Inter - integrated circuit (I 2 c)

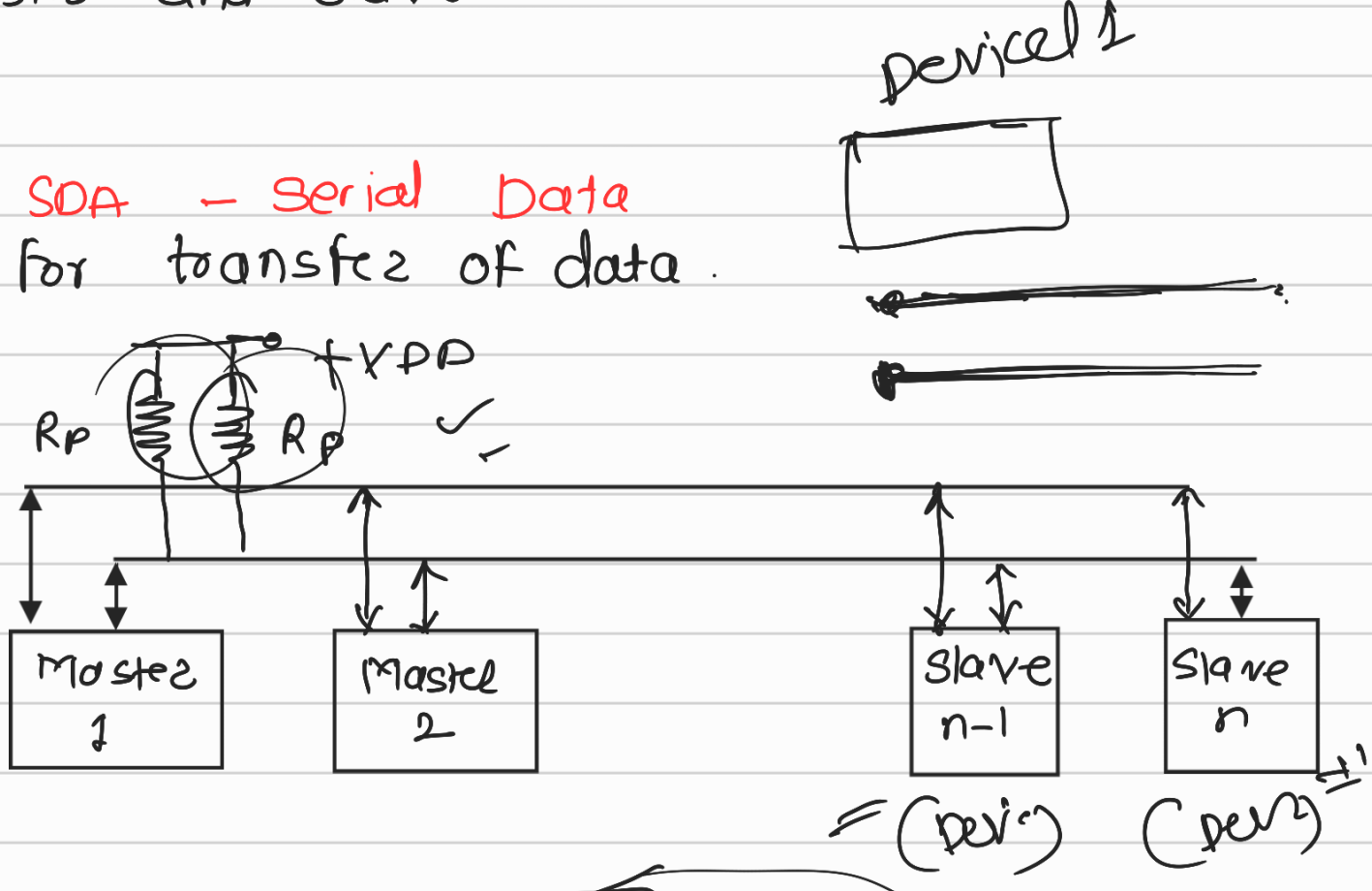
- Names:-

1) Inter integrated circuits
2) I2C
3) I^2C
4) Two wire interface
5) TWI

- I2c has two wire lines :

1) SCL – serial clock
→ for synchronizing data transfer between master and slave

2) SDA – Serial Data
→ for transfer of data.



