FIB UPC UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

Serial Communications Interfaces

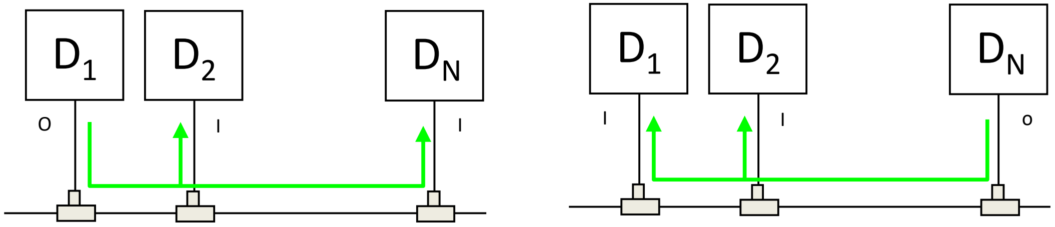
Dpt. Enginyeria de Sistemes, Automàtica i Informàtica Industrial

Communication interfaces types

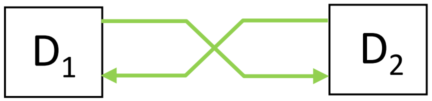
Serial

Asynchronous

Half-Duplex

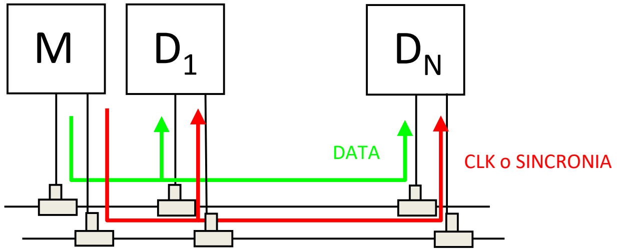


Full-Duplex
(generally point to point (PTP) only)

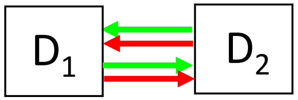


Synchronous

Half-Duplex



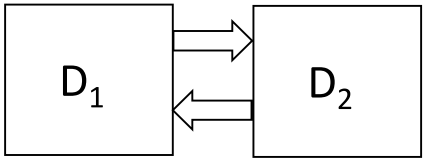
Full-Duplex
(generally point to point (PTP) only)



Parallel

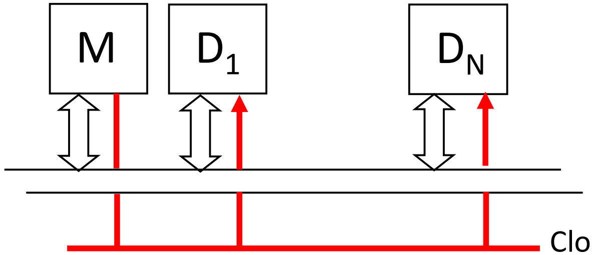
Asynchronous (not very common)

(generally point to point PTP only)



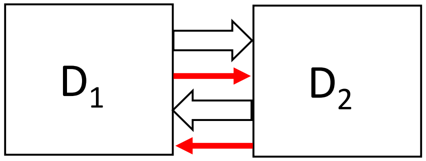
Synchronous

Half-Duplex



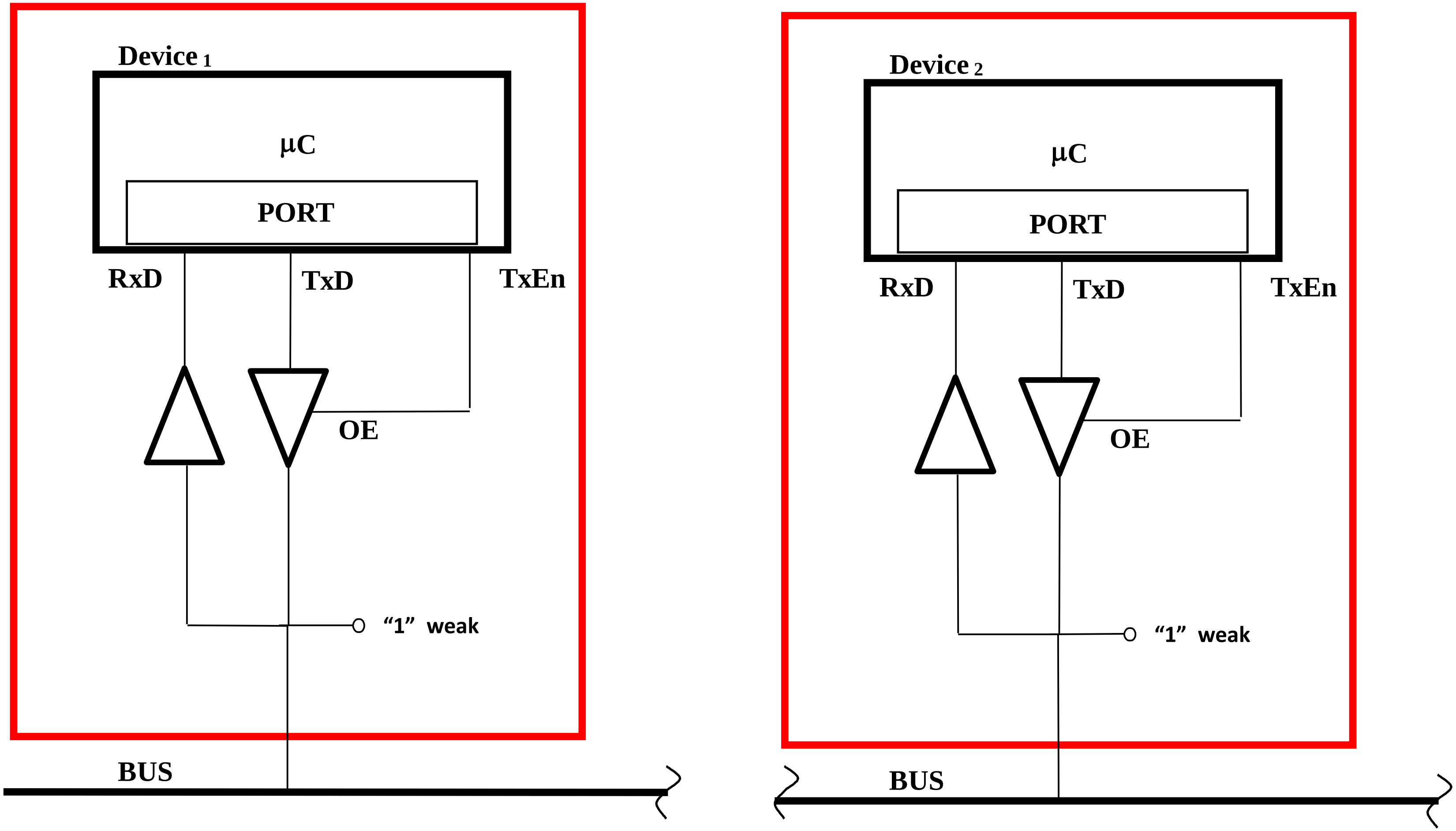
Clock or synchrony
plus other control signals

Full-Duplex

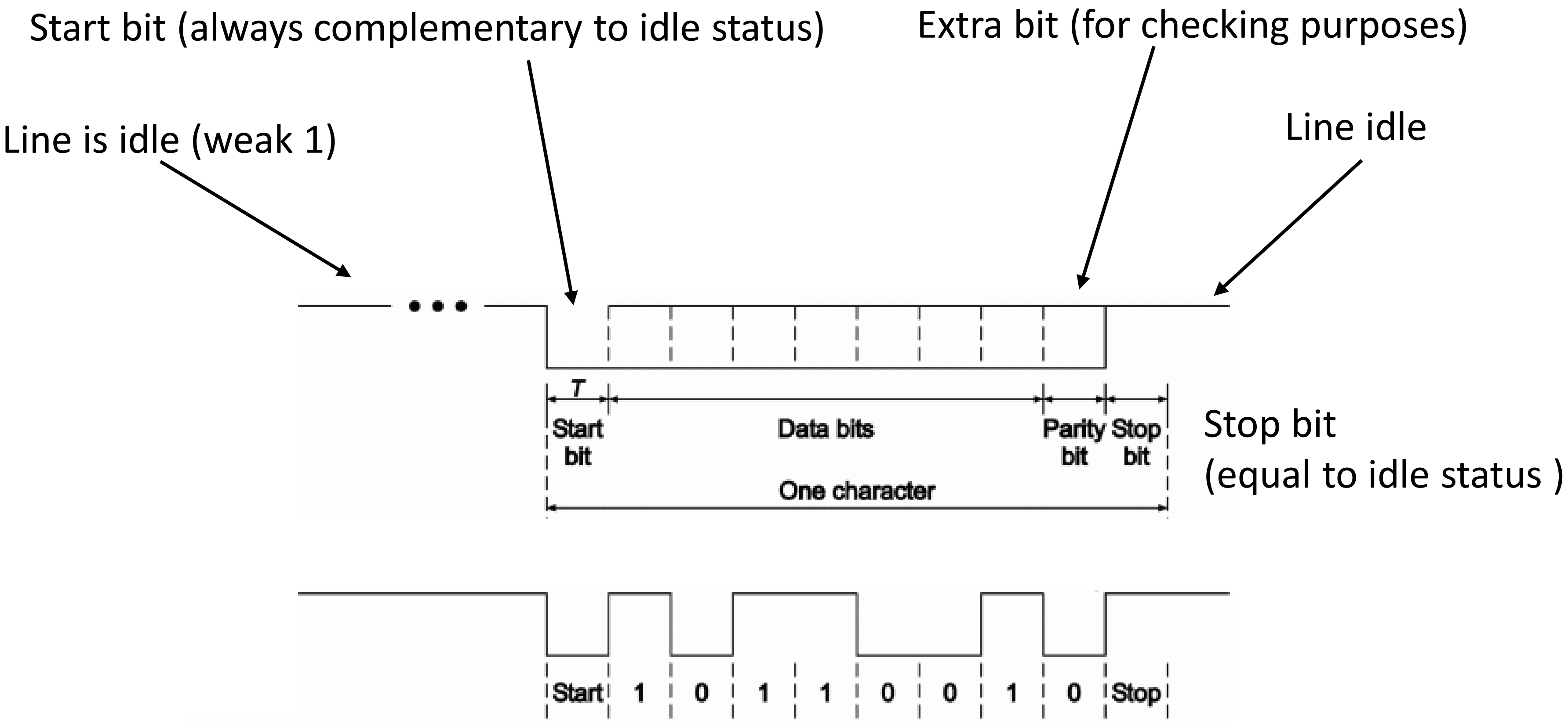


(generally point to point PTP only)

Serial communication basic concepts



Simple serial interface


Start bit (always complementary to idle status)

Extra bit (for checking purposes)

Line is idle (weak 1)

Line idle

Stop bit
(equal to idle status)

Serial asynchronous communication chronogram (8 bits)

Data Transmission Errors

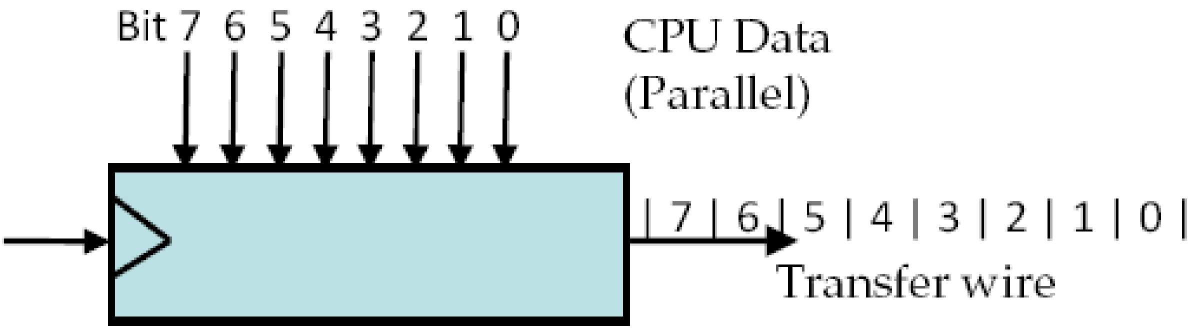
1. Framing error
   - May occur due to clock synchronization problem
   - Can be detected by the missing stop bit
2. Receiver overrun
   - May occur when the CPU did not read the received data for a while
3. Parity errors
   - Occur due to odd number of bits change values

