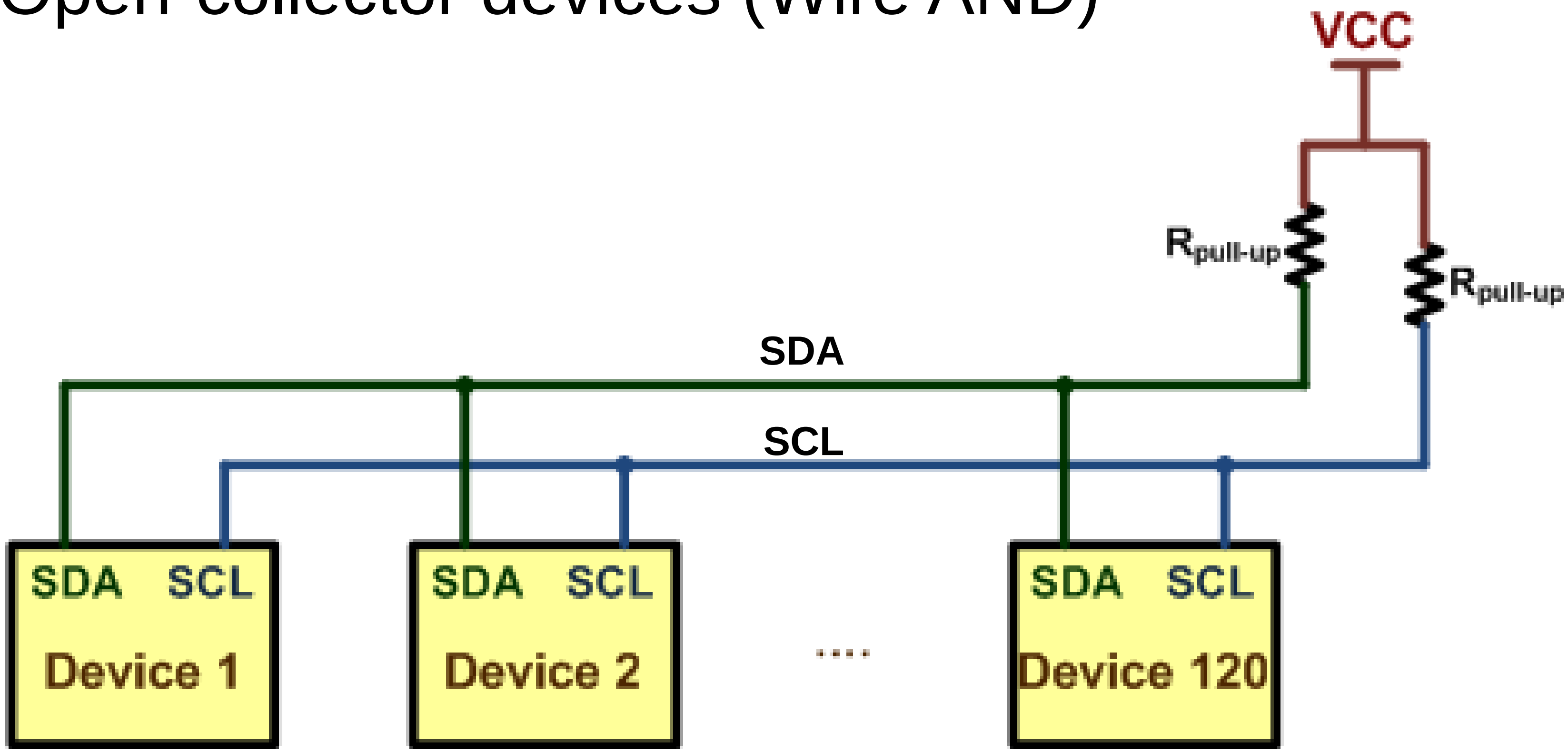
I2C

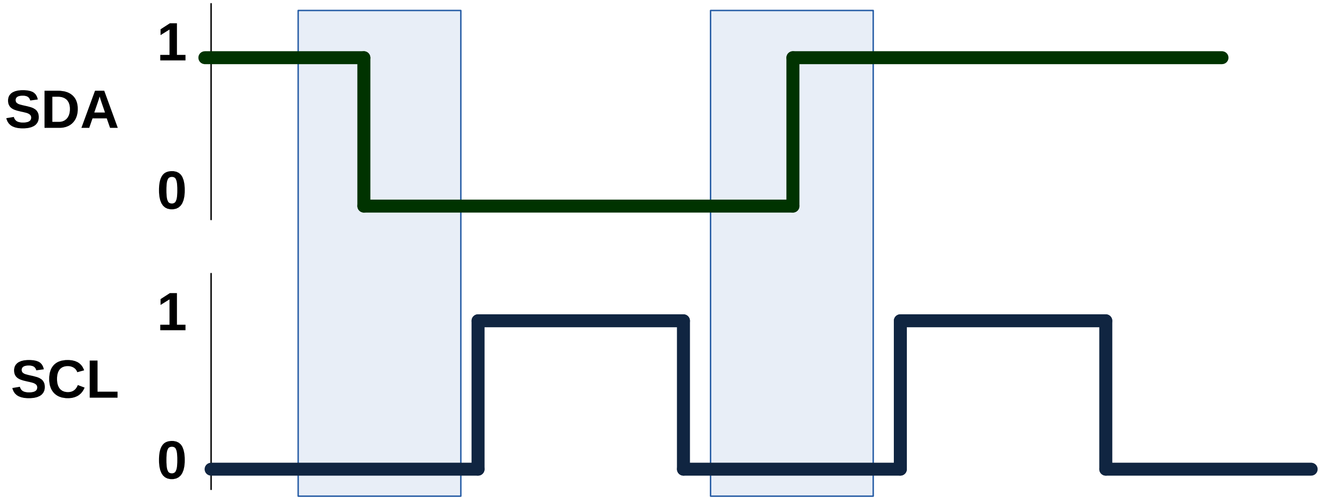
Connecting devices using I2C

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



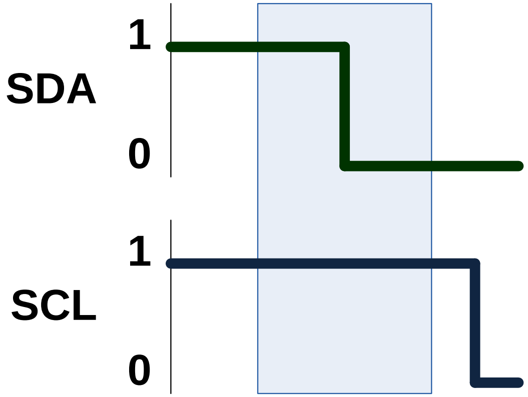
Sending bits of data

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

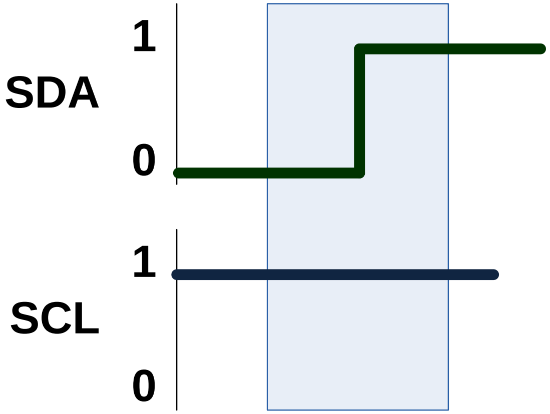


Start and Stop conditions

- Start