→ We have to connect resistors externally or

we shall enable them internally.

⊙ Operation modes:

i) Master transmitter
ii) Master Receiver
iii) Slave transmitter
iv) Slave Receiver

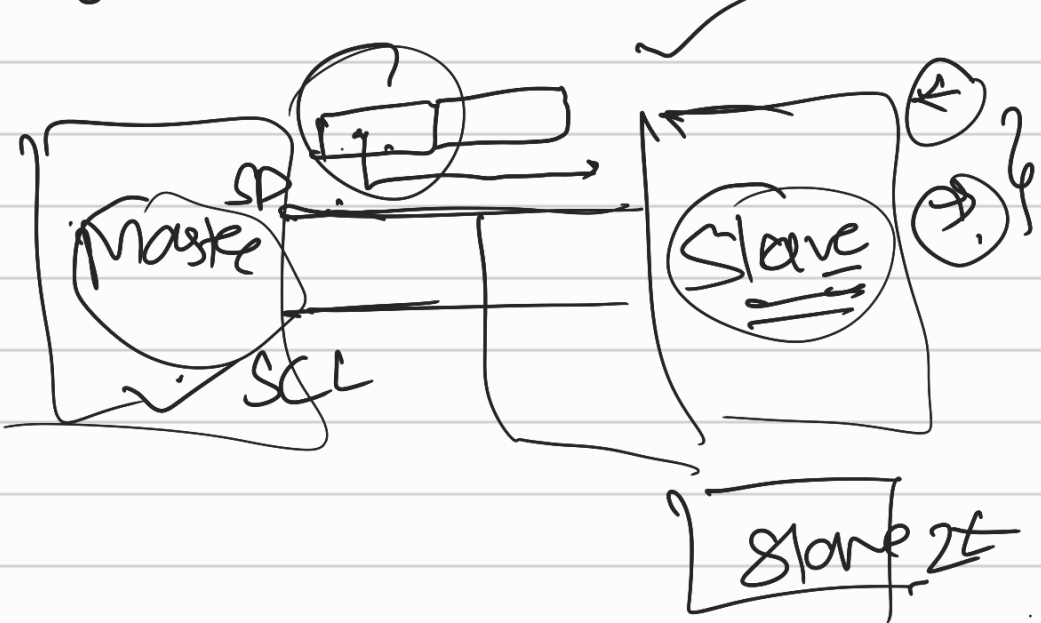
So

⊙ key Factors:

