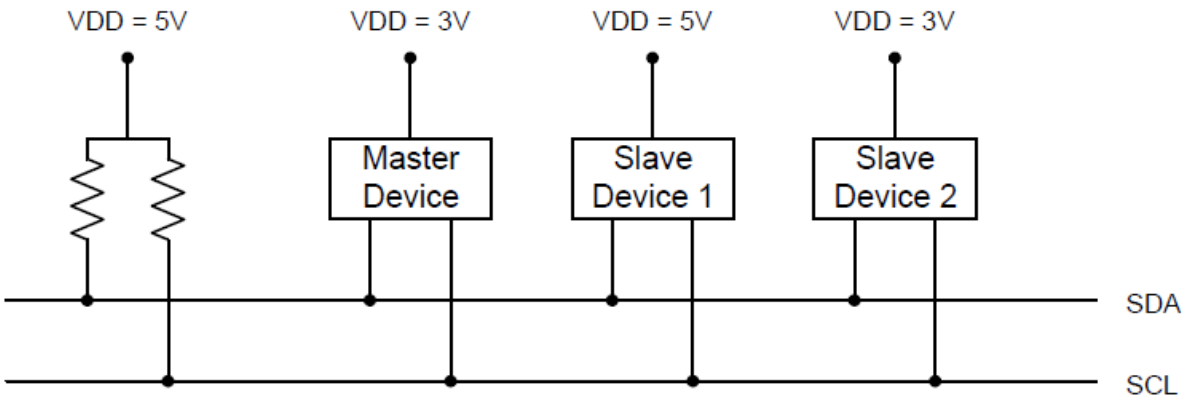
I^2C (Inter IC)-



Important features of I^2C configuration:

1. Only two lines are required SDA (serial data line) and SCL (serial clock line) for communication.
2. Each device is identified by its unique 7-bit address.
3. Any device with capability of generating clock, START and STOP conditions can act as MASTER.
4. MASTER controls the SCL line and initiates the data transfer.
5. The 8-bit bidirectional serial data transfer occurs at 100Kbits/s in standard mode and 400 Kbits/s in fast mode. (DS1307 only supports standard mode.)
6. The no. of devices that can be connected is limited by maximum bus capacitance of 400pF.
7. Each device has to have open-drain/open-collector output configuration. The lines are pulled up as shown in figure.

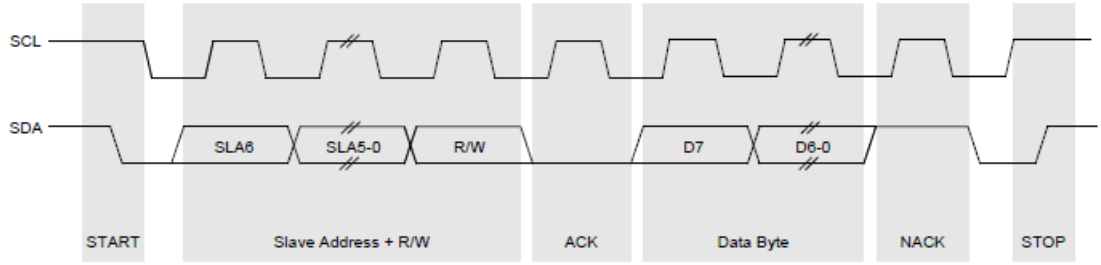
I^2C bus states

START : HIGH to LOW transition on SDA line when SCL is HIGH defines the START condition.

STOP : LOW TO HIGH transition on SDA line when SCL is HIGH defines the STOP condition.

ACK : The receiver has to hold the SDA line low during 9^{th} cycle after 8 bit transfer from transmitter to receiver.

NACK : The receiver has to release the SDA line during 9^{th} cycle after 8 bit transfer from transmitter to receiver.



There can be 2 possible way of data transfer.