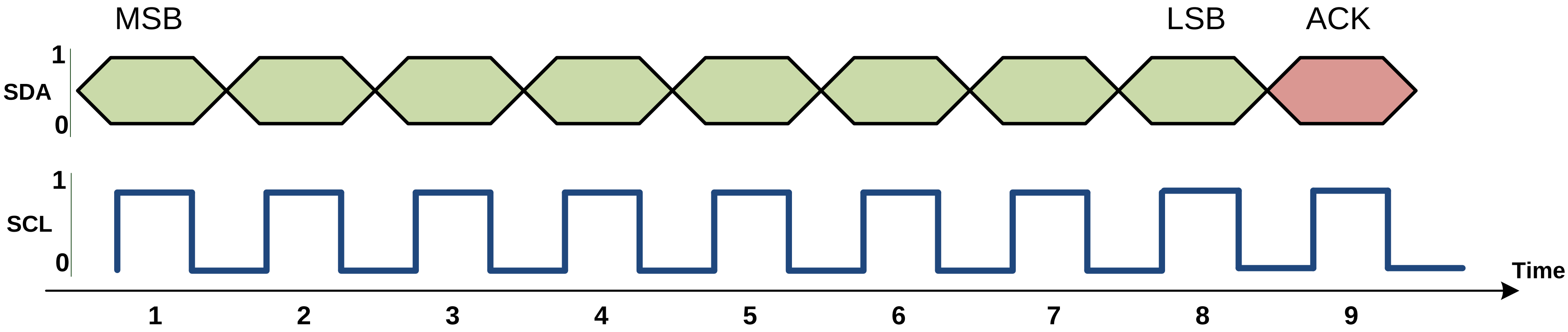


- Stop



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)



🟩	Sent by sender
🟥	Sent by receiver

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

Master vs. Slave (Cont.)

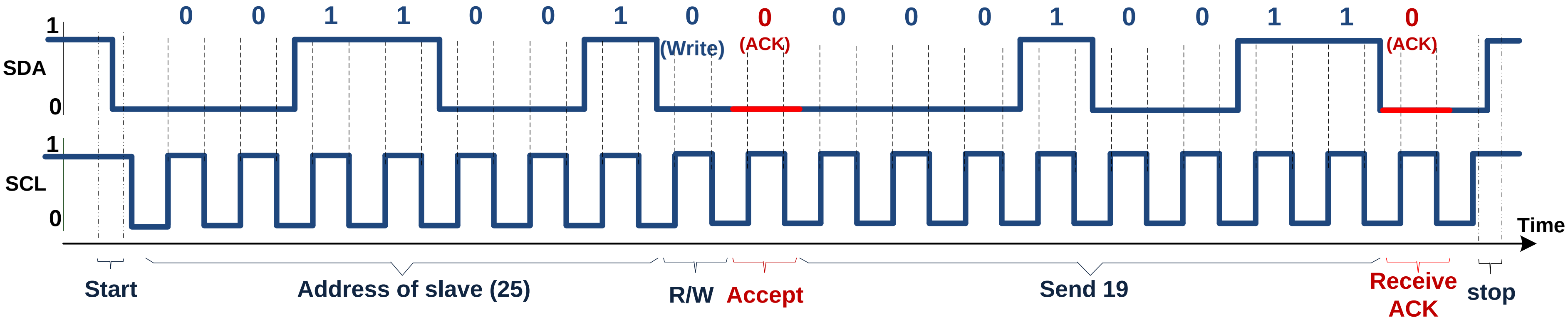
- There might be more than 1 master on an I2C bus
- Each device can be both Master and Slave

