Embedded Systems and Software

Serial Interconnect Buses—I^{2}C and SPI

Purpose of Serial Interconnect Buses

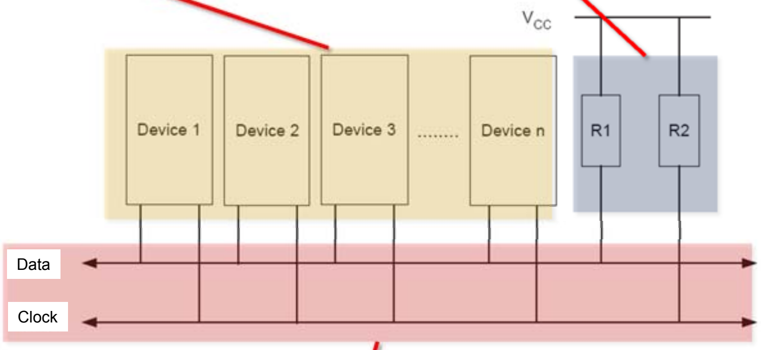
- Provide low-cost—i.e., low wire/pin count—connection between IC devices
- There are many serial bus "standards"
  - I2C (Inter-Integrated Circuit)
  - SMB (System Management Bus)
  - SPI (Serial Peripheral Interface)
  - Microwire
  - Maxim 3-wire
  - Maxim/Dallas 1-wire
  - CAN (controller area network)
  - etc.
- We will focus on I^{2}C and SPI

Overview

Generic Serial Interconnect Bus

Devices on bus. Can be one or multiple micros + one or more peripherals

Pullup resistors ensure idle state of bus is HIGH. Devices pull line low when signaling. Wired-OR arrangement



Bus. 1, 2, or 3 wires. This bus has 2 lines: **Data** and **Clock**. Bus carries data and commands.

Commonly Encountered Terminology

Term	Description
Transmitter	The device which sends the data to the bus.
Receiver	The device which receives the data from the bus.
Master	The device which initiates a transfer, generates clock signals and terminates a transfer.
Slave	The device addressed by a master.
Multi-Master	More than one master can attempt to control the bus.
Arbitration	Only one master can control the bus.
Synchronization	Procedure to sync. the clock signal.

I^{2}C (Inter-IC)

- **I^{2}C, "Eye-Square-See", I2C, "Eye-Two-See"**
  - **Two-wire serial bus protocol developed by Philips Semiconductors ~ 20 years ago**
  - **Enables peripheral ICs to communicate using simple communication hardware**
  - ***Data transfer rates up to 100 kbits/s and 7-bit addressing possible in normal mode***
  - **3.4 Mbits/s and 10-bit addressing in fast-mode**
  - **Common devices capable of interfacing to I^{2}C bus:**

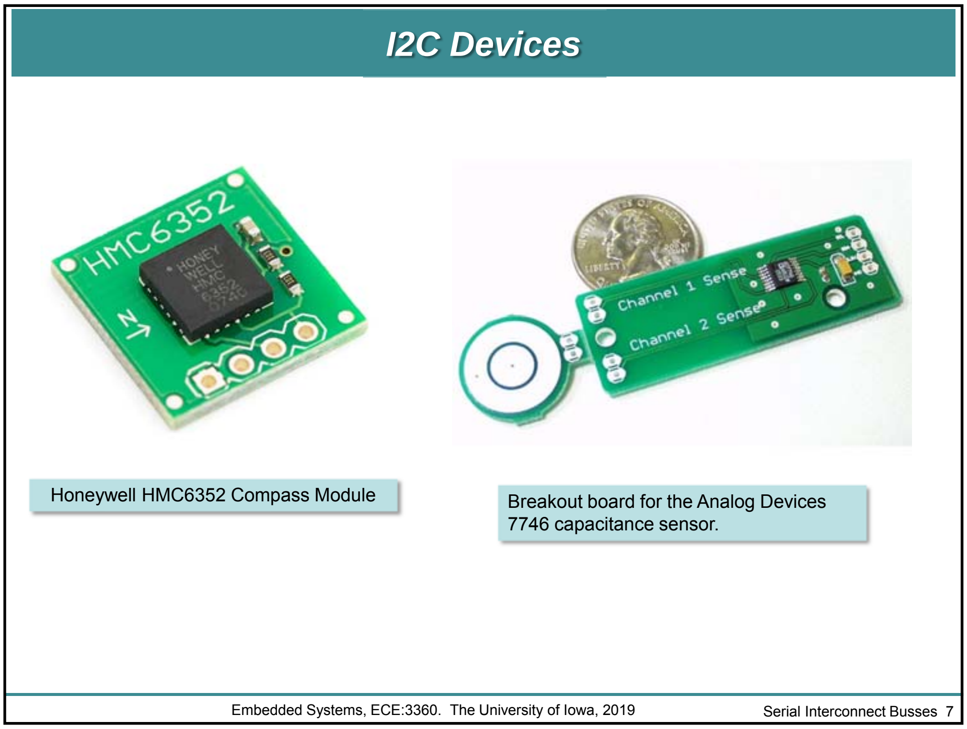
    EPROM, Flash, and some RAM memory, real-time clocks, watchdog timers, and microcontrollers
- **Many microcontrollers, including ATmega88PA, have Two-Wire Interface (TWI) hardware**
- **AVR's TWI can be used to implement I2C, SMB, etc.**

I2C Devices

BlinkM®is a "Smart LED", a networkable and programmable full-color RGB LED for hobbyists, industrial designers, and experimenters.

MCP4725 is an I2C controlled Digital-to-Analog converter (DAC).

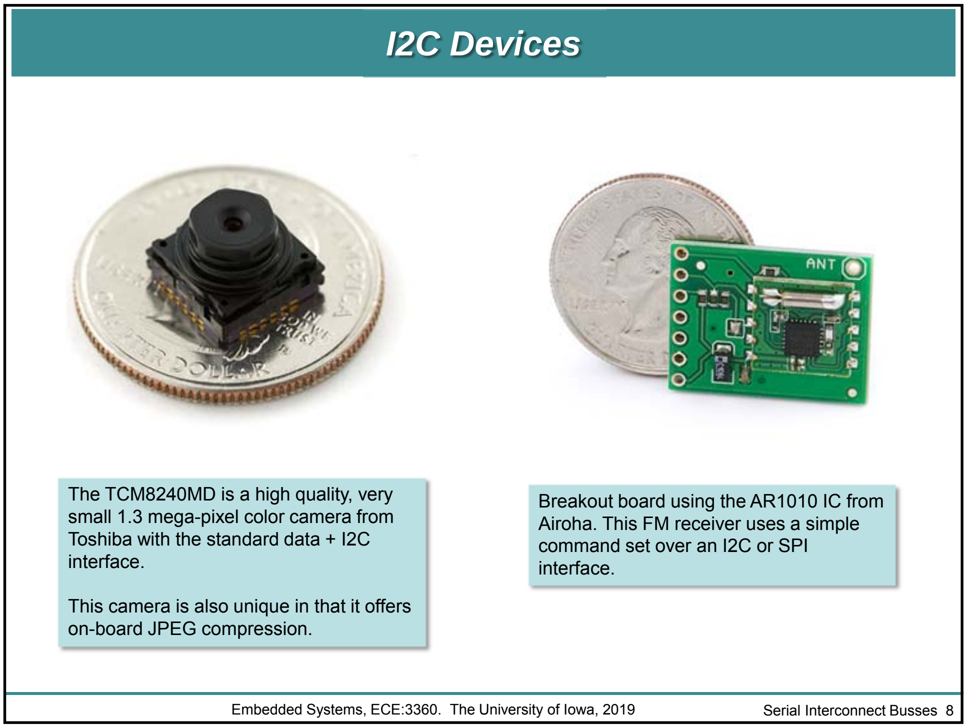
A DAC allows a microcontroller to output analog values like a sine wave. Digital to analog converters are used sound generation, musical instruments, filtering, etc.



I2C Devices

Honeywell HMC6352 Compass Module

Breakout board for the Analog Devices 7746 capacitance sensor.



I2C Devices

The TCM8240MD is a high quality, very small 1.3 mega-pixel color camera from Toshiba with the standard data + I2C interface.

This camera is also unique in that it offers on-board JPEG compression.

Breakout board using the AR1010 IC from Airoha. This FM receiver uses a simple command set over an I2C or SPI interface.

I2C

The I2C-bus is a multi-master bus. This means that more than one device capable of controlling the bus can be connected to it.

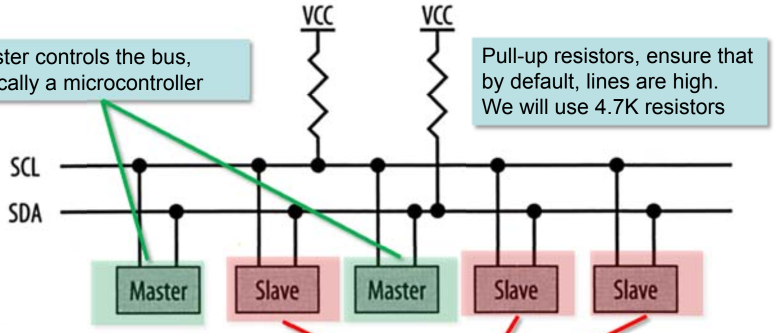
Masters are usually microcontrollers, slaves are peripherals

Often there is one master (Atmega88PA) and one or more slaves (RTC, ADC, DAC, …)

I2C Structure

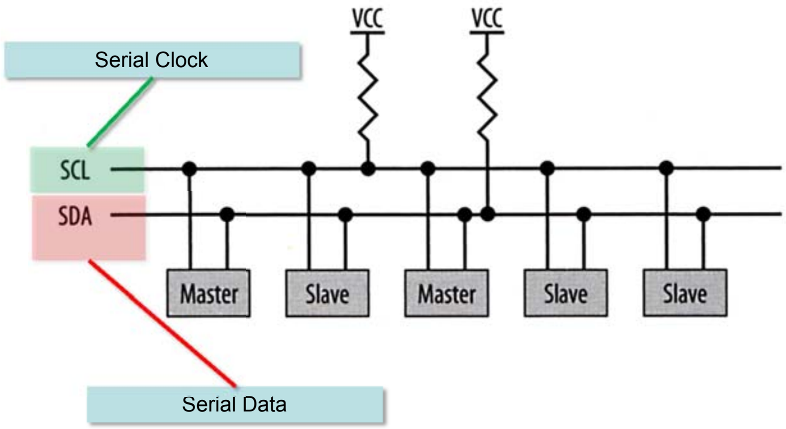
Master controls the bus, typically a microcontroller

Pull-up resistors, ensure that by default, lines are high. We will use 4.7K resistors



Slave responds to master(s), typically sensors, memory, etc.

