Embedded Systems and Software

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

Purpose of Serial Interconnect Buses

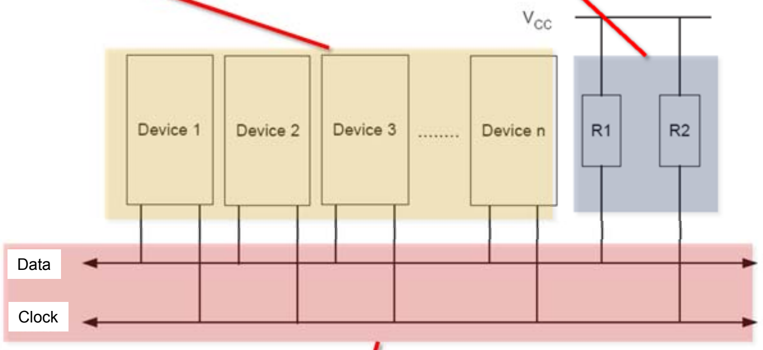
- Provide low-cost—i.e., low wire/pin count—connection between IC devices
- There are many serial bus "standards"
  - I2C (Inter-Integrated Circuit)
  - SMB (System Management Bus)
  - SPI (Serial Peripheral Interface)
  - Microwire
  - Maxim 3-wire
  - Maxim/Dallas 1-wire
  - CAN (controller area network)
  - 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.

I²C (Inter-IC)

- I²C, “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 I²C 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.**

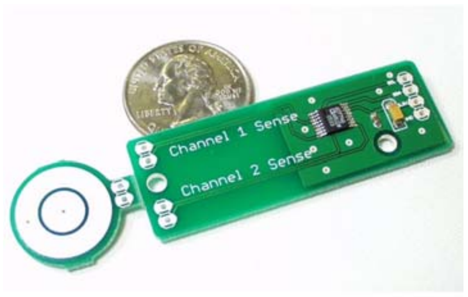
I2C Devices

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.

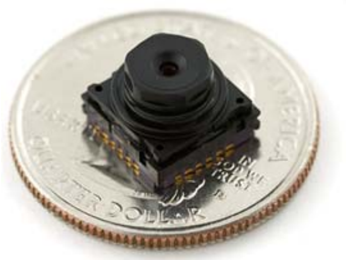
I2C Devices

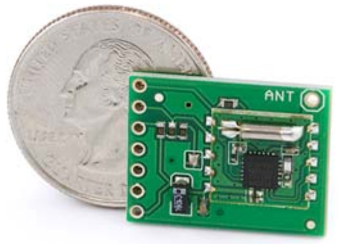


Honeywell HMC6352 Compass Module

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.

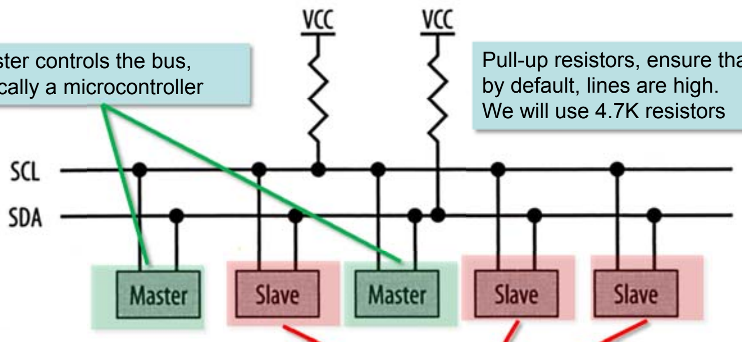
I2C

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

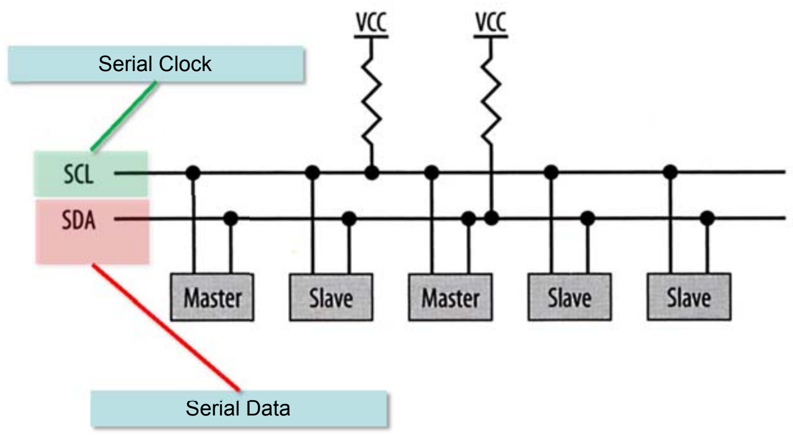


Master controls the bus, typically a microcontroller

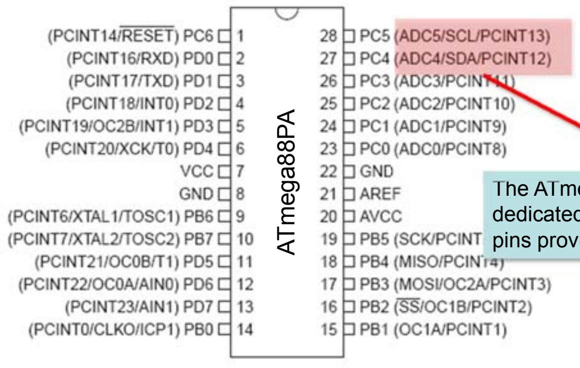
Pull-up resistors, ensure that by default, lines are high. We will use 4.7K resistors

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

I2C Structure



TWI Hardware on ATmega88PA



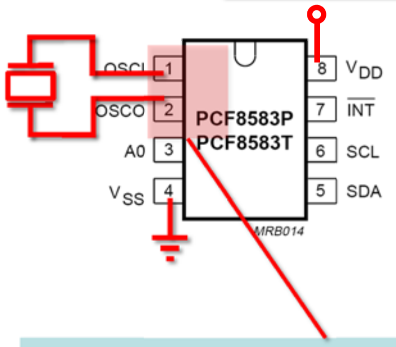
The ATmega88PA controller has dedicated TWI hardware. These pins provide access to the hardware

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



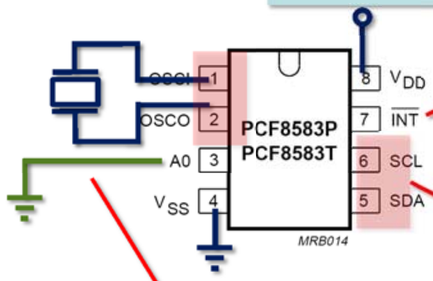
SYMBOL	PIN	DESCRIPTION
OSCI	1	oscillator input, 50 Hz or event-pulse input
OSCO	2	oscillator output
A0	3	address input
V_{SS}	4	negative supply
SDA	5	serial data line
SCL	6	serial clock line
$\overline{INT}$	7	open drain interrupt output (active LOW)
V_{DD}	8	positive supply

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



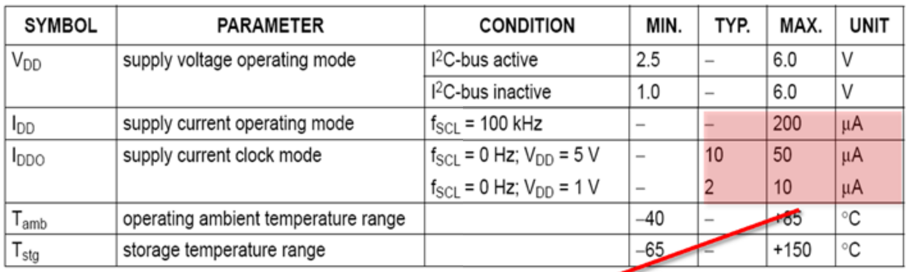
I2C Bus Lines

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

It can be "1" or "0", we choose "0"

Example - I2C RTC

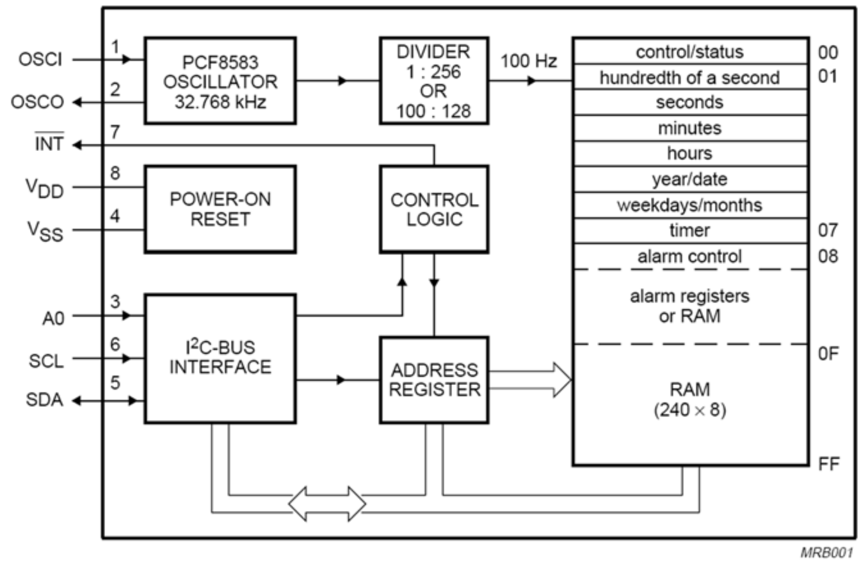
PCF8583 Clock/calendar with 240×8-bit RAM



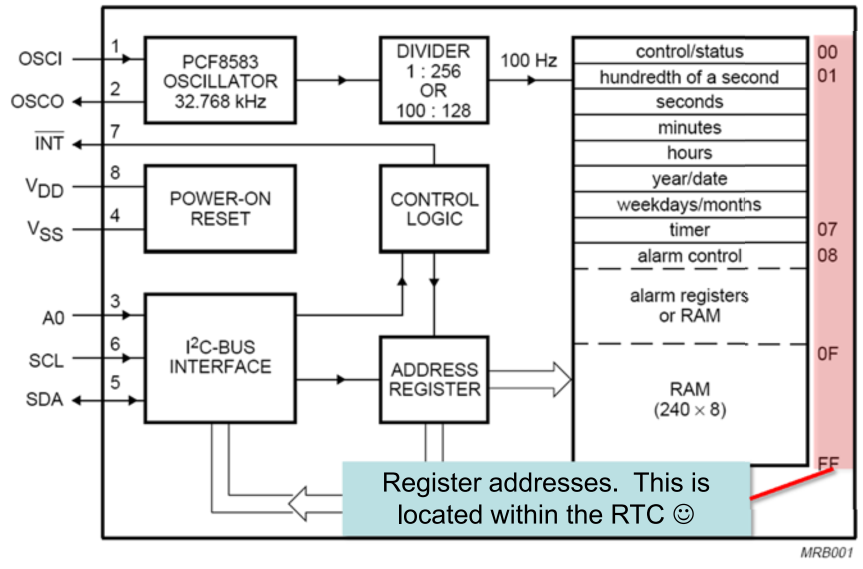
SYMBOL	PARAMETER	CONDITION	MIN.	TYP.	MAX.	UNIT
V_{DD}	supply voltage operating mode	I^2C-bus active	2.5	–	6.0	V
		I^2C-bus inactive	1.0	–	6.0	V
I_{DD}	supply current operating mode	f_{SCL} = 100 kHz	–	–	200	µA
I_{DDO}	supply current clock mode	f_{SCL} = 0 Hz; V_{DD} = 5 V	–	10	50	µA
		f_{SCL} = 0 Hz; 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

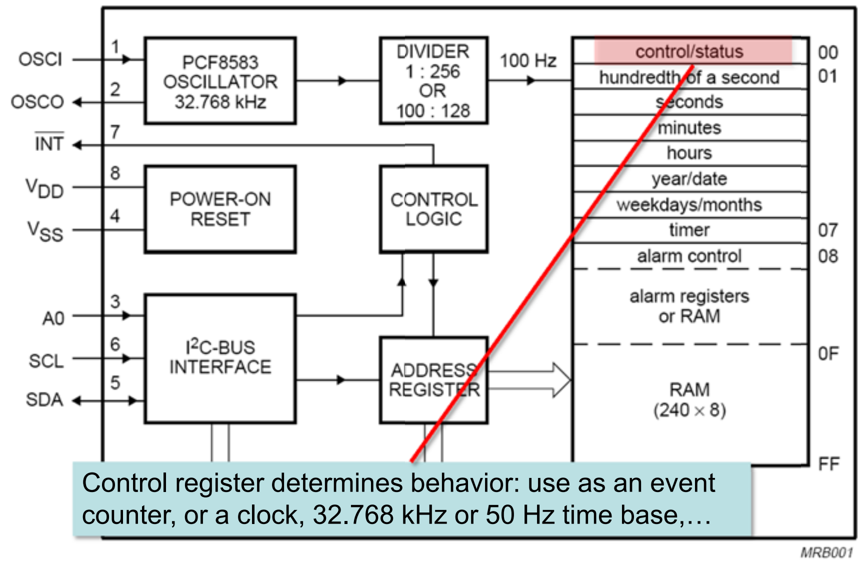


I2C RTC Used in Lab 6



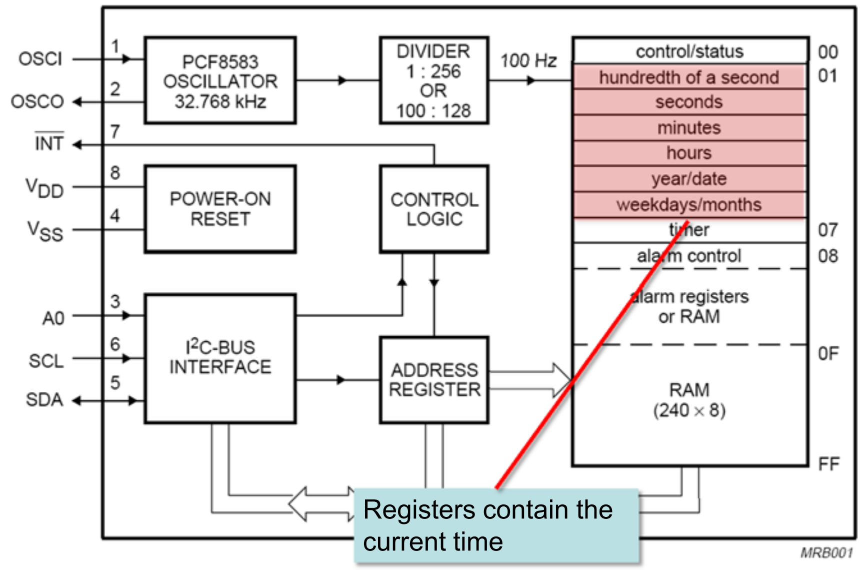
Register addresses. This is located within the RTC ☺