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

Sending a byte

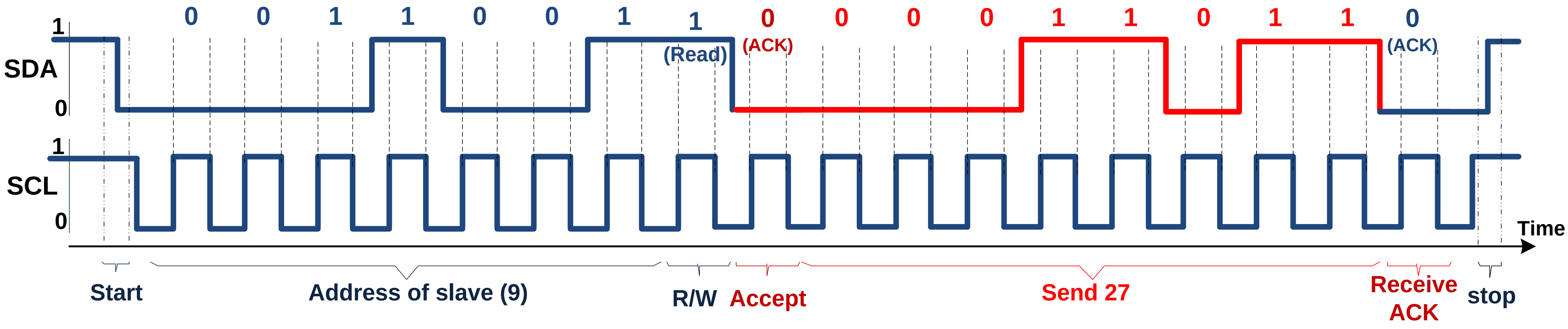
- Sending 19 to device 25.



🟦 Sent by master

🟥 Sent by slave

Receiving a byte from Device 9



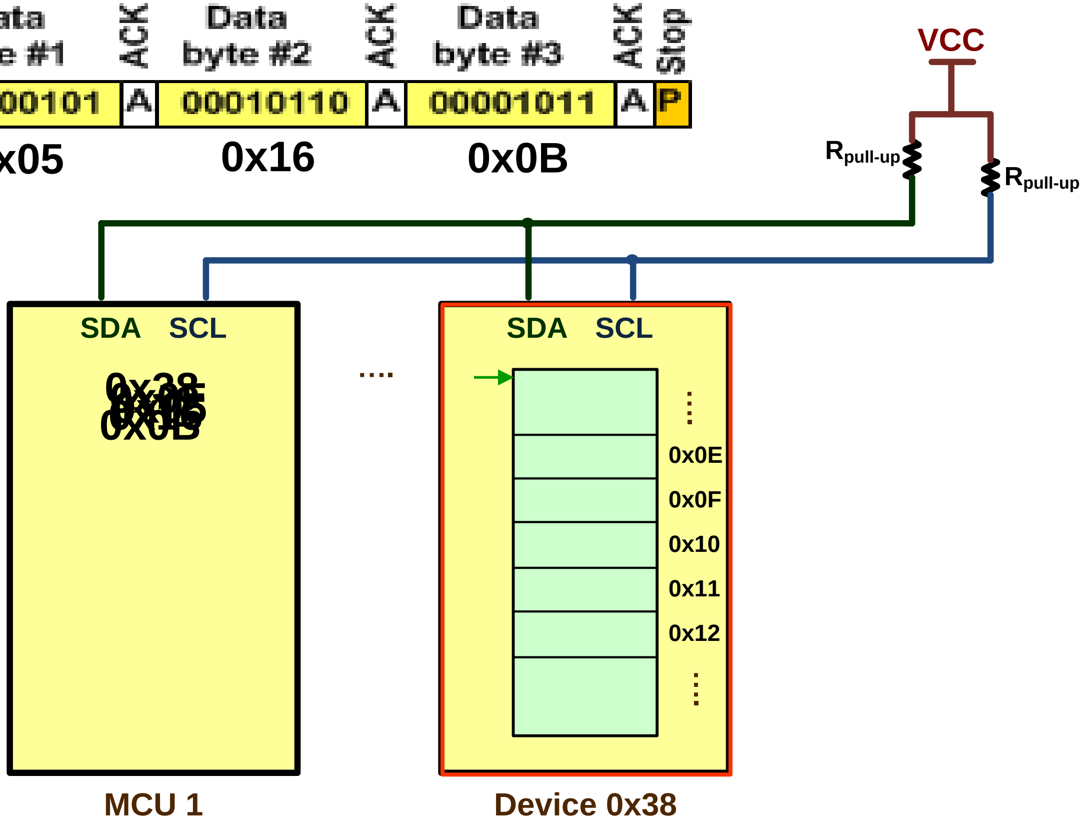
🟦 Sent by master

🟥 Sent by slave

Multi-byte Burst Write

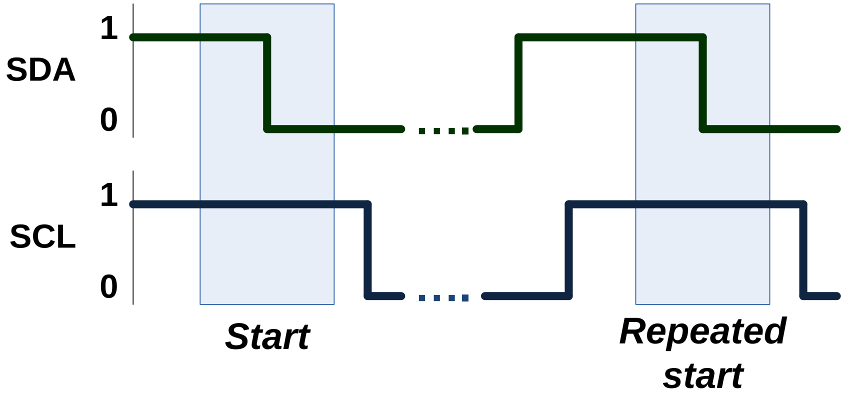
- Master can send multiple bytes of data to slaves.
  - E.g. To store data in memories with I2C interface, first the address and then data is sent.