Serial Comms, Fundamentals

Key element

Function to perform: PARALLEL to SERIAL converter

Key component: Shift Register

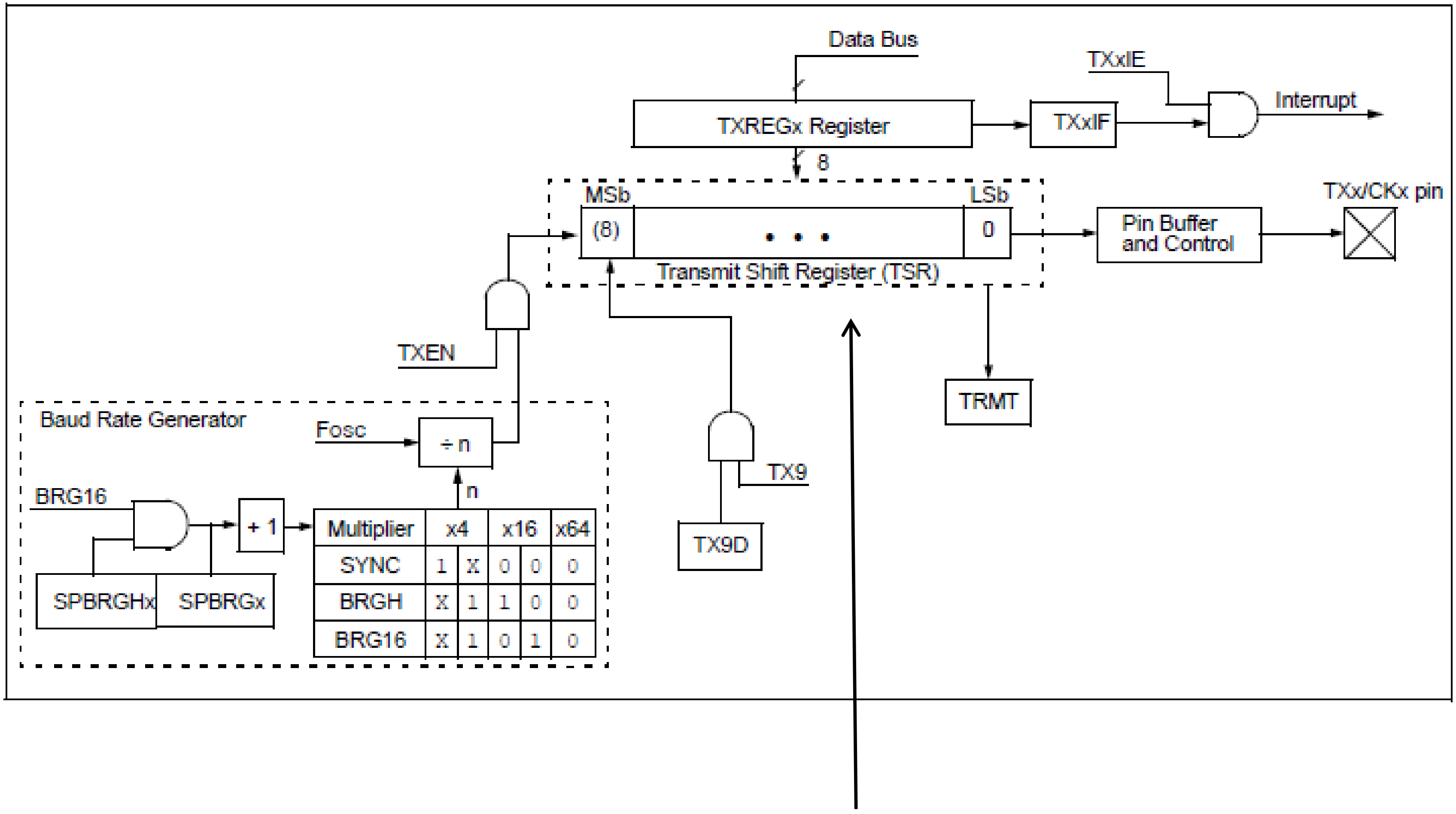


Output Port (Transmitter)
unidirectional !!!

The **information unit** is the *character*, no the byte.

18F45K22 USART Tx Block

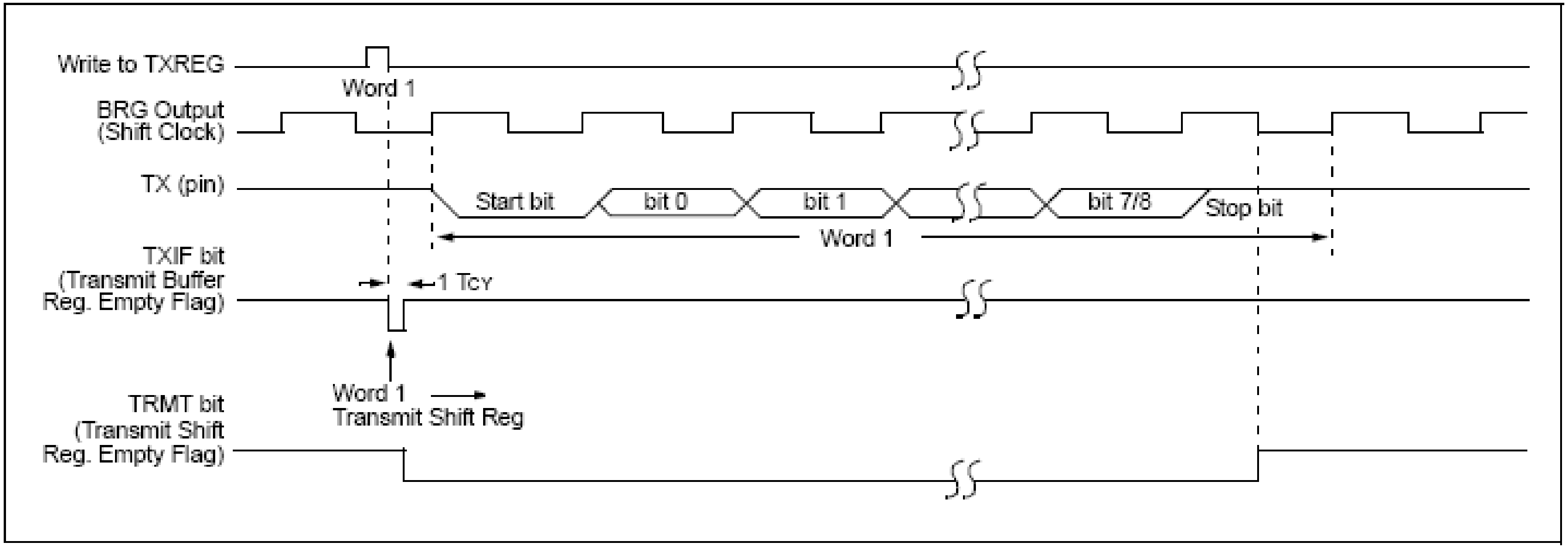
FIGURE 16-1: EUSART TRANSMIT BLOCK DIAGRAM



Multiplier	x4		x16		x64
SYNC	1	X	0	0	0
BRGH	X	1	1	0	0
BRG16	X	1	0	1	0

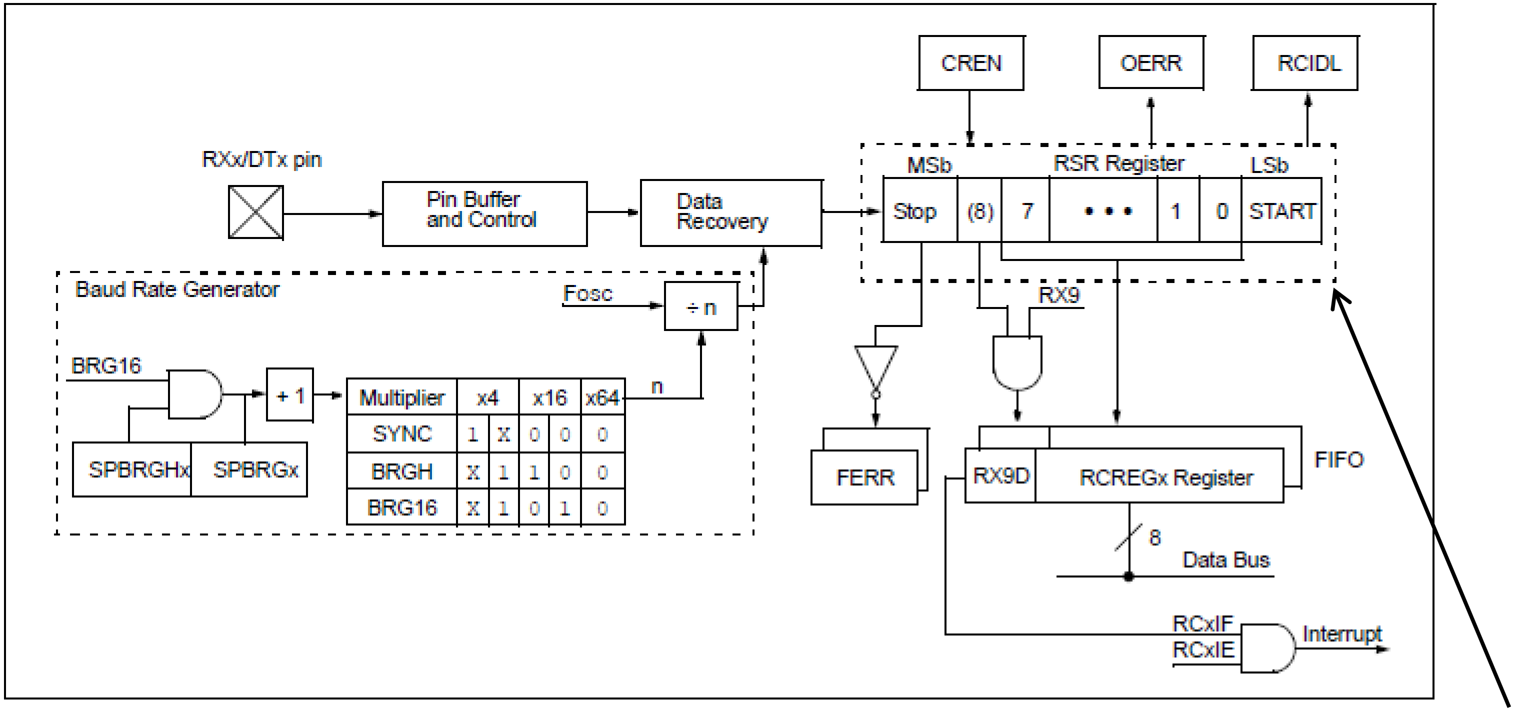
Shift Register (Parallel Input/Serial Output)

FIGURE 20-4: ASYNCHRONOUS TRANSMISSION, TXCKP = 0 (TX NOT INVERTED)



18F45K22 USART Rx Block

FIGURE 16-2: EUSART RECEIVE BLOCK DIAGRAM



Multiplier	x4		x16		x64
SYNC	1	X	0	0	0
BRGH	X	1	1	0	0
BRG16	X	1	0	1	0

Shift Register (Serial Input/Parallel Output)

Reception starts upon detection of a START bit

- Transmission integrity is checked testing STOP bit level
- Received data in Shift register is stored in FIFO (One single address, RCREG)
- If three characters are received in a row, FIFO overflows Overrun error
- RCIF is set HIGH until FIFO is empty

The PIC18 USART Serial Communication Interface

USART-Related Pins

- RC6/TX1/CK1 and RC7/RX1/DT1 (USART1)
- RD6/TX2/CK2 and RD7/RX2/DT2 (USART2)

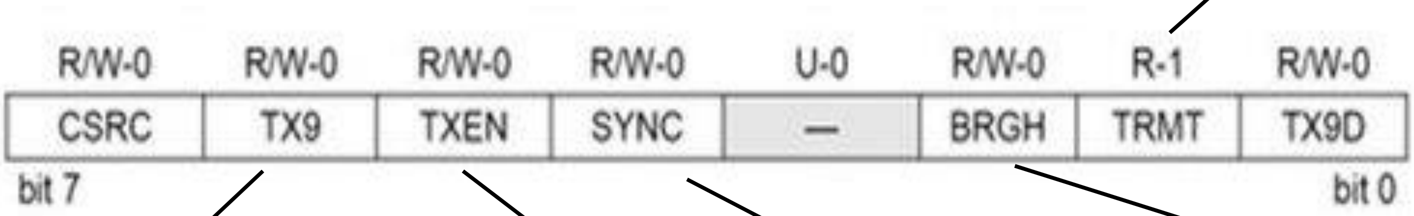
USART-Related Registers

- Transmit status register (TXSTA) - Transmit register (TXREG)
- Receive status register (RCSTA) - Receive register (RCREG)
- Baud rate Control register (BAUDCON)
- Baud rate generator register (SPBRG)

The PIC18 USART Transmit and receive control registers

1.TXSTAx - Transmit Status and Control Register

TSR
register is empty



R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R-1	R/W-0
CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D
bit 7							bit 0

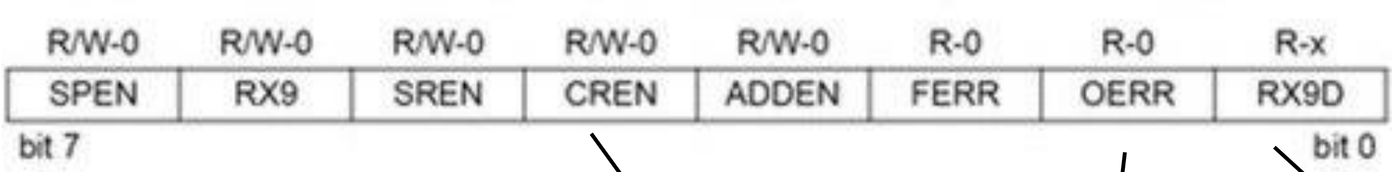
8 or 9 bits

Enable Tx.

Synchronous/Asynchr.

High Baud Rate

2. RCSTAx - Receive Status and Control Register



R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-x
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
bit 7							bit 0

Serial Port Enable !

Enable Rx.

9th data bit

Overrun Rx. Error Status

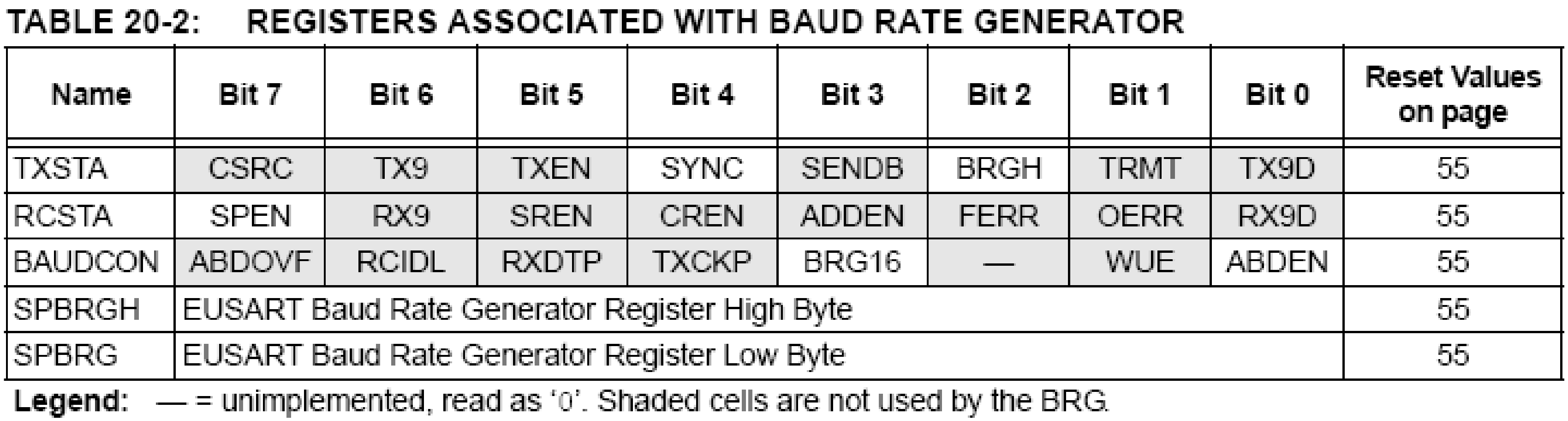


TABLE 20-2: REGISTERS ASSOCIATED WITH BAUD RATE GENERATOR

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	55
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	55
BAUDCON	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN	55
SPBRGH	EUSART Baud Rate Generator Register High Byte								55
SPBRG	EUSART Baud Rate Generator Register Low Byte								55

Legend: — = unimplemented, read as '0'. Shaded cells are not used by the BRG.