Example - I2C RTC



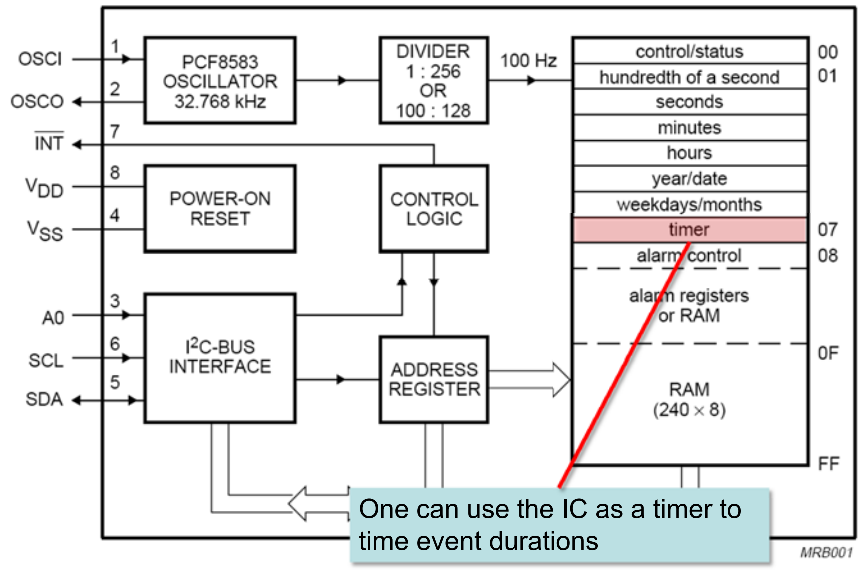
Control register determines behavior: use as an event counter, or a clock, 32.768 kHz or 50 Hz time base,…

Example - I2C RTC



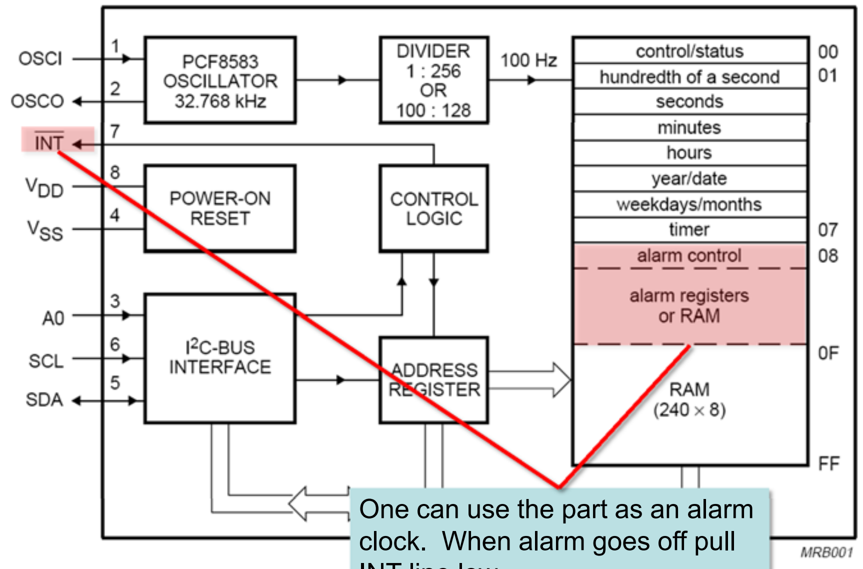
Registers contain the current time

Example - I2C RTC



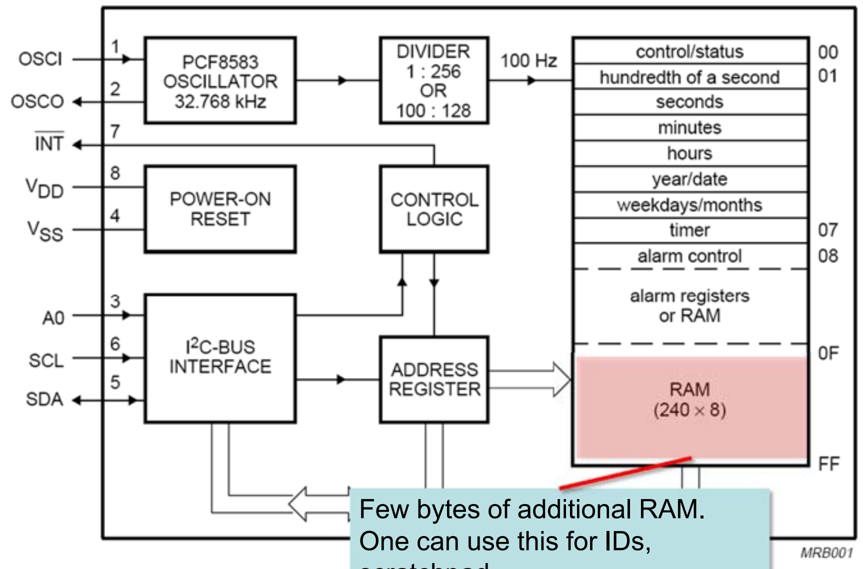
One can use the IC as a timer to time event durations

Example - I2C RTC



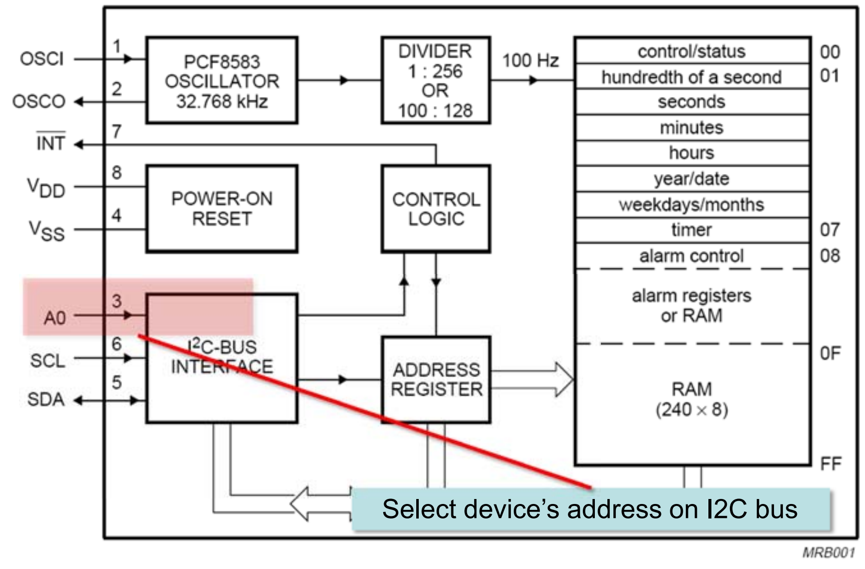
One can use the part as an alarm clock. When alarm goes off pull INT line low

Example - I2C RTC



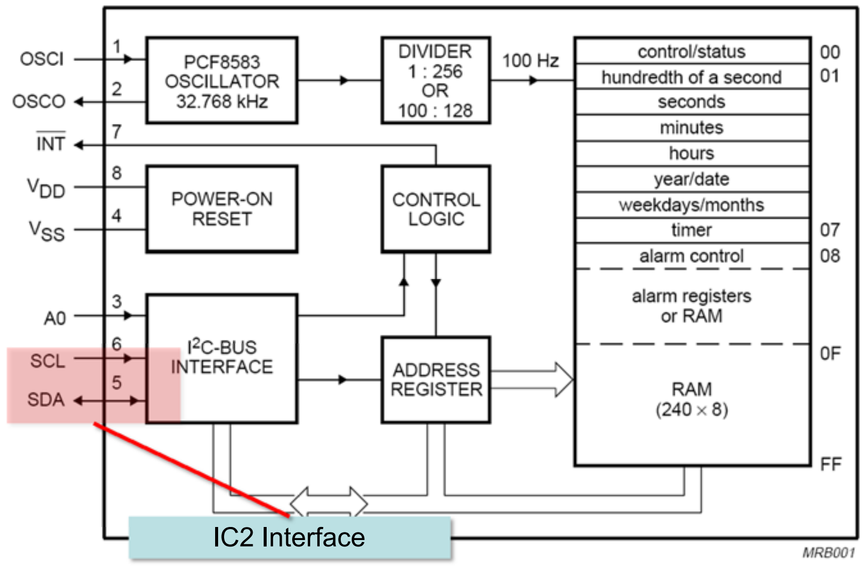
Few bytes of additional RAM. One can use this for IDs, scratchpad, …

Example - I2C RTC



Select device's address on I2C bus

Example - I2C RTC

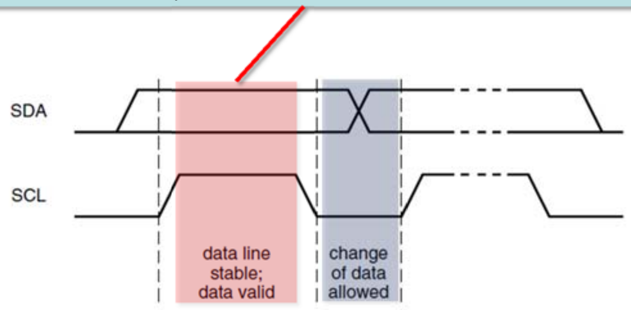


IC2 Interface

I2C Protocol

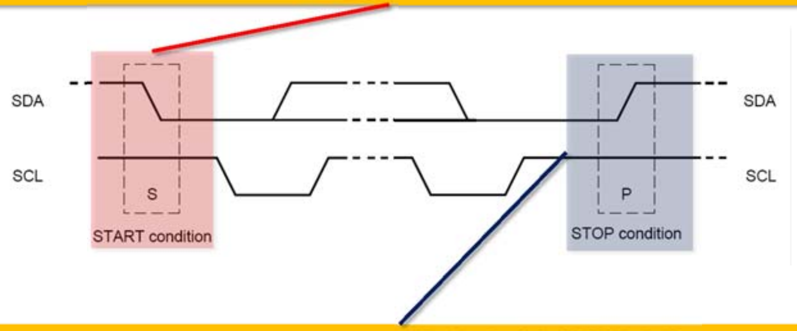
- The clock signal is always generated by the current bus master
- One exception: "clock stretching" by slave devices
  → e.g., force the clock low at times to delay the master sending more data

- 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**).

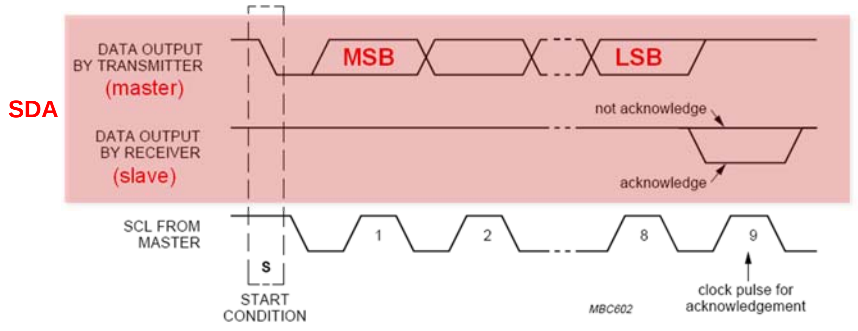


A LOW-to-HIGH transition of the data line while the clock is HIGH is defined as the stop condition (**P**).

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.

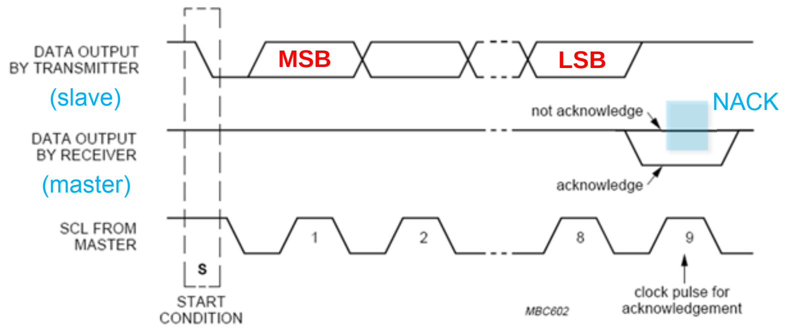


b) Master receiver (the master addresses a slave and receives data from it)

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

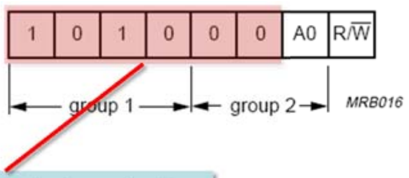
Acknowledgement on the I2C Bus

- 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

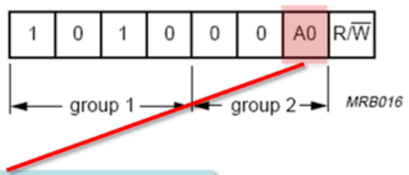
- Before any data is transmitted on the I2C-bus, the device which should respond is addressed first.
- The addressing is always carried out with the first byte transmitted after the start procedure.