Start	Slave address	Write	ACK	First location address	ACK	Data byte #1	ACK	Data byte #2	ACK	Data byte #3	ACK	Stop
S	1111000	0	A	00001111	A	00000101	A	00010110	A	00001011	A	P
	0x38			0x0F		0x05		0x16		0x0B		



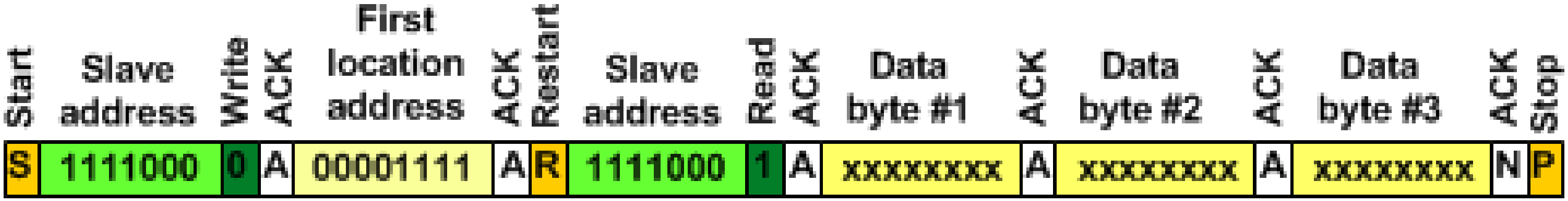
Repeated start

- A new Start condition before the Stop condition



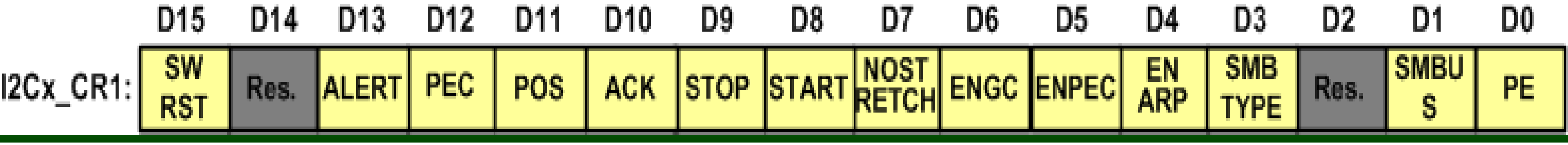
Repeated-Start and Multi-byte Burst Read

- Reading from location 0x0F of memory:
  - Send the address of memory for write
  - Write the address of memory location to be read (0x0F)
  - Make a repeated-start and send the address of memory for read
  - Get data and send Ack. as long as, you want to get the next byte



Start	Slave address	Write	ACK	First location address	ACK	Restart	Slave address	Read	ACK	Data byte #1	ACK	Data byte #2	ACK	Data byte #3	ACK	Stop
S	1111000	0	A	00001111	A	R	1111000	1	A	xxxxxxxx	A	xxxxxxxx	A	xxxxxxxx	N	P

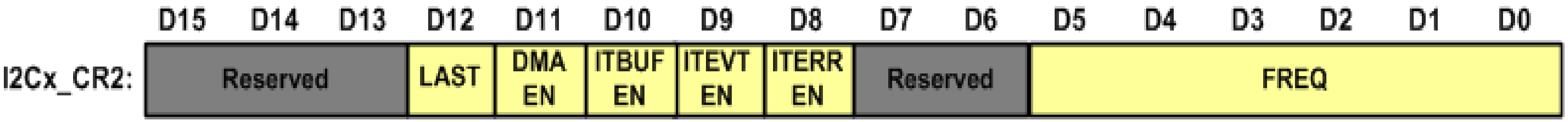
I2C in STM32F10x



	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
I2Cx_CR1:	SW RST	Res.	ALERT	PEC	POS	ACK	STOP	START	NOST RETCH	ENGC	ENPEC	EN ARP	SMB TYPE	Res.	SMBU S	PE

Field	Bit	Descriptions
SWRST	15	Software Reset
ALERT	13	SMBus alert (It is used in SMBus.)
PEC	12	Packet Error Checking
POS	11	PEC Position
ACK	10	Acknowledge enable (0: No Ack, 1: ACK) The bit is set/cleared by software to send ACK/NACK. Hardware clears the bit after sending ACK.
STOP	9	Stop generation If software sets the bit, the I2C hardware generates a Stop condition and switches to slave mode (The MSL bit of I2C_SR2 is cleared.)
START	8	Start generation When software sets the bit, the I2C interface generates a Start condition and switches to master mode (MSL is set). If it is already in master mode, setting the START bit generates a repeated start.
NO STRETCH	7	No clock stretching in slave mode (0: clock stretching enabled, 1: disabled) Software can set the bit to disable clock stretching.
ENGC	6	General Call Enable (0: General call disabled, 1: enabled (Addr. 0 is ACKed))
ENPEC	5	PEC enable (0: PEC calculation disabled, 1: PEC calculation enabled)
ENARP	4	ARP enable (It is used in SMBus.)
SMBTYPE	3	SMB Type (It is used in SMBus.)
SMBUS	1	SMB Bus (0: I2C mode, 1: SMBus mode)
PE	0	Peripheral Enable (0: peripheral disable, 1: enable)

I2C_CR2 (Control Register 2)



	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
I2Cx_CR2:	Reserved			LAST	DMA EN	ITBUF EN	ITEVT EN	ITERR EN	Reserved		FREQ					

Field	Bit	Descriptions
LAST	12	DMA Last Transfer
DMAEN	11	DMA request Enable (0: disabled, 1: enabled)
ITBUFEN	10	Buffer Interrupt Enable (0: Interrupt disabled, 1: enabled) If ITBUFEN is set, an interrupt is generated when TxE or RxNE flags of I2C_SR1 are set.
ITEVTEN	9	Event Interrupt Enable (0: Interrupt disabled, 1: enabled) If ITEVTEN is set, an interrupt is generated when any of the event flags (SB, ADDR, ADD10, STOPF, or BTF) are set.
ITERREN	8	Error Interrupt Enable (0: Interrupt disabled, 1: enabled) If ITERREN is set, an interrupt is generated when any of the error flags (BERR, ARLO, AF, OVR, PECERR, TIMEOUT, or SMBALERT) are set.
FREQ	5-0	PCLK (Peripheral Clock) Frequency