SPBRG register

The rate selection is made by the BRGH bit in TXSTA register:
1 = High speed
0 = Low speed

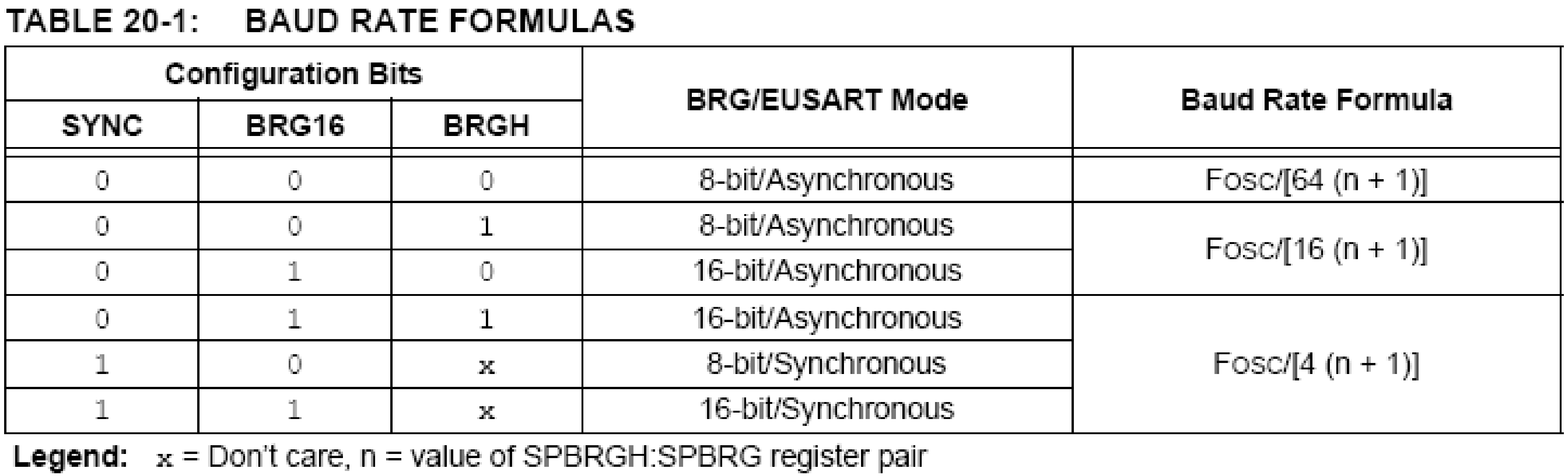


TABLE 20-1: BAUD RATE FORMULAS

Configuration Bits			BRG/EUSART Mode	Baud Rate Formula
SYNC	BRG16	BRGH		
0	0	0	8-bit/Asynchronous	Fosc/[64 (n + 1)]
0	0	1	8-bit/Asynchronous	Fosc/[16 (n + 1)]
0	1	0	16-bit/Asynchronous	
0	1	1	16-bit/Asynchronous	Fosc/[4 (n + 1)]
1	0	x	8-bit/Synchronous	
1	1	x	16-bit/Synchronous	

Legend: x = Don't care, n = value of SPBRGH:SPBRG register pair

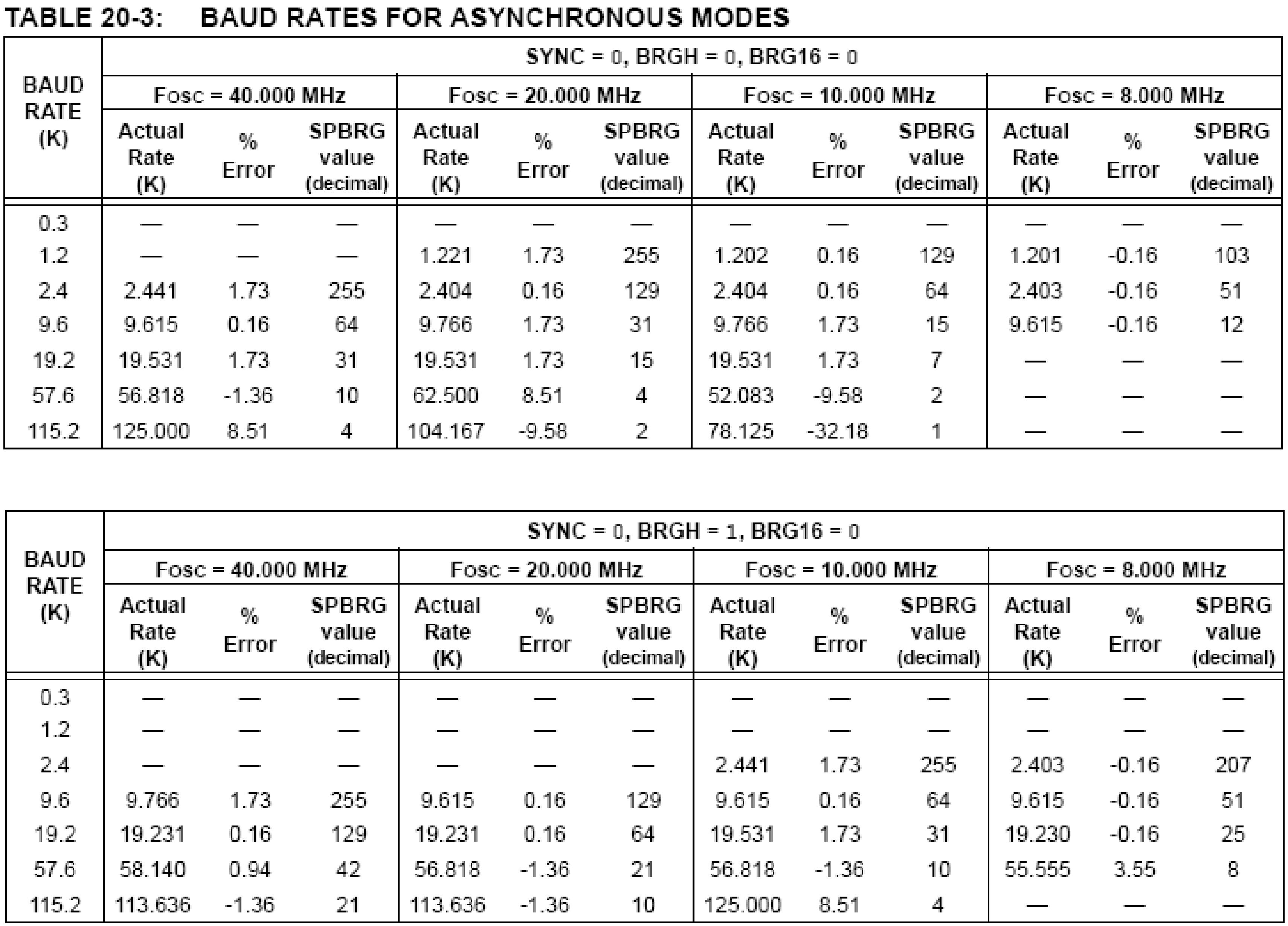
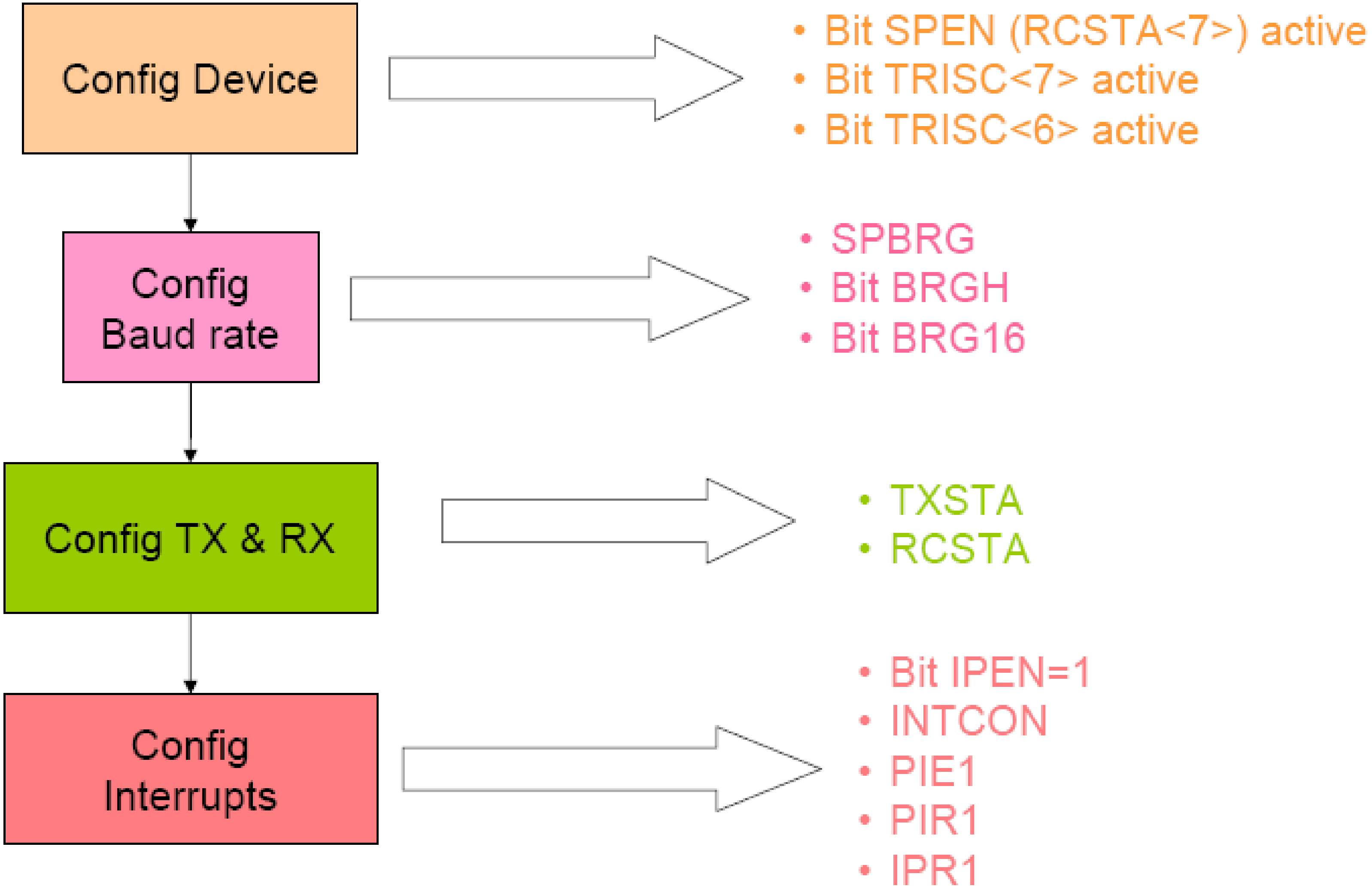


TABLE 20-3: BAUD RATES FOR ASYNCHRONOUS MODES

BAUD RATE (K)	SYNC = 0, BRGH = 0, BRG16 = 0											
	Fosc = 40.000 MHz			Fosc = 20.000 MHz			Fosc = 10.000 MHz			Fosc = 8.000 MHz		
	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)
0.3	—	—	—	—	—	—	—	—	—	—	—	—
1.2	—	—	—	1.221	1.73	255	1.202	0.16	129	1.201	-0.16	103
2.4	2.441	1.73	255	2.404	0.16	129	2.404	0.16	64	2.403	-0.16	51
9.6	9.615	0.16	64	9.766	1.73	31	9.766	1.73	15	9.615	-0.16	12
19.2	19.531	1.73	31	19.531	1.73	15	19.531	1.73	7	—	—	—
57.6	56.818	-1.36	10	62.500	8.51	4	52.083	-9.58	2	—	—	—
115.2	125.000	8.51	4	104.167	-9.58	2	78.125	-32.18	1	—	—	—

BAUD RATE (K)	SYNC = 0, BRGH = 1, BRG16 = 0											
	Fosc = 40.000 MHz			Fosc = 20.000 MHz			Fosc = 10.000 MHz			Fosc = 8.000 MHz		
	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)
0.3	—	—	—	—	—	—	—	—	—	—	—	—
1.2	—	—	—	—	—	—	—	—	—	—	—	—
2.4	—	—	—	—	—	—	2.441	1.73	255	2.403	-0.16	207
9.6	9.766	1.73	255	9.615	0.16	129	9.615	0.16	64	9.615	-0.16	51
19.2	19.231	0.16	129	19.231	0.16	64	19.531	1.73	31	19.230	-0.16	25
57.6	58.140	0.94	42	56.818	-1.36	21	56.818	-1.36	10	55.555	3.55	8
115.2	113.636	-1.36	21	113.636	-1.36	10	125.000	8.51	4	—	—	—

Serial Comms, Recommended Initialization



Config Device

- Bit SPEN (RCSTA<7>) active
- Bit TRISC<7> active
- Bit TRISC<6> active

Config Baud rate

- SPBRG
- Bit BRGH
- Bit BRG16

Config TX & RX

- TXSTA
- RCSTA

Config Interrupts

- Bit IPEN=1
- INTCON
- PIE1
- PIR1
- IPR1

Ex. Write a subroutine to configure the USART1 transmitter to transmit data in asynchronous mode using 8-bit data format, disable interrupt, set baud rate to 9600. Assume the frequency of the crystal oscillator is 16 MHz.

```
void usart1_Init(void)
{
        TXSTA1              = 0x24; /* USART Configuration Register */
        SPBRG1              = 103;    /* Set de Baud rate */
        TRISCbits.RC7       = 1;      /* configure RX1 pin for input */
        TRISCbits.RC6       = 1;      /* configure TX1 pin for output */
        PIE1bits.TXIE       = 0;      /* disable transmit interrupt */
        RCSTA1bits.SPEN     = 1;      /* enable USART port */
}
```

Ex. Write a subroutine to output a character to USART1 using the polling method.