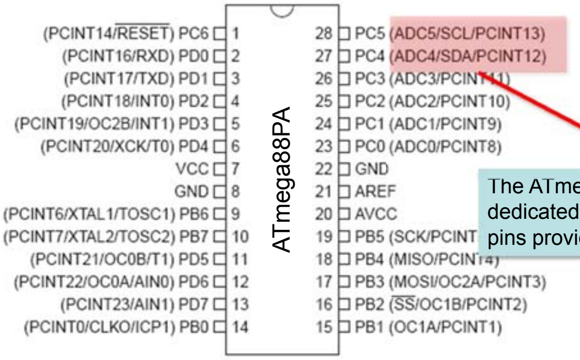
I2C Structure



Serial Clock

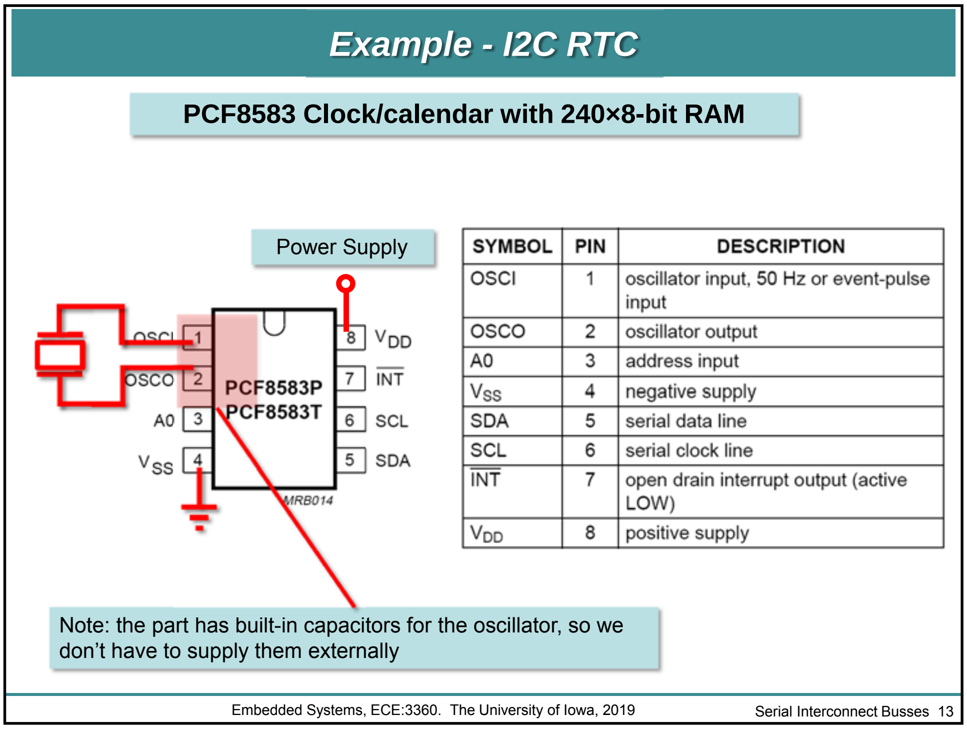
Serial Data

TWI Hardware on ATmega88PA



The ATmega88PA controller has dedicated TWI hardware. These pins provide access to the hardware

One can also implement a Two-Wire serial interface in software on other pins. In other words, by *bit-banging*.



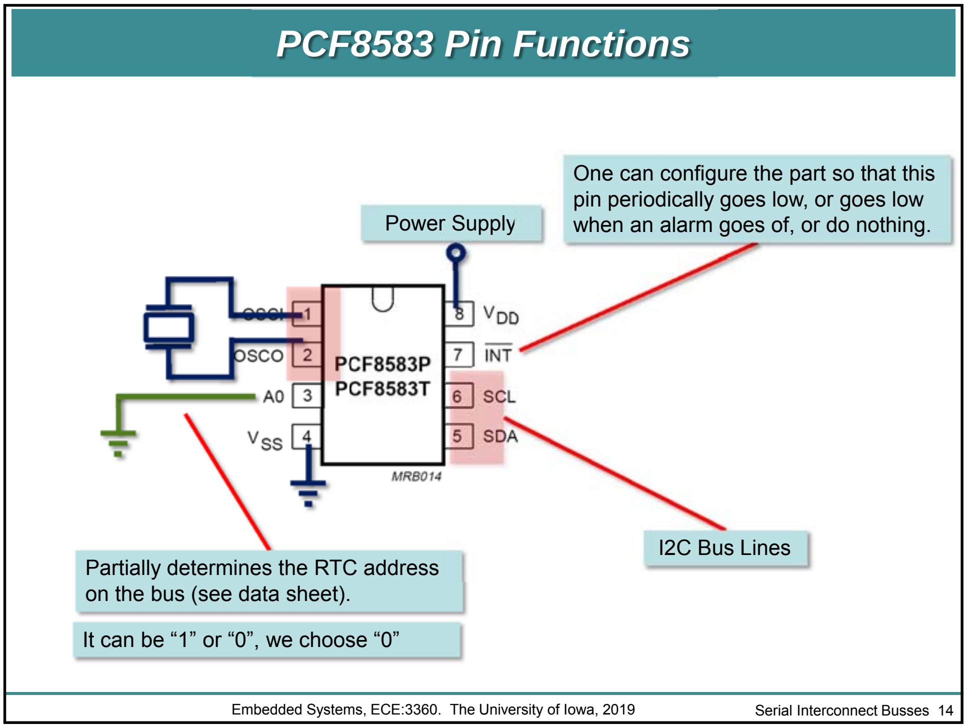
Example - I2C RTC

PCF8583 Clock/calendar with 240×8-bit RAM

Power Supply

SYMBOL	PIN	DESCRIPTION
OSCI	1	oscillator input, 50 Hz or event-pulse input
OSCO	2	oscillator output
A0	3	address input
V_{SS}	4	negative supply
SDA	5	serial data line
SCL	6	serial clock line
$\overline{INT}$	7	open drain interrupt output (active LOW)
V_{DD}	8	positive supply

Note: the part has built-in capacitors for the oscillator, so we don’t have to supply them externally



PCF8583 Pin Functions

One can configure the part so that this pin periodically goes low, or goes low when an alarm goes of, or do nothing.

Power Supply

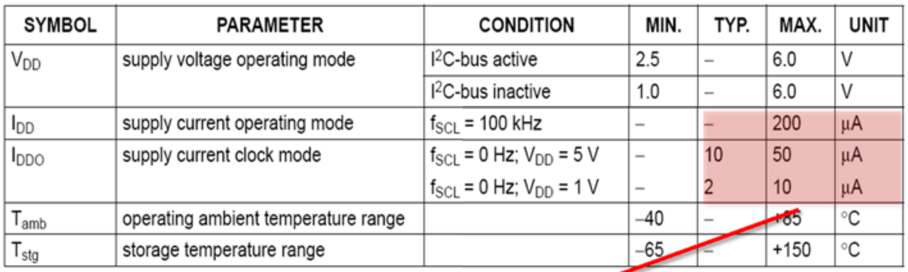
I2C Bus Lines

Partially determines the RTC address on the bus (see data sheet).

It can be “1” or “0”, we choose “0”

Example - I2C RTC

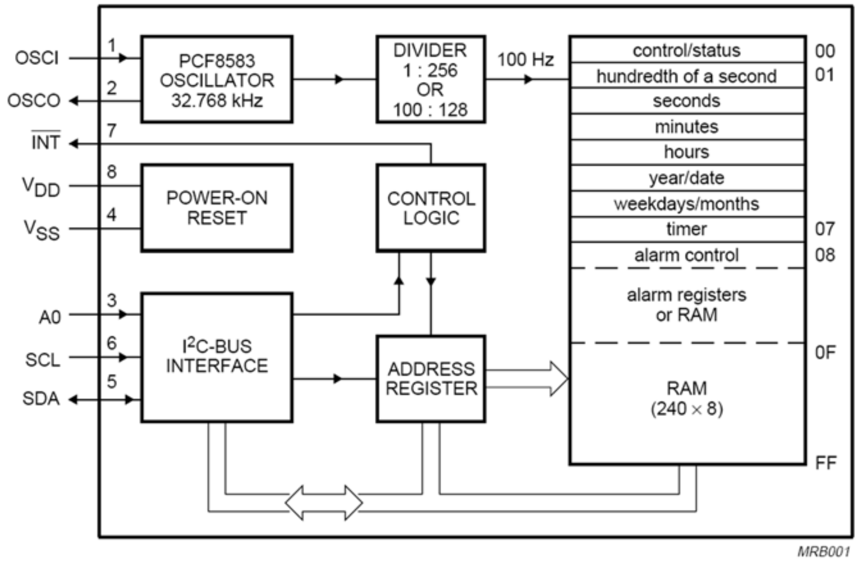
PCF8583 Clock/calendar with 240×8-bit RAM