I2Cx_CCR (Clock Control Register)

	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
I2Cx_CCR:	F/S	DUTY	Reserved		CCR											

Field	Bit	Descriptions
F/S	15	Master mode selection (0: Standard mode, 1: Fast mode)
DUTY	14	SCL clock duty cycle in Fast mode
CCR	11-0	Clock Control in master mode

I2C grade	Baud rate
Standard	100Kbps
Fast	Up to 400Kbps
Fast plus	Up to 1Mbps
High Speed	Up to 3.2Mbps

F/S	DUTY	Duty cycle for SCL	t_{low}	t_{high}	T_{I2C} $(=t_{low}+t_{high})$	Baud rate $(1/T_{I2C})$
0 (Standard)	X	50%	CCR × T_{PCLK}	CCR × T_{PCLK}	2×CCR×T_{PCLK}	F_{PCLK}/(2×CCR)
1 (Fast)	0	33.3%	2×CCR×T_{PCLK}	CCR×T_{PCLK}	3×CCR×T_{PCLK}	F_{PCLK}/(3×CCR)
1 (Fast)	1	36%	16×CCR×T_{PCLK}	9×CCR×T_{PCLK}	25×CCR×T_{PCLK}	F_{PCLK}/(25×CCR)

I2C_TRISE (T_{Rise})

- Using the I2C_TRISE, we mention the amount of time that the rise time might take.
  - In Standard mode
    - I2C_TRISE is usually set to (PCLK/1M) + 1
    - E.g.: PCLK = 32M ➔ TRISE = 32 + 1 = 33
  - In Fast mode
    - I2C_TRISE = 0.3 × (PCLK/1M) + 1
    - E.g.: PCLK = 40M ➔ TRISE = (0.3 × 40) +1 = 13

Example

- Assuming PCLK1 is 36MHz. Find I2C_CR2, I2C_CCR and I2C_TRISE for speed of 100Kbps.

Solution:

PCLK1 is 36MHz. So, FREQ should be set to 36 (100100 in binary)

I2C_CR2	Res.	LAST	DMAEN	ITBUFEN	ITEVTEN	ITERREN	Res.	FREQ
	000	0	0	0	0	0	00	100100

I2C_CR2 = 0x0024

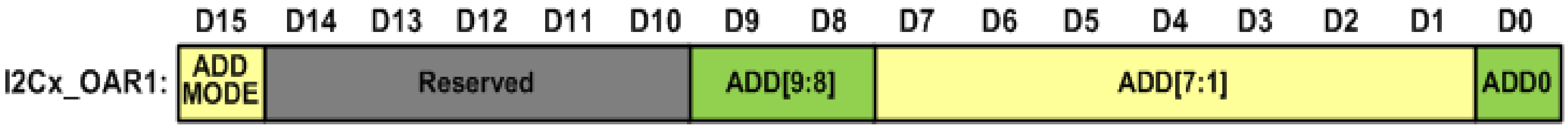
100Kbps is standard. baud rate = PCLK/(2×CCR) ➔ 100K = 36M/(2×CCR) ➔ CCR = 180. So, F/S = 0 and CCR = 000010110100. DUTY can be 0 or 1.

I2C_CCR	F/S	DUTY	Res.	CCR
	0	X	00	0000 1011 0100

I2C_CCR = 0x00B4

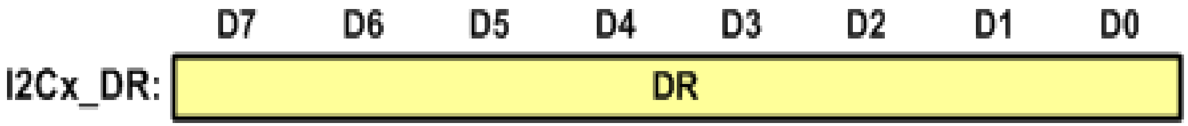
I2C_TRISE = (PCLK/1M) + 1 = 36 + 1 = 37.

I2Cx_OAR1 (Own Address Register)



	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
I2Cx_OAR1:	ADD MODE	Reserved					ADD[9:8]		ADD[7:1]							ADD0

I2C_DR (Data Register)



	D7	D6	D5	D4	D3	D2	D1	D0
I2Cx_DR:	DR							

	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
I2Cx_SR1:	SMB ALERT	TIME OUT	Res.	PEC ERR	OVR	AF	ARLO	BERR	TxE	RxNE	Res.	STOPF	ADD10	BTF	ADDR	SB

Field	Bit	Descriptions
SMBALERT	15	SMB Alert (Not used in I2C)
TIMEOUT	14	Time out (0: No timeout, 1: SCL remained LOW for 25ms)
PECERR	12	PEC Error in reception
OVR	11	Overrun/Underrun (0: No overrun/underrun, 1: overrun or underrun) The flag might rise when NOSTRETCH=1. See the reference manual.
AF	10	Acknowledge Failure (0: No Acknowledge failure, 1: Acknowledge failure)
ARLO	9	Arbitration lost in master mode (0: no arbitration lost, 1: arbitration lost)
BERR	8	Bus Error (0: no bus error, 1: bus error) As discussed earlier, when SCL is high, SDA should not change. Otherwise, it is considered as STOP or START conditions. While transferring data, if the value of SDA changes when SCL is high, the BERR flag sets.
TxE	7	Transmit Empty (0: Transmit not empty, 1: Empty) The flag is set by hardware when it is waiting for us to write data to the I2C_DR register.
RxNE	6	Receive Not Empty The flag is set when a new byte of data is in I2C_DR waiting to be read.
STOPF	4	Stop detection in slave mode (0: No Stop condition, 1: Stop condition)
ADD10	3	10-bit header sent (Used in 10-bit address mode)
BTF	2	Byte transfer finished (0: transfer not done, 1: transfer successfully finished)
ADDR	1	Address sent (master mode)/matched (slave mode) In slave mode, if the received address matches with I2C_OAR, the flag sets. In master mode, when the address is sent, the flag sets. To clear the flag, read I2C_SR1 and then I2C_SR2.
SB	0	Start bit (0: No start generated, 1: start condition generated) In master mode, the flag sets as soon as a Start condition is generated. To clear the flag, read I2C_SR1 and then write to I2C_DR.