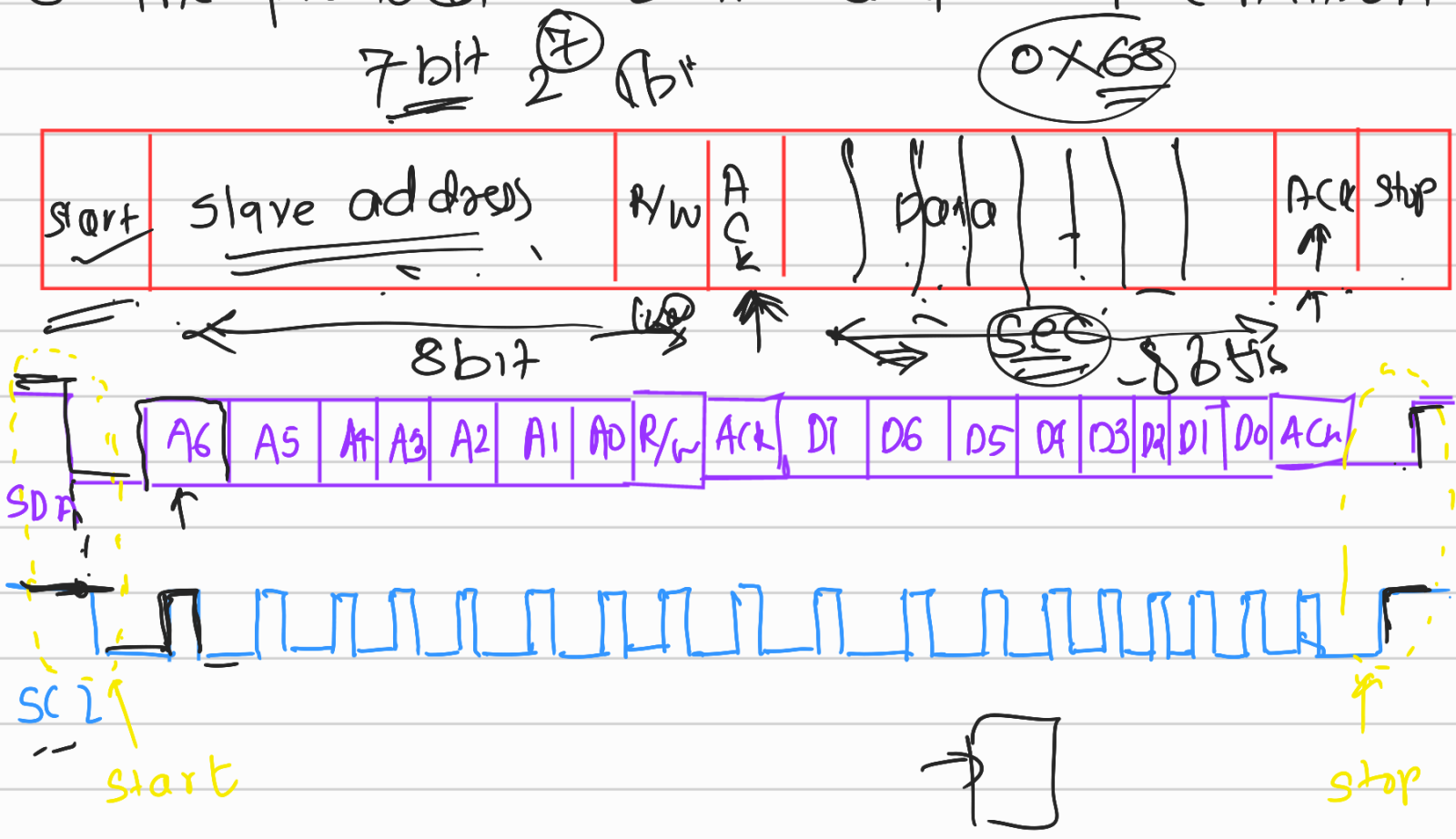
→ Transaction are initiated and completed by master.

→ All message have address frame and data Frame.

→ Data is placed on SDA line after SCL goes Low and it is sampled after the line goes high



⊙ The protocol - START and STOP condition



Point 1:
All transaction begins with START and terminated by STOP.

Point 2:
High to Low on SDA line while SCL = High
∴ START

Point 3:
Low to High on SDA + SCL High
= STOP

Point 4:
START and STOP condition always generated by master)

Points;

A bus is considered as free again ~ retain time after STOP condition.

Point6:

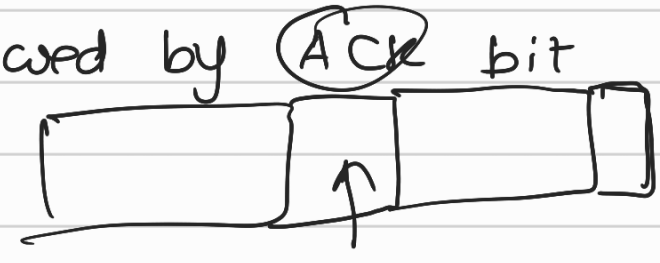
The bus stays busy if repeated START is generated instead of STOP condition.

Point7:

Any information transmitted on SDA line must be eight bit long. The No of bytes that can be transmitted per transfer is unrestricted.

Point8:

Each byte must be followed by ACK bit (Acknowledge bit)



Point9:

Data is transmitted with MSB first.

Point 10:

The Address frame is first in any communication.