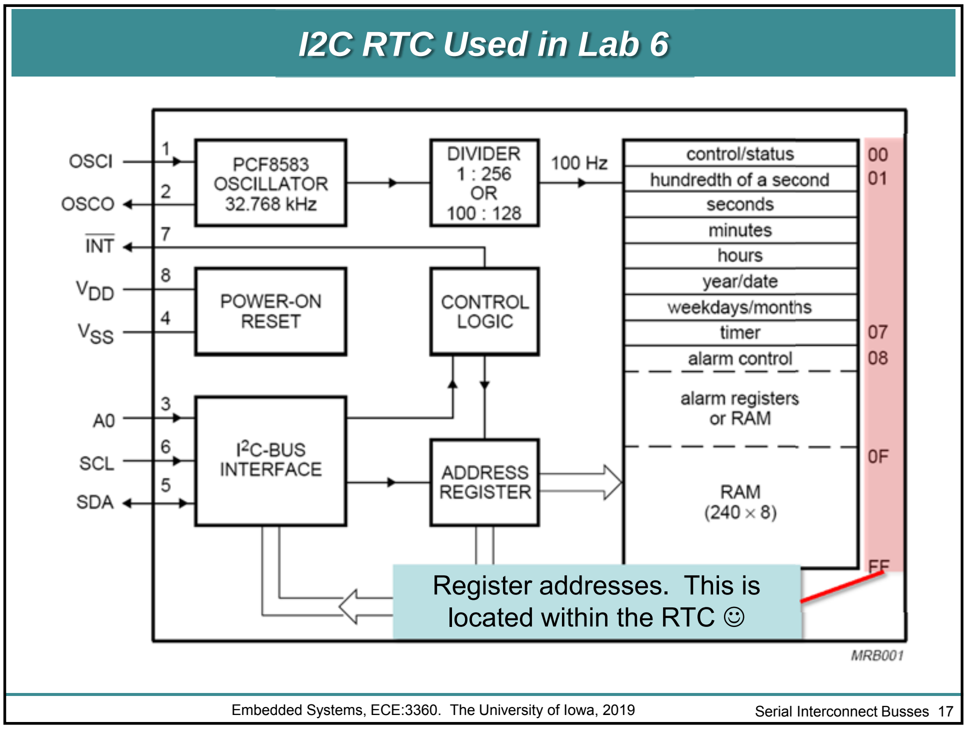


SYMBOL	PARAMETER	CONDITION	MIN.	TYP.	MAX.	UNIT
V_{DD}	supply voltage operating mode	I^{2}C-bus active	2.5	–	6.0	V
		I^{2}C-bus inactive	1.0	–	6.0	V
I_{DD}	supply current operating mode	f_{SCL} = 100 kHz	–	–	200	µA
I_{DDO}	supply current clock mode	f_{SCL} = 0 Hz; V_{DD} = 5 V	–	10	50	µA
		f_{SCL} = 0 Hz; V_{DD} = 1 V	–	2	10	µA
T_{amb}	operating ambient temperature range		−40	–	+85	°C
T_{stg}	storage temperature range		−65	–	+150	°C

Notice, this does not use much current, one reason is because the clock frequency is low: 32.768 kHz

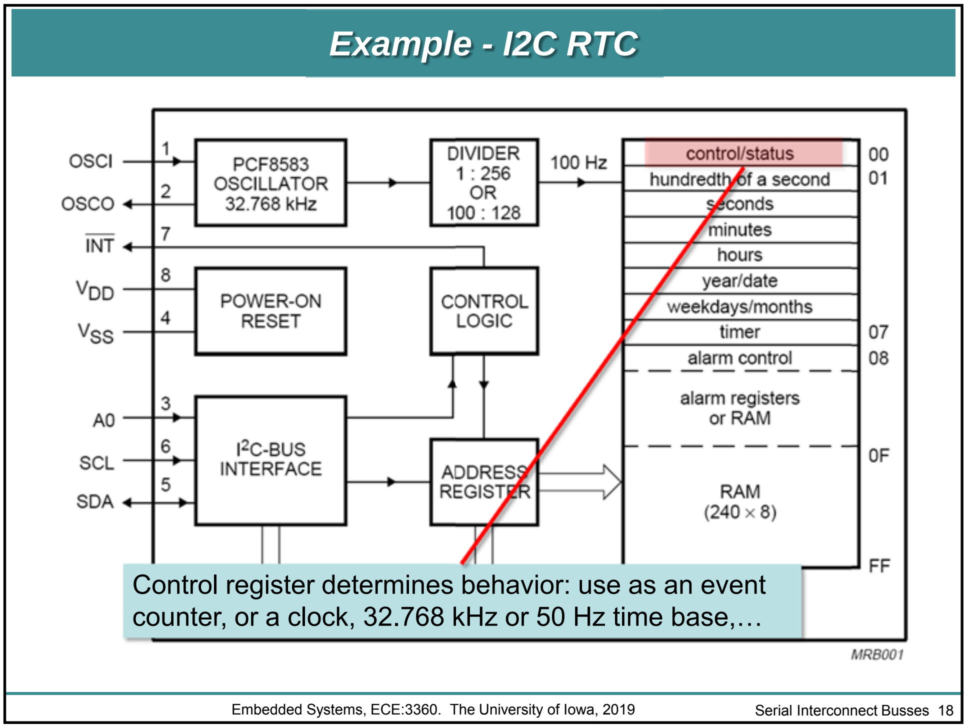
Example - I2C RTC





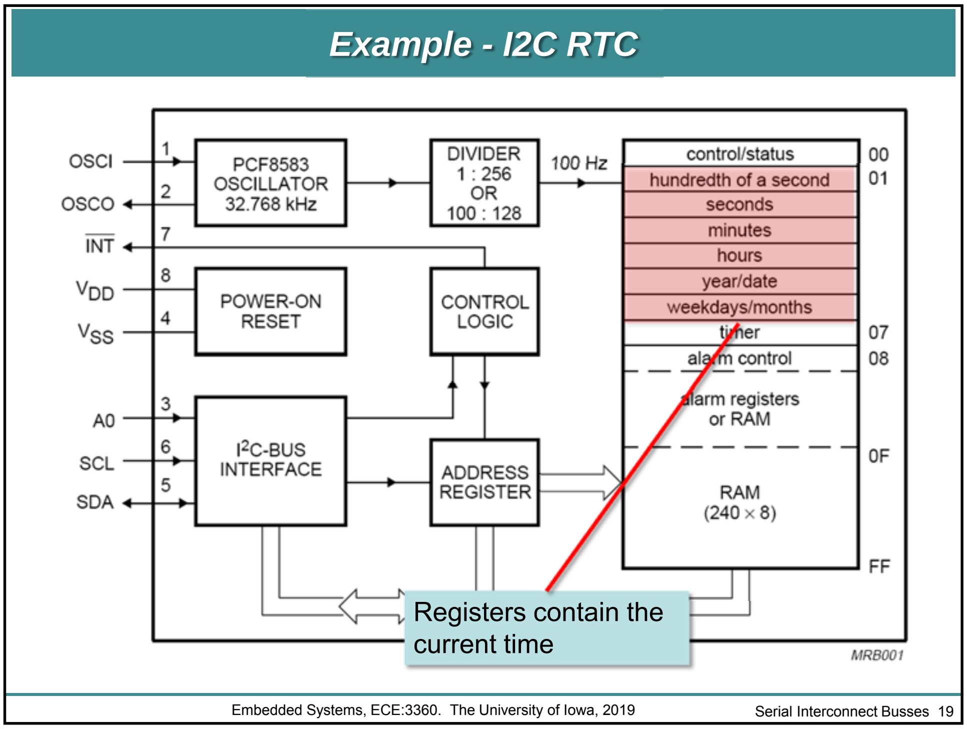
I2C RTC Used in Lab 6

Register addresses. This is located within the RTC ☺



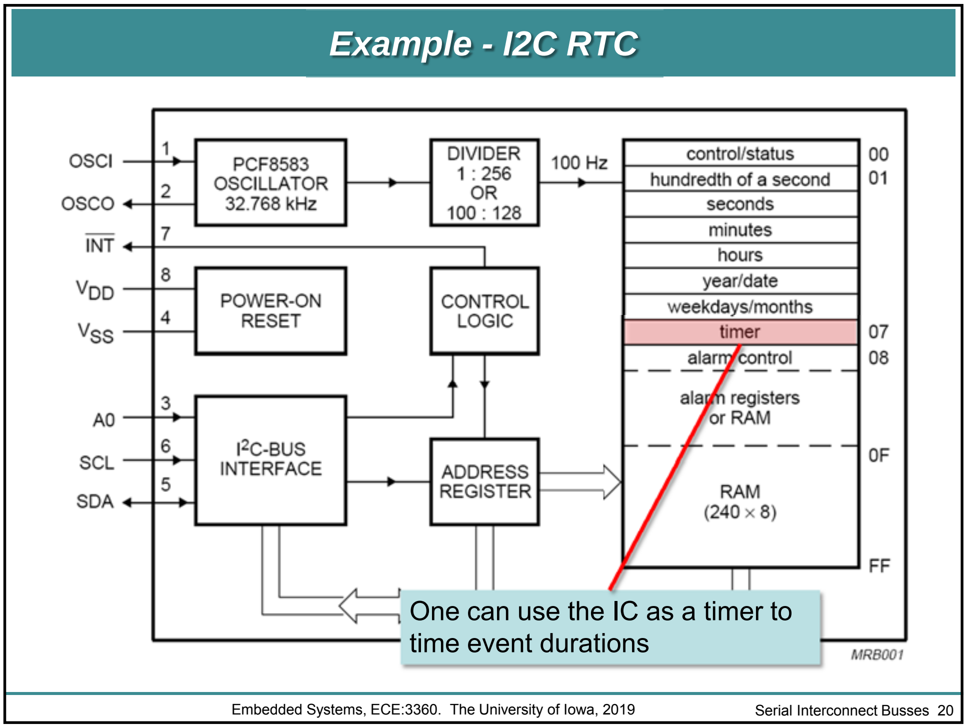
Example - I2C RTC

Control register determines behavior: use as an event counter, or a clock, 32.768 kHz or 50 Hz time base,…



