I2C and RTC
Chapter 18

Sepehr Naimi

www.NicerLand.com

1

I2C History

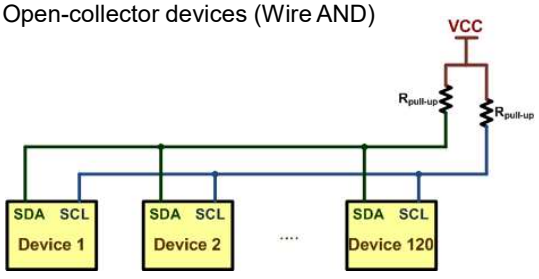
- IIC: Inter-Integrated Circuit
- Philips
- 1982
- Aim: connecting many devices (around 128 devices) to the MCU using two wires

2

2

Connecting devices using I2C

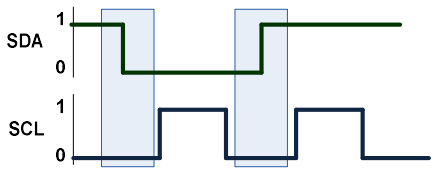
- SDA: Serial Data
- SCL: Serial Clock
- Open-collector devices (Wire AND)



3

Sending bits of data

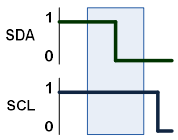
- The SDA values changes when SCL is low.
- The receiver reads SDA on the falling edge of SCL.



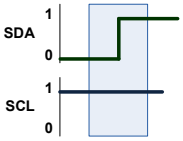
4

Start and Stop conditions

- Start



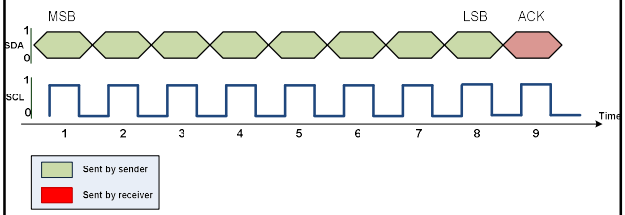
- Stop



5

Packet Format

- Each packet is 9 bits long.
- First 8 bits are put on SDA by the transmitter
- The 9th bit is an acknowledge by the receiver
  - NACK (leave high) or ACK (pull down)



6

Master vs. Slave

- Master
  - Begins the communication
  - Chooses the slave
  - Makes clock
  - Sends or receives data
- Slave
  - Responds to the master
  - Each slave has a unique 7-bit address

7

Master vs. Slave (Cont.)

- There might be more than 1 master on an I2C bus
- Each device can be both Master and Slave

8

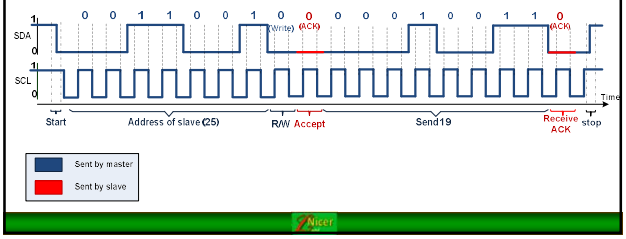
Steps of a communication

1. Start
2. Address
3. Send or Receive (Write or read)
4. Acknowledge
5. Send/receive a byte of data
6. Acknowledge
7. Stop

9

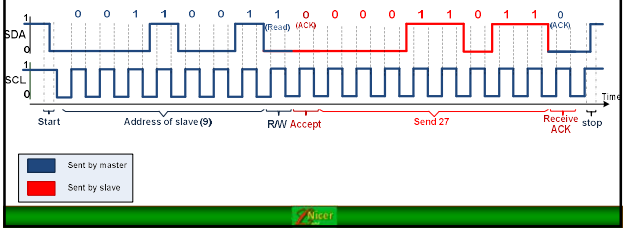
Sending a byte

- Sending 19 to device 25.