For 7-bit address, the address is sent on MSB first, followed by R/W

⊙ I2C clock speed: (bus speed)

Modes:

a) Standard mode : 100KHz max
b) Fast Mode : 400KHz max
c) Fast-mode Plus : 1MHz
d) High-speed Mode : 3.4 MHz

⊙ I2C Duty cycle:

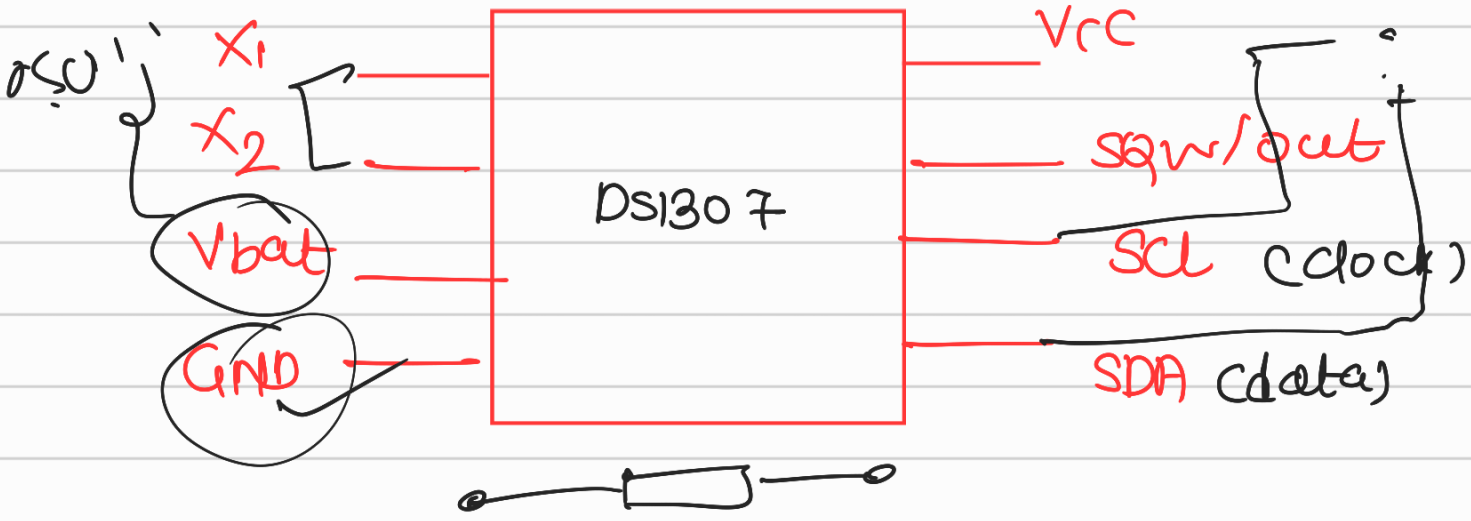
→ specifies ratio between t_{LOW} and t_{high} of I2C SCL line

→ Possible values:

I2C_Dutycycle_2 = 2:1
I2C_Dutycycle_16_9 = 16:9

→ by choosing the appropriate duty cycle we can prescale the peripheral clock to achieve desired I2C speed.