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

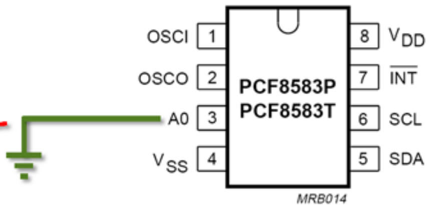
Addressing on the I2C Bus

- 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.



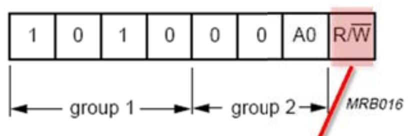
This part of the address is determined by the state of the IC's A0 pin

It can be "1" or "0", we choose "0"



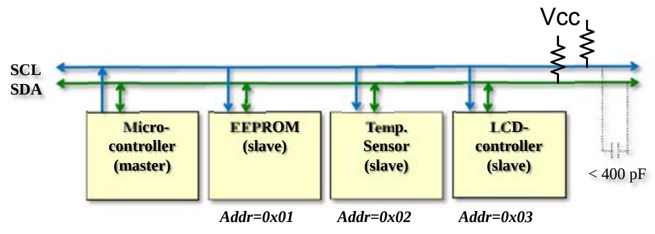
Addressing on the I2C Bus

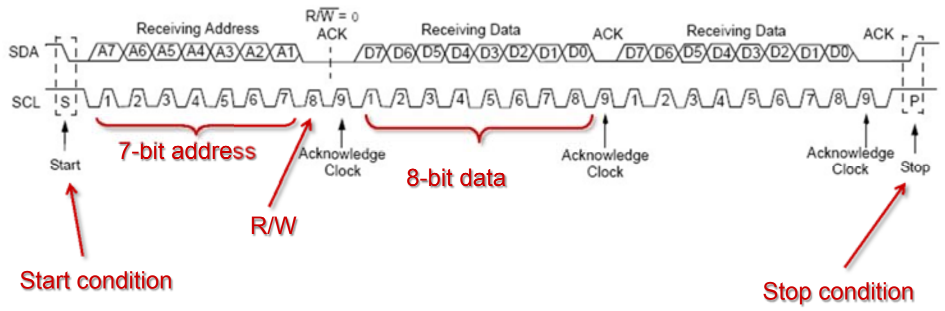
- Before any data is transmitted on the I2C-bus, the device which should respond is addressed first.
- The addressing is always carried out with the first byte transmitted after the start procedure.



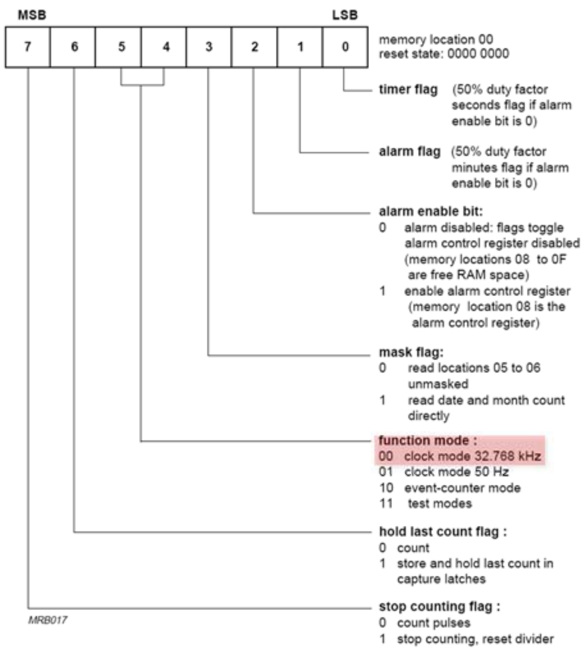
This bit determines if we are reading (= 1) from or writing to (= 0) to the devices

I2C Structure



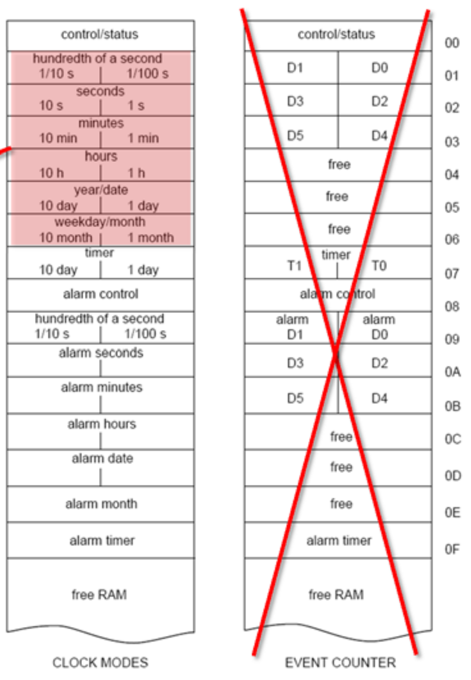


PCF8583 Control Status Register


MSB
LSB
7 6 5 4 3 2 1 0
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
MRB017

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**

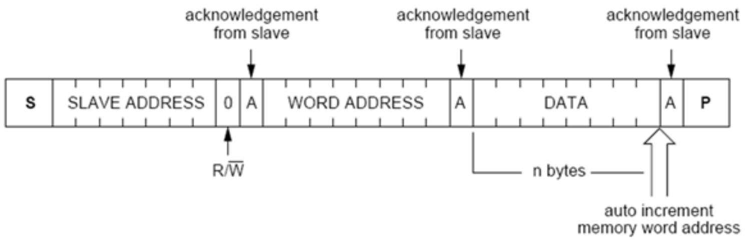
1 2

Packed BCD: **0001 0010**

1 2

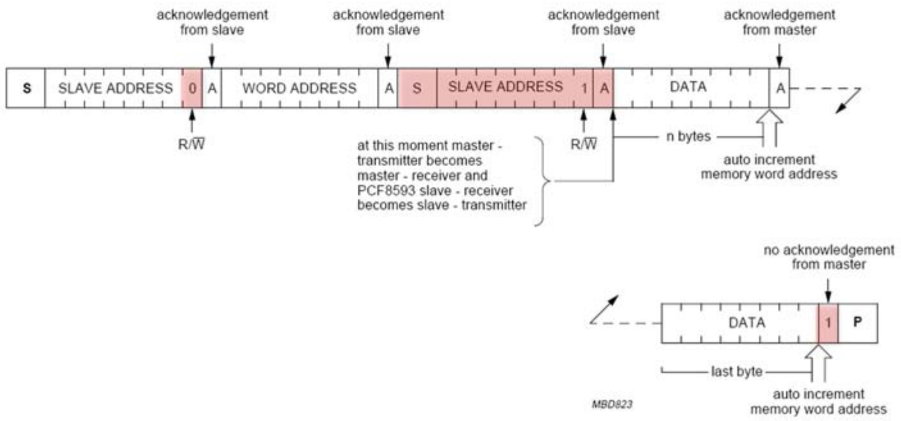
Example - PCF8583

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

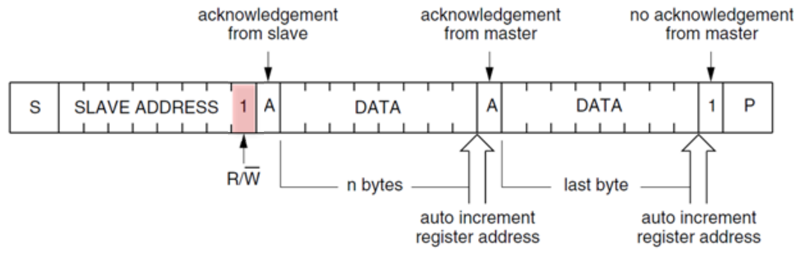
Example - PCF8583

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

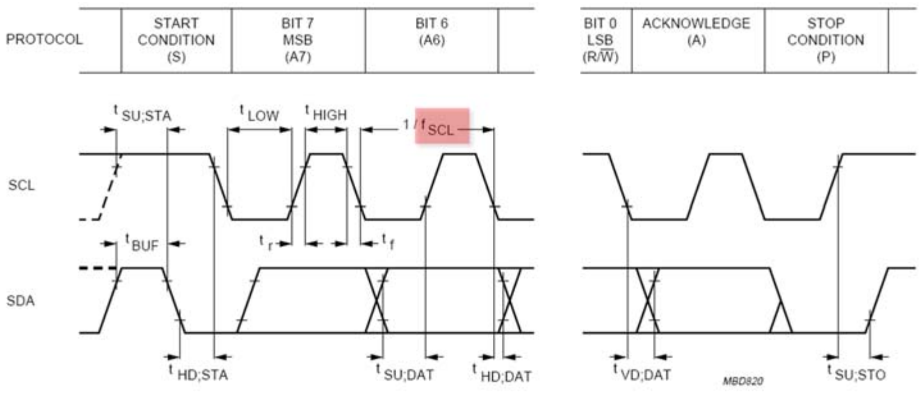

acknowledgement from slave
acknowledgement from slave
acknowledgement from slave
acknowledgement from master
S
SLAVE ADDRESS
0
A
WORD ADDRESS
A
S
SLAVE ADDRESS
1
A
DATA
A
R/W̄
R/W̄
at this moment master - transmitter becomes master - receiver and PCF8593 slave - receiver becomes slave - transmitter
n bytes
auto increment memory word address
no acknowledgement from master
DATA
1
P
last byte
auto increment memory word address
MBD823

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/W̄
n bytes
last byte
auto increment register address
auto increment register address