Example - I2C RTC

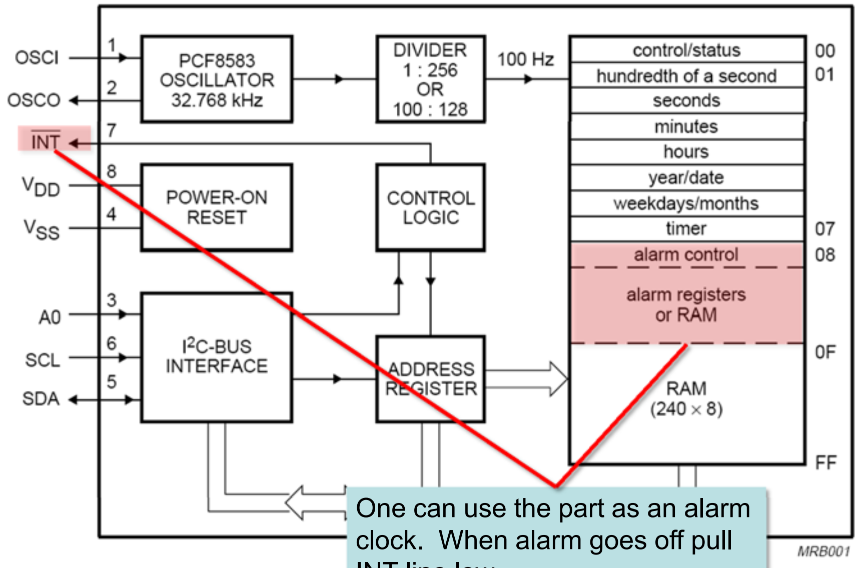
Registers contain the current time



Example - I2C RTC

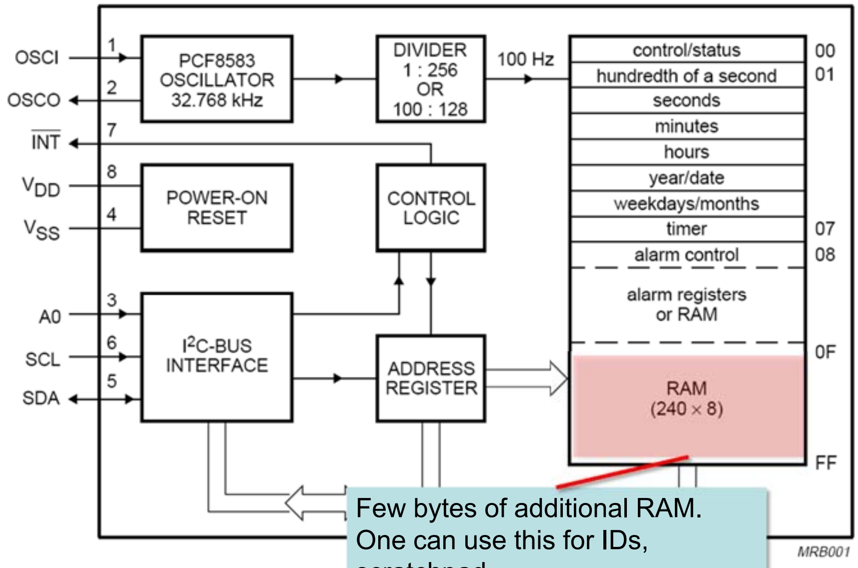
One can use the IC as a timer to time event durations