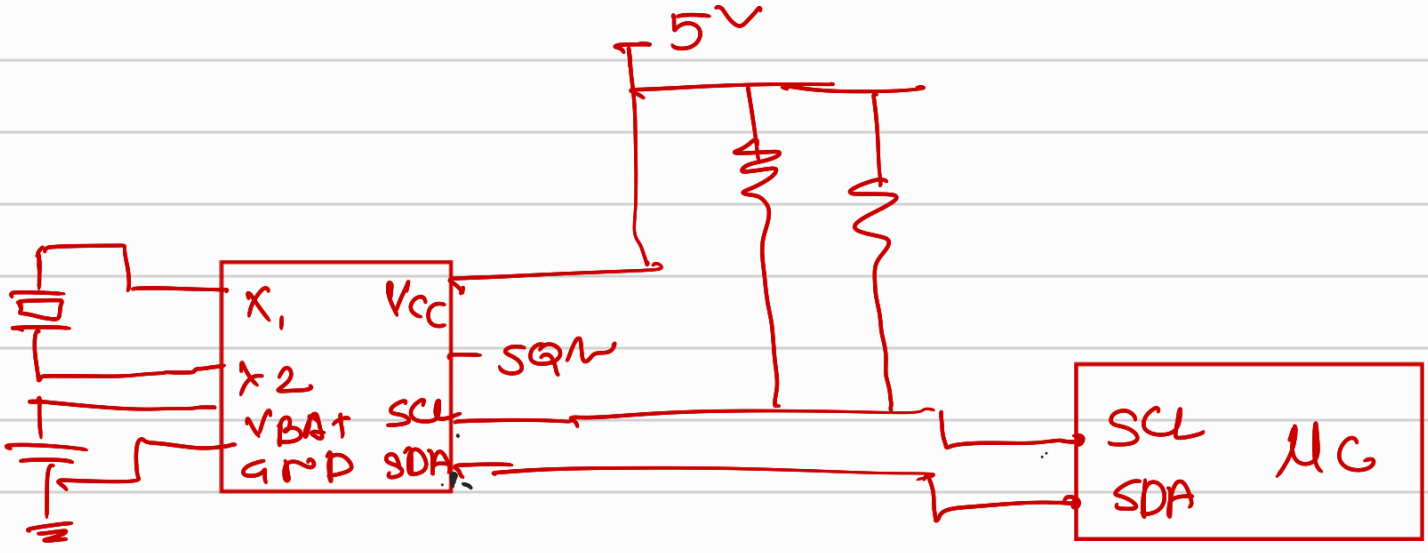
⊙ RTC (DS 1307) (I2C bus)



X1, X2 } External Crystal oscillator — 32.768 KHz Quartz crystal ✓

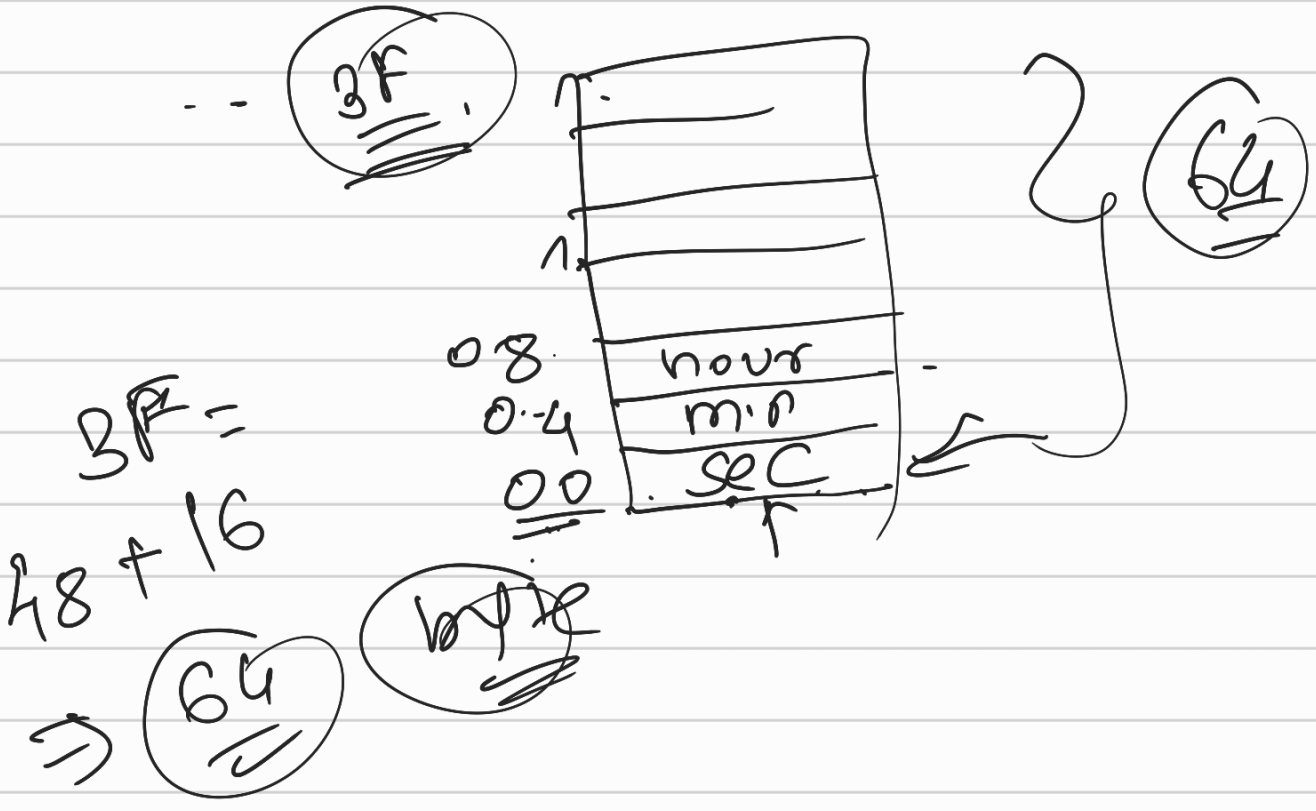
VBAT → 3V battery — external battery ✓
GND — Ground