10

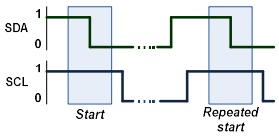
Receiving a byte from Device 9



11

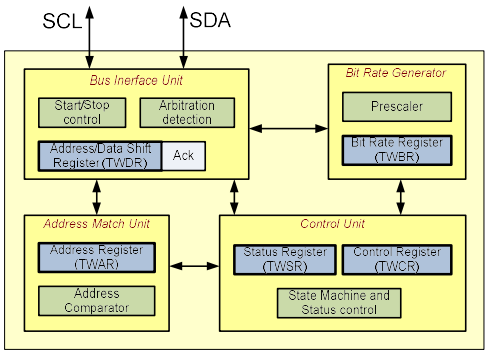
Repeated start

- A new Start condition before the Stop condition



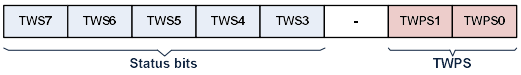
12

I2C (TWI) Unit in AVR



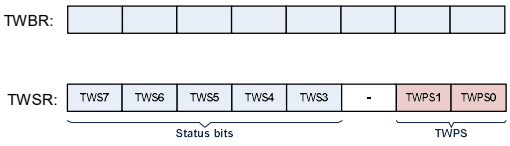
13

TWSR (TWI Status Register)



14

TWBR (TWI Bit Rate) Register



$$frequency = \frac{XTAL}{16 + 2 \times TWBR \times 4^{TWPS}}$$

15

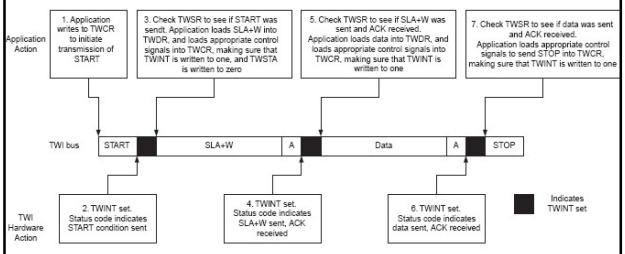
TWCR

TWCR:	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE

- TWINT: TWI Interrupt flag
- TWEA: TWI Enable Acknowledge bit
  - 1:ACK, 0:NACK
- TWSTA: TWI Start condition bit
- TWSTO: TWI Stop condition bit
- TWWC: TWI Write Collision flag
- TWEN: TWI Enable bit
- TWIE: TWI Interrupt Enable

16

TWI Control Register



17

TWAR (TWI Address Register)

TWAR:	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE

- TWA6-0 (TWI slave Address)
- TWGCE (TWI General Call Recognition Enable bit)
  - 1: Answer to general call

18

TWDR (TWI Data Register)

19

TWI, Master Mode programming

- Initializing
  - Set the TWI module clock frequency by setting the values of the TWBR register and the TWPS bits in the TWSR register.
  - Set the TWEN bit in TWCR to one to enable the TWI module
- Transmit START condition
  - Set TWEN, TWSTA, and TWINT bits of TWCR to one.

20

TWI, Master Mode programming

- Send Data
  - Copy the data byte to the TWDR
  - Set the TWEN and TWINT bits of the TWCR to one to start sending the byte.
  - Poll TWINT flag in TWCR register to see whether the byte transmitted completely
- Receive Data
  - Set TWEN and TWINT bits of TWCR to one to start receiving a byte.
  - Poll TWINT flag in TWCR to see whether a byte has been received completely
  - read the received byte from the TWDR

21

TWI, Master Mode programming

- Transmit STOP condition
  - Set TWEN, TWSTO, and TWINT bits of TWCR to one
    - Note: we cannot poll the TWINT flag after transmitting the STOP condition

22

Writing and reading a byte in master mode

```
#include <avr/io.h>

void i2c_write(unsigned char data)
{
  TWDR = data ;
  TWCR = (1<< TWINT)|(1<<TWEN);
  while ((TWCR & (1 <<TWINT)) == 0);
}

//************************************************
unsigned char i2c_read(unsigned char isLast)
{
  if (isLast == 0)  //send ACK
    TWCR = (1<< TWINT)|(1<<TWEN)|(1<<TWEA); //send ACK
  else
    TWCR = (1<< TWINT)|(1<<TWEN); //send NACK
  while ((TWCR & (1 <<TWINT)) == 0);
  return TWDR;
}

void i2c_start(void)
```