Example - I2C RTC

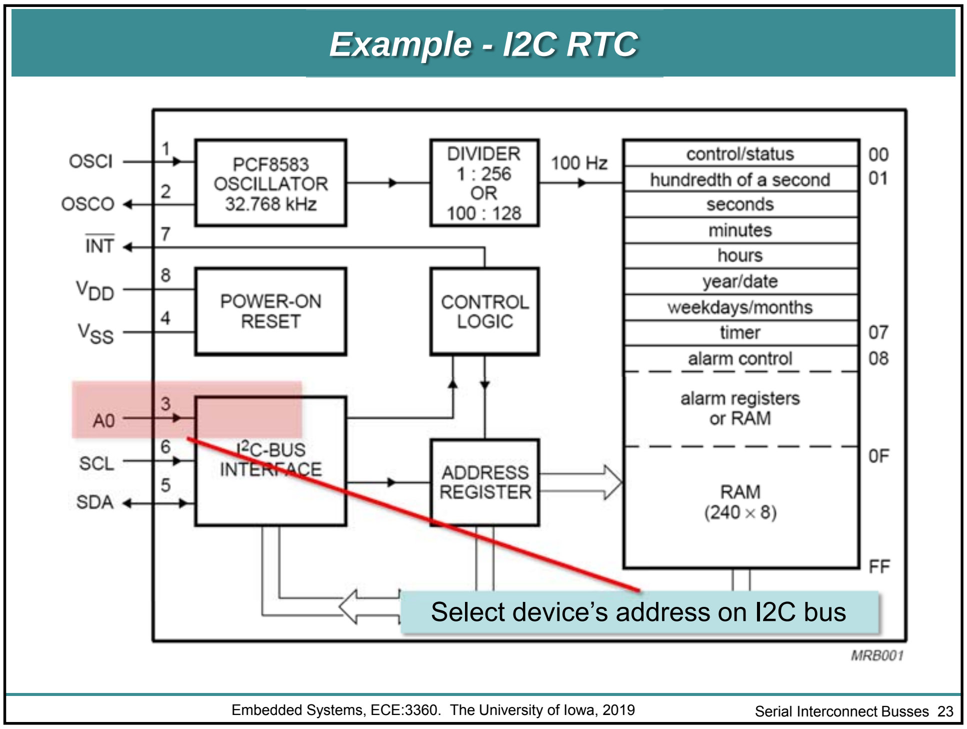


One can use the part as an alarm clock. When alarm goes off pull INT line low

Example - I2C RTC

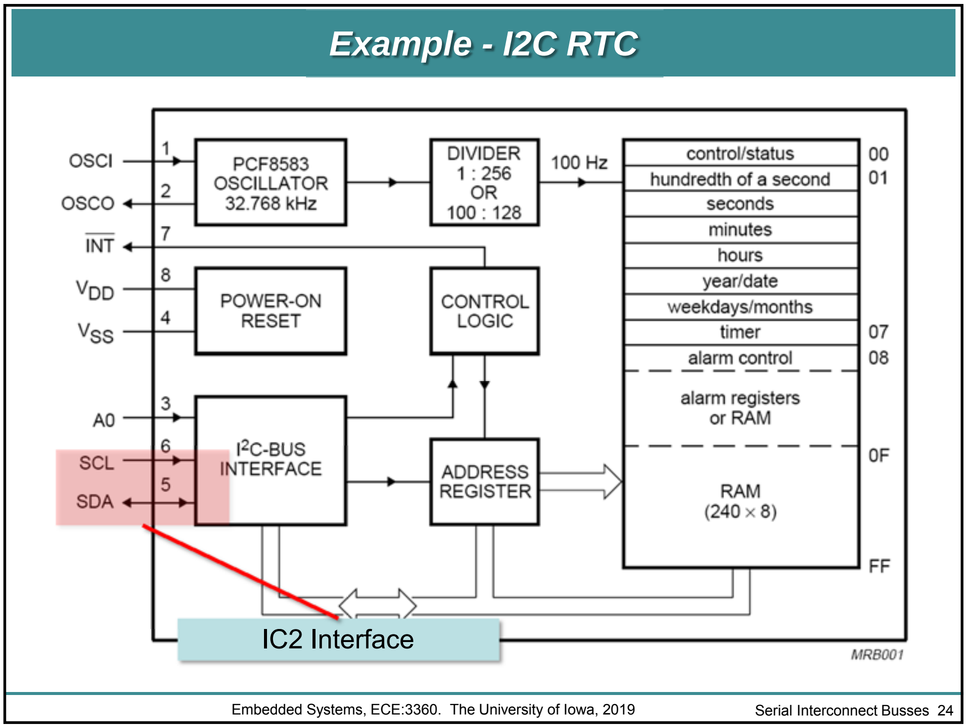


Few bytes of additional RAM. One can use this for IDs, scratchpad, …



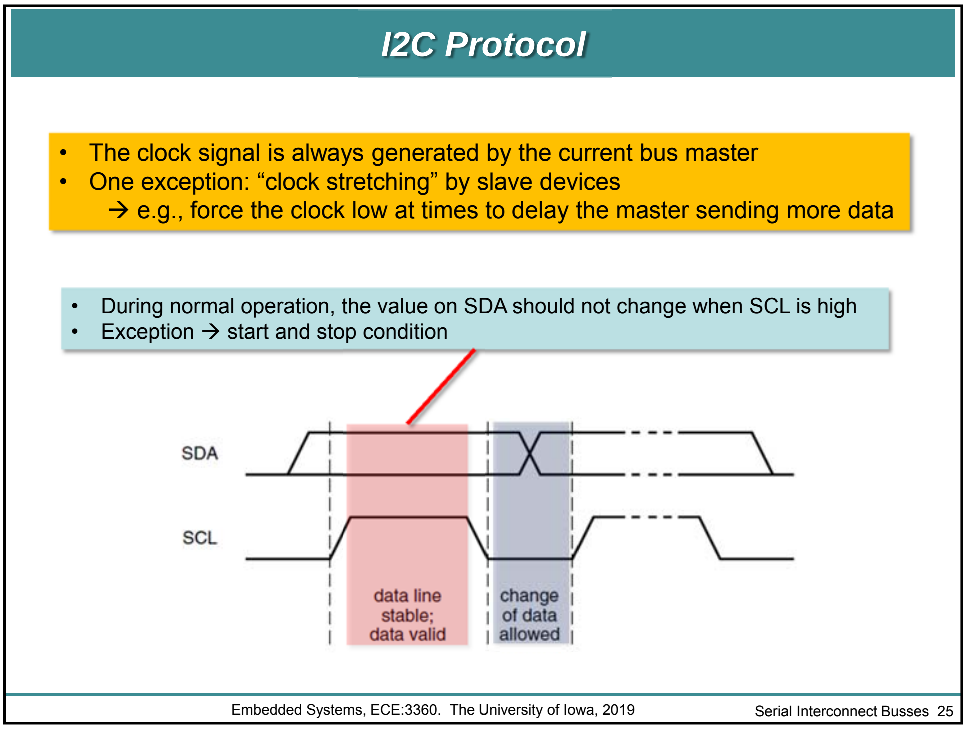
Example - I2C RTC

Select device’s address on I2C bus



Example - I2C RTC

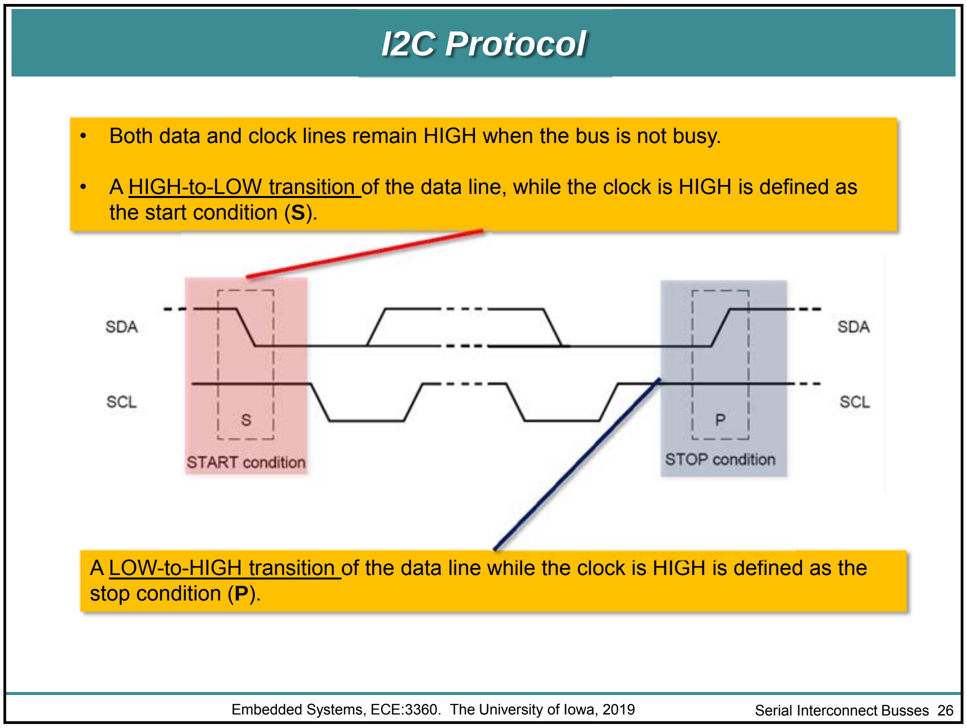
IC2 Interface



I2C Protocol

- The clock signal is always generated by the current bus master
- One exception: “clock stretching” by slave devices
  → e.g., force the clock low at times to delay the master sending more data

- During normal operation, the value on SDA should not change when SCL is high
- Exception → start and stop condition



I2C Protocol

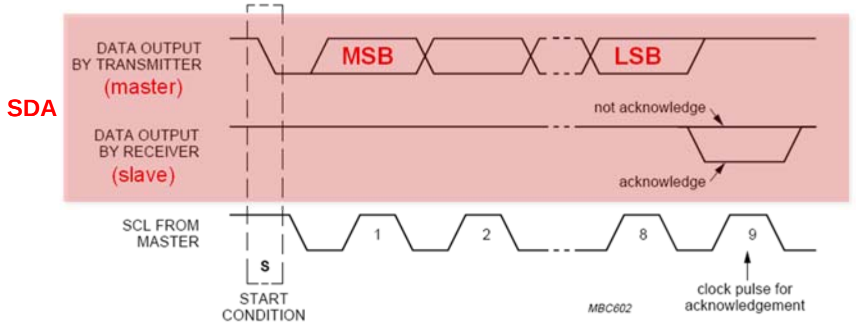
- Both data and clock lines remain HIGH when the bus is not busy.
- A HIGH-to-LOW transition of the data line, while the clock is HIGH is defined as the start condition (**S**).

A LOW-to-HIGH transition of the data line while the clock is HIGH is defined as the stop condition (**P**).

Acknowledgement on the I2C Bus

a) Master transmitter (the master addresses a slave and transmits data to it)

Each byte of eight bits is followed by an acknowledge bit (**ACK**). Upon transmission of the 8th bit, the master releases the SDA line, which goes HIGH, the master generates an **ACK** clock pulse, and the slave acknowledges by pulling the SDA line low.

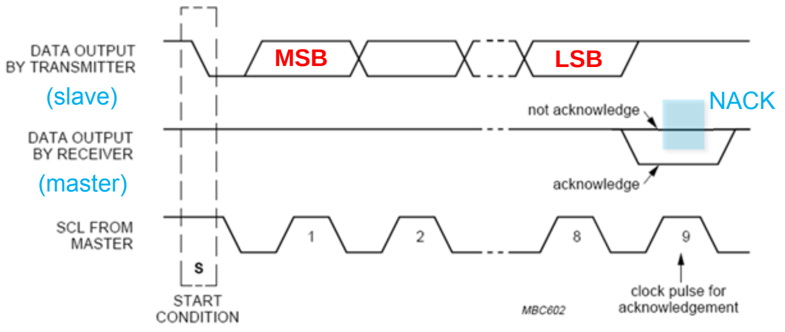


b) Master receiver (the master addresses a slave and receives data from it)

A **master receiver** must generate an acknowledge after the reception of each byte that has been clocked out of the slave transmitter.

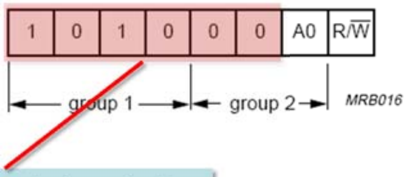
Acknowledgement on the I2C Bus

- A **master receiver** must signal an **end of data** to the transmitter by **not generating** an acknowledge on the last byte that has been clocked out of the slave (**NACK)**.
- In this event, the transmitter must leave the data line HIGH to enable the master to generate a stop condition.



Addressing on the I2C Bus

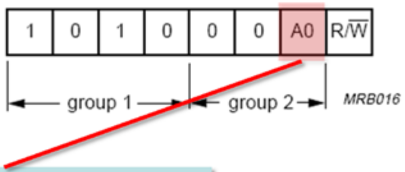
- Before any data is transmitted on the I2C-bus, the device which should respond is addressed first.
- The addressing is always carried out with the first byte transmitted after the start procedure.



This part of the address is determined by the manufacturer of the PCF8583, and is fixed.

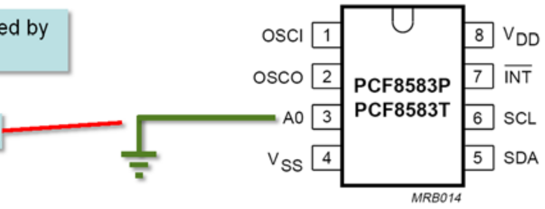
Addressing on the I2C Bus

- Before any data is transmitted on the I2C-bus, the device which should respond is addressed first.
- The addressing is always carried out with the first byte transmitted after the start procedure.



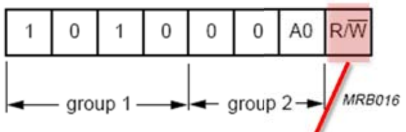
This part of the address is determined by the state of the IC’s A0 pin

It can be “1” or “0”, we choose “0”



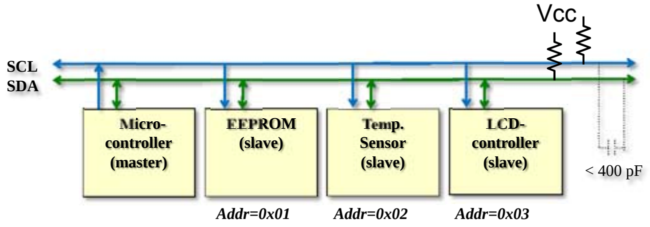
Addressing on the I2C Bus

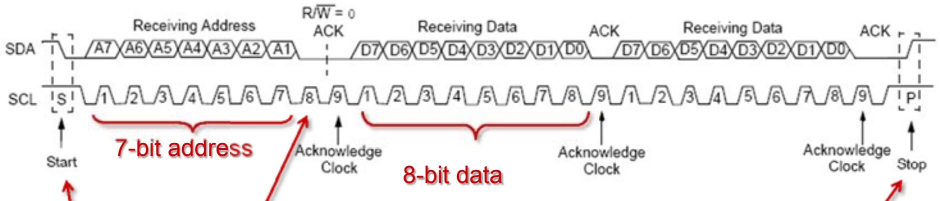
- Before any data is transmitted on the I2C-bus, the device which should respond is addressed first.
- The addressing is always carried out with the first byte transmitted after the start procedure.



