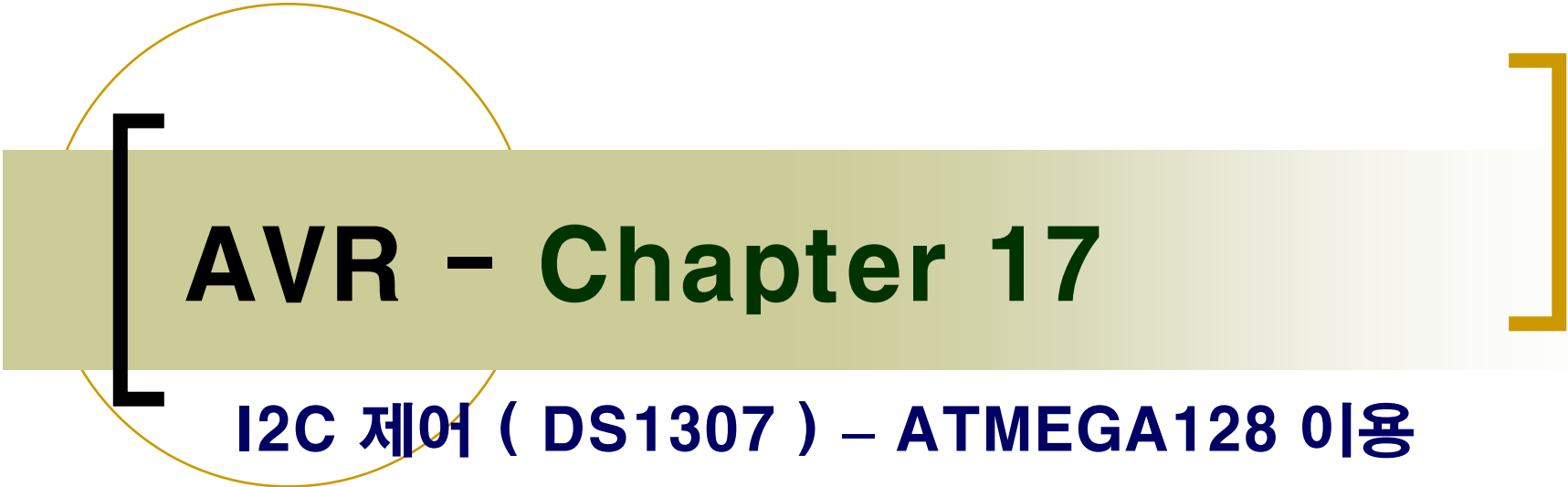




AVR - Chapter 17

I2C 제어 (DS1307) – ATMEGA128 이용

류 대 우

davidryu@newtc.co.kr

[DS1307 RTC를 I2C로 제어]

■ I2C란?

- SCL, SDA 두 선으로 시리얼 통신을 통하여 데이터를 주고 받는 프로토콜
- SCL은 클럭 신호로 동기를 맞추고, SDA에서 데이터를 실어 주고 받는다. 순서는 타이밍도에 따른다.