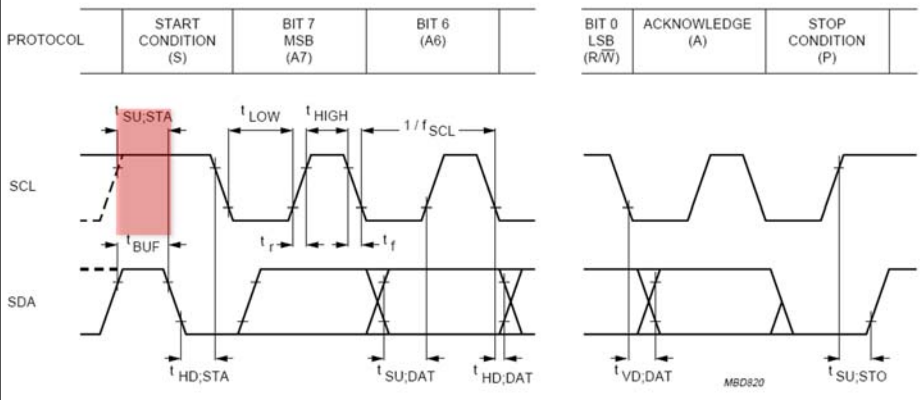
Example - PCF8583 - I2C Bus Timing



f_{scl} = 100 kHz

SCL clock frequency

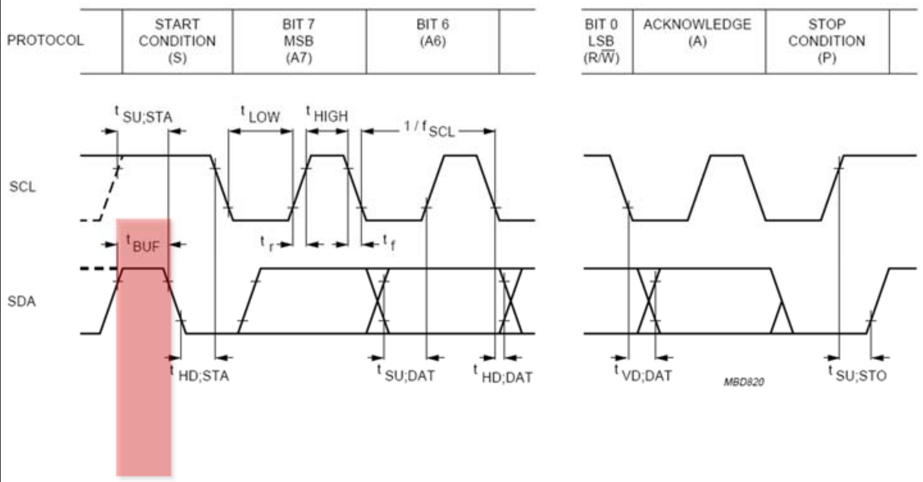
PCF8583 - I2C Bus Timing



$t_{SU;STA}$ = 4.7 μs min

START setup time

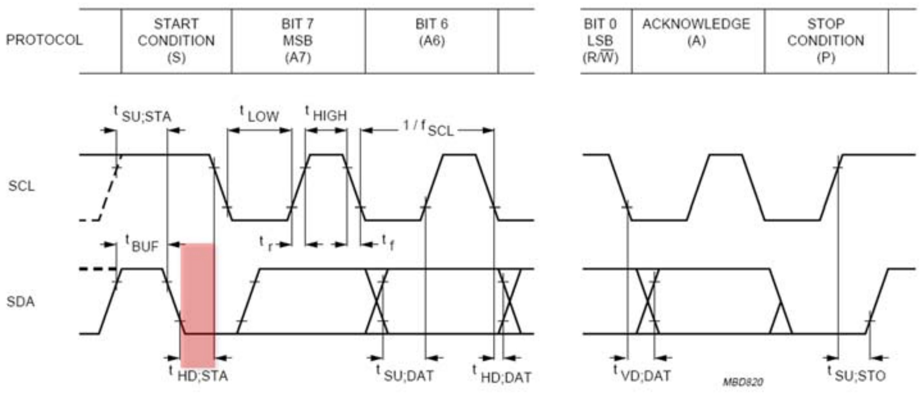
PCF8583 - I2C Bus Timing



t_{BUF} = 4.7 μs min

Bus free time

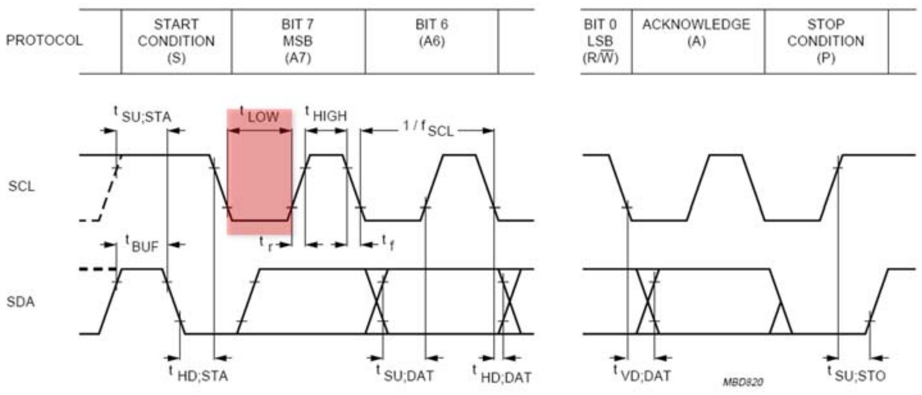
PCF8583 - I2C Bus Timing



$t_{HD;STA}$ = 4 μs min

START Hold Time

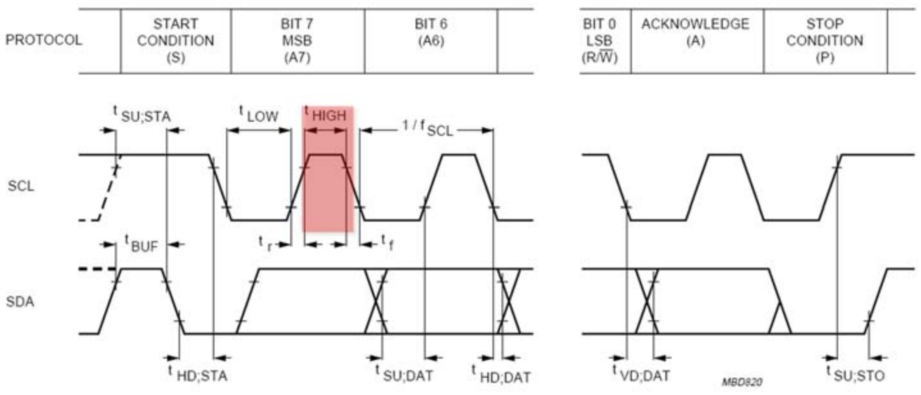
PCF8583 - I2C Bus Timing



t_{LOW} = 4.7 μs min

SCL LOW time

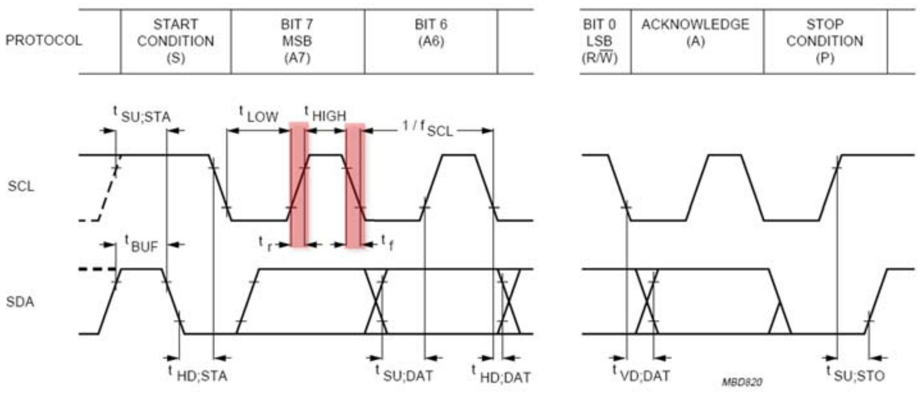
PCF8583 - I2C Bus Timing



t_{HIGH}= 4.7 μs min

SCL HIGH time

PCF8583 - I2C Bus Timing

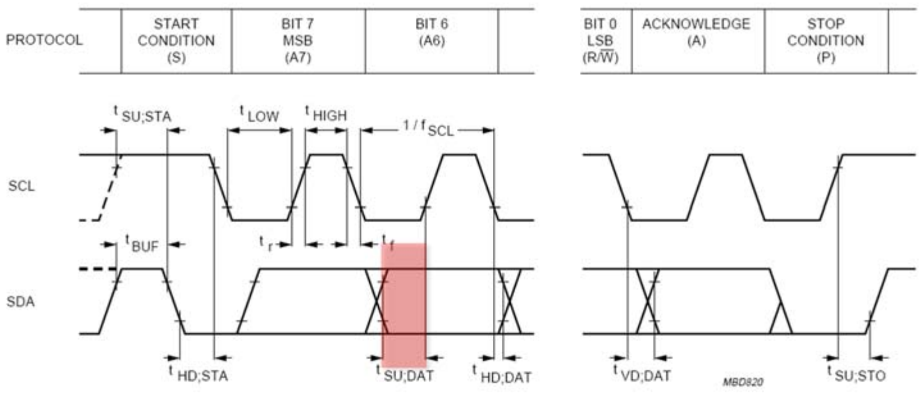


t_r = 1.0 μs max

t_f = 0.3 μs max

SDA and SCL rise and fall times

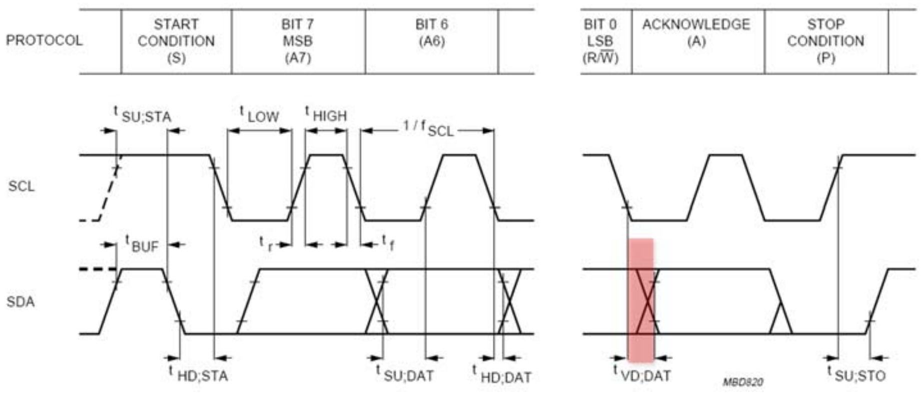
PCF8583 - I2C Bus Timing



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

Data setup time

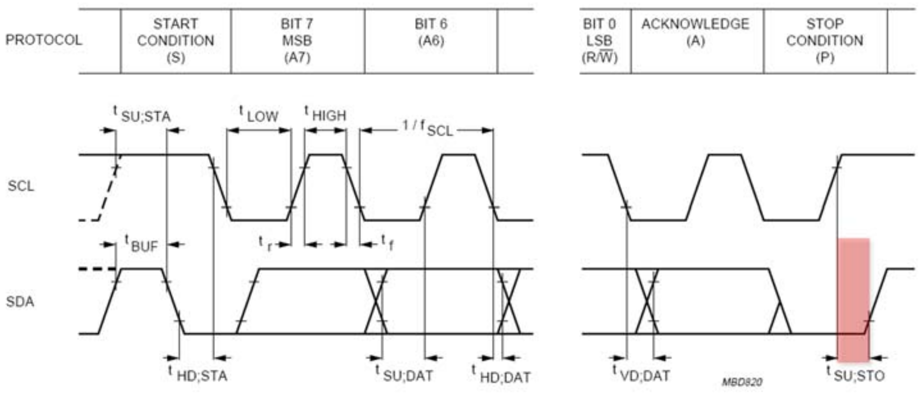
PCF8583 - I2C Bus Timing



$t_{VD;DAT}$ = 3.4 μs

SCL LOW to data out valid

PCF8583 - I2C Bus Timing



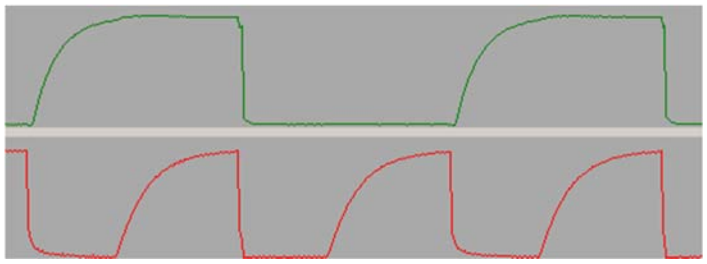
$t_{SU;STO}$ = 3.4 μs

STOP setup time

Actual Bus Signals

SDA

SCL



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

Serial Peripheral Interface (SPI)

Serial Peripheral Interface (SPI)

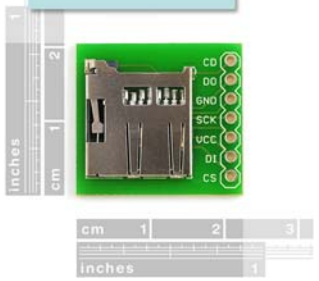
- **Originally developed by Motorola**
- **Synchronous, serial protocol**
  - **Data timing is controlled by an explicit clock signal (SCK)**
- **Master-slave**
  - **Master device controls the clock**
- **Bi-directional data exchange**
  - **Data clocked into and out-of device at same time**

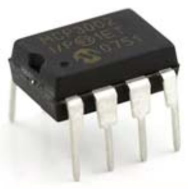
SPI Devices

A/D Converters

Barometric Pressure Sensor

SD Card





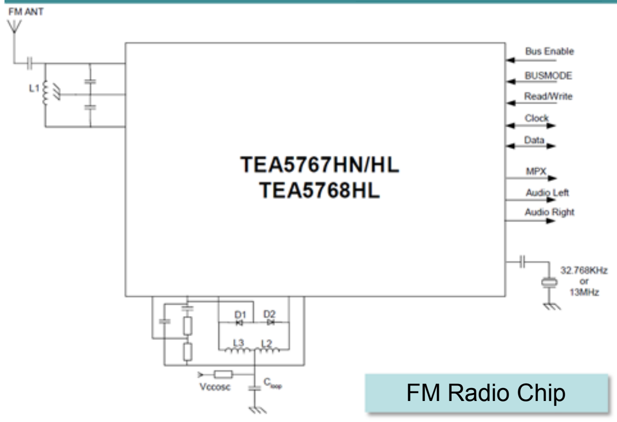
EEPROMs

SPI Devices

Compass

Ethernet

SPI Devices



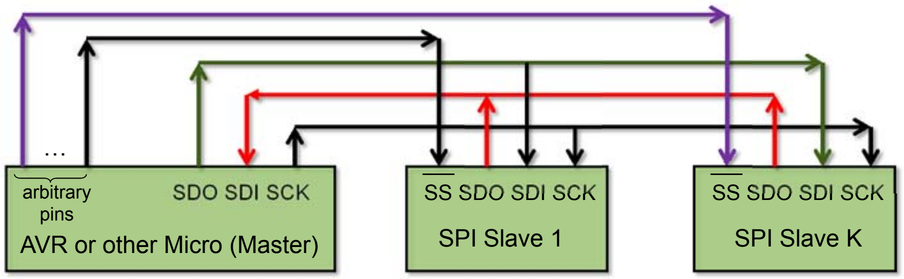
FM Radio Chip

FM Radio Module

SPI signals

- **$\overline{SS}$ ($\overline{CS}$) (Slave Select, Chip Select)**
  - **When SS is low the slave is enabled**
- **SCK (Serial Clock)**
  - **Controls the transfer of data**
- **SDO (Serial Data Out)**
  - **Carries data OUT of the device**
- **SDI (Serial Data In)**
  - **Carries data INTO the device**

Connecting Multiple SPI Devices


…
arbitrary pins
SDO SDI SCK
AVR or other Micro (Master)
$\overline{SS}$ SDO SDI SCK
SPI Slave 1
$\overline{SS}$ SDO SDI SCK
SPI Slave K

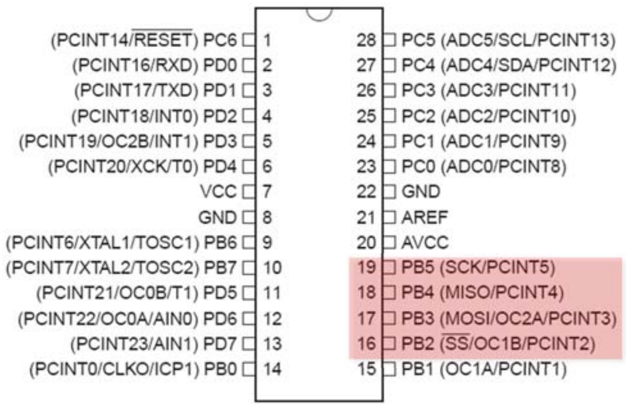
SPI on AVRs

AVR microcontrollers have built-in hardware support for SPI

AVRs can be bus masters (common), but one can also configure them to be bus slaves

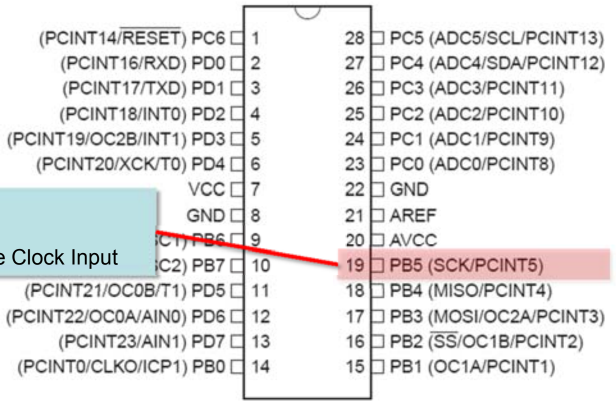
AVRs use SPI pins for ISP

What is ISP?