This bit determines if we are reading (= 1) from or writing to (= 0) to the devices

I2C Structure





7-bit address

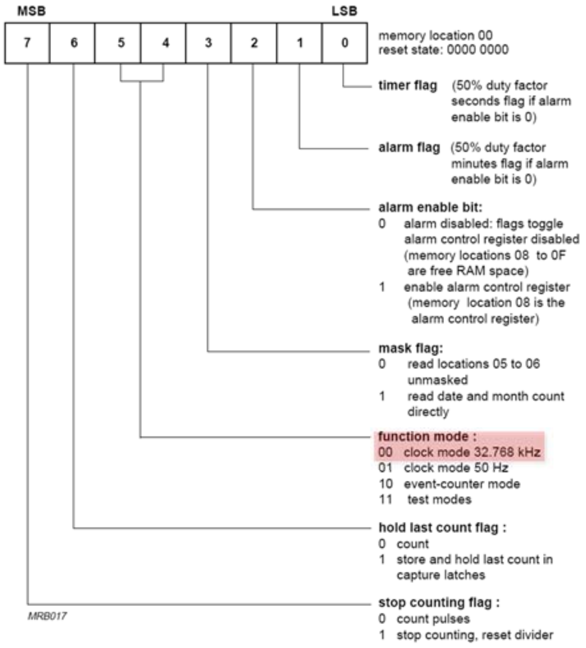
8-bit data

R/W

Start condition

Stop condition

PCF8583 Control Status Register



memory location 00
reset state: 0000 0000

timer flag (50% duty factor seconds flag if alarm enable bit is 0)

alarm flag (50% duty factor minutes flag if alarm enable bit is 0)

alarm enable bit:
0 alarm disabled: flags toggle alarm control register disabled (memory locations 08 to 0F are free RAM space)
1 enable alarm control register (memory location 08 is the alarm control register)

mask flag:
0 read locations 05 to 06 unmasked
1 read date and month count directly

function mode :
00 clock mode 32.768 kHz
01 clock mode 50 Hz
10 event-counter mode
11 test modes

hold last count flag :
0 count
1 store and hold last count in capture latches

stop counting flag :
0 count pulses
1 stop counting, reset divider

PCF8583 Register Arrangement

Values are stored in in BCD

What does “BCD” mean?

Answer

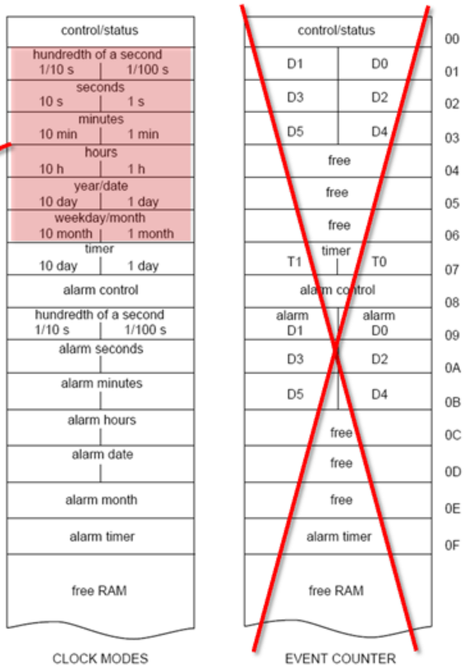
Binary-Coded Decimal.
There are two variants:
Unpacked and packed

Example

12d = 0x0C = 0000 1100b

Unpacked BCD: **0000 0001 0000 0010**
1 2

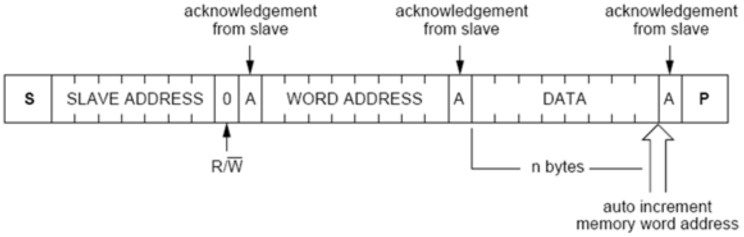
Packed BCD: **0001 0010**
1 2



CLOCK MODES	EVENT COUNTER	
control/status	~~control/status~~	00
hundredth of a second 1/10 s | 1/100 s	~~D1 | D0~~	01
seconds 10 s | 1 s	~~D3 | D2~~	02
minutes 10 min | 1 min	~~D5 | D4~~	03
hours 10 h | 1 h	~~free~~	04
year/date 10 day | 1 day	~~free~~	05
weekday/month 10 month | 1 month	~~free~~	06
timer 10 day | 1 day	~~timer T1 | T0~~	07
alarm control	~~alarm control~~	08
hundredth of a second 1/10 s | 1/100 s	~~alarm D1 | alarm D0~~	09
alarm seconds	~~D3 | D2~~	0A
alarm minutes	~~D5 | D4~~	0B
alarm hours	~~free~~	0C
alarm date	~~free~~	0D
alarm month	~~free~~	0E
alarm timer	~~alarm timer~~	0F
free RAM	~~free RAM~~	

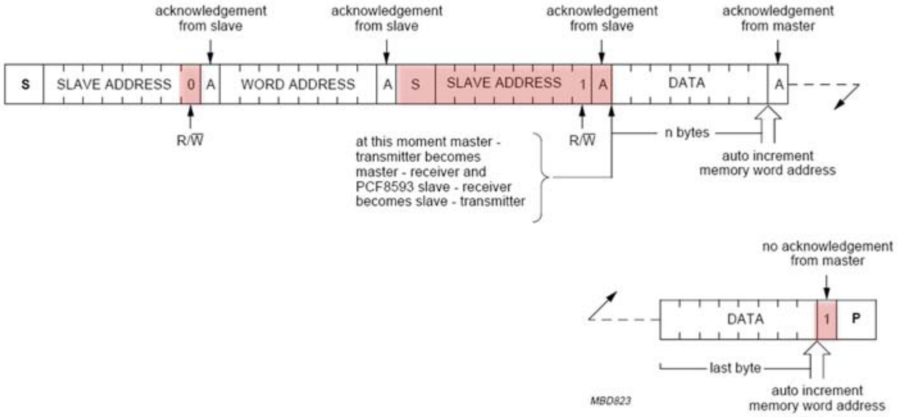
Example - PCF8583

Master transmits to slave receiver (WRITE) mode.


acknowledgement from slave
acknowledgement from slave
acknowledgement from slave
S
SLAVE ADDRESS
0
A
WORD ADDRESS
A
DATA
A
P
R/W̄
n bytes
auto increment memory word address

Example - PCF8583

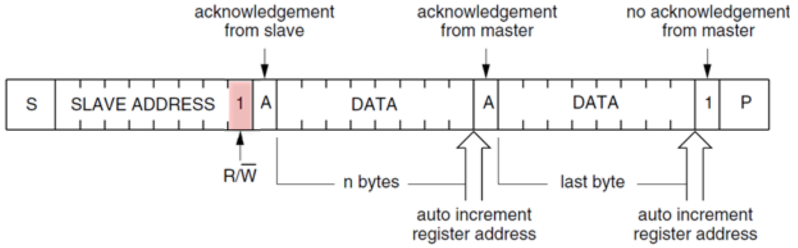
Master reads after setting word address (write word address; READ data).


acknowledgement from slave
acknowledgement from slave
acknowledgement from slave
acknowledgement from master
S
SLAVE ADDRESS
0
A
WORD ADDRESS
A
S
SLAVE ADDRESS
1
A
DATA
A
R/W̄
R/W̄
n bytes
auto increment memory word address
no acknowledgement from master
DATA
1
P
last byte
auto increment memory word address
MBD823

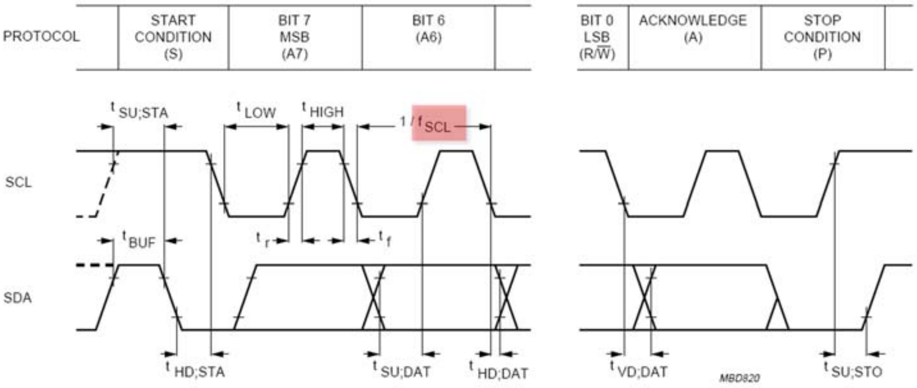
at this moment master - transmitter becomes master - receiver and PCF8593 slave - receiver becomes slave - transmitter

Example - PCF8583

Master reads slave immediately after first byte (READ mode).


acknowledgement from slave
acknowledgement from master
no acknowledgement from master
S
SLAVE ADDRESS
1
A
DATA
A
DATA
1
P
$R/\overline{W}$
n bytes
last byte
auto increment register address
auto increment register address