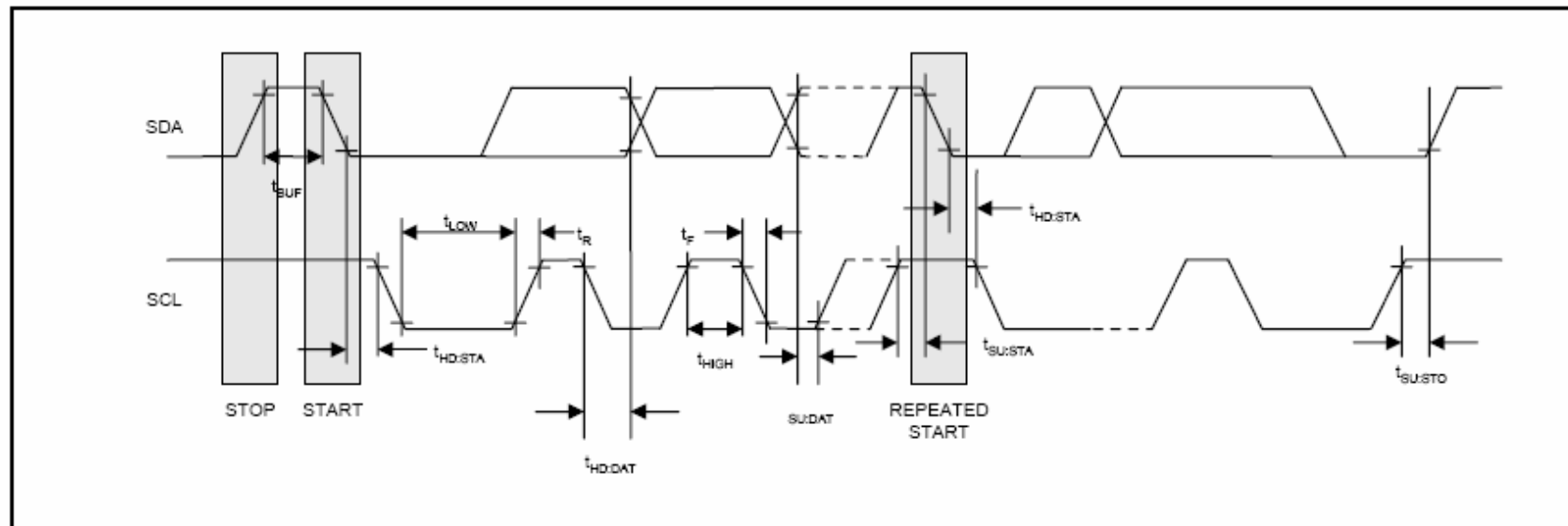
■ RTC란?

- Real Time Clock의 줄임말
- Embedded 장비에서 시간을 체크할 때 사용하며, 배터리가 내장된 것, 내장되지 않은 것, 통신 방식에 따라 여러가지 RTC가 있으며, 필요에 따라 골라서 사용하면 된다.