SQW out } Provide frequency IF enable
connect to pull up resistor to generate frequency ✓

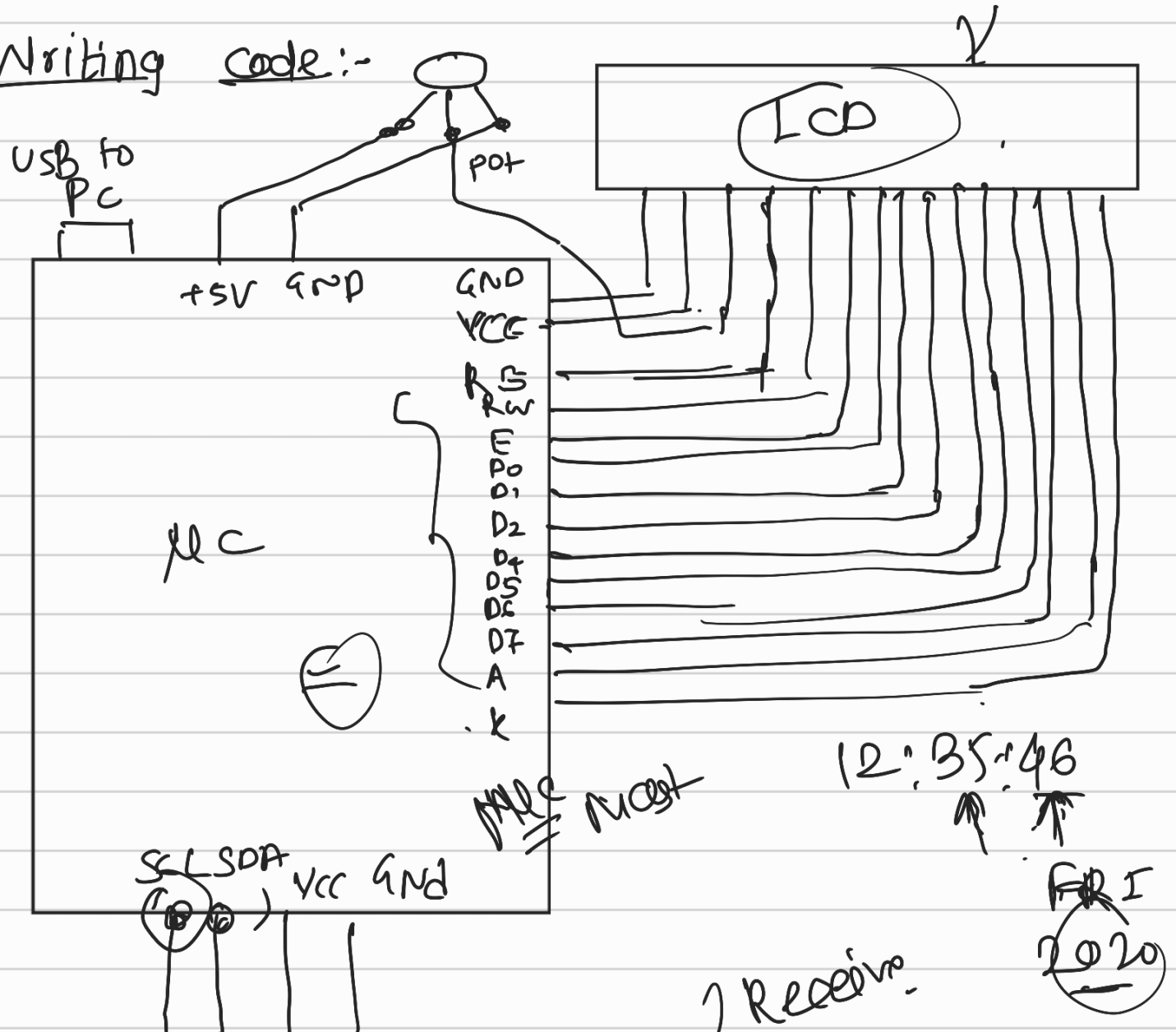


Specification: (datasheet) DS1307

- 64 bytes of RAM space
- Address 00-3FH
- 00-06 for RTC values of time and date
- Control register 07
- Data storage 07H - 3FH



Writing code :-



Register ⇒ low level ⇒ bare-metal

```
#include <stdio.h>
#include "tm4c123gh6pm.h"

//SCL   } datasheet
//SDA

#define Slave_address 0x68

void I2C_init(void)
{
```

1) Enable clock to peripheral

2) Enable clock to I2C1 module

3) // Pins have alternate function
   a) AFSEL = enable alternate function
   b) GPIOPCTL = set alternate function
   c) disable AMSEL ⇒ disable

4) Set up pins as open drain configuration
   a) open drain
   b) Pull up

SDA

5) Now Initialization
   a) disable I2C ⇒ I2C control register

   b) put clock to I2C control register
      i.e 16 MHz will go in CR2
      = 0x0010;

c) get Mode (speed)

```