23

TWI, Slave Mode programming

- Initializing
  - Set the slave address by setting the values for the TWAR register.
    - 7 bits for address
    - 8th bit is TWGCE (1 = answer general calls)
  - Set the TWEN bit in TWCR to one to enable the TWI module
  - Set the TWEN, TWINT, and TWEA bits of TWCR to one to enable the TWI and acknowledge generation
- Listening
  - poll the TWINT flag to see when the slave is addressed by a master device or use its interrupt

24

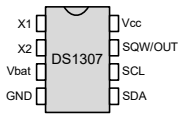
TWI, Slave Mode programming

- Send Data
  - Copy the data byte to the TWDR
  - Set the TWEN, TWEA, and TWINT bits of the TWCR register to one to start sending the byte.
  - Poll TWINT flag in TWCR register to see whether the byte transmitted completely
- Receive Data
  - Set TWEN and TWINT bits of TWCR to one to start receiving a byte.
  - Poll TWINT flag in TWCR to see whether a byte has been received completely
  - read the received byte from the TWDR

25

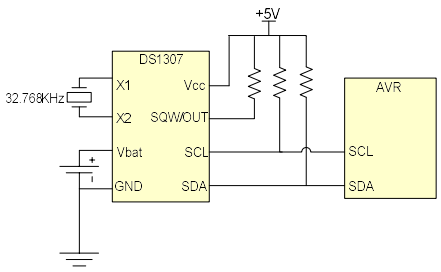
DS1307

- RTC (Real-time clock)
- Keeps time and date



26

Connecting DS1307 to the AVR



27

DS1307 address map

- 64 bytes of RAM
- BCD format is used
- Bit7 of address location 0 (CH) should be zero to enable the oscillator
- Address location 07 is the control register

ADDRESS	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	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

28

Register Pointer in DS1307

Register pointer:

- In DS1307 there is a register pointer that specifies the byte that will be accessed in the next read or write command.
- After each read or write operation, the content of the register pointer is automatically incremented. It is useful in multi-byte read or write.

29

Write to DS1307

- Transmit START condition
- Transmit the address of DS1307 (1001101) followed by 0 to indicate a write operation
- Transmit the address of location you want to access (it set the value of Register Pointer)
- Transmit one or more bytes of date
- Transmit STOP condition

30

Read from DS1307

- Transmit START condition
- Transmit the address of DS1307 (1001101) followed by 1 to indicate a read operation
- Receive one or more bytes of date
- Transmit STOP condition
  - Note: the register pointer indicates which address will be read (you should set it using a write operation)

31

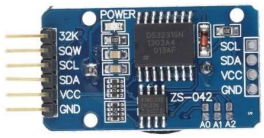
Set Time and Get Time

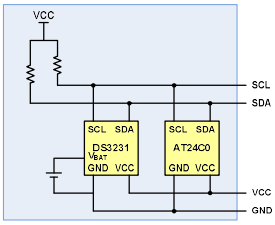
```
void rtc_setTime(unsigned char h,unsigned char m,unsigned char s)
{
        i2c_start();                    //transmit START condition
        i2c_write(0xD0);                //address DS1307 for write
        i2c_write(0);                   //set register pointer to 0
        i2c_write(s);                   //set seconds
        i2c_write(m);                   //set minutes
        i2c_write(h);                   //set hour
        i2c_stop();                     //transmit STOP condition
}

void rtc_getTime(unsigned char *h,unsigned char *m,unsigned char *s)
{
        i2c_start();                    //transmit START condition
        i2c_write(0xD0);                //address DS1307 for write
        i2c_write(0);                   //set register pointer to 0
        i2c_stop();                     //transmit STOP condition

        i2c_start();                    //transmit START condition
        i2c_write(0xD1);                //address DS1307 for read
```

32

DS3231 module

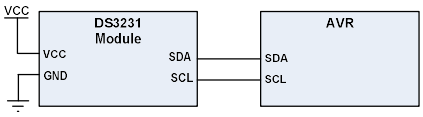
- Similar to DS1307





33

Connecting DS3231 to AVR



34