[I2C 타이밍도

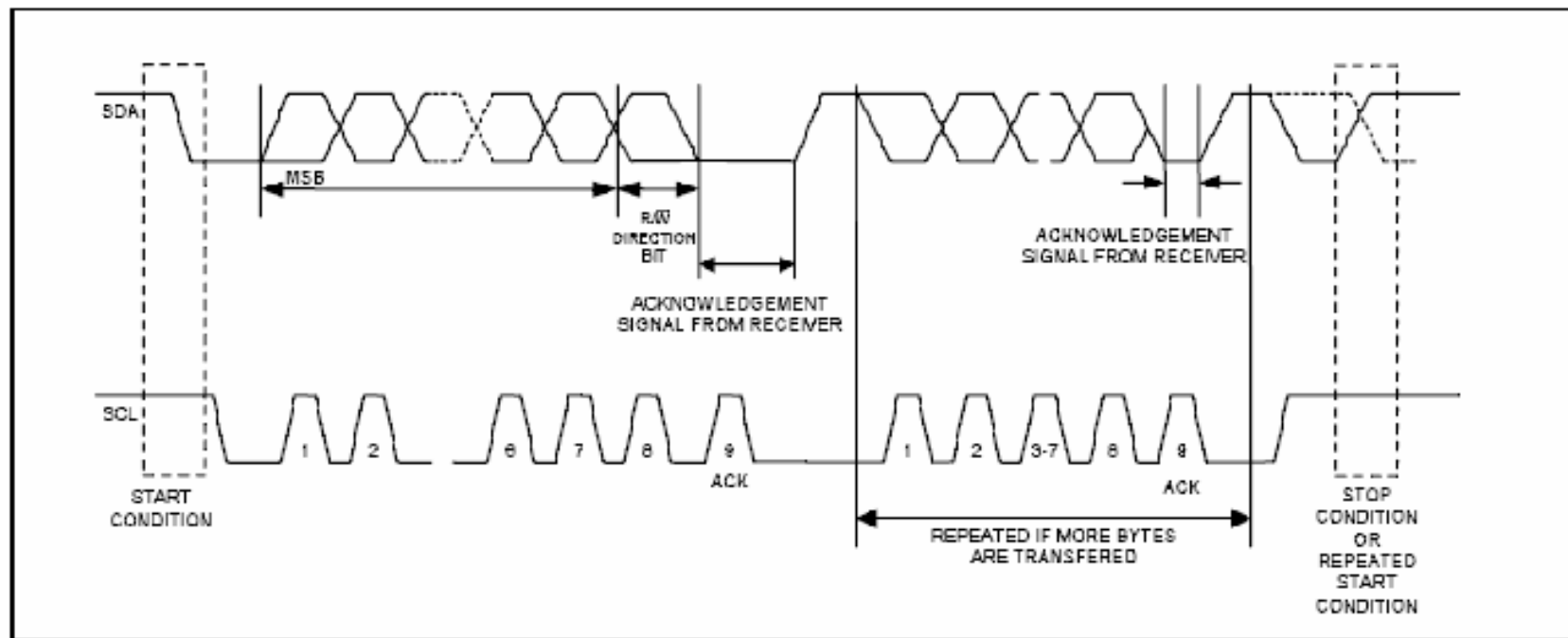


TIMING DIAGRAM



[DS1307시계의 I2C 타이밍도]

Data Transfer on I²C Serial Bus



Depending upon the state of the R/W bit, two types of data transfer are possible:

[Control SPEC



(Over the operating range*)

PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS	NOTES
SCL Clock Frequency	f_{SCL}	0		100	kHz	
Bus Free Time Between a STOP and START Condition	t_{BUF}	4.7			μs	
Hold Time (Repeated) START Condition	$t_{HD:STA}$	4.0			μs	3
LOW Period of SCL Clock	t_{LOW}	4.7			μs	
HIGH Period of SCL Clock	t_{HIGH}	4.0			μs	
Set-up Time for a Repeated START Condition	$t_{SU:STA}$	4.7			μs	
Data Hold Time	$t_{HD:DAT}$	0			μs	4,5
Data Set-up Time	$t_{SU:DAT}$	250			ns	
Rise Time of Both SDA and SCL Signals	t_R			1000	ns	
Fall Time of Both SDA and SCL Signals	t_F			300	ns	
Set-up Time for STOP Condition	$t_{SU:STO}$	4.7			μs	
Capacitive Load for each Bus Line	C_B			400	pF	6
I/O Capacitance ($T_A = 25^\circ C$)	C_{IO}		10		pF	
Crystal Specified Load Capacitance ($T_A = 25^\circ C$)			12.5		pF	

*Unless otherwise specified.

[소스 분석]

■ I2C_start();

- SDA 출력 방향을 AVR0이 출력하는 것으로 함
- SCL, SDA : High로
- SDA : Low로 떨어뜨림
- void I2C_start(void)
- {
- I2C_DIR=SDIROUT;
- sda(1);
- scl(1);I2C_delay();
- sda(0);I2C_delay();
- }

[소스분석 - 계속]

■ I2C_stop();

- SDA 출력 방향을 AVR0이 출력하는 것으로 함
- SCL을 High로
- SDA를 Low로
- SDA를 High로

```
○ void I2C_stop(void)
○ {
○     I2C_DIR=SDIROUT;
○     scl(1);I2C_delay();
○     sda(0);I2C_delay();
○     sda(0);I2C_delay();
○     sda(1);I2C_delay();
○ }
```

[소스분석 - 계속]

- `init_ds1307();`
 - `I2C_start();` 를 실행한다.
 - DS1307 칩을 초기화 한다.
 - 시간, 분, 초 등을 초기 값으로 초기화 한다.
 - 동작 확인용 LED를 달기 위하여,
1Hz를 사용 가능하도록 설정한다.

- `void init_ds1307(void){`
 - `I2C_start();`
 - `I2C_write(ADDR_RTC);`
 - `I2C_write(0x00);`
 - `I2C_write(0x00);`
 - `I2C_stop();`

[소스분석 - 계속]

- I2C_start();
- I2C_write(ADDR_RTC);
- I2C_write(0x00);
- I2C_write(sec);
- I2C_write(min);
- I2C_write(hr);
- I2C_write(dy);
- I2C_write(dt);
- I2C_write(mn);
- I2C_write(yr);
- I2C_write(0x10); // sqwe:enable 1Hz output
- I2C_stop();
- }

[소스분석 - 계속

- **writebyte(어드레스, 데이터)**
 - 설명 : 어드레스에 데이터를 쓴다.
 - SDA 출력을 AVR에서 출력하는 것으로 한다.
 - 1.SCL을 Low로 떨어뜨린다.
 - 2.최상위 비트부터 한 비트 쓴다.
 - 3.SCL을 High로 올린다.
 - 4.답 비트 준비
 - 1,2,3,4을 8번 반복한다.
 - Ack를 체크한다.(그냥 무시하고 지나가고 있다.)