I2Cx_SR2 (Status Register 2)

	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
I2Cx_SR2:	PEC								DUALF	SMB HOST	SMBDE FAULT	GEN CALL	Res.	TRA	BUSY	MSL

Field	Bit	Descriptions
PEC	15-8	Packet Error Checking (For more information, see the manual.)
DUALF	7	Dual Flag (It is used when dual address is enabled. See the manual.)
SMBHOST	6	Used in SMBus. (For more information, see the manual.)
SMBDEFAULT	5	Used in SMBus. (For more information, see the manual.)
GENCALL	4	General call detected (0: no general call, 1: General call address received)
TRA	2	Transmitter/receiver (0: receiver, 1: transmitter) In slave mode, the flag shows if the device is in receiver or transmitter mode. The hardware sets the flag according to the R/W signal.
BUSY	1	Bus Busy (0: no communication on the bus, 1: communication on the bus) The flag indicates that a communication is in progress. Hardware sets the flag when SCL or SDA become low. The flag clears when a stop condition is detected.
MSL	0	Master/Slave (0: Slave, 1: Master) The hardware sets the flag when the I2C module is in master mode. The flag is set when a stop condition or arbitration lost is detected.

Configuring the I2C

- We need to take the following steps to configure the I2C:
  - Enable the clock to I2C module and the GPIO using APB1ENR and APB2ENR,
  - Configure I2C pins as alternate function open-drain output,
  - Initialize the FREQ field of CR2 with the PCLK frequency,
  - Initialize CCR to make proper baud rate in master mode,
  - Initialize the TRISE register,
  - Enable the I2C module by setting the PE bit of CCR1.

Sending data in master mode

1. Check the busy flag of SR2 to make sure the bus is not busy.
2. Set the START bit of CR1 to make a start condition.
3. Monitor the SB bit of SR1 until the start condition is generated.
4. Put the slave address in the data register (DR). Bits 1 to 7 should contain the slave address and the bit 0 is R/W. To send data the R/W needs to be 0.
5. Monitor the status registers. If the address is sent successfully the ADDR flag sets and you can continue the progress. If the ARLO (Arbitration Lost) is set, you should wait until the bus becomes free and you should repeat steps 1 to 5.
6. Load the data register with the data to be sent.
7. Monitor the TxE flag. The flag is set when an ACK is received.
8. Repeat steps 6 and 7 if you have more bytes to send. Otherwise, set the STOP bit of CR1 to make a stop condition.

Receiving Data

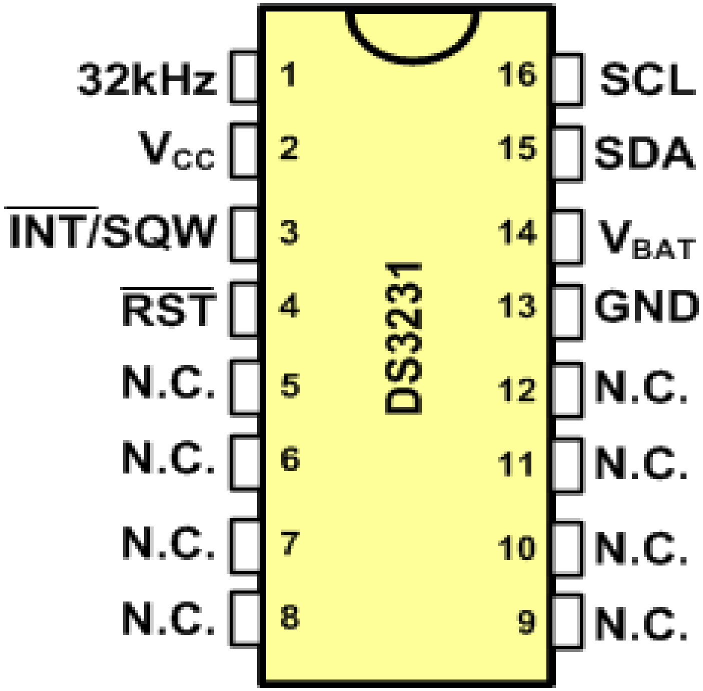
- Receiving data is similar to sending. To receive data in master mode we should do the followings:
  1. Check the busy flag of SR2 to make sure the bus is not busy.
  2. Set the START bit of CR1 to make a start condition.
  3. Monitor the SB bit of SR1 until the start condition is generated.
  4. Put the slave address in the data register (DR). Bits 1 to 7 should contain the slave address and the bit 0 is R/W. To receive data, the R/W needs to be 1.
  5. Monitor the status registers. If the address is sent successfully the ADDR flag sets and you can continue the progress. If the ARLO (Arbitration Lost) is set, you should wait until the bus becomes free and you should repeat steps 1 to 5.
  6. If you want to send an ACK in response, set the ACK bit of CR1.
  7. Monitor the RxNE flag. The flag is set when a byte is received. Then, read the data register to get the received byte.
  8. Repeat steps 6 and 7 if you want to receive more bytes. Otherwise, set the STOP bit of CR1 to make a stop condition.