1. MASTER TRANSMITTER - SLAVE RECEIVER
2. MASTER RECIVER - SLAVE TRANSMITTER

C8051F380 has two SMB (system management bus) modules SMB0 and SMB1. They are referred as SMBn where n=0, 1. SMBUSn is **I^2C compatible.**

SMBUS interrupt

The SMBUS interrupt is generated when

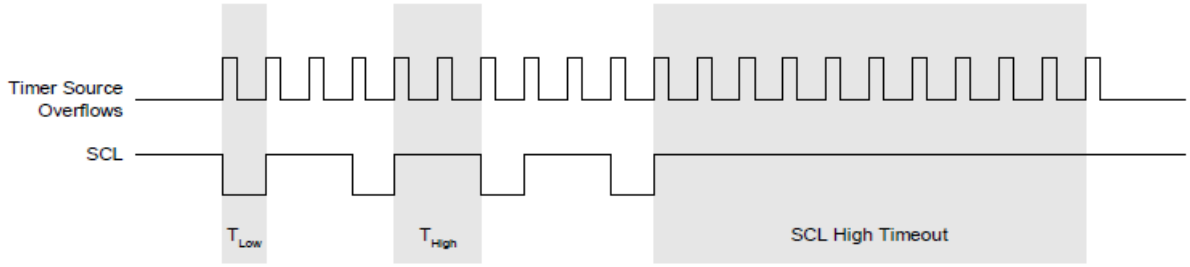
1. START condition generated on SDA line.
2. ACK has been received.
3. Data byte has been received and ACKRQ is generated. (EHACK*=0).
4. Data byte has been received and acknowledgement as per ACK (SMBnCN.1) is sent (EHACK*=1).

EHACK* =>Enable hardware acknowledgment.

When the interrupt is generated SI flag is set to 1. The clock is disabled for that time. SI has to be cleared in software before leaving the ISR routine.

SMBUS clock selection

The SMBUS uses timer overflows to generate SCL. Which timer overflow to be used is decided by setting appropriate values in SMBnCF.



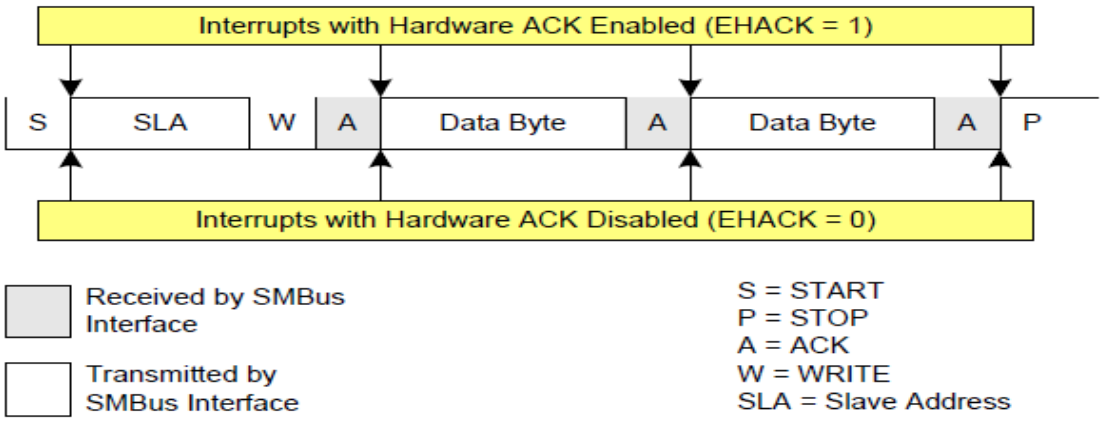
The bitrate of communication can be approximated as

$$BitRate = \frac{f_{ClockSourceOverflow}}{3}$$

SMBUS operation

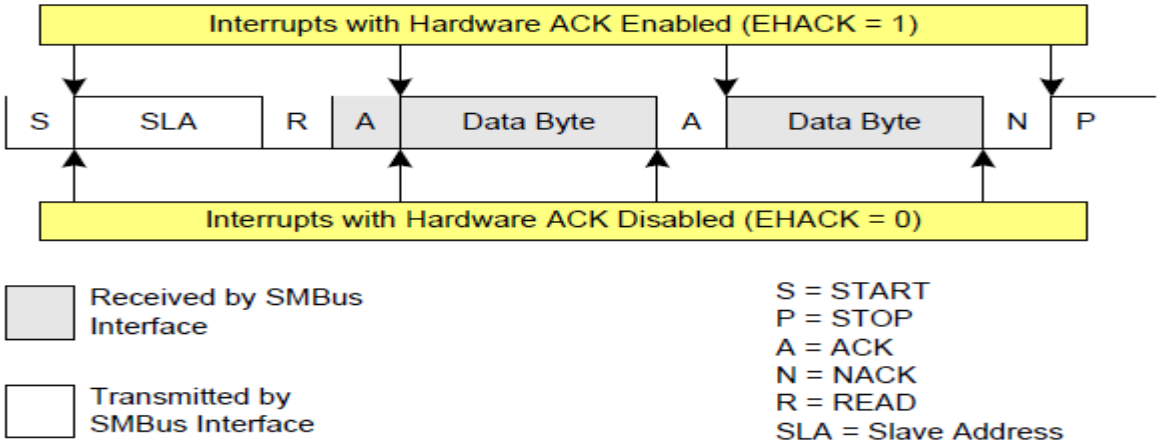
WRITE:

Just after the START condition is generated by MASTER, in next byte it transmits the SLAVE address + R/W bit. For this operation R/W bit will be 0. After address byte + R/W is acknowledged by slave, in the next byte MASTER starts sending the data in 8-bit format. After a byte is transferred transmitter will wait for the ACK from receiver. Note that MASTER is always a transmitter in this function.



READ:

Just after the START condition is generated by MASTER, in next byte it transmits the SLAVE address + R/W bit. For this operation R/W bit will be 1. After address byte is acknowledged by SLAVE, in the next byte MASTER waits for the data from receiver in 8-bit format. After byte is transferred transmitter will wait for the ACK from receiver. Note that MASTER is transmitter during address byte and receiver during data bytes.



Steps to configure I^2C for write:
(If using SMB0 module)

1. Initialize the port registers.
2. Route the I^2C module to ports by configuring XBR0 register.
3. Enable crossbar in XBR1.
4. Set the timer registers in appropriate mode.
5. Enable the global and SMB0 interrupt. (IE and EIE1).
6. For automatic HW acknowledgement enable the EHACK0 bit. (SMB0ADM)
7. *If automatic HW acknowledgement enable is not used then received byte has to be acknowledged by SW by writing thee ACK value.(SMB0CN).
8. Enable SMB0 and select the timer clock source. (SMB0CF)
9. Generate start condition.(STA0)
10. Handle the interrupt generated. (To know what caused the interrupt **monitor the values in SMB0CN.)

*It is better if we use automatic HW acknowledgement. It eases the task.
** Look at the table 22.5 in Silicon lab datasheet on page no. 226. It eases the task quite a lot.
The sample code, (Here some values are intentionally left 'XX'; you will have to figure it out.)

```
void main (void)  {

        P0MDIN = 0xFF;                  //DIGITAL MODE (After reset they become digital, so not really required)

        P2MDIN = 0xFF;                  //DIGITAL MODE (After reset they become digital, so not really required)
```

```
        P0MDOUT = 0x3F;          // Open drain for P0.7=SCL, P0.6=SDA and push-pull for others for connecting it to LCD
      P2MDOUT = 0xFF;            // Push-pull for connecting it to LCD
        P0SKIP = 0x3F;           // Skip pins so SDA=P0.6 & SCL=P0.7
        XBR0 = 0xXX;             // SMB0 selected
        XBR1 = 0xXX;             // enable crossbar and disable weak pull-ups
        CKCON = 0xXX;            // Use system clock for timer 0
        TH0 = 0xXX;
        TL0 = 0xXX;
        TMOD = 0xXX;             // Timer 0 mode XX
        TR0 = 1;
        EA = 1;                  //Global interrupt
        EIE1 = 0xXX;             // Enable SMB0 interrupt
        EIP1 = 0x01;             // Make SMB0 high priority (May not require)
      SMB0ADM = 0xXX;        // Enable automatic H/W
      SMB0CF = 0xXX;         // Enable SMB0 and select timer 0 as clk source
      P0=0xFF;
      STA0 = 1;            //START condition will get generated.
while(1){
}
}
```