SPI on AVRs

AVR microcontrollers have built-in hardware support for SPI

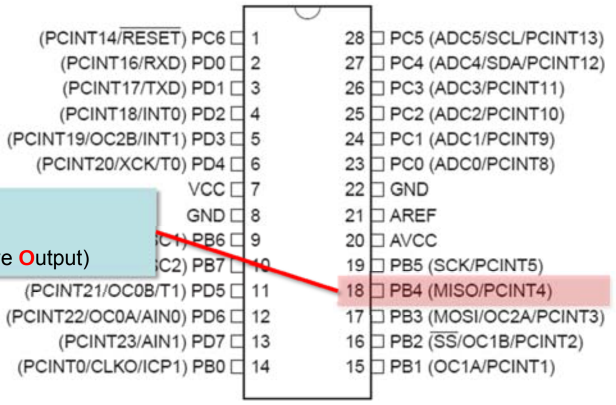


SCK

Master Clock Output, Slave Clock Input

SPI on AVRs

AVR microcontrollers have built-in hardware support for SPI

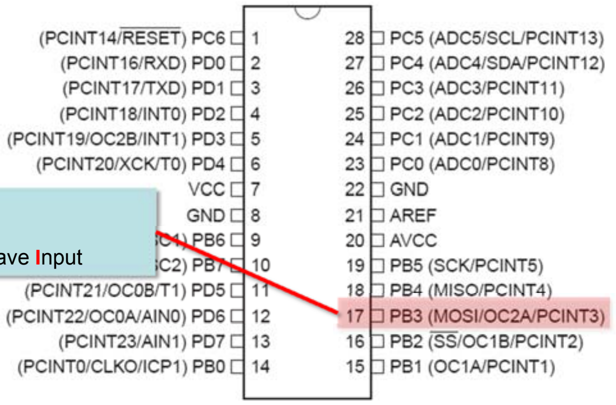


MISO

SPI Bus **M**aster **I**nput, **S**lave **O**utput)

SPI on AVRs

AVR microcontrollers have built-in hardware support for SPI

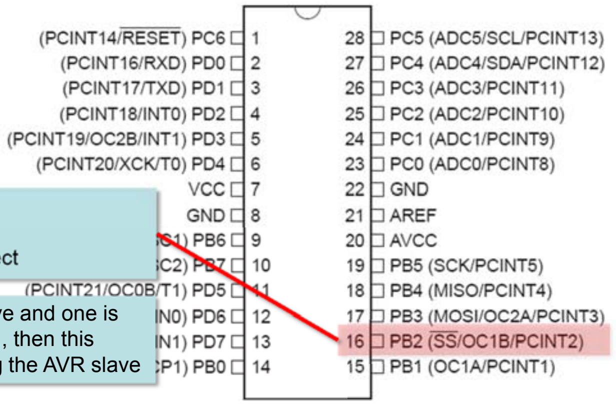


MOSI

SPI Bus **M**aster **O**utput, **S**lave **I**nput

SPI on AVRs

AVR microcontrollers have built-in hardware support for SPI

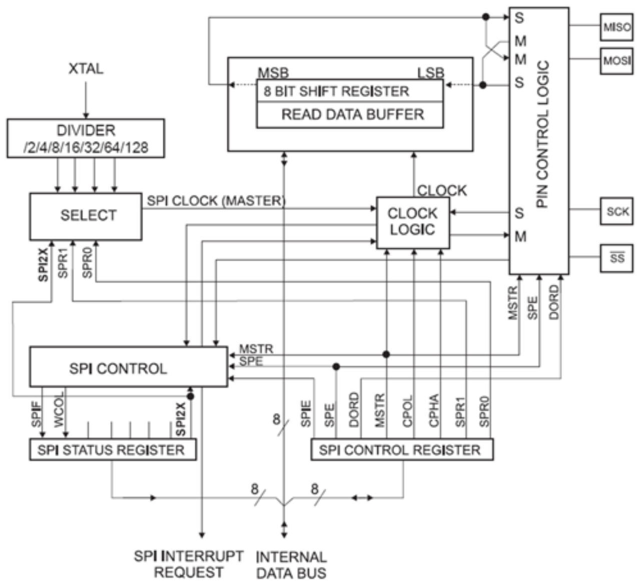


$\overline{\text{SS}}$

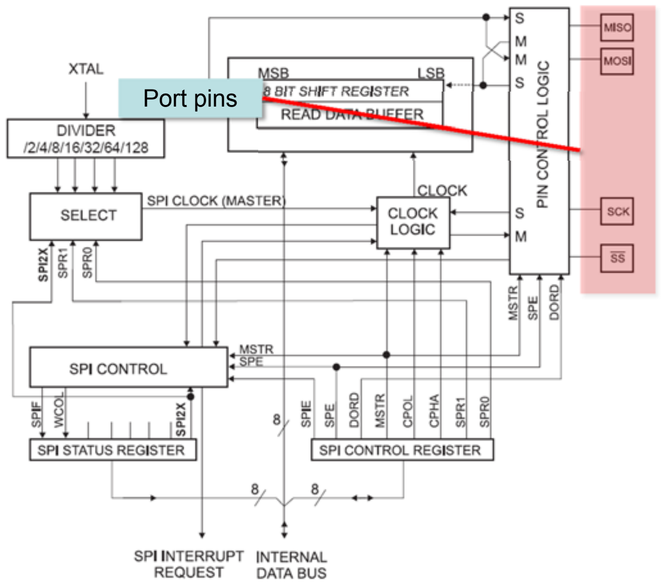
SPI Bus Master **S**lave **S**elect

If AVR is configured as slave and one is using SPI hardware for SPI, then this would be used for selecting the AVR slave

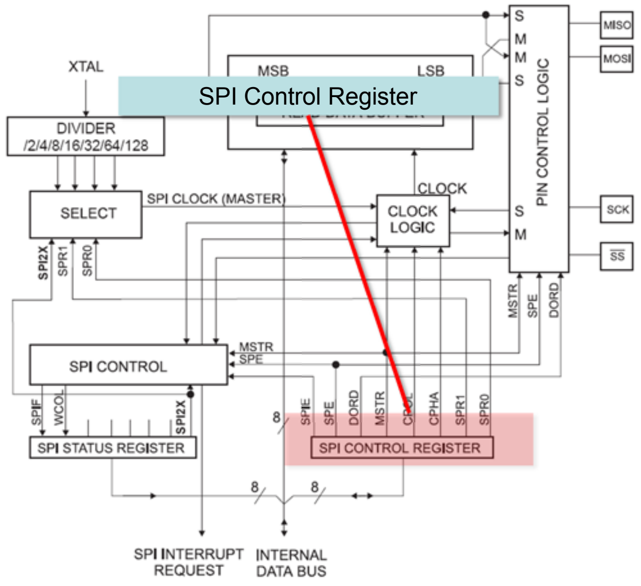
SPI on AVRs



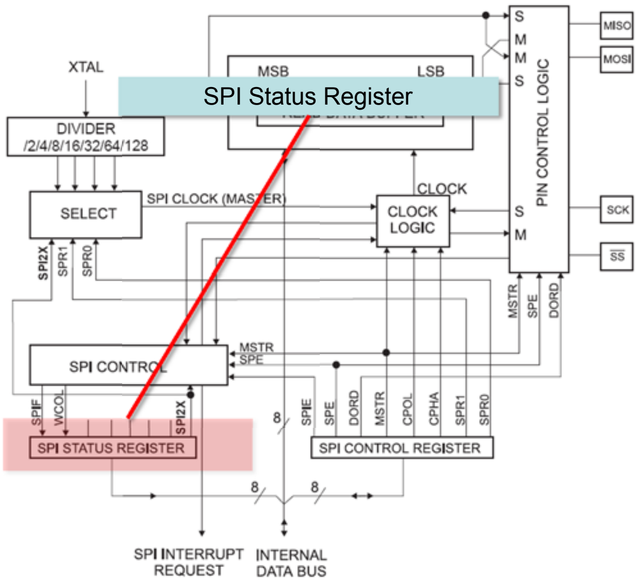
SPI on AVRs



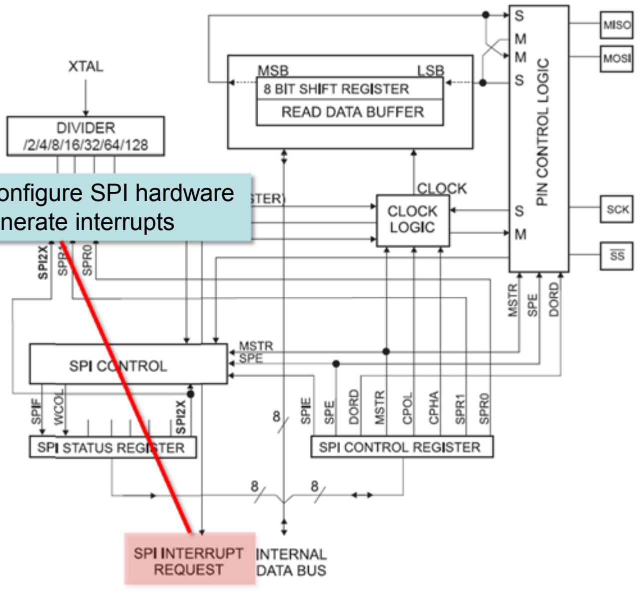
SPI on AVRs



SPI on AVRs

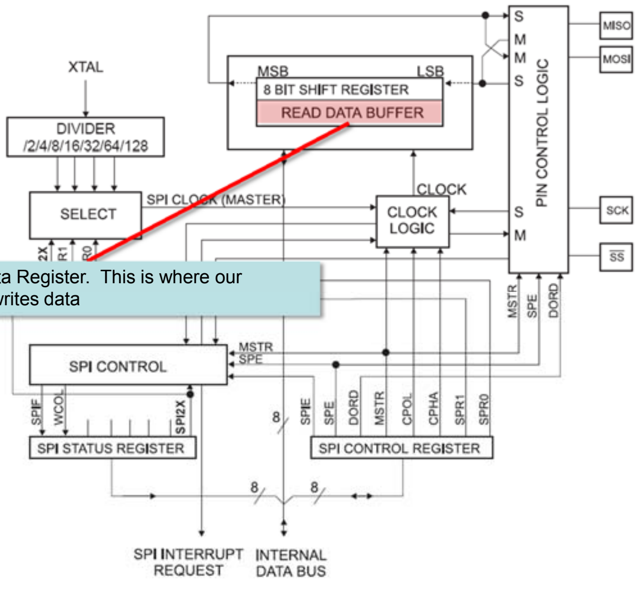


SPI on AVRs



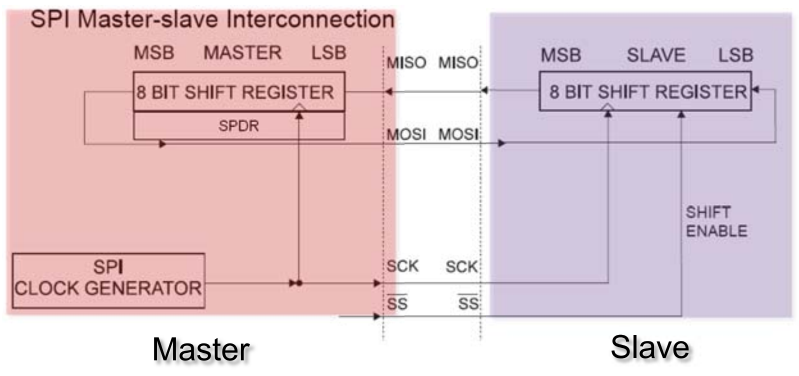
One can configure SPI hardware to generate interrupts