I2C->CCR = 80 ; // standard mode 100KHz
```

d) Rise time

```
I2C_TRISE = 17; // max rise time
```

e) Enable Re
CR1

}

// Read a byte of data from memory location of slave device.

```
int i2c_readbyte ( char slave_addr ,
                   char maddr, char
                   *data)
{
    volatile int tmp;
    while ( I2C1 -> SR2 & 2)   // make sure bus not busy
      {
      }
    // start condition
```

```
I2C1 -> CR1 |= 0x100;
while (!(I2C -> SR1 & 1)) { };
                    // start flag to set

I2C1 -> DR = slave_addr << 1;
while (! (I2C1 -> SR1 & 2));
tmp = I2C1 -> SR2;
while (! (I2C1 -> SR1 & 0x80)) { };
I2C1 -> DR = maddr;   // (wait until data register is empty)
while (!(I2C1 -> SR1 & 0x80)) { };
I2C1 -> CR1 |= 0x100;   // Restart
while (! (I2C1 -> SR1 & 2)) { };
I2C1 -> DR = slave_addr << 1 | 1;
                    (read)
while (! (I2C1 -> SR1 & 2));
I2C1 -> CR1 &= ~0x400;
                    // disable the acknowledge
tmp = I2C1 -> SR2;
// Stop condition
```