Solution:

- Data can be sent to the transmitter only when it is idle.

```
void putc_usart1 (char xc);
{
    while (! TXSTA1.TRMT);
    TXREG1 = xc;
}
```

Ex. Write a subroutine to output a string (in program memory) pointed to by TBLPTR and terminated by a NULL character from USART1.

Solution:

```
void puts_usart1 (unsigned rom char *cptr)
{
    while(*cptr)
              putc_usart1 (*cptr++);
}
```

Ex. Write an instruction sequence to configure the USART1 to receive data in asynchronous mode using 8-bit data format, disable interrupt, set baud rate to 9600. Assume that the frequency of the crystal oscillator is 16 MHz.

Solution:

```
RCSTA1 = 0x90;
SPBRG   = 103;
TRISC    |= 0xC0;                /* configure RC7/RX1 & RC6/TX1 pin */
```

Ex. Write a subroutine to read a character from USART1 and return the character in WREG using the polling method. Ignore any errors.

Solution: A new character is received if the RCIF flag of the PIR1 register is set to 1.

```
unsigned char getc_usart1 (void)
{
        while (! PIR1bits.RCIF);
        return RCREG1;                          // RCIF clears automatically
}
```

Flow Control of USART in Asynchronous Mode

- In some circumstances, the software cannot read the received data and needs to inform the transmitter to stop.
- In some other situation, the transmitter may need to be told to suspend transmission because the receiver is too busy to read data.
- Both situations are handled by flow control.
- There are two flow control methods: hardware and XON/XOFF.
- XON and XOFF are two standard ASCII characters.
- The ASCII code for XON and XOFF are 0x11 and 0x13, respectively.
- Whenever a microcontroller cannot handle the incoming data, it sends the XOFF to the transmitter.
- When the microcontroller can handle incoming characters, it sends out XON character.

Asynchronous serial interface RS-232

The EIA232 Standard

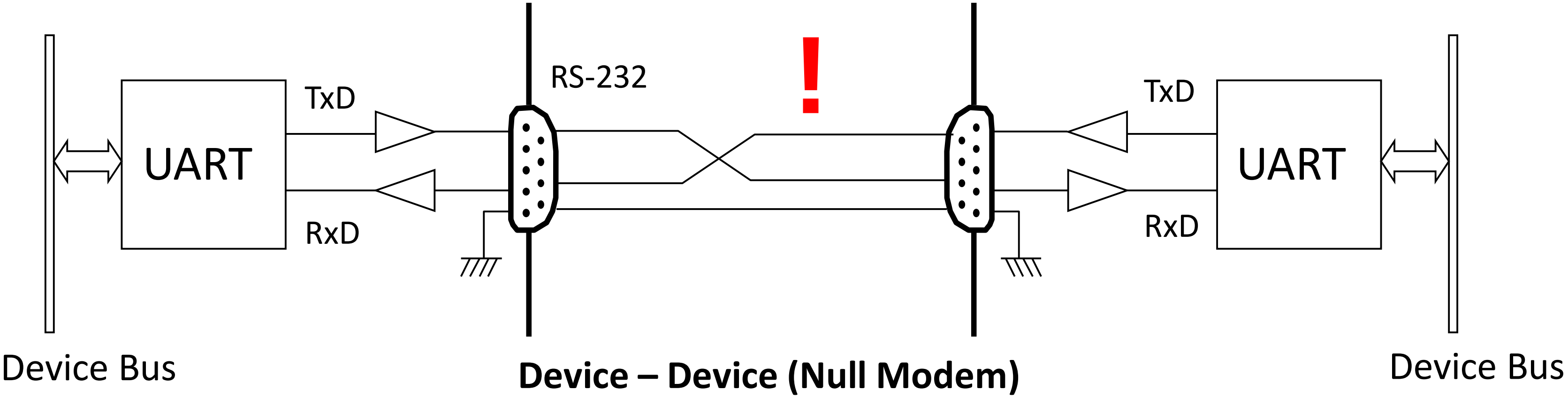
- Developed in 1960, **RS-232** (Recommended Standard 232) is a standard for serial binary single-ended data and control signals connecting between a *DTE* (Data Terminal Equipment) and a *DCE* (Data Circuit-terminating Equipment or modem).

- The standard requires the transmitter to use +12 V and −12 V, but requires the receiver to distinguish voltages as low as +3 V and -3 V

- Common asynchronous speeds: 200, 2.400, 4.800, 9.600, 19.200, 57.600, 115.200 bauds (for a binary two-level signal transmissions, one baud is equal to one bit per second).

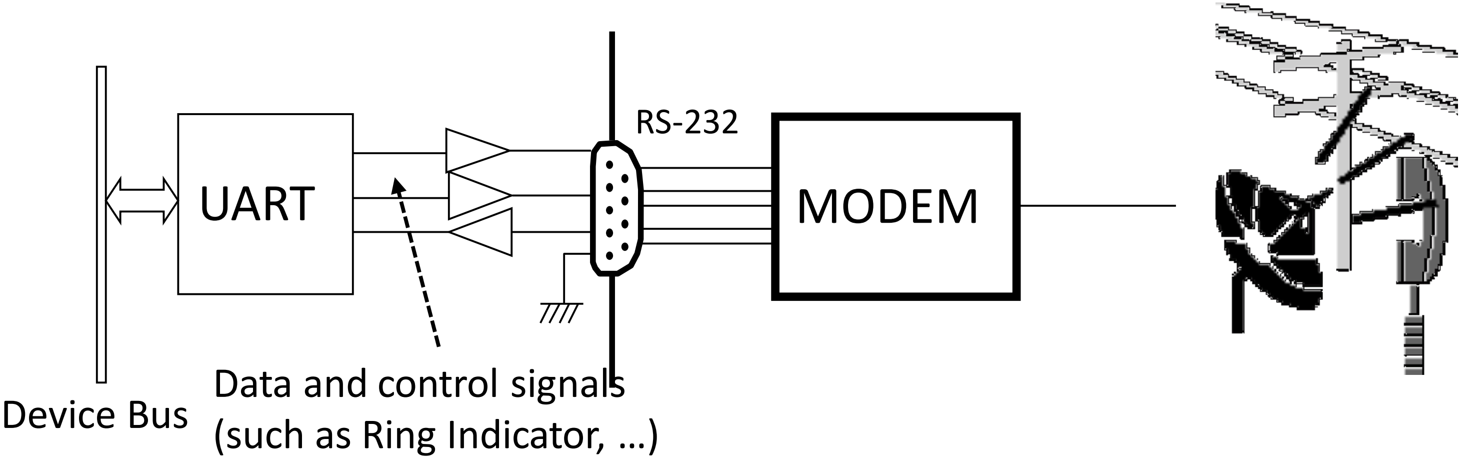
- A male DB-9 connector for a serial port

The UART is used to communicate directly two devices **or** a device with a modem.



Device – Device (Null Modem)

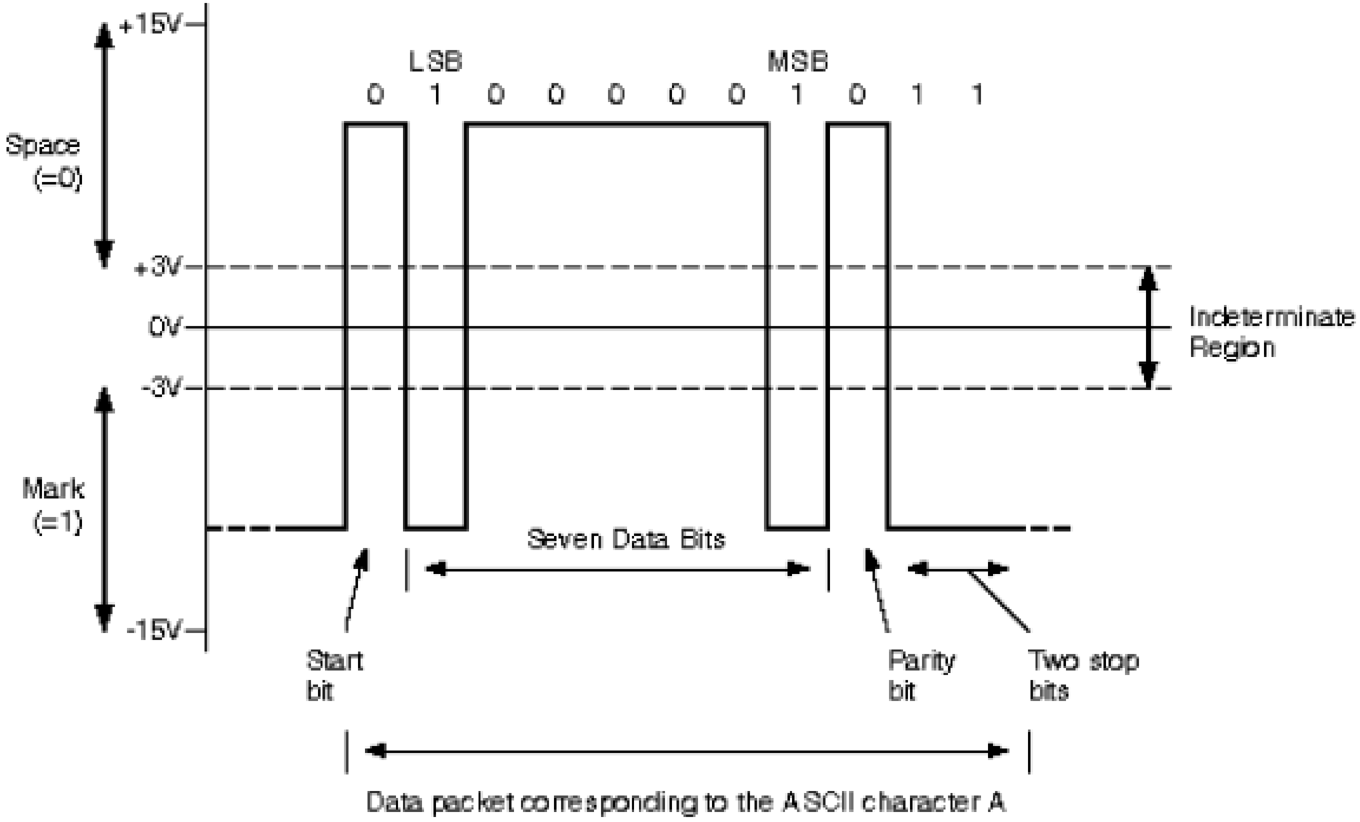
! **Tx wire and Rx wire must be cross connected**



Device – Modem

RS232 standard

Electrical Level: RS232 Signals

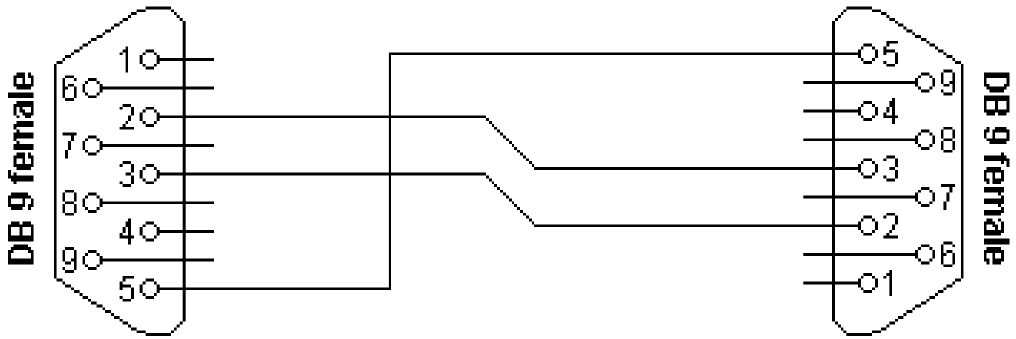


RS232 standard

Logical Level: Signals

Nombre	Dirección DTE ↔ DCE	Función	Comentario
TD	⇒	Transmitted data	Par de Datos
RD	⇐	Received Data	
RTS	⇒	Request to Send	Par de Handshake
CTS	⇐	Clear to Send	
DTR	⇒	Data Terminal Ready	Par de Handshake
DSR	⇐	Data Set Ready	
DCD	⇐	Data Carrier Detect	Habilitan DTE
RI	⇐	Ring Indicator	