SPI on AVRs

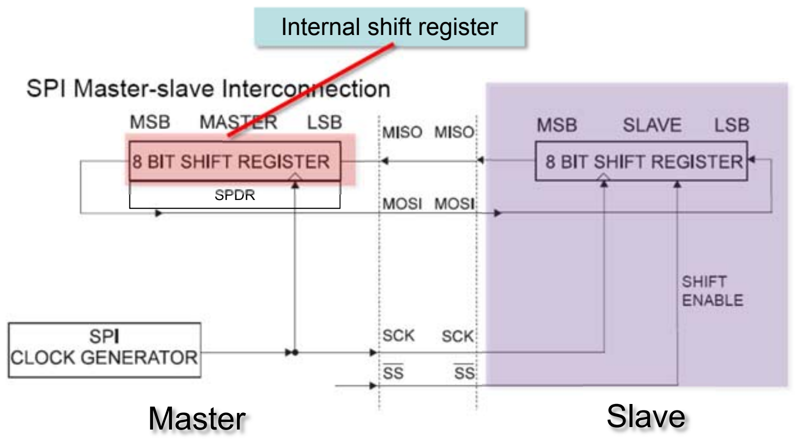


SPDR – SPI Data Register. This is where our program reads/writes data

SPI Data Loop

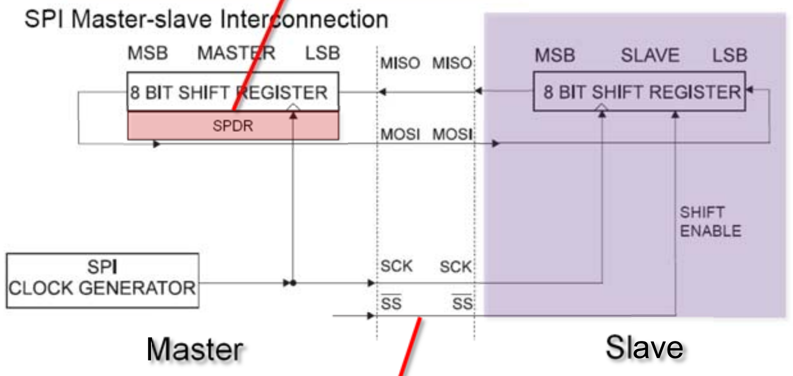


SPI Data Loop



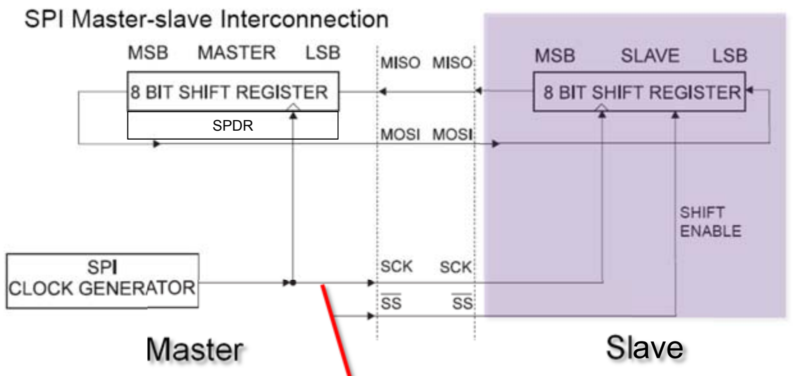
SPI Data Loop

SPDR – SPI Data Register. This is where our program reads/writes data



Master selects slave on bus to talk to

SPI Data Loop



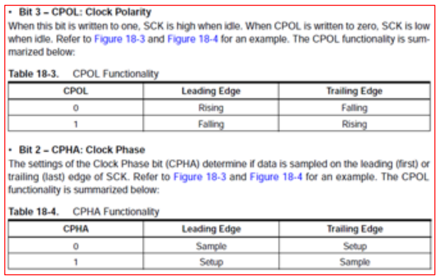
Master generates the clock that controls the data transfer

SPI Modes

SPI has several *modes* that determine when data is valid with respect to the clock

Table 18-2. SPI Modes

SPI Mode	Conditions	Leading Edge	Trailing eDge
0	CPOL=0, CPHA=0	Sample (Rising)	Setup (Falling)
1	CPOL=0, CPHA=1	Setup (Rising)	Sample (Falling)
2	CPOL=1, CPHA=0	Sample (Falling)	Setup (Rising)
3	CPOL=1, CPHA=1	Setup (Falling)	Sample (Rising)



- **Bit 3 – CPOL: Clock Polarity**

When this bit is written to one, SCK is high when idle. When CPOL is written to zero, SCK is low when idle. Refer to Figure 18-3 and Figure 18-4 for an example. The CPOL functionality is summarized below:

Table 18-3. CPOL Functionality

CPOL	Leading Edge	Trailing Edge
0	Rising	Falling
1	Falling	Rising

- **Bit 2 – CPHA: Clock Phase**

The settings of the Clock Phase bit (CPHA) determine if data is sampled on the leading (first) or trailing (last) edge of SCK. Refer to Figure 18-3 and Figure 18-4 for an example. The CPOL functionality is summarized below:

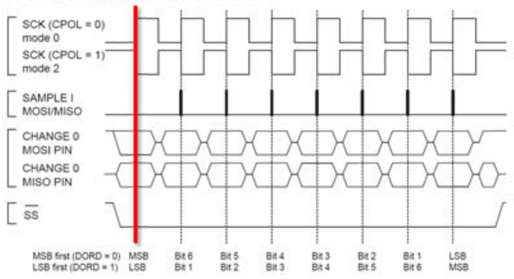
Table 18-4. CPHA Functionality

CPHA	Leading Edge	Trailing Edge
0	Sample	Setup
1	Setup	Sample

Caution: read SPI peripheral (i.e., RTC, sensor, …) datasheet carefully, and make sure your AVR uses the same mode…

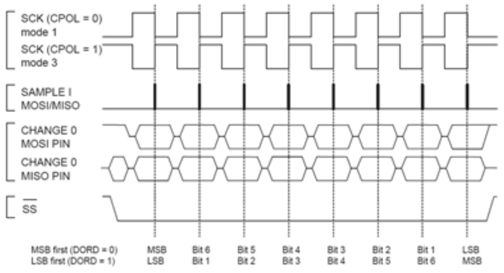
SPI Transfer Modes

Figure 18-3. SPI Transfer Format with CPHA = 0


SCK (CPOL = 0) mode 0
SCK (CPOL = 1) mode 2
SAMPLE I MOSI/MISO
CHANGE 0 MOSI PIN
CHANGE 0 MISO PIN
SS
MSB first (DORD = 0) MSB Bit 6 Bit 5 Bit 4 Bit 3 Bit 2 Bit 1 LSB
LSB first (DORD = 1) LSB Bit 1 Bit 2 Bit 3 Bit 4 Bit 5 Bit 6 MSB

SPI Transfer Format with CPHA = 0

Figure 18-4. SPI Transfer Format with CPHA = 1


SCK (CPOL = 0) mode 1
SCK (CPOL = 1) mode 3
SAMPLE I MOSI/MISO
CHANGE 0 MOSI PIN
CHANGE 0 MISO PIN
SS
MSB first (DORD = 0) MSB Bit 6 Bit 5 Bit 4 Bit 3 Bit 2 Bit 1 LSB
LSB first (DORD = 1) LSB Bit 1 Bit 2 Bit 3 Bit 4 Bit 5 Bit 6 MSB

SPI Transfer Format with CPHA = 1

SPI Transfer Modes

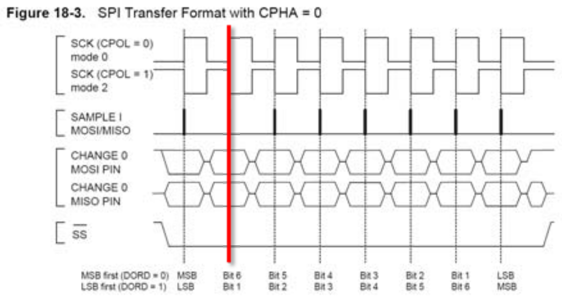


Figure 18-3. SPI Transfer Format with CPHA = 0
SCK (CPOL = 0) mode 0
SCK (CPOL = 1) mode 2
SAMPLE I MOSI/MISO
CHANGE 0 MOSI PIN
CHANGE 0 MISO PIN
SS
MSB first (DORD = 0) MSB Bit 6 Bit 5 Bit 4 Bit 3 Bit 2 Bit 1 LSB
LSB first (DORD = 1) LSB Bit 1 Bit 2 Bit 3 Bit 4 Bit 5 Bit 6 MSB

SPI Transfer Format with CPHA = 0

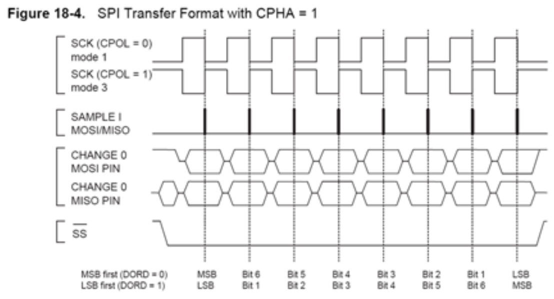


Figure 18-4. SPI Transfer Format with CPHA = 1
SCK (CPOL = 0) mode 1
SCK (CPOL = 1) mode 3
SAMPLE I MOSI/MISO
CHANGE 0 MOSI PIN
CHANGE 0 MISO PIN
SS
MSB first (DORD = 0) MSB Bit 6 Bit 5 Bit 4 Bit 3 Bit 2 Bit 1 LSB
LSB first (DORD = 1) LSB Bit 1 Bit 2 Bit 3 Bit 4 Bit 5 Bit 6 MSB

SPI Transfer Format with CPHA = 1

SPI Transfer Modes

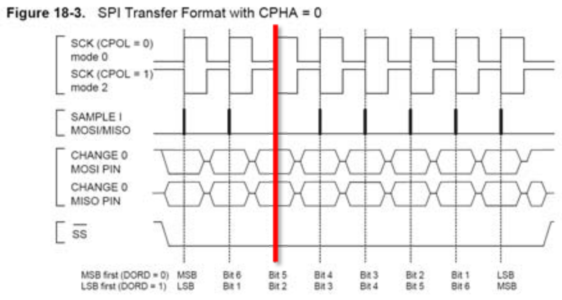


Figure 18-3. SPI Transfer Format with CPHA = 0
SCK (CPOL = 0) mode 0
SCK (CPOL = 1) mode 2
SAMPLE I MOSI/MISO
CHANGE 0 MOSI PIN
CHANGE 0 MISO PIN
SS
MSB first (DORD = 0) MSB Bit 6 Bit 5 Bit 4 Bit 3 Bit 2 Bit 1 LSB
LSB first (DORD = 1) LSB Bit 1 Bit 2 Bit 3 Bit 4 Bit 5 Bit 6 MSB

SPI Transfer Format with CPHA = 0

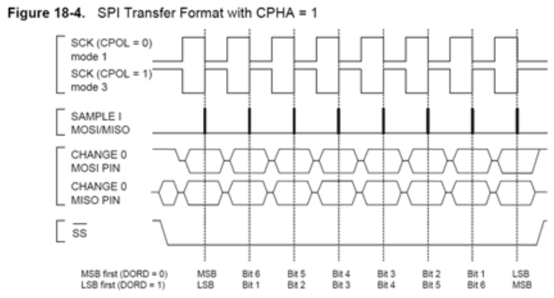
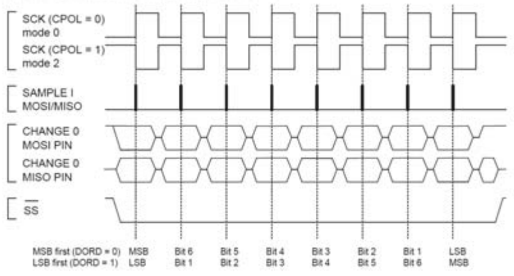


Figure 18-4. SPI Transfer Format with CPHA = 1
SCK (CPOL = 0) mode 1
SCK (CPOL = 1) mode 3
SAMPLE I MOSI/MISO
CHANGE 0 MOSI PIN
CHANGE 0 MISO PIN
SS
MSB first (DORD = 0) MSB Bit 6 Bit 5 Bit 4 Bit 3 Bit 2 Bit 1 LSB
LSB first (DORD = 1) LSB Bit 1 Bit 2 Bit 3 Bit 4 Bit 5 Bit 6 MSB

SPI Transfer Format with CPHA = 1

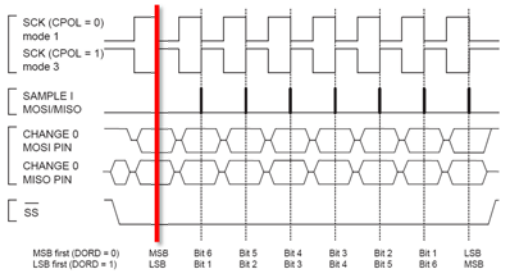
SPI Transfer Modes

Figure 18-3. SPI Transfer Format with CPHA = 0


SCK (CPOL = 0) mode 0
SCK (CPOL = 1) mode 2
SAMPLE I MOSI/MISO
CHANGE 0 MOSI PIN
CHANGE 0 MISO PIN
SS
MSB first (DORD = 0) MSB Bit 6 Bit 5 Bit 4 Bit 3 Bit 2 Bit 1 LSB
LSB first (DORD = 1) LSB Bit 1 Bit 2 Bit 3 Bit 4 Bit 5 Bit 6 MSB

SPI Transfer Format with CPHA = 0

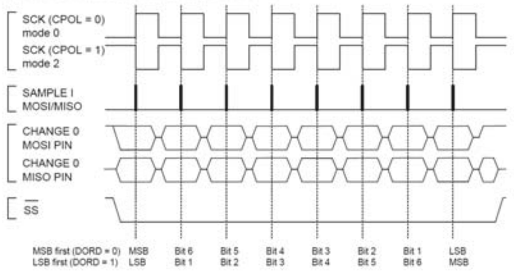
Figure 18-4. SPI Transfer Format with CPHA = 1


SCK (CPOL = 0) mode 1
SCK (CPOL = 1) mode 3
SAMPLE I MOSI/MISO
CHANGE 0 MOSI PIN
CHANGE 0 MISO PIN
SS
MSB first (DORD = 0) MSB Bit 6 Bit 5 Bit 4 Bit 3 Bit 2 Bit 1 LSB
LSB first (DORD = 1) LSB Bit 1 Bit 2 Bit 3 Bit 4 Bit 5 Bit 6 MSB