Program: Sending 0x0E and 0 to slave 0x68

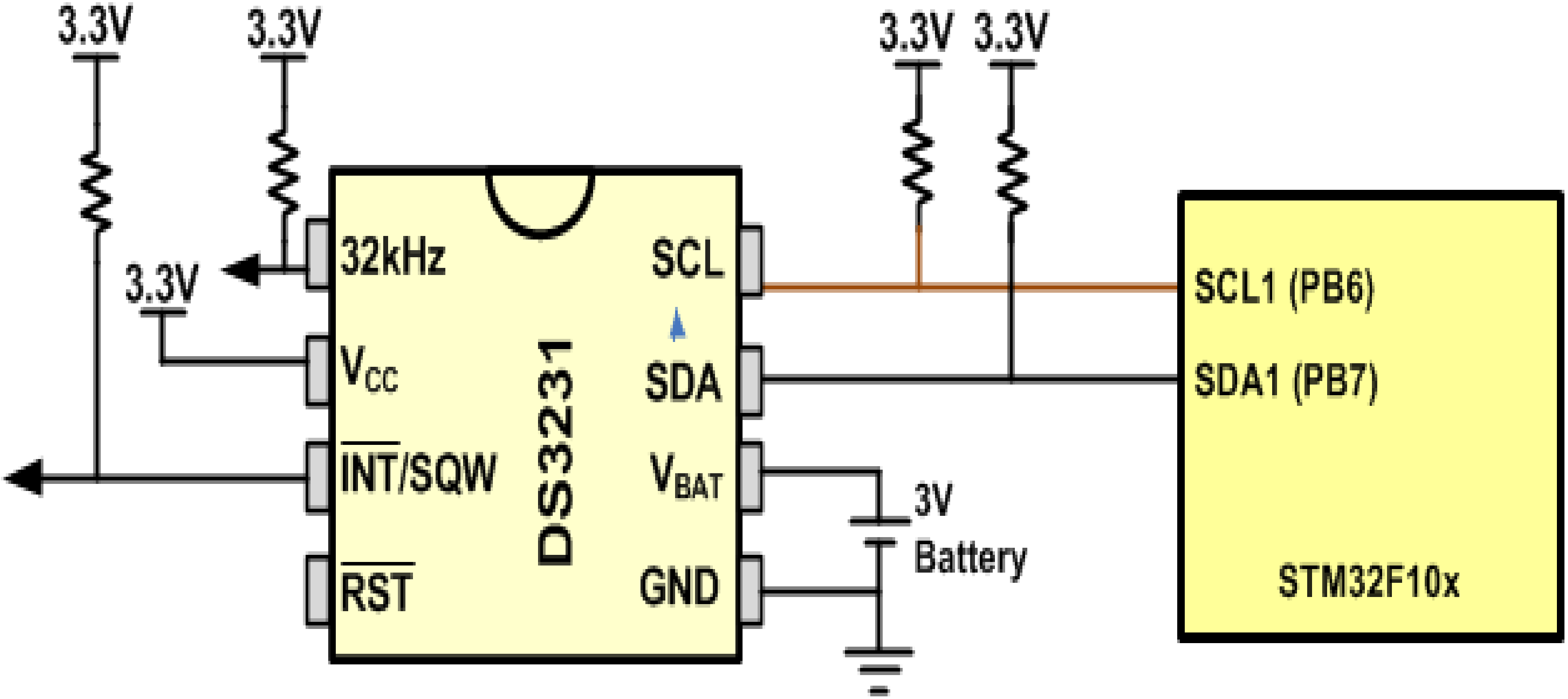
```
#include <stm32f10x.h>

void i2c_init(void);
void i2c_waitForReady(void);
void i2c_sendStart(void);
uint8_t i2c_sendAddrForWrite(uint8_t addr);
uint8_t i2c_sendData(uint8_t data);
void i2c_sendStop(void);

int main( ) {
        i2c_init();

        do{
                i2c_waitForReady();
                i2c_sendStart();
        }while(i2c_sendAddrForWrite(0x68) != 0);

        i2c_sendData(0x0E);
        i2c_sendData(0);
        i2c_sendStop();
        while(1);
}
```

DS3231

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



Connecting DS3231 to the MCU



DS3231 address map

Addr.	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Function	Range
00H	0	10 Seconds			Seconds				Seconds	00-59
01H	0	10 Minutes			Minutes				Minutes	00-59
02H	0	12/24	PM/AM 20 hour	10hour	Hours				Hours	1-12+AM/PM 0-23
03H	0	0	0	0	0	Day			Day	1-7
04H	0	0	10 Date		Date				Date	01-31
05H	Century	0	0	10Month	Month				Month Century	1-12+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 00-23
0AH	A1M4	DY/DT	10 Date		Day Date				Alarm 1 Day Alarm 1 Date	1-7 1-31
0BH										0-59
0CH										1-12 0-23
0DH										1-7 1-31
0EH										-
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	-

- 19 bytes
- BCD format is used
- Addresses 0x00 to 0x06 give time and date.

Register Pointer in DS3231

Register pointer:

- In DS3231 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.