DS 1307

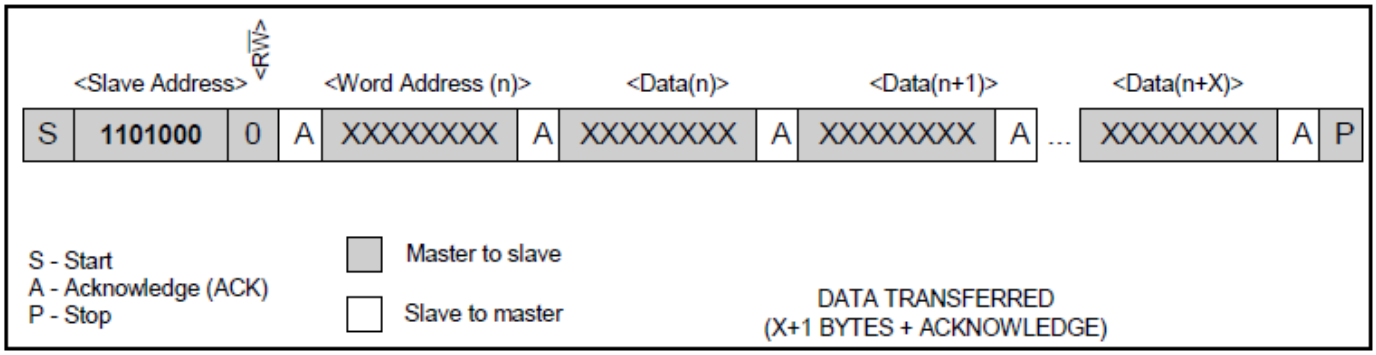
It's a real time clock. The slave address of DS1307 is 1011000. So for WRITE after START condition 1011000 + 0 = 0xD0 has to be transmitted. And for read 0xD1 has to be transmitted. **DS1307 can only act as slave. DS1307 has the memory map as follows.

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

When DS1307 is powered on the data and time registers are reset to 01/01/00 01 00:00:00 (MM/DD/YY DOW HH:MM:SS). The CH bit is seconds register is set to 1. CH=1 disables the clock. ***So, CH has to be cleared before beginning any operation using DS1307***. The data is

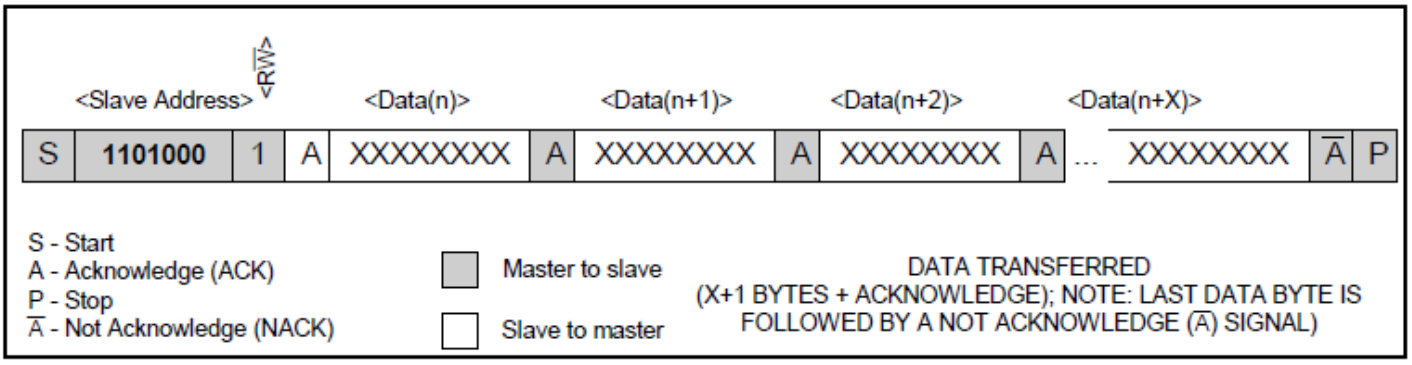
stored in BCD format. When bit 5 of hour register is set to 1, 12 hour format is selected. Otherwise 24 hour format is selected. In 12 hour format bit 6 defines AM/PM. Logic high indicates PM. When a WRITE operation is performed at particular address the pointer is set to that location. After which if READ operation is performed then data at that particular location is transmitted. Once a read/write operation is performed address pointer automatically gets *incremented. So, in next cycle value at next location is transmitted. It becomes clearer in the following figures.

The WRITE cycle is,



Here ‘n‘denotes the location at which address pointer is pointing. Data (n) will WRITE the data at location ‘n’.

The READ cycle is,



Here ‘n‘denotes the location at which address pointer is pointing. Data (n) will READ the data at location ‘n’.