SPI Transfer Format with CPHA = 1

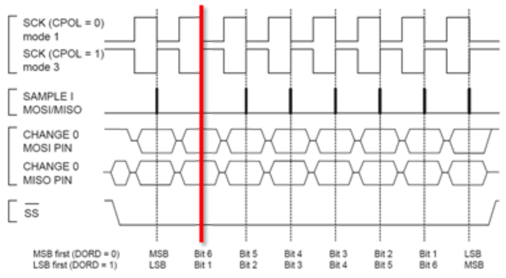
SPI Transfer Modes

Figure 18-3. SPI Transfer Format with CPHA = 0


SCK (CPOL = 0) mode 0
SCK (CPOL = 1) mode 2
SAMPLE I MOSI/MISO
CHANGE 0 MOSI PIN
CHANGE 0 MISO PIN
SS
MSB first (DORD = 0) MSB Bit 6 Bit 5 Bit 4 Bit 3 Bit 2 Bit 1 LSB
LSB first (DORD = 1) LSB Bit 1 Bit 2 Bit 3 Bit 4 Bit 5 Bit 6 MSB

SPI Transfer Format with CPHA = 0

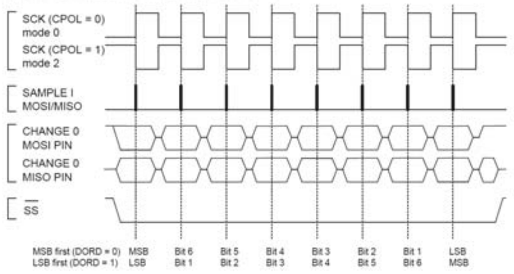
Figure 18-4. SPI Transfer Format with CPHA = 1


SCK (CPOL = 0) mode 1
SCK (CPOL = 1) mode 3
SAMPLE I MOSI/MISO
CHANGE 0 MOSI PIN
CHANGE 0 MISO PIN
SS
MSB first (DORD = 0) MSB Bit 6 Bit 5 Bit 4 Bit 3 Bit 2 Bit 1 LSB
LSB first (DORD = 1) LSB Bit 1 Bit 2 Bit 3 Bit 4 Bit 5 Bit 6 MSB

SPI Transfer Format with CPHA = 1

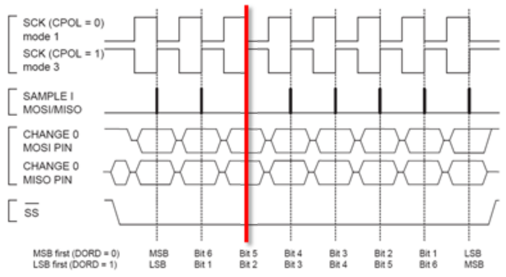
SPI Transfer Modes

Figure 18-3. SPI Transfer Format with CPHA = 0


SCK (CPOL = 0) mode 0
SCK (CPOL = 1) mode 2
SAMPLE I MOSI/MISO
CHANGE 0 MOSI PIN
CHANGE 0 MISO PIN
SS
MSB first (DORD = 0) MSB Bit 6 Bit 5 Bit 4 Bit 3 Bit 2 Bit 1 LSB
LSB first (DORD = 1) LSB Bit 1 Bit 2 Bit 3 Bit 4 Bit 5 Bit 6 MSB

SPI Transfer Format with CPHA = 0

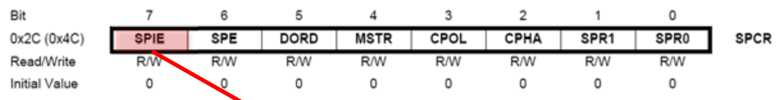
Figure 18-4. SPI Transfer Format with CPHA = 1


SCK (CPOL = 0) mode 1
SCK (CPOL = 1) mode 3
SAMPLE I MOSI/MISO
CHANGE 0 MOSI PIN
CHANGE 0 MISO PIN
SS
MSB first (DORD = 0) MSB Bit 6 Bit 5 Bit 4 Bit 3 Bit 2 Bit 1 LSB
LSB first (DORD = 1) LSB Bit 1 Bit 2 Bit 3 Bit 4 Bit 5 Bit 6 MSB

SPI Transfer Format with CPHA = 1

Configuring SPI

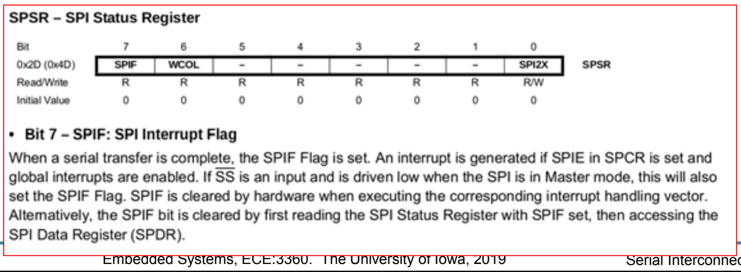
SPCR – SPI Control Register



Bit	7	6	5	4	3	2	1	0	
0x2C (0x4C)	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	SPCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

SPIE: SPI Interrupt Enable

This bit causes the SPI interrupt to be executed if **SPIF** bit in the **SPSR** register is set and the if the Global Interrupt Enable bit in **SREG** is set.



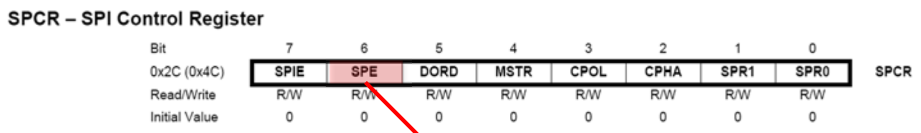
SPSR – SPI Status Register

Bit	7	6	5	4	3	2	1	0	
0x2D (0x4D)	SPIF	WCOL	–	–	–	–	–	SPI2X	SPSR
Read/Write	R	R	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – SPIF: SPI Interrupt Flag**

When a serial transfer is complete, the SPIF Flag is set. An interrupt is generated if SPIE in SPCR is set and global interrupts are enabled. If $\overline{SS}$ is an input and is driven low when the SPI is in Master mode, this will also set the SPIF Flag. SPIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, the SPIF bit is cleared by first reading the SPI Status Register with SPIF set, then accessing the SPI Data Register (SPDR).

Configuring SPI



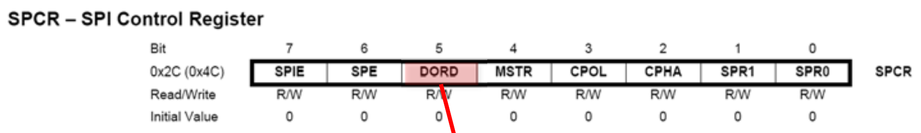
SPCR – SPI Control Register

Bit	7	6	5	4	3	2	1	0	
0x2C (0x4C)	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	SPCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

SPE: SPI Enable

When the **SPE** bit is written to one, the SPI is enabled. This bit must be set to enable any SPI operations.

Configuring SPI



SPCR – SPI Control Register

Bit	7	6	5	4	3	2	1	0	
0x2C (0x4C)	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	SPCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

DORD: Data Order

When the **DORD** bit is written to one, the LSB of the data word is transmitted first.
When the **DORD** bit is written to zero, the MSB of the data word is transmitted first.

Configuring SPI

SPCR – SPI Control Register

Bit	7	6	5	4	3	2	1	0	
0x2C (0x4C)	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	SPCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

MSTR: Master/Slave Select

This bit selects Master SPI mode when written to one, and Slave SPI mode when written logic zero.

Configuring SPI

SPCR – SPI Control Register

Bit	7	6	5	4	3	2	1	0	
0x2C (0x4C)	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	SPCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

CPOL: Clock Polarity

When this bit is written to one, **SCK** is high when idle. When **CPOL** is written to zero, **SCK** is low when idle.

Table 18-3. CPOL Functionality

CPOL	Leading Edge	Trailing Edge
0	Rising	Falling
1	Falling	Rising

Configuring SPI

SPCR – SPI Control Register

Bit	7	6	5	4	3	2	1	0	
0x2C (0x4C)	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	SPCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

CPHA: Clock Phase

The settings of the Clock Phase bit (**CPHA**) determine if data is sampled on the leading (first) or trailing (last) edge of **SCK**.

Table 18-4. CPHA Functionality

CPHA	Leading Edge	Trailing Edge
0	Sample	Setup
1	Setup	Sample

Configuring SPI

SPCR – SPI Control Register

Bit	7	6	5	4	3	2	1	0	
0x2C (0x4C)	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	SPCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

SPR1, SPR0: SPI Clock Rate Select 1 and 0

These two bits control the **SCK** rate of the device configured as a Master. **SPR1** and **SPR0** have no effect on the Slave. See data sheet for relationship between these bits and clock frequency.

SPI2X	SPR1	SPR0	SCK Frequency
0	0	0	$f_{osc}/4$
0	0	1	$f_{osc}/16$
0	1	0	$f_{osc}/64$
0	1	1	$f_{osc}/128$
1	0	0	$f_{osc}/2$
1	0	1	$f_{osc}/8$
1	1	0	$f_{osc}/32$
1	1	1	$f_{osc}/64$

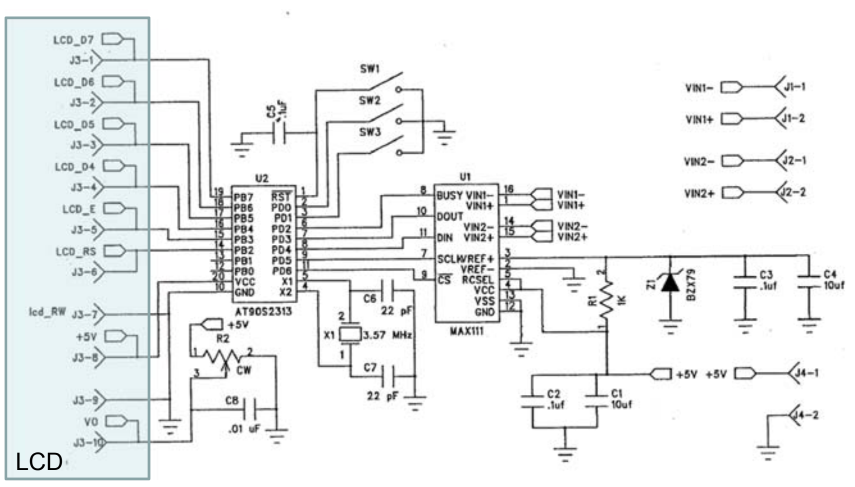
SPI

… more information and configuration examples:

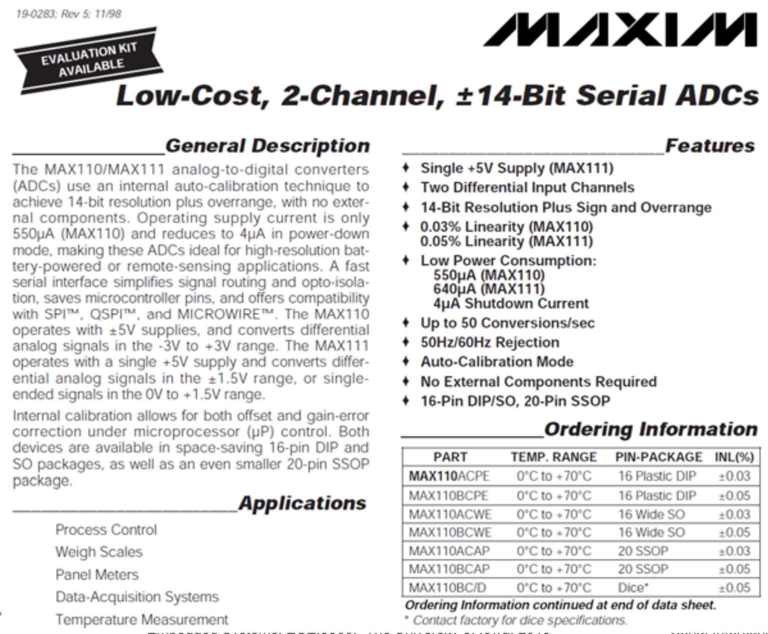
See ATmega88PA datasheet

ES with SPI - Example

+/- 14 bit Dual-Channel Voltmeter with LCD



Dual-Channel Voltmeter with LCD



19-0283; Rev 5; 11/98

EVALUATION KIT AVAILABLE

MAXIM

Low-Cost, 2-Channel, ±14-Bit Serial ADCs

General Description

The MAX110/MAX111 analog-to-digital converters (ADCs) use an internal auto-calibration technique to achieve 14-bit resolution plus overrange, with no external components. Operating supply current is only 550µA (MAX110) and reduces to 4µA in power-down mode, making these ADCs ideal for high-resolution battery-powered or remote-sensing applications. A fast serial interface simplifies signal routing and opto-isolation, saves microcontroller pins, and offers compatibility with SPI™, QSPI™, and MICROWIRE™. The MAX110 operates with ±5V supplies, and converts differential analog signals in the -3V to +3V range. The MAX111 operates with a single +5V supply and converts differential analog signals in the ±1.5V range, or single-ended signals in the 0V to +1.5V range.

Internal calibration allows for both offset and gain-error correction under microprocessor (µP) control. Both devices are available in space-saving 16-pin DIP and SO packages, as well as an even smaller 20-pin SSOP package.