[소스분석 - 계속]

- `void writebyte(unsigned char addr, unsigned char data){`
- `I2C_start();`
- `I2C_write(ADDRRTC);`
-
- `I2C_DIR=SDIROUT;`
- `I2C_write(addr);`
- `I2C_write(data);`
- `I2C_stop();`
- `}`

[소스분석 - 계속]

■ 데이터 = readbyte(어드레스)

- 설명 : 어드레스에서 데이터를 읽는다.
- SDA핀을 AVR에서 입력 받는 것으로 한다.
- 1. SCL을 High로 올린다.(I2C에서 데이터 수신 받을 때 안정적인 쪽은 클럭이 High일 때이므로)
- 2. 최상위비트부터 한 비트 받는다.
- 3. SCL을 Low로 떨어뜨린다.
- 4. 담 비트 받을 준비한다.
- 1,2,3,4를 8번 반복한다.
- Ackn을 받는다.

[소스분석 - 계속]

- unsigned char readbyte(unsigned char addr){
- unsigned char byte;
- I2C_start();
- I2C_write(ADDR_RTC);
- I2C_write(addr);
-
- I2C_start();
- I2C_write(ADDR_RTC|1);
- byte = I2C_read(NACK);
- I2C_stop();
- return byte;
- }

[소스분석 - 계속]

- 데이터 = I2C_read();
 - 현재 번지에서 데이터를 읽어 온다.
 - ACK를 굳이 읽을 이유가 없어서, ACK는 강제로 1로 주고 있다.

[소스분석 - 계속]

- I2C_read(NACK)
 - unsigned char I2C_read(unsigned char b)
 - {
 - unsigned char data, i;
 - sda(1);I2C_delay();I2C_delay();I2C_delay();
 - scl(0);I2C_delay();I2C_delay();
 - for(i=1;i<=8;i++){
 - scl(1);I2C_delay();
 - I2C_DIR=SDIRIN;
 - data=data<<1;
 - data = data | ((I2C_IN>>1)&0x01);
 - I2C_DIR=SDIROUT;
 - scl(0);I2C_delay();
 - }
 - sda(b);I2C_delay();
 - scl(0);I2C_delay();
 - scl(1);I2C_delay();
 - if(b==NACK)
 - sda(1);I2C_delay();
 - scl(0);I2C_delay();
 - sda(1);I2C_delay();
 - return data;
 - }

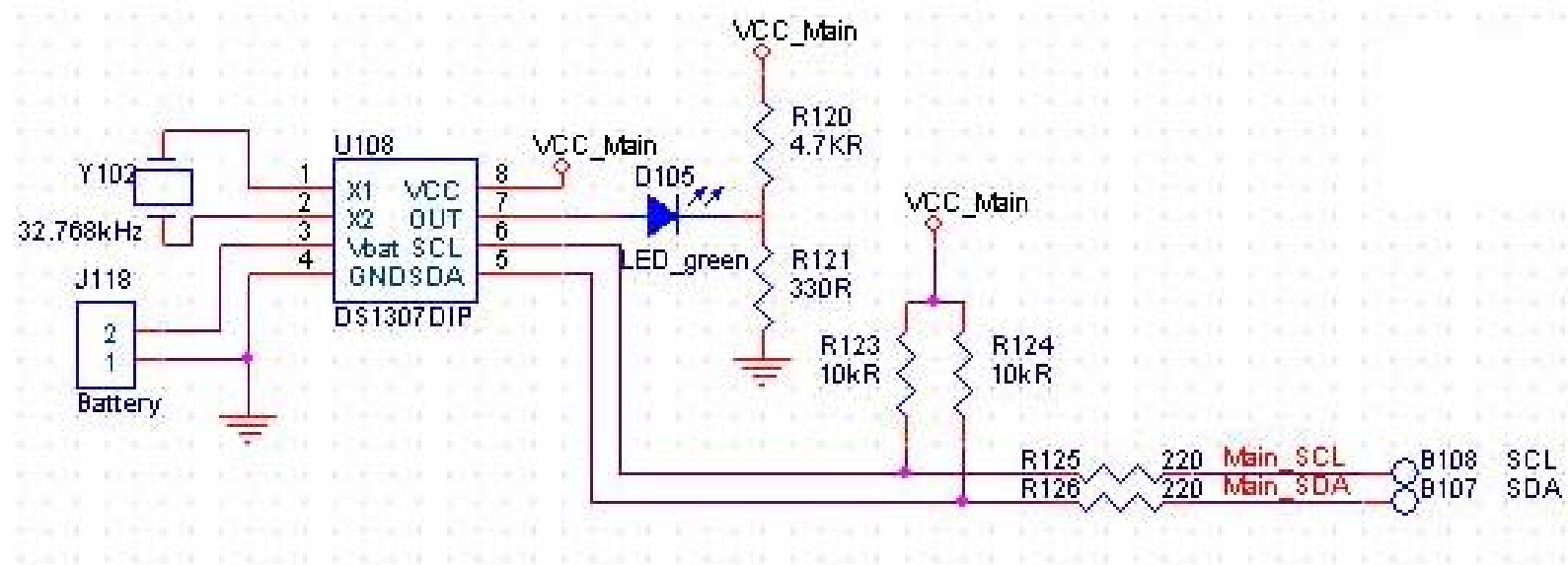
[소스분석 - 계속]