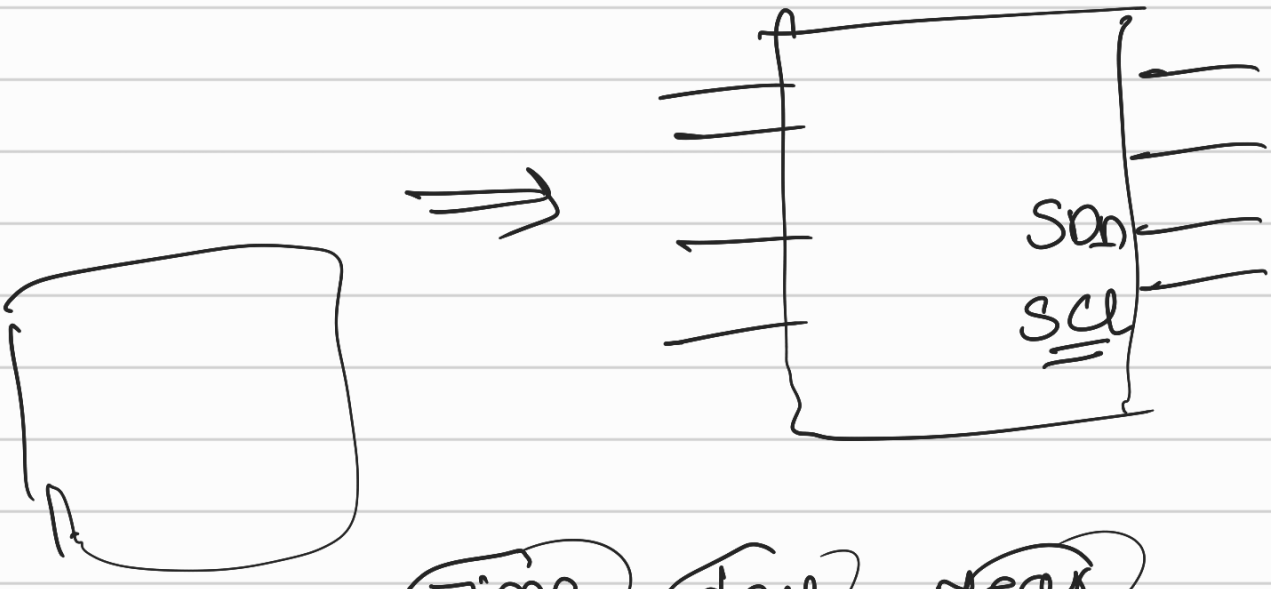
```
I2C1 -> CR1 |= 0x200;
// wait stop flag to set
while( !(I2C1->SR1 &0x40)); };
*data++ = I2C1->DR;
return 0;
}

void delayMS(int n) {
    for( i=0; i<delayMS; i++);
    {
    }

Int main()
{   i2c_init();
    clock;
  [ Mode select for anyother port;
    output pin
  ]
  char data;
```

```
while (1)
{
    i2c_readbyte ( slave_address , 0,
                   &data);
    if ( data &1)
        GPIOF -> blinx led;
    else
        no led
}
```

RTC mod

DS1307

Clock

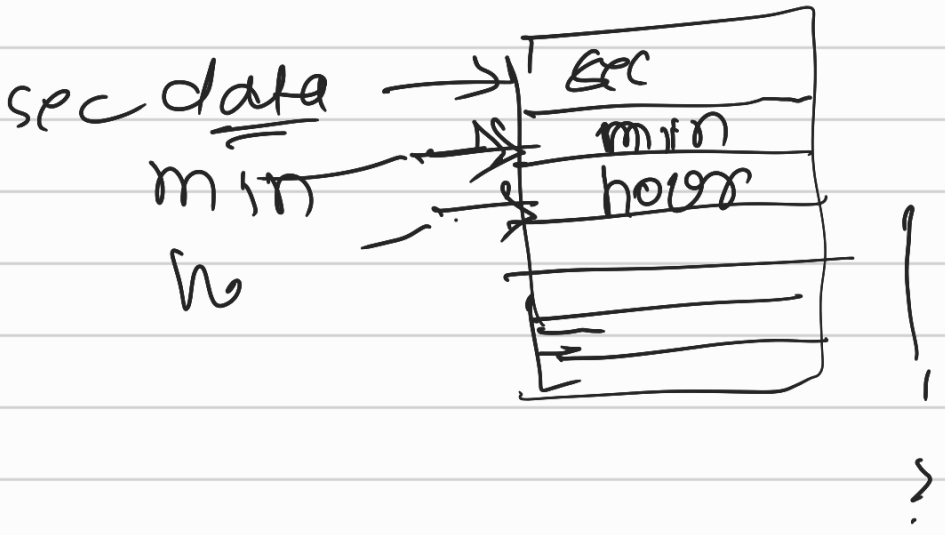


Time, day, year

Alarm ✓

I2C ⇐ DS1307 ⇒

sec



⊙ 1 ⇒ present
2 ⇒ present
3 ⇒ present
4 ⇒ present
5 ⇒ present
6 ⇒ —
7 ⇒ present
8 ⇒ —
9 ⇒ present
10 ⇒ present
11 ⇒ present
12 ⇒ present
13 ⇒ present
14 ⇒ present
15 ⇒ present
16 ⇒ present
17 ⇒ present
18 ⇒ present
19 ⇒ present

20 ⇒ Reset

21 ⇒