Applications

Process Control
Weigh Scales
Panel Meters
Data-Acquisition Systems
Temperature Measurement

Features

- Single +5V Supply (MAX111)
- Two Differential Input Channels
- 14-Bit Resolution Plus Sign and Overrange
- 0.03% Linearity (MAX110)
  0.05% Linearity (MAX111)
- Low Power Consumption:
  550µA (MAX110)
  640µA (MAX111)
  4µA Shutdown Current
- Up to 50 Conversions/sec
- 50Hz/60Hz Rejection
- Auto-Calibration Mode
- No External Components Required
- 16-Pin DIP/SO, 20-Pin SSOP

Ordering Information

PART	TEMP. RANGE	PIN-PACKAGE	INL(%)
MAX110ACPE	0°C to +70°C	16 Plastic DIP	±0.03
MAX110BCPE	0°C to +70°C	16 Plastic DIP	±0.05
MAX110ACWE	0°C to +70°C	16 Wide SO	±0.03
MAX110BCWE	0°C to +70°C	16 Wide SO	±0.05
MAX110ACAP	0°C to +70°C	20 SSOP	±0.03
MAX110BCAP	0°C to +70°C	20 SSOP	±0.05
MAX110BC/D	0°C to +70°C	Dice*	±0.05

Ordering Information continued at end of data sheet.

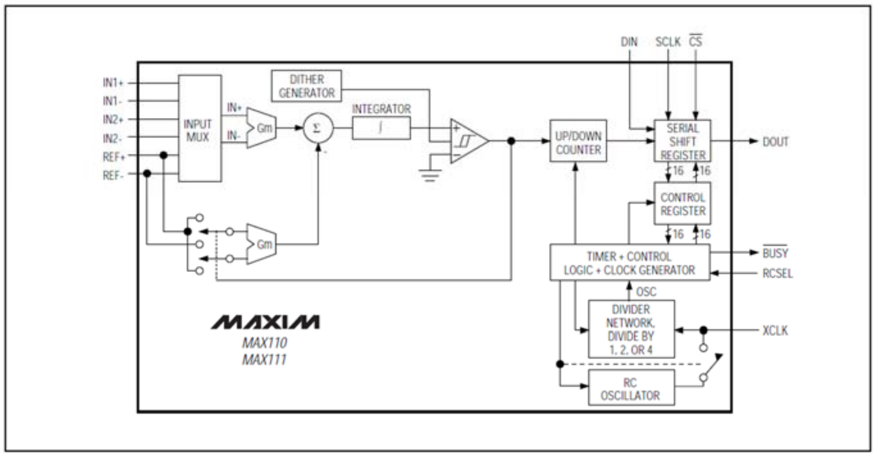
** Contact factory for dice specifications.*

MAX110/MAX111

Dual-Channel Voltmeter with LCD

Low-Cost, 2-Channel, ±14-Bit Serial ADCs

MAX110/MAX111



Interface MAX110, MAX111

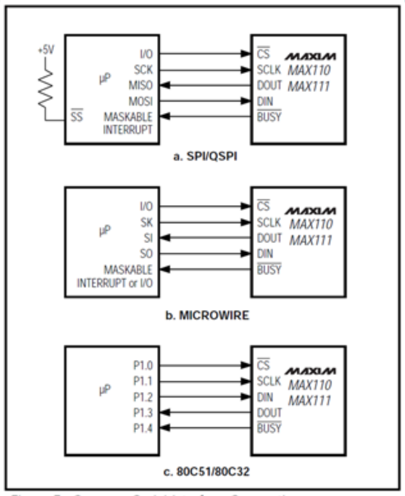
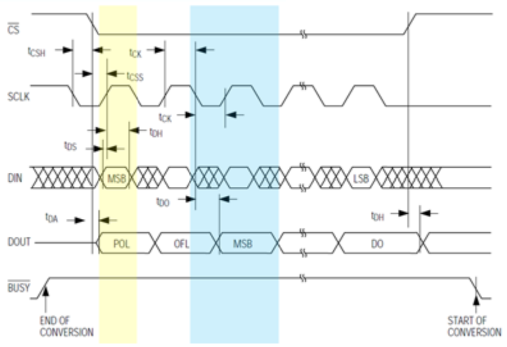


Figure 7. Common Serial-Interface Connections



- Output data from the ADC is clocked out on SCLK's falling edge and **should be read on SCLK's rising edge**.
- Input data to the ADC at DIN is **clocked in on SCLK's rising edge**.

- **SPI … Serial Peripheral Interface – Requires CPU Intervention**
- **QSPI … Queued Serial Peripheral Interface – Does not require CPU Intervention**

Interface MAX110, MAX111

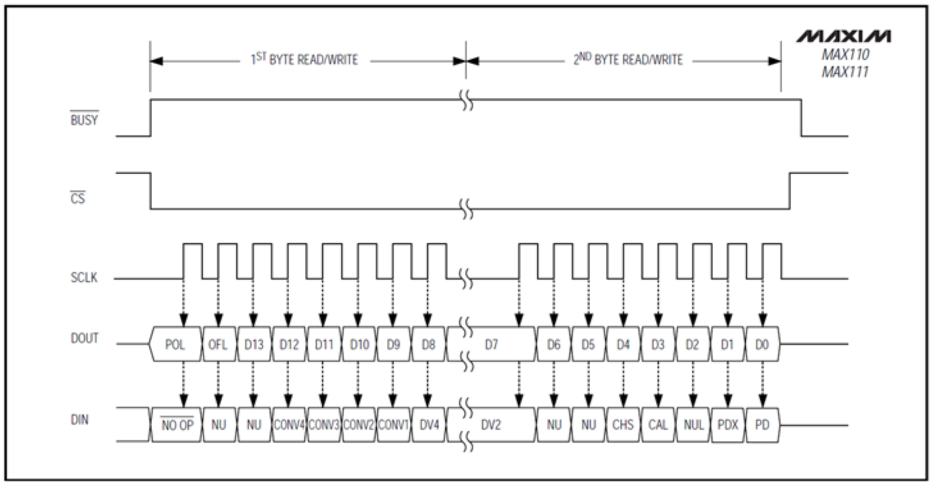


Figure 8a. SPI/MICROWIRE-Interface Timing

Interface MAX110, MAX111

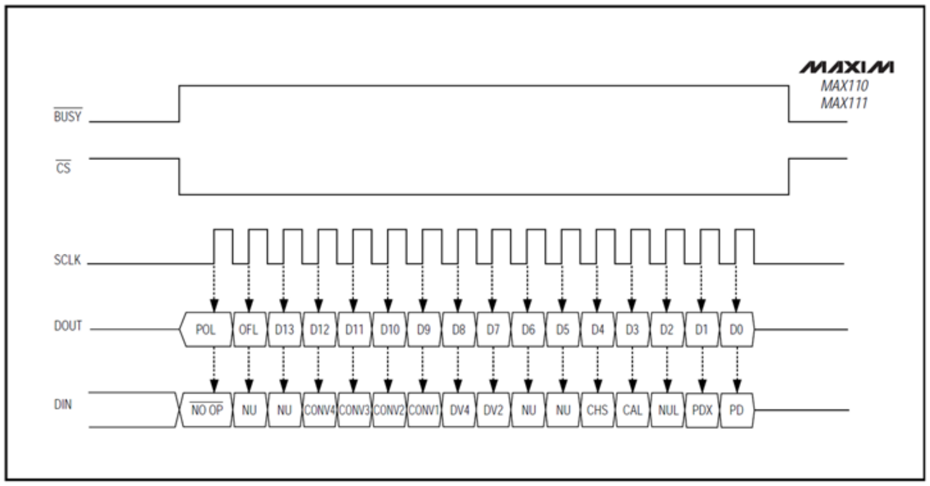


Figure 8b. QSPI Serial-Interface Timing

Interface MAX110, MAX111

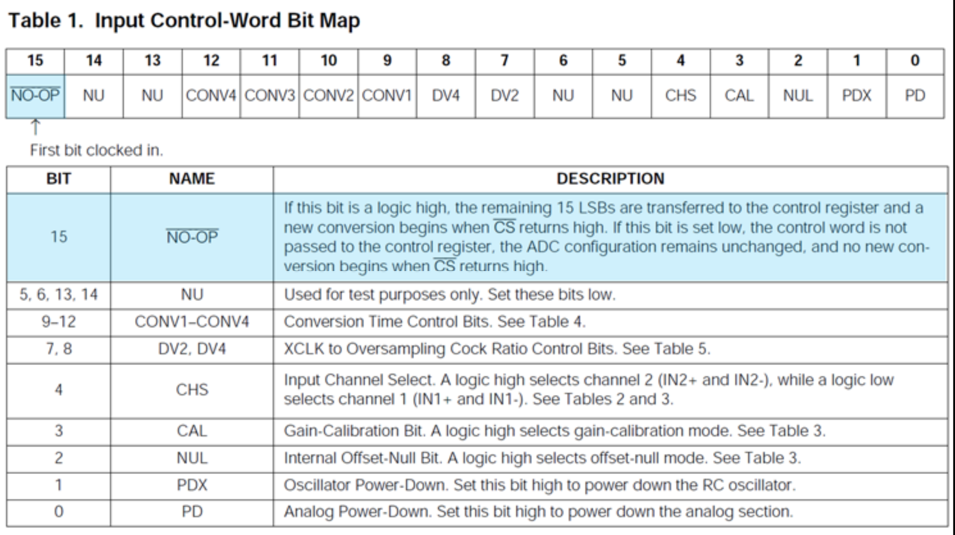


Table 1. Input Control-Word Bit Map

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$\overline{\text{NO-OP}}$	NU	NU	CONV4	CONV3	CONV2	CONV1	DV4	DV2	NU	NU	CHS	CAL	NUL	PDX	PD

↑
First bit clocked in.

BIT	NAME	DESCRIPTION
15	$\overline{\text{NO-OP}}$	If this bit is a logic high, the remaining 15 LSBs are transferred to the control register and a new conversion begins when $\overline{\text{CS}}$ returns high. If this bit is set low, the control word is not passed to the control register, the ADC configuration remains unchanged, and no new conversion begins when $\overline{\text{CS}}$ returns high.
5, 6, 13, 14	NU	Used for test purposes only. Set these bits low.
9–12	CONV1–CONV4	Conversion Time Control Bits. See Table 4.
7, 8	DV2, DV4	XCLK to Oversampling Cock Ratio Control Bits. See Table 5.
4	CHS	Input Channel Select. A logic high selects channel 2 (IN2+ and IN2-), while a logic low selects channel 1 (IN1+ and IN1-). See Tables 2 and 3.
3	CAL	Gain-Calibration Bit. A logic high selects gain-calibration mode. See Table 3.
2	NUL	Internal Offset-Null Bit. A logic high selects offset-null mode. See Table 3.
1	PDX	Oscillator Power-Down. Set this bit high to power down the RC oscillator.
0	PD	Analog Power-Down. Set this bit high to power down the analog section.

MAX111 – Grounding

- **Board layout and grounding is important for ADC performance!**

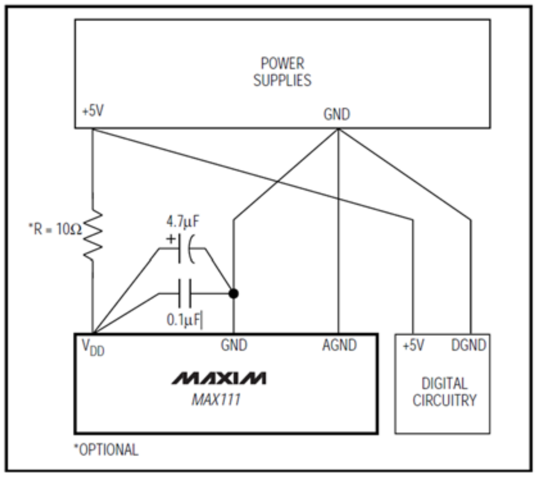


Figure 10b. MAX111 Power-Supply Grounding Connections

MAX111 – Application Example

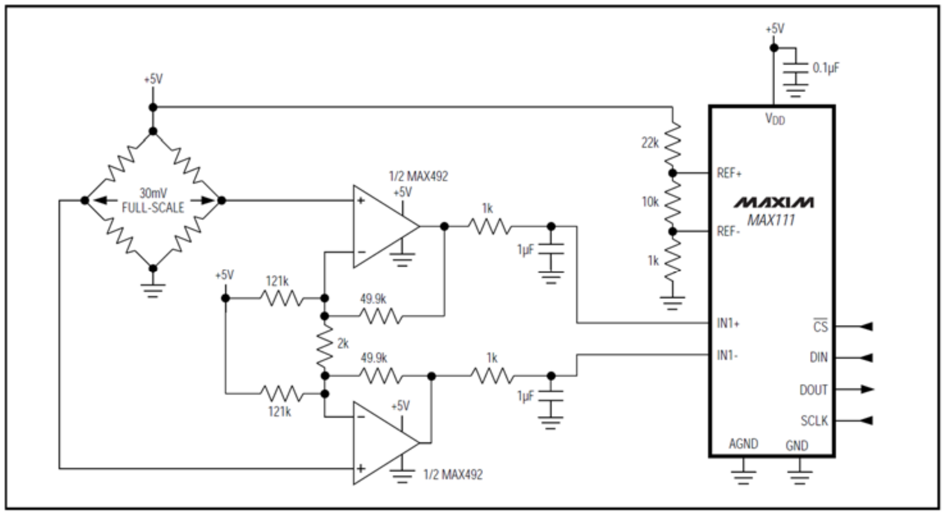


Figure 11. Weigh Scale Application

… EOL