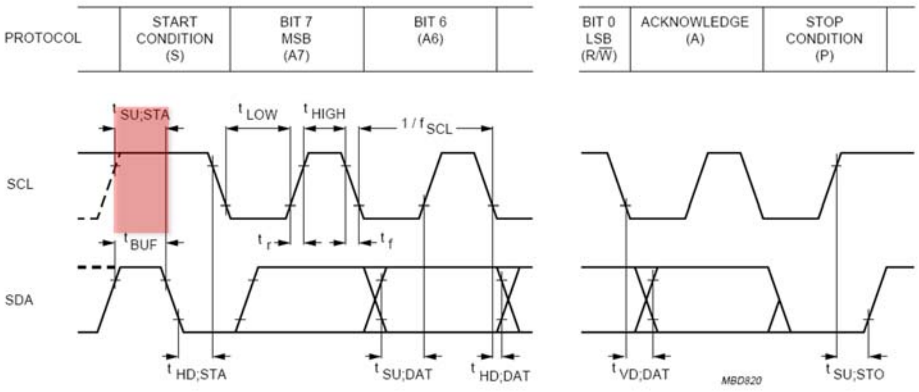
Example - PCF8583 - I2C Bus Timing



f_{scl} = 100 kHz

SCL clock frequency

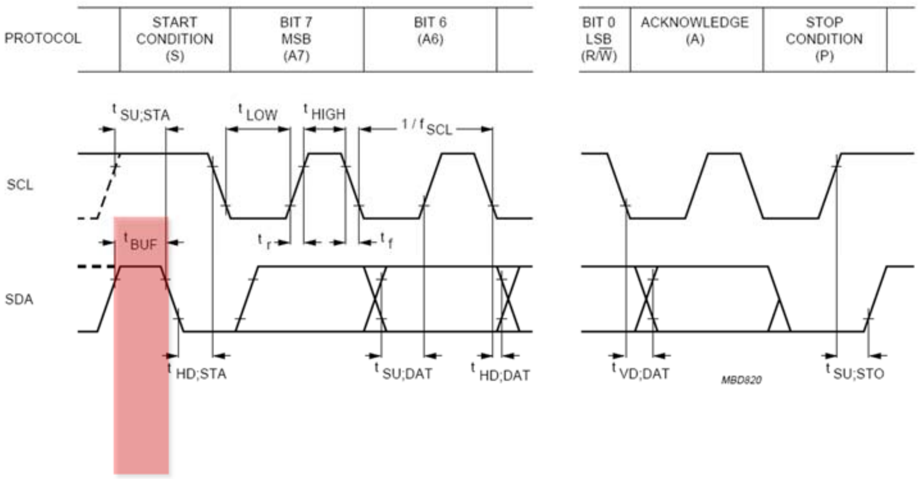
PCF8583 - I2C Bus Timing



$t_{SU;STA}$ = 4.7 μs min

START setup time

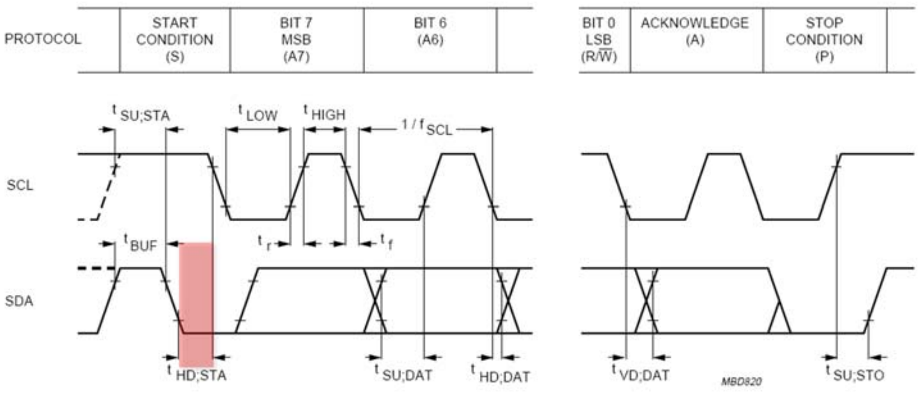
PCF8583 - I2C Bus Timing



t_{BUF} = 4.7 μs min

Bus free time

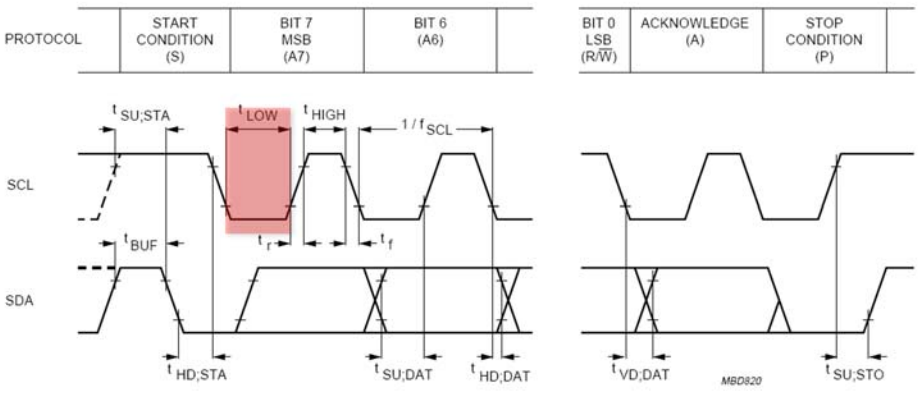
PCF8583 - I2C Bus Timing



$t_{HD;STA}$ = 4 μs min

START Hold Time

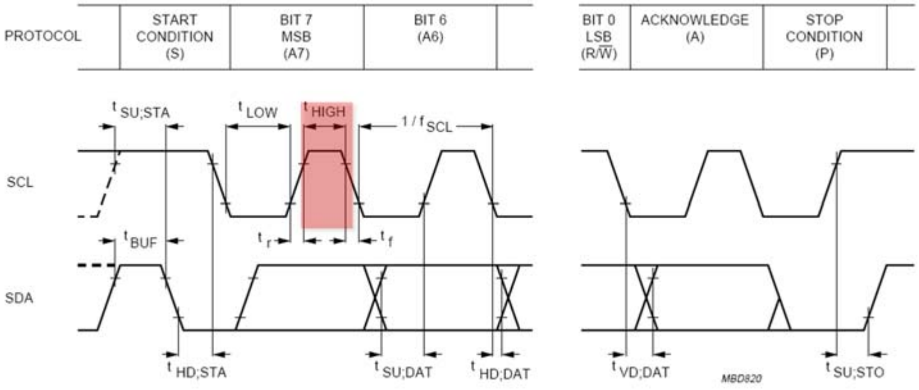
PCF8583 - I2C Bus Timing



t_{LOW} = 4.7 μs min

SCL LOW time

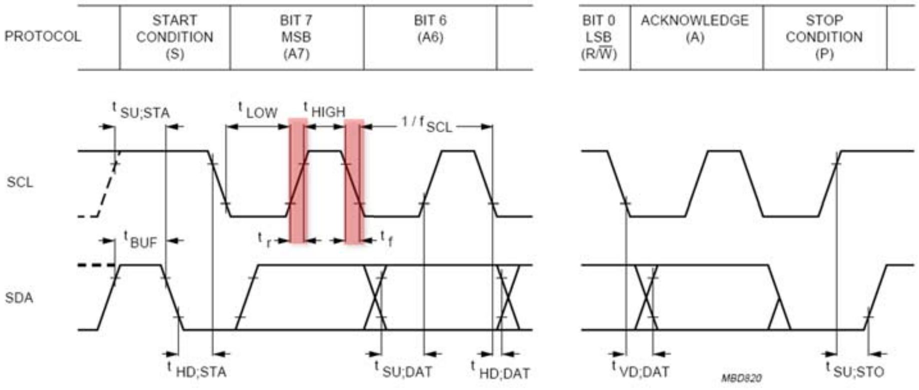
PCF8583 - I2C Bus Timing



t_{HIGH}= 4.7 μs min

SCL HIGH time

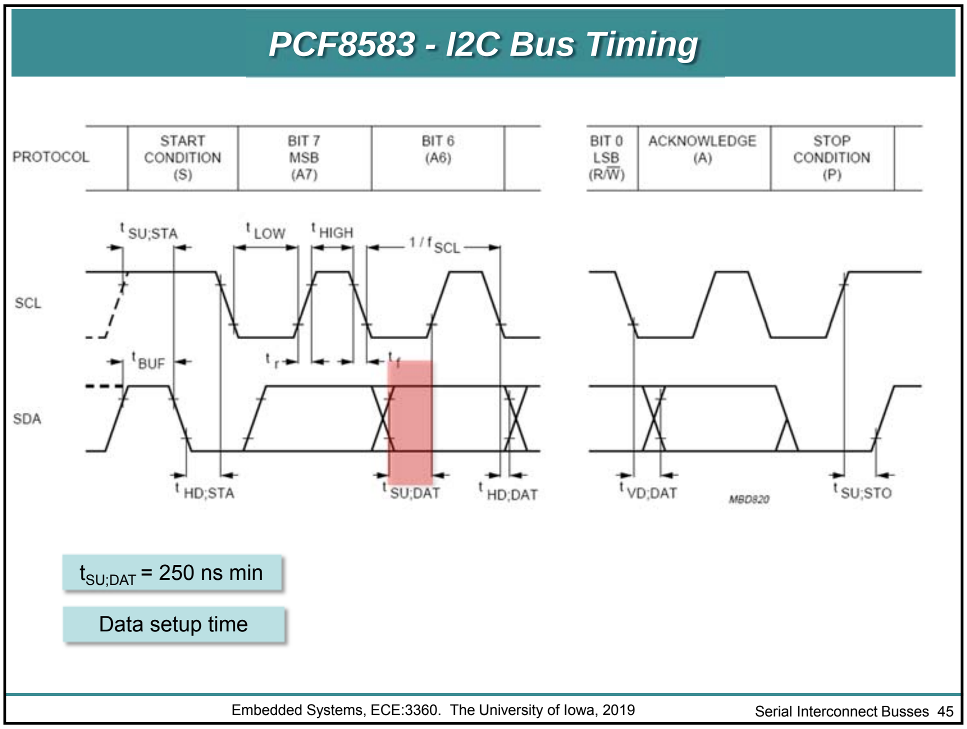
PCF8583 - I2C Bus Timing



t_r = 1.0 μs max

t_f = 0.3 μs max

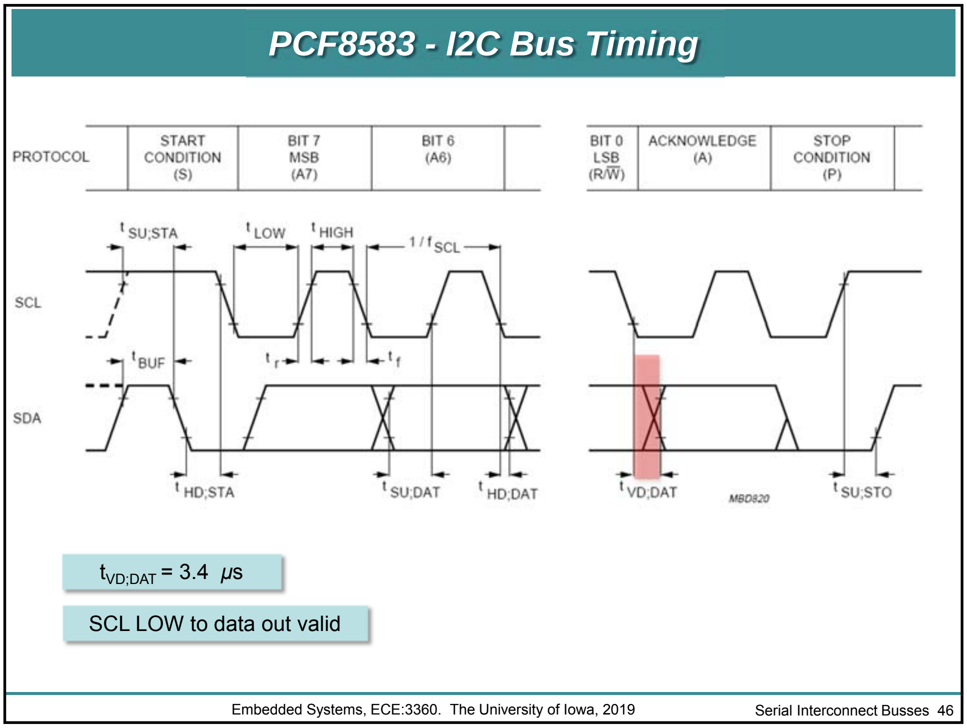
SDA and SCL rise and fall times



PCF8583 - I2C Bus Timing

$t_{SU;DAT}$ = 250 ns min

Data setup time

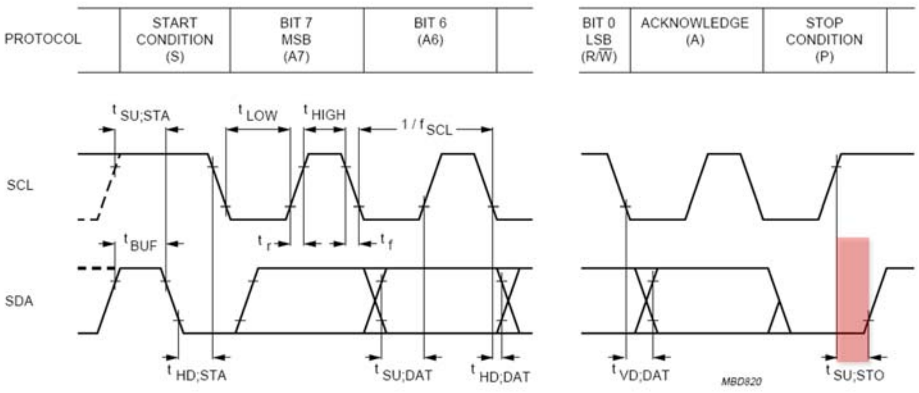


PCF8583 - I2C Bus Timing

$t_{VD;DAT}$ = 3.4 μs

SCL LOW to data out valid

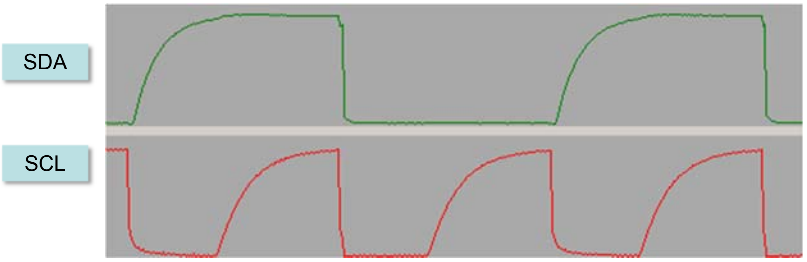
PCF8583 - I2C Bus Timing



$t_{SU;STO}$ = 3.4 μs

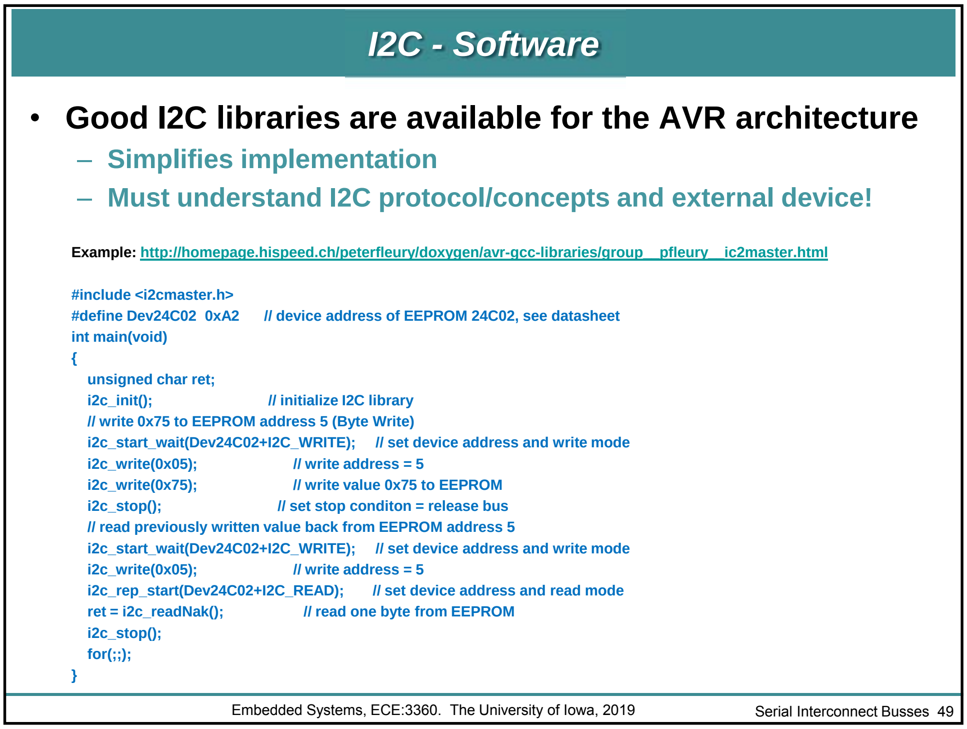
STOP setup time

Actual Bus Signals



SDA (above) and SCL (below) with R_p = 10 kΩ and C_p = 300 pF. The SCL clock runs at 100 kHz (nominal).

One can influence rise- and fall times with resistor values!