- **I2C_write**
 - 현재의 어드레스에 데이터를 쓴다.
 - ACK에 0을 실어준다.

[소스분석 - 계속]

```
○ void I2C_write(unsigned char data)
○ {
○     unsigned char i;
○     I2C_DIR=SDIROUT;
○     scl(0);I2C_delay();
○     for(i=1;i<=8;i++){
○         sda(data>>7);I2C_delay();
○         scl(1);I2C_delay();
○         data=data<<1;
○         scl(0);I2C_delay();
○     }
○
○     sda(0);I2C_delay();
○     scl(0);I2C_delay();
○     scl(1);I2C_delay();
○     scl(0);I2C_delay();
○     sda(1);I2C_delay();
○ }
```

[DS1307 기본 회로도]



[Main 함수에서 소스 사용법]

```
■ void main(void)
■ {
■     unsigned char i;
■     init_devices();
■     init_ds1307();

■     writebyte(0x00,0x00); // 00번지(초)에 0 값을 쓴다.
■
■     while(1){
■         Display();
■         FourDigit=readbyte(0); // 초값을 읽어온다.
■     }
■ }
```

[어드레스와 그곳 데이터 의미]

CHIP : DS1307

Timekeeper Registers

ADDRESS	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	FUNCTION	RANGE
00h	CH	10 Seconds			Seconds				Seconds	00–59
01h	0	10 Minutes			Minutes				Minutes	00–59
02h	0	12	10 Hour	10 Hour	Hours				Hours	1–12 +AM/PM 00–23
		24	PM/ AM							
03h	0	0	0	0	0	DAY			Day	01–07
04h	0	0	10 Date		Date				Date	01–31
05h	0	0	0	10 Month	Month				Month	01–12
06h	10 Year				Year				Year	00–99
07h	OUT	0	0	SQWE	0	0	RS1	RS0	Control	—
08h–3Fh									RAM 56 x 8	00h–FFh

0 = Always reads back as 0.

[실습하면서 만들 함수]

- **WatchONOFF(int onoff)**
 - Onoff값에 따라 시계를 켜다 켜다 할 수있다.
- **SetSeconds(UC seconds)**
 - 초 설정
 - 비슷한 부류로.....분,시간,일, 요일,월,년 설정
- **Seconds()**
 - 현재의 초 값을 리턴한다.
 - 비슷한 부류로.....분,시간,일, 요일,월,년 읽음

[실습하면서 만들 함수-계속]

- **SetH12()**
 - 12시간제로 변경
 - 유사함수 SetH24() 및 AM/PM 등 설정 함수가 있다.
- **WatchWriteTest();**
 - 시계 영역에 데이터를 0~5까지 쓴다.
- **WatchReadTest();**
 - 시계 영역에서 데이터를 읽어온다.
- **RAMWriteTest();**
 - 시계의 램 영역에 데이터를 쓴다.
- **RAMReadTest();**
 - 시계의 램 영역에서 데이터를 읽어온다.