Writing to DS3231

- Transmit START condition
- Transmit the address of DS3231 (1001101) followed by 0 to indicate a write operation
- Transmit the address of location you want to access (it sets the value of Register Pointer)
- Transmit one or more bytes of data
- Transmit STOP condition

Reading from DS3231

- 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 before reading)

Example: Setting Time and Date

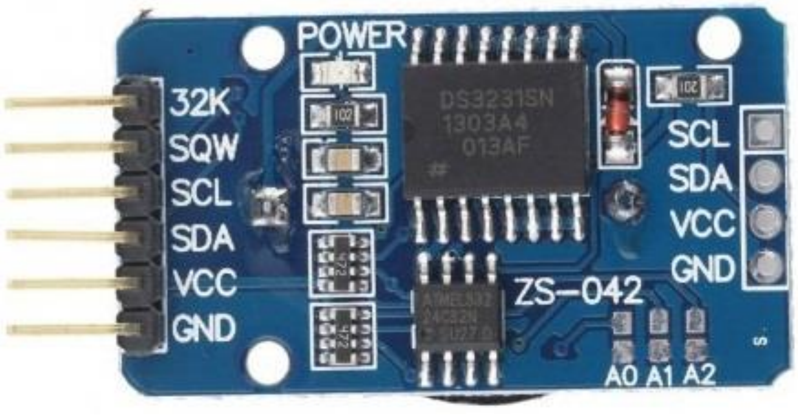
```
/* The program sets time and date to 15/9/2019 19:14:35 */
#include <stm32f10x.h>

void i2c_init(void);
void i2c_waitForReady(void);
void i2c_sendStart(void);
uint8_t i2c_sendAddrForWrite(uint8_t addr);
uint8_t i2c_sendData(uint8_t data);
void i2c_sendStop(void);

int main( ) {
        i2c_init();

        do{
                i2c_waitForReady();/* wait while the bus is busy */
                i2c_sendStart();   /* generate a start condition */
        /* send slave addr. 0x68 for write. repeat from beginning if arbitration lost */
        }while(i2c_sendAddrForWrite(0x68) != 0);

        i2c_sendData(0x0); /* set addr. pointer to 0 */
```

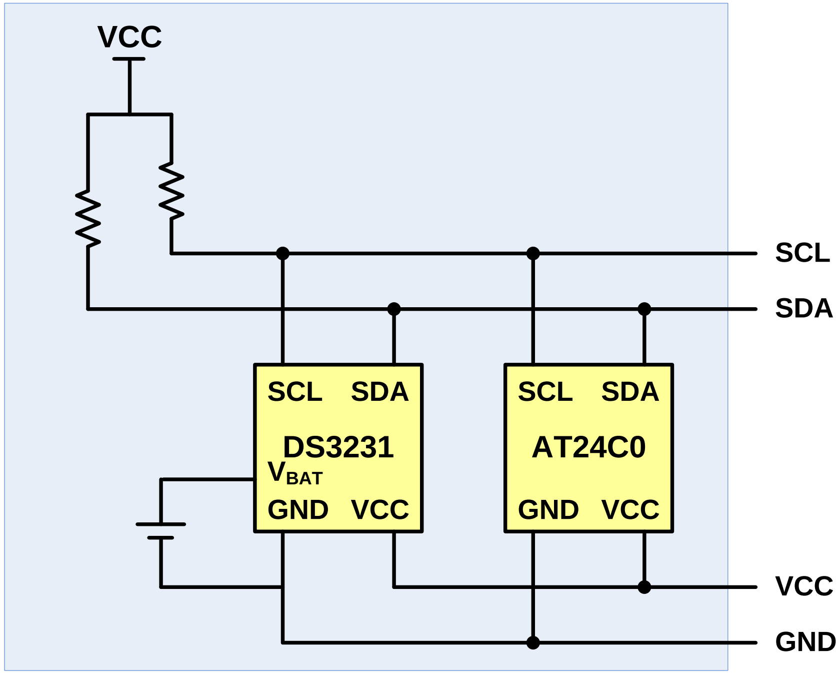
Example: Reading Time and Date and sending via USART1

```
#include <stm32f10x.h>
#include <stdio.h>

void i2c_init(void);
void i2c_waitForReady(void);
void i2c_sendStart(void);
void i2c_sendStop(void);
uint8_t i2c_sendAddrForRead(uint8_t addr);
uint8_t i2c_sendAddrForWrite(uint8_t addr);

void getTime(uint8_t *year, uint8_t *month, uint8_t *day, uint8_t *hour, uint8_t *min, uint8_t *sec);

void usart1_init(void);
void usart1_sendByte(unsigned char c);
void usart1_sendStr(char *str);

void delay_ms(uint16_t t);

int main()
```

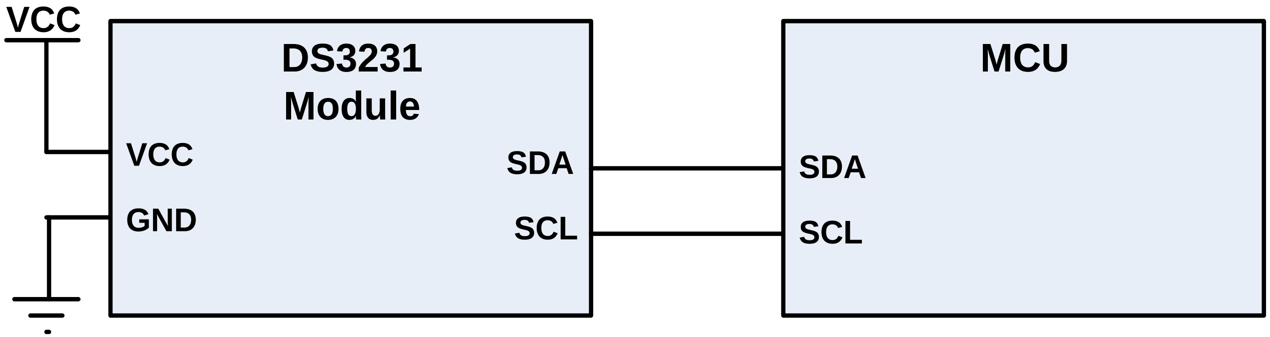
DS3231 module

- It contains:
  - – DS3231
  - – A backup battery
  - – An AT24C0 EEPROM





Connecting DS3231 to the MCU


VCC
DS3231
Module
VCC
GND
SDA
SCL
MCU
SDA
SCL