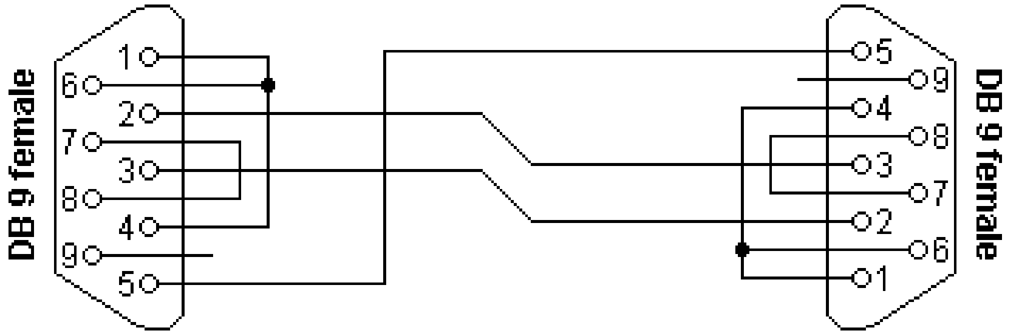
Null modem connection



Without Handshaking

Connector 1	Connector 2	Function
2	3	Rx ← Tx
3	2	Tx → Rx
5	5	Signal Ground

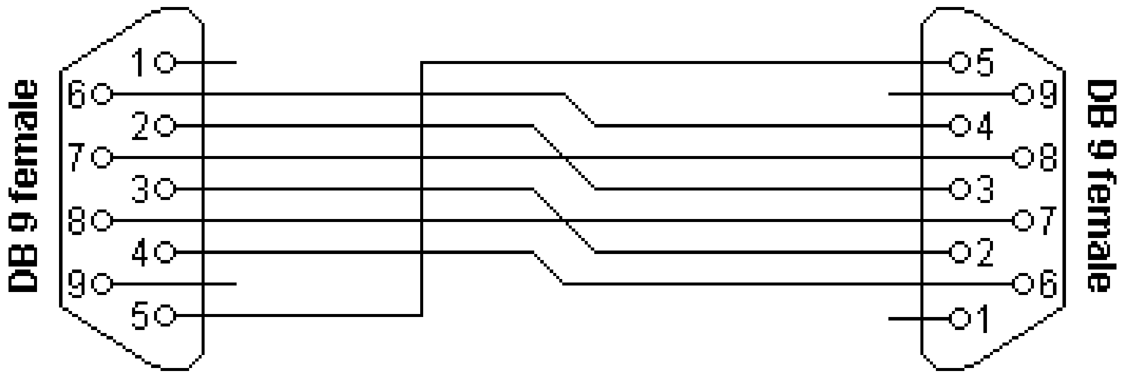
Null modem connection



With loop back
Handshaking

Connector 1	Connector 2	Function
2	3	Rx ← Tx
3	2	Tx → Rx
5	5	Signal ground
1 + 4 + 6	-	DTR → CD + DSR
-	1 + 4 + 6	DTR → CD + DSR
7 + 8	-	RTS → CTS
-	7 + 8	RTS → CTS

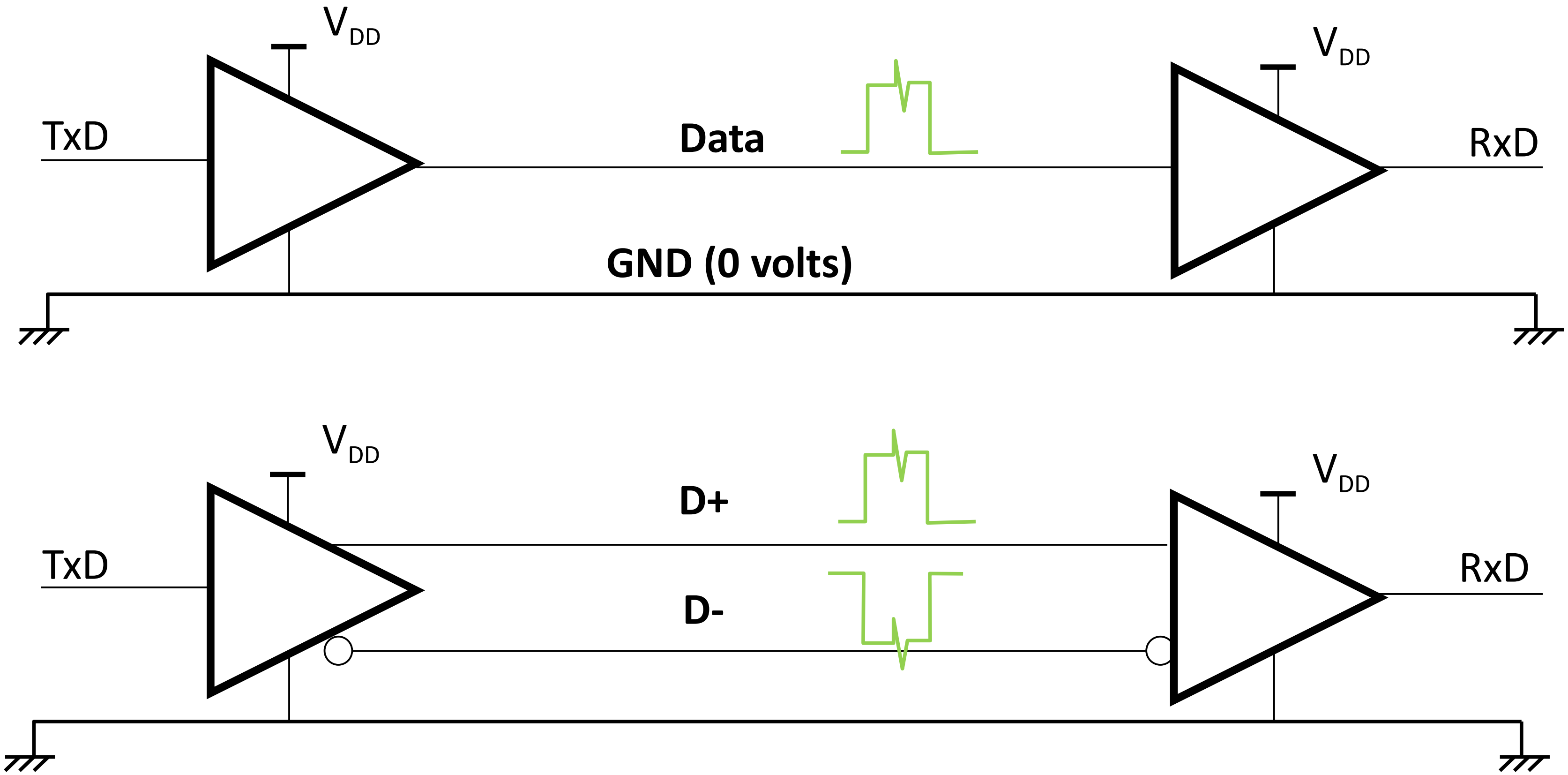
Null modem connection



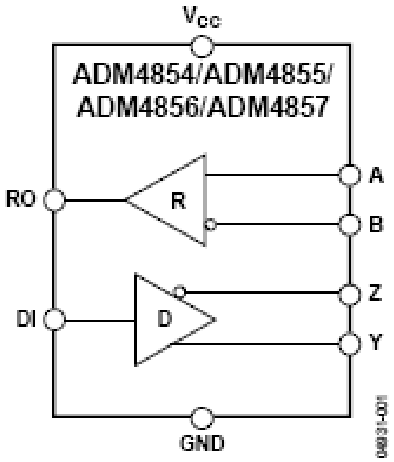
With full Handshaking

Connector 1	Connector 2	Function
2	3	Rx ← Tx
3	2	Tx → Rx
4	6	DTR → DSR
5	5	Signal ground
6	4	DSR ← DTR
7	8	RTS → CTS
8	7	CTS ← RTS

Noise immunity



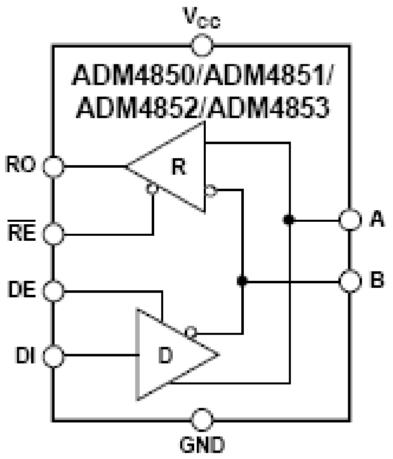
Single ended **and** ***Differential*** **transmission**



Full-duplex

Differential transmission electrical interfaces (RS-422)

Data reception enabling control



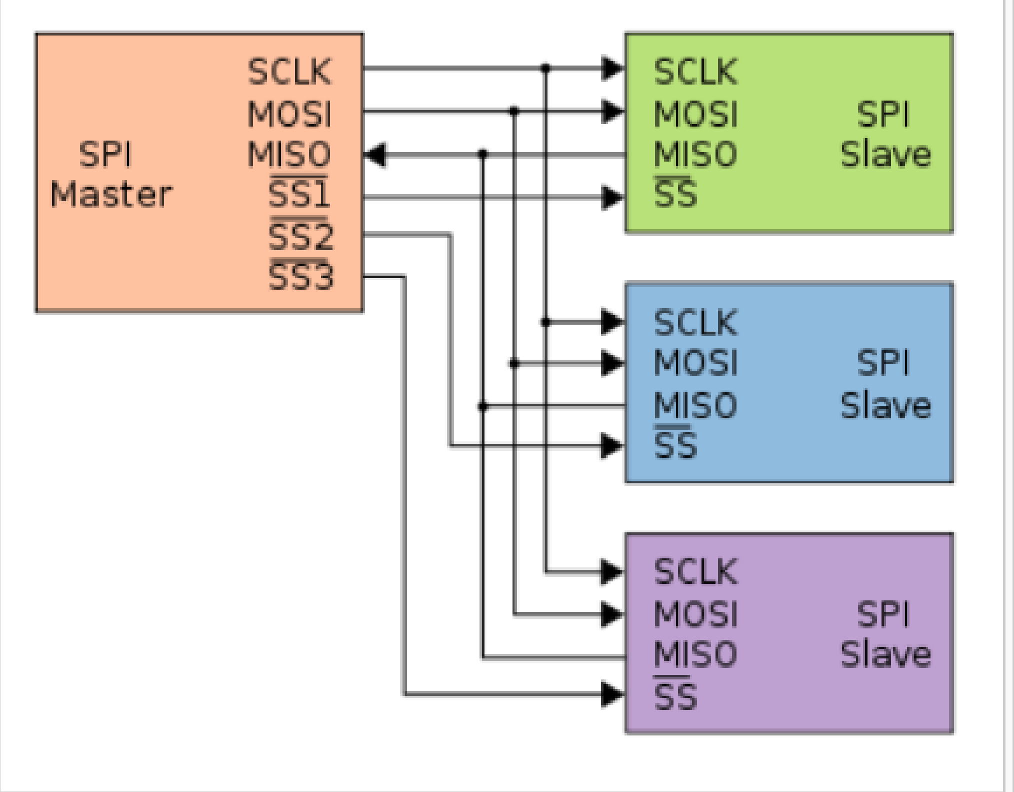
Half-duplex

Data transmission enabling control

Differential transmission electrical interfaces (RS-485)

Synchronous Serial Peripheral Interface: SPI

The **Serial Peripheral Interface Bus** or **SPI** bus is a synchronous serial data link standard that operates in full duplex mode. Devices communicate in master/slave mode where the master device initiates the data frame. Multiple slave devices are allowed with individual slave select (chip select) lines.



Typical SPI bus: master and three independent slaves

The PIC18 MSSP Module

- Has two modes of operation:
  1. Serial peripheral interface (SPI)
  2. Inter-integrated circuit (I^2C)

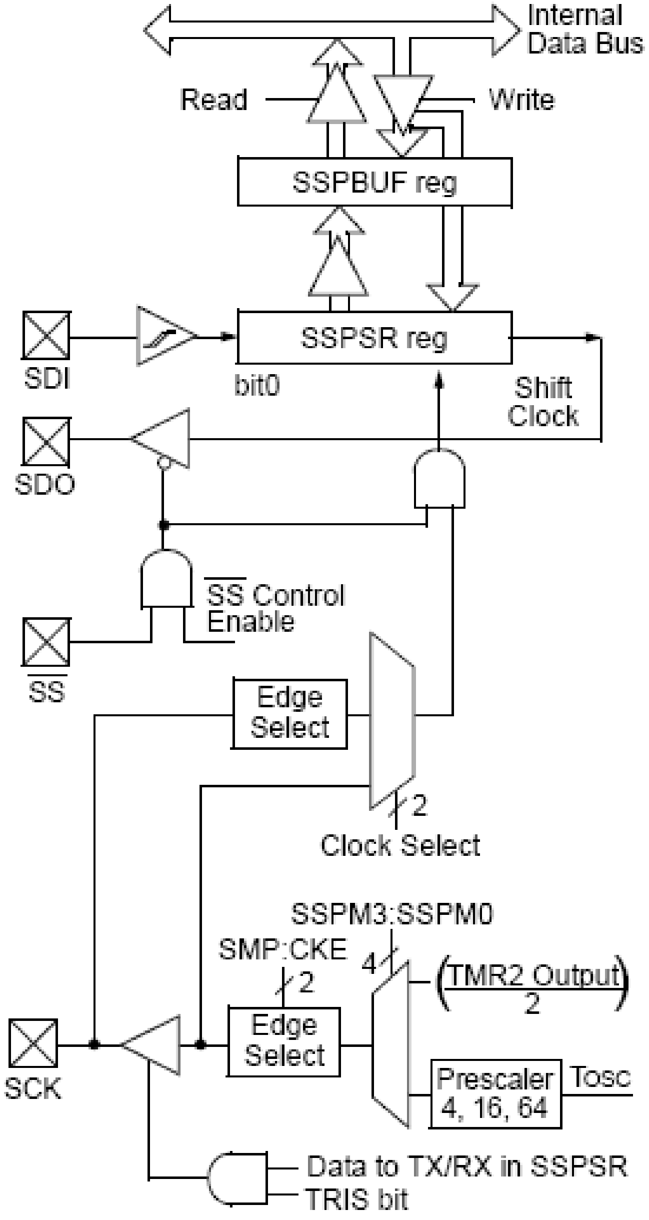
- Can be used to interface with serial EEPROM, shift registers, display drivers, A/D converters, D/A converters, digital temperature sensors, time-of-day chips, etc.

- Devices are divided into the master and slaves in a system that uses either the SPI or I^2C protocol to exchange data.

- The SPI and I^2C module share the same signal pins and cannot to be active at the same time.