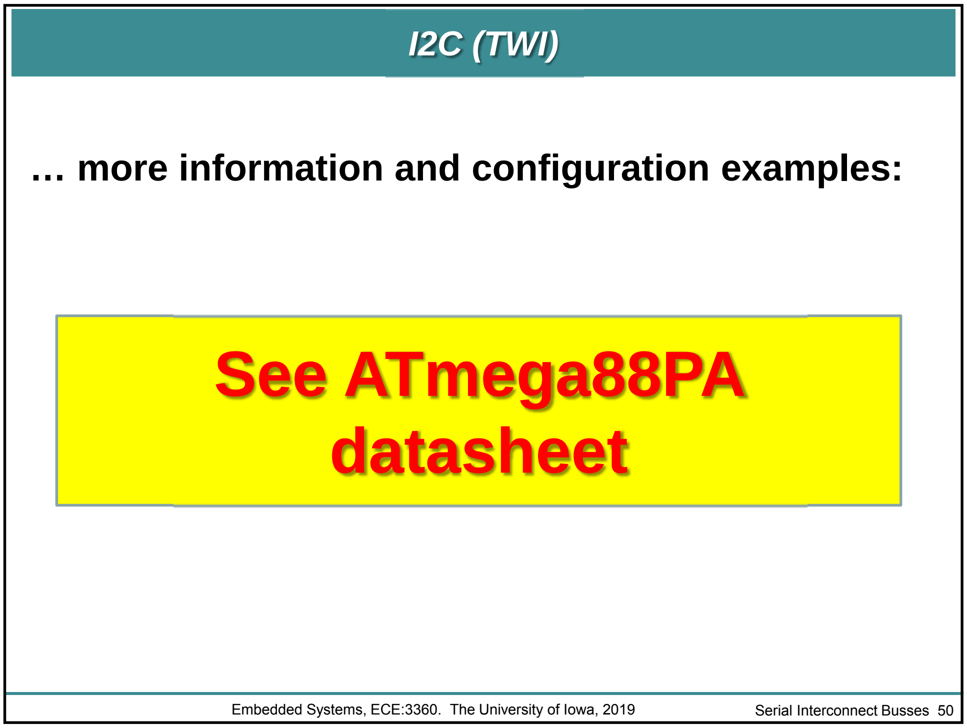


I2C - Software

- Good I2C libraries are available for the AVR architecture
  - Simplifies implementation
  - Must understand I2C protocol/concepts and external device!

Example: http://homepage.hispeed.ch/peterfleury/doxygen/avr-gcc-libraries/group__pfleury__ic2master.html

```
#include <i2cmaster.h>
#define Dev24C02  0xA2      // device address of EEPROM 24C02, see datasheet
int main(void)
{
   unsigned char ret;
   i2c_init();                          // initialize I2C library
   // write 0x75 to EEPROM address 5 (Byte Write)
   i2c_start_wait(Dev24C02+I2C_WRITE);     // set device address and write mode
   i2c_write(0x05);                        // write address = 5
   i2c_write(0x75);                        // write value 0x75 to EEPROM
   i2c_stop();                          // set stop conditon = release bus
   // read previously written value back from EEPROM address 5
   i2c_start_wait(Dev24C02+I2C_WRITE);     // set device address and write mode
   i2c_write(0x05);                        // write address = 5
   i2c_rep_start(Dev24C02+I2C_READ);       // set device address and read mode
   ret = i2c_readNak();                    // read one byte from EEPROM
   i2c_stop();
   for(;;);
}
```



I2C (TWI)

… more information and configuration examples:

See ATmega88PA datasheet