- Three pins are used by this module:
  1. Serial data out (SDO)—RC5/SDO
  2. Serial data in (SDI)—RC4/SDI/SDA
  3. Serial clock (SCK)—RC3/SCK/SCL

  A fourth signal pin, RA5/SS (Slave Select), may be used in slave mode



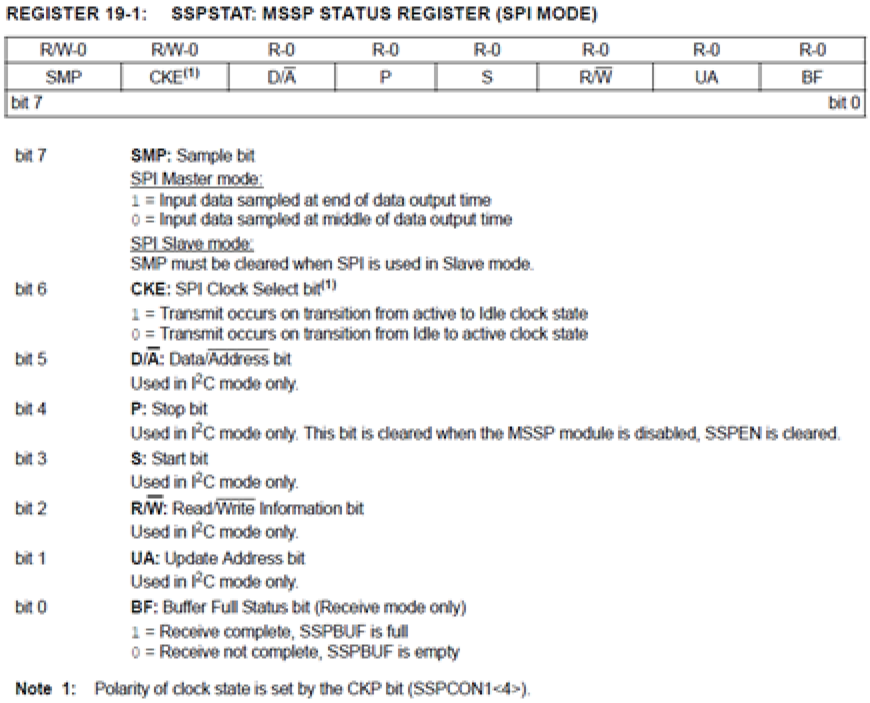
Note: Only those pin functions relevant to SPI operation are shown here.

The SPI Mode

- Eight bits of data are exchanged synchronously in one operation.
- In slave mode, all four signals are used.
- In master mode, the SS pin is not needed.
- Registers for SPI mode operation:

  1. MSSP control register 1 (SSPCON1)
  2. MSSP status register (SSPSTAT)
  3. Serial receive/transmit buffer (SSPBUF)
  4. MSSP shift register (SSPSR) -not directly accessible by the user-

- A write to SSPBUF will also write into the SSPSR register



REGISTER 19-1: SSPSTAT: MSSP STATUS REGISTER (SPI MODE)

R/W-0	R/W-0	R-0	R-0	R-0	R-0	R-0	R-0
SMP	CKE[1]	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF
bit 7							bit 0

bit 7 **SMP:** Sample bit
SPI Master mode:
1 = Input data sampled at end of data output time
0 = Input data sampled at middle of data output time
SPI Slave mode:
SMP must be cleared when SPI is used in Slave mode.

bit 6 **CKE:** SPI Clock Select bit[1]
1 = Transmit occurs on transition from active to Idle clock state
0 = Transmit occurs on transition from Idle to active clock state

bit 5 **D/$\overline{A}$:** Data/$\overline{\text{Address}}$ bit
Used in I²C mode only.

bit 4 **P:** Stop bit
Used in I²C mode only. This bit is cleared when the MSSP module is disabled, SSPEN is cleared.

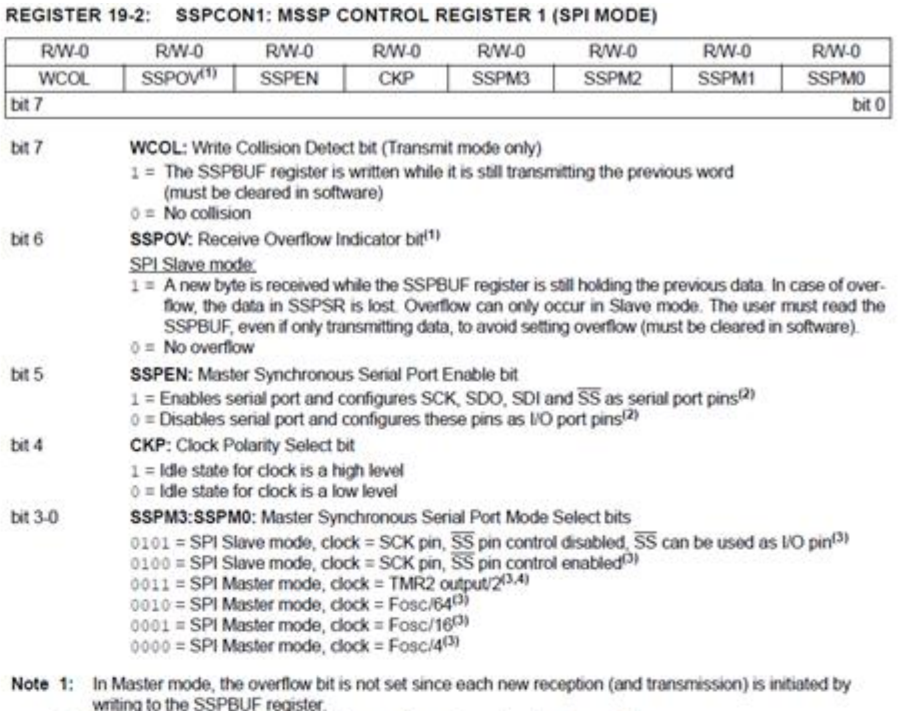
bit 3 **S:** Start bit
Used in I²C mode only.

bit 2 **R/$\overline{W}$:** Read/$\overline{\text{Write}}$ Information bit
Used in I²C mode only.

bit 1 **UA:** Update Address bit
Used in I²C mode only.

bit 0 **BF:** Buffer Full Status bit (Receive mode only)
1 = Receive complete, SSPBUF is full
0 = Receive not complete, SSPBUF is empty

Note 1: Polarity of clock state is set by the CKP bit (SSPCON1<4>).



REGISTER 19-2: SSPCON1: MSSP CONTROL REGISTER 1 (SPI MODE)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
WCOL	SSPOV[1]	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0
bit 7							bit 0

bit 7 **WCOL:** Write Collision Detect bit (Transmit mode only)
1 = The SSPBUF register is written while it is still transmitting the previous word (must be cleared in software)
0 = No collision

bit 6 **SSPOV:** Receive Overflow Indicator bit[1]
SPI Slave mode:
1 = A new byte is received while the SSPBUF register is still holding the previous data. In case of overflow, the data in SSPSR is lost. Overflow can only occur in Slave mode. The user must read the SSPBUF, even if only transmitting data, to avoid setting overflow (must be cleared in software).
0 = No overflow

bit 5 **SSPEN:** Master Synchronous Serial Port Enable bit
1 = Enables serial port and configures SCK, SDO, SDI and $\overline{SS}$ as serial port pins[2]
0 = Disables serial port and configures these pins as I/O port pins[2]

bit 4 **CKP:** Clock Polarity Select bit
1 = Idle state for clock is a high level
0 = Idle state for clock is a low level

bit 3-0 **SSPM3:SSPM0:** Master Synchronous Serial Port Mode Select bits
0101 = SPI Slave mode, clock = SCK pin, $\overline{SS}$ pin control disabled, $\overline{SS}$ can be used as I/O pin[3]
0100 = SPI Slave mode, clock = SCK pin, $\overline{SS}$ pin control enabled[3]
0011 = SPI Master mode, clock = TMR2 output/2[3,4]
0010 = SPI Master mode, clock = Fosc/64[3]
0001 = SPI Master mode, clock = Fosc/16[3]
0000 = SPI Master mode, clock = Fosc/4[3]

Note 1: In Master mode, the overflow bit is not set since each new reception (and transmission) is initiated by writing to the SSPBUF register.

SPI Operation

- A simplified circuit connection between a SPI master and a slave is shown (conceptually is a 16 bits shift register divided in two parts)

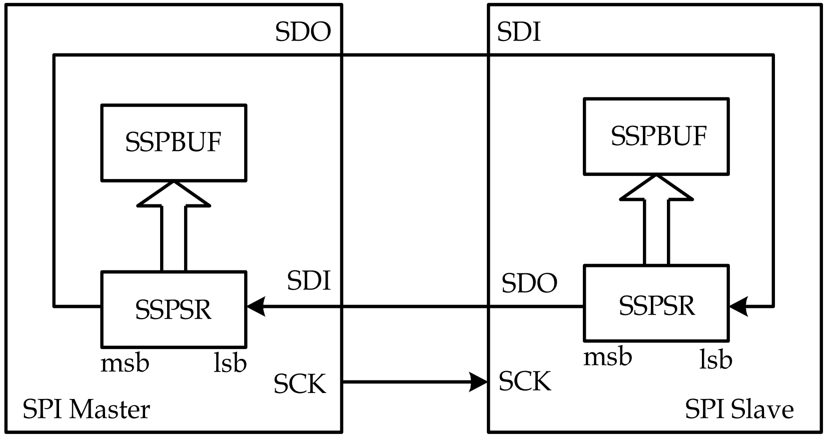


Figure 10.3 Connection between an SPI master and an SPI slave

- The SDO pin of the master is connected to the SDI pin of the slave.
- The SDI pin of the master is connected to the SDO pin of the slave.
- To send data to the slave, the master writes data to the SSPBUF register, after which, eight clock pulses are triggered and data is shifted to the slave (SSPIF and BF bits are set).
- To read data from the slave, the master makes (possibly a dummy) write into the SSPBUF register to trigger eight clock pulses to shift in data, following a SSPBUF read.

Data Shift Rate

- In master mode, the SPI clock rate is programmable to one of the following:

  1. $F_{OSC}/4$ (or F_{CY})
  2. $F_{OSC}/16$ (or $F_{CY}/4$)
  3. $F_{OSC}/64$ (or $F_{CY}/16$)
  4. Timer2 output/2

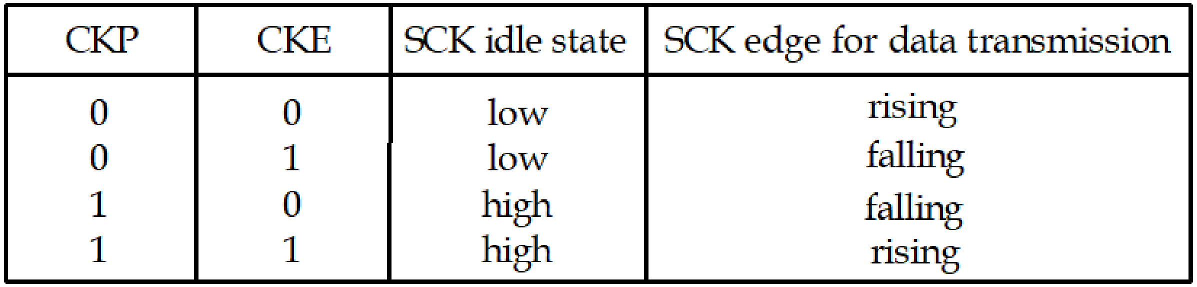
- Data rate is configured by the lowest four bits of the SSPCON1 register.

- The highest data rate is 10 Mbps for 40 MHz crystal oscillator

Clock Edge for Shifting Data

- When the SPI module is not transmitting data, it is referred to as **idle**.
- One can set the SCK signal to be idle low or idle high.
- Setting the **CKP** bit of the SSPCON1 register to 1, makes the SCK signal idle high.
- The **CKE** bit of the SSPSTAT register and the **CKP** bit of the SSPCON1 register together select the edge of the SCK signal for shifting the data:

Table 10.0 SCK idle state and data shifting edge selection



CKP	CKE	SCK idle state	SCK edge for data transmission
0	0	low	rising
0	1	low	falling
1	0	high	falling
1	1	high	rising

- One can choose to use the middle or the end of a bit time to sample the incoming data.
- When the **SMP** bit of the SSPSTAT register is 1, incoming data is sampled at the end of the bit time. Otherwise, incoming data is sampled at the middle of a bit time.

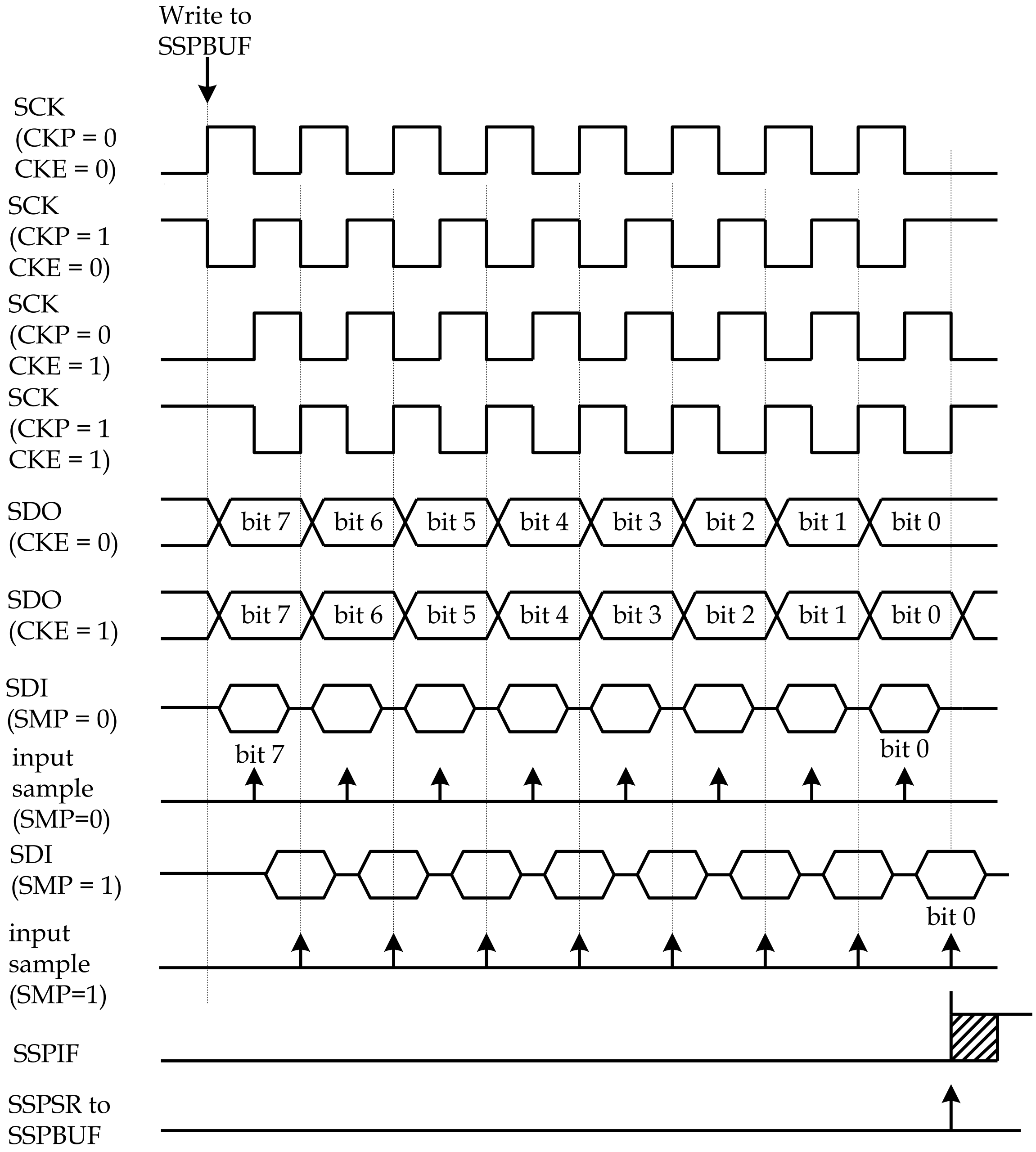


Figure 10.4a SPI Mode waveform (master mode) (redraw with permission of Microchip)

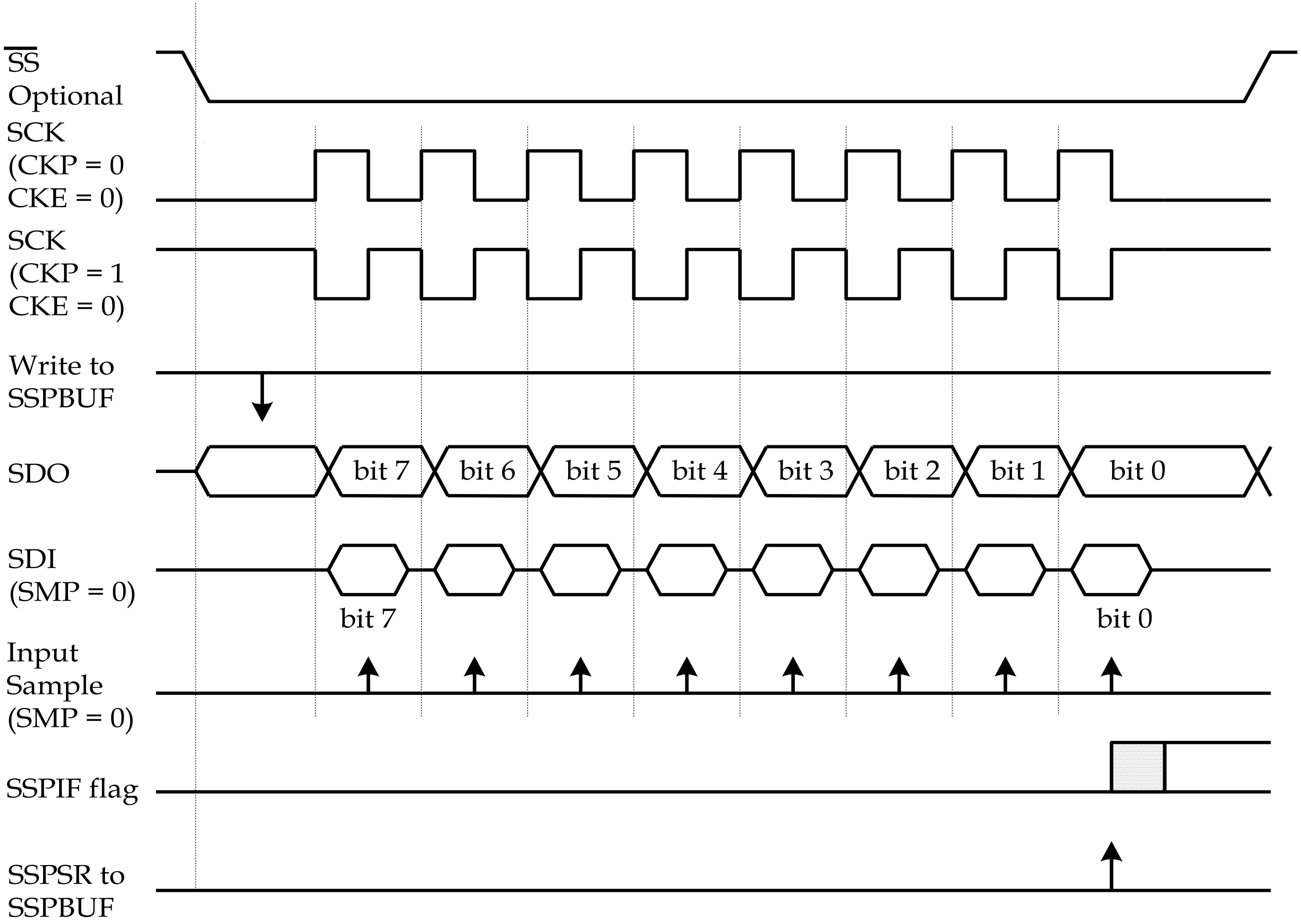


Figure 10.4b SPI clock format (slave mode with CKE = 0) (redraw with permission of Microchip)

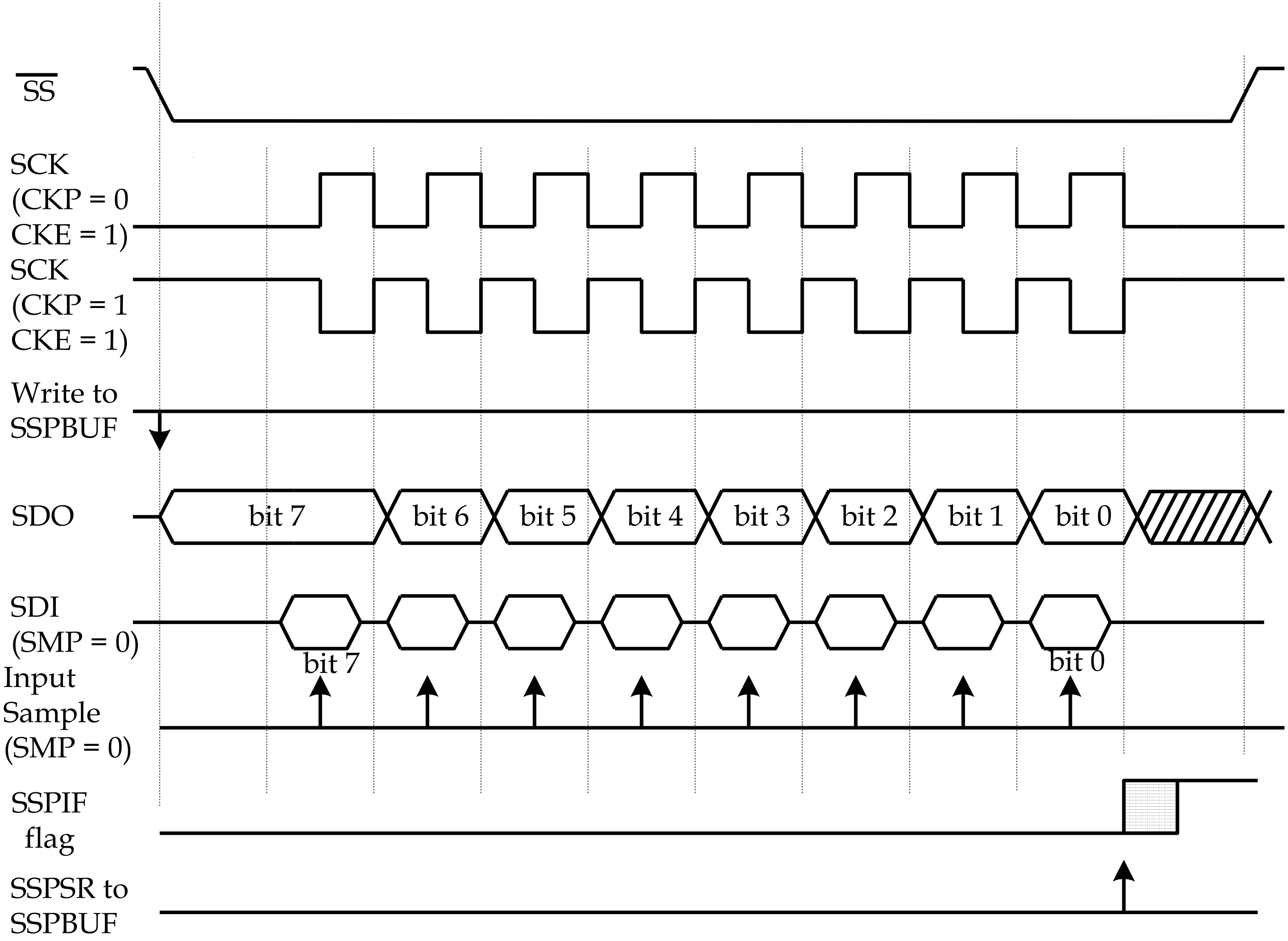
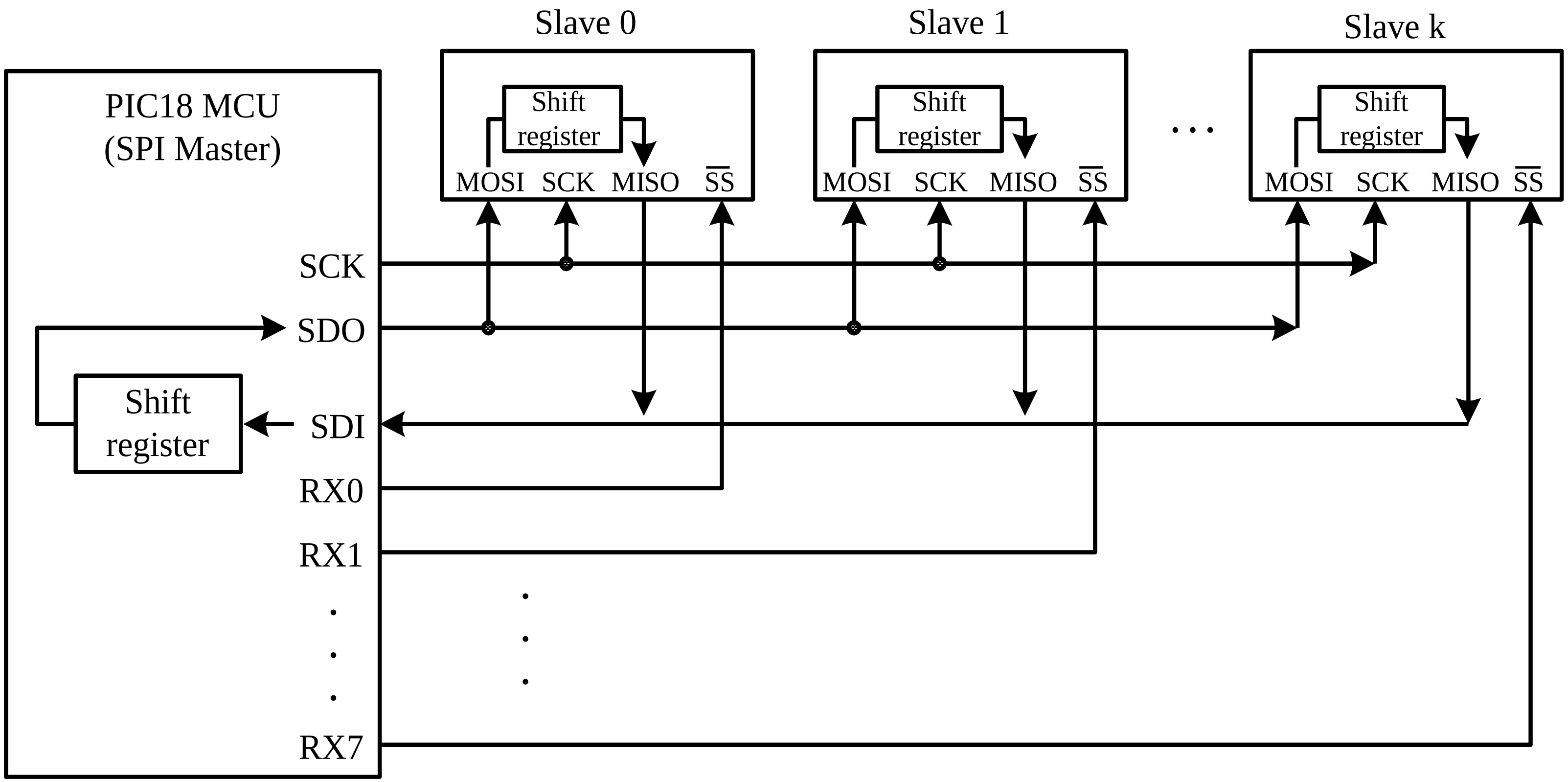


Figure 10.4c SPI clock format (slave mode with CKE = 1)
(redraw with permission of Microchip)

SPI Circuit Connection

- There are many possibilities for connecting an SPI master to multiple SPI slaves.
- Two connection methods are shown in the next slides
- The method shown in the next slide requires the use of port pins to select one of the SPI slave to perform the data transfer.
- The method shown in second slide concatenate all the slaves into a single ring. This last method does not require the use of port pins to select SPI slave device. !



Note: RX is an unused I/O port
MOSI stands for master out, slave in
MISO stands for master in, slave out

Figure 10.5 Single-master and multiple-slave device connection (method 1)

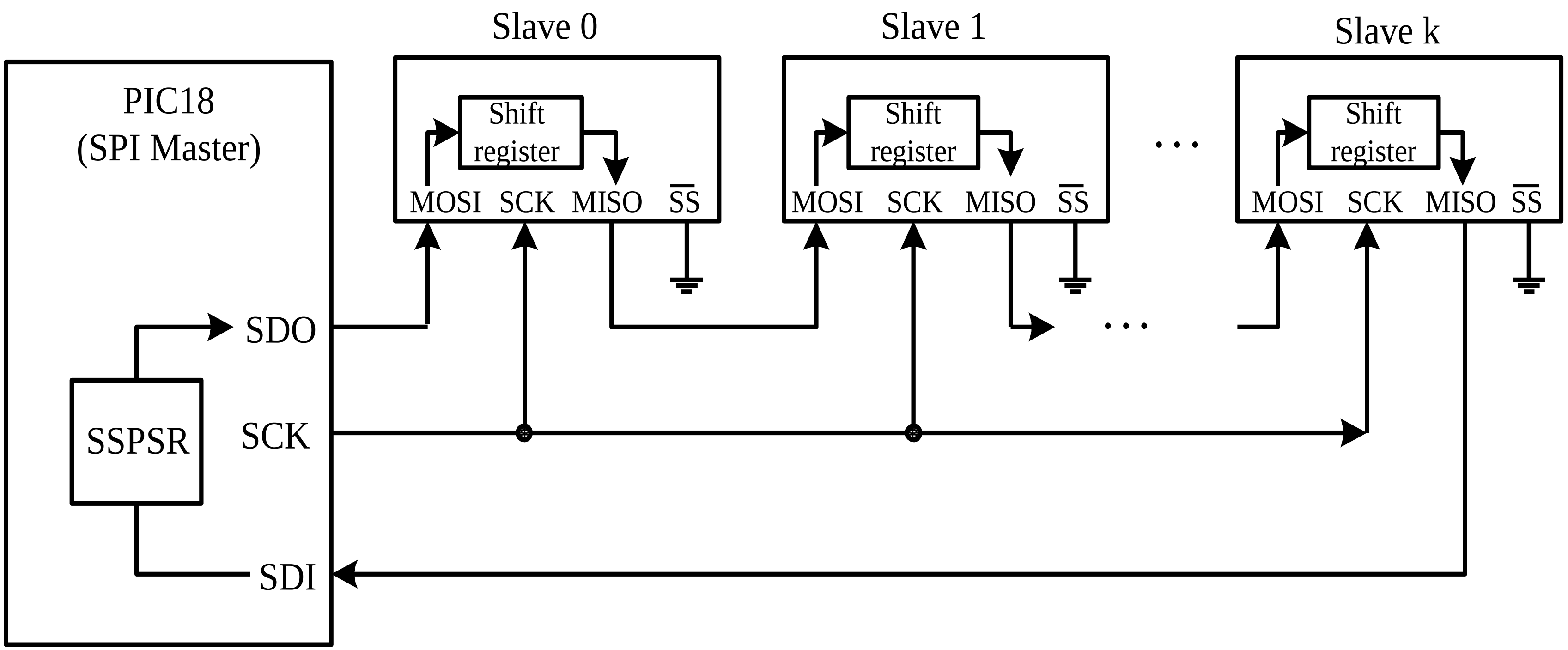


Figure 10.6 Single-master and multiple-slave device connection (method 2) !

Ex. The next figure shows PIC18 MCU and TC72 chip for digital temperature reading. Write a C program to read the temperature every 200 ms. Convert the temperature value into a string so that it can be displayed in an appropriate output device. A pointer to the buffer to hold the string will be passed to this function. The crystal oscillator of the PIC18 is assumed to be 16 MHz.

See TC72 datasheet at
http://ww1.microchip.com/downloads/en/devicedoc/21743a.pdf

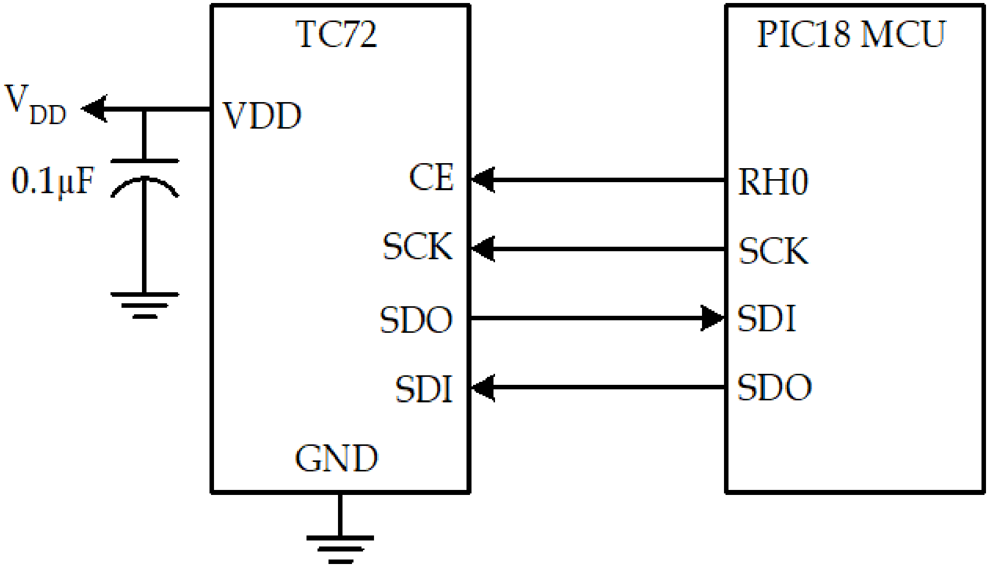
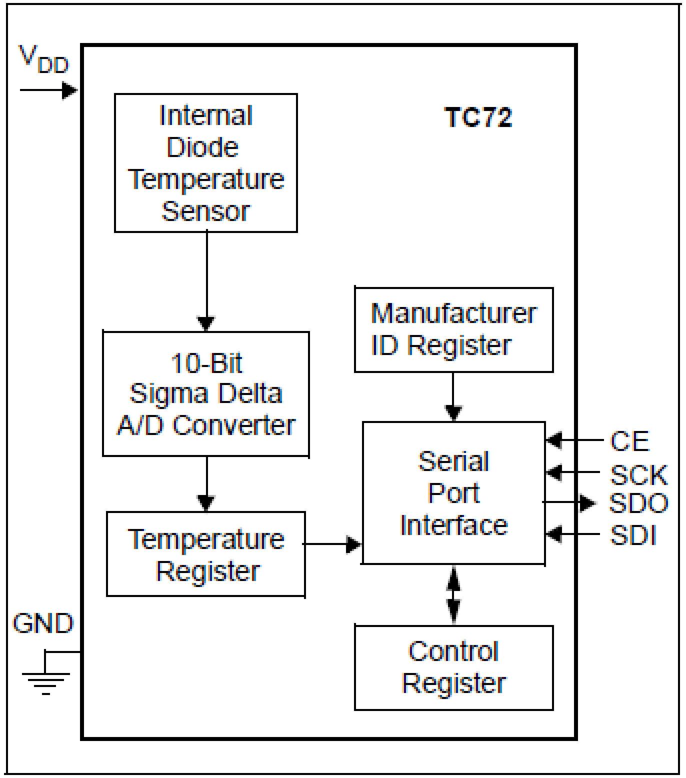


Figure 10.18 Circuit connection between the TC72 and PIC18 MCU on the SSE8720 demo board

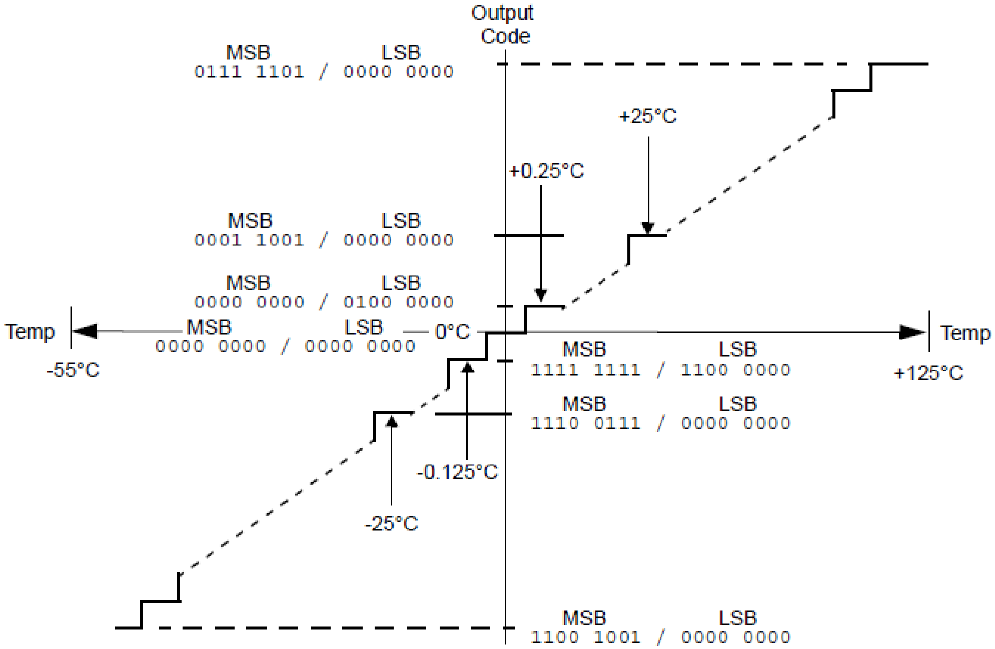
TC72 datasheet

Block Diagram



Typical Applications

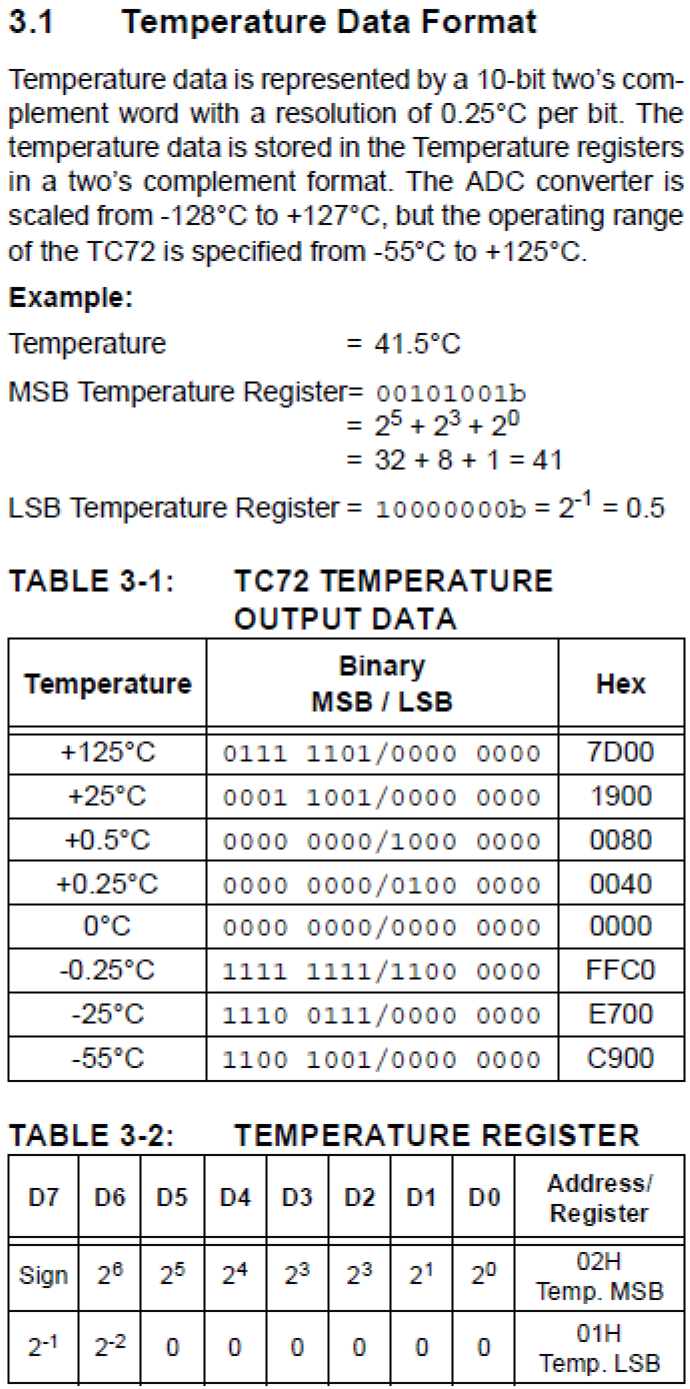
- Personal Computers and Servers
- Hard Disk Drives and Other PC Peripherals
- Entertainment Systems
- Office Equipment
- Datacom Equipment
- Mobile Phones



Note: The ADC converter is scaled from -128°C to -127°C, but the operating range of the TC72 is specified from -55°C to +125°C.

TC72 datasheet.

(cont.)



3.1 Temperature Data Format

Temperature data is represented by a 10-bit two's complement word with a resolution of 0.25°C per bit. The temperature data is stored in the Temperature registers in a two's complement format. The ADC converter is scaled from -128°C to +127°C, but the operating range of the TC72 is specified from -55°C to +125°C.

Example:

Temperature = 41.5°C

MSB Temperature Register= `00101001b`
= $2^5 + 2^3 + 2^0$
= 32 + 8 + 1 = 41

LSB Temperature Register = `10000000b` = 2^{-1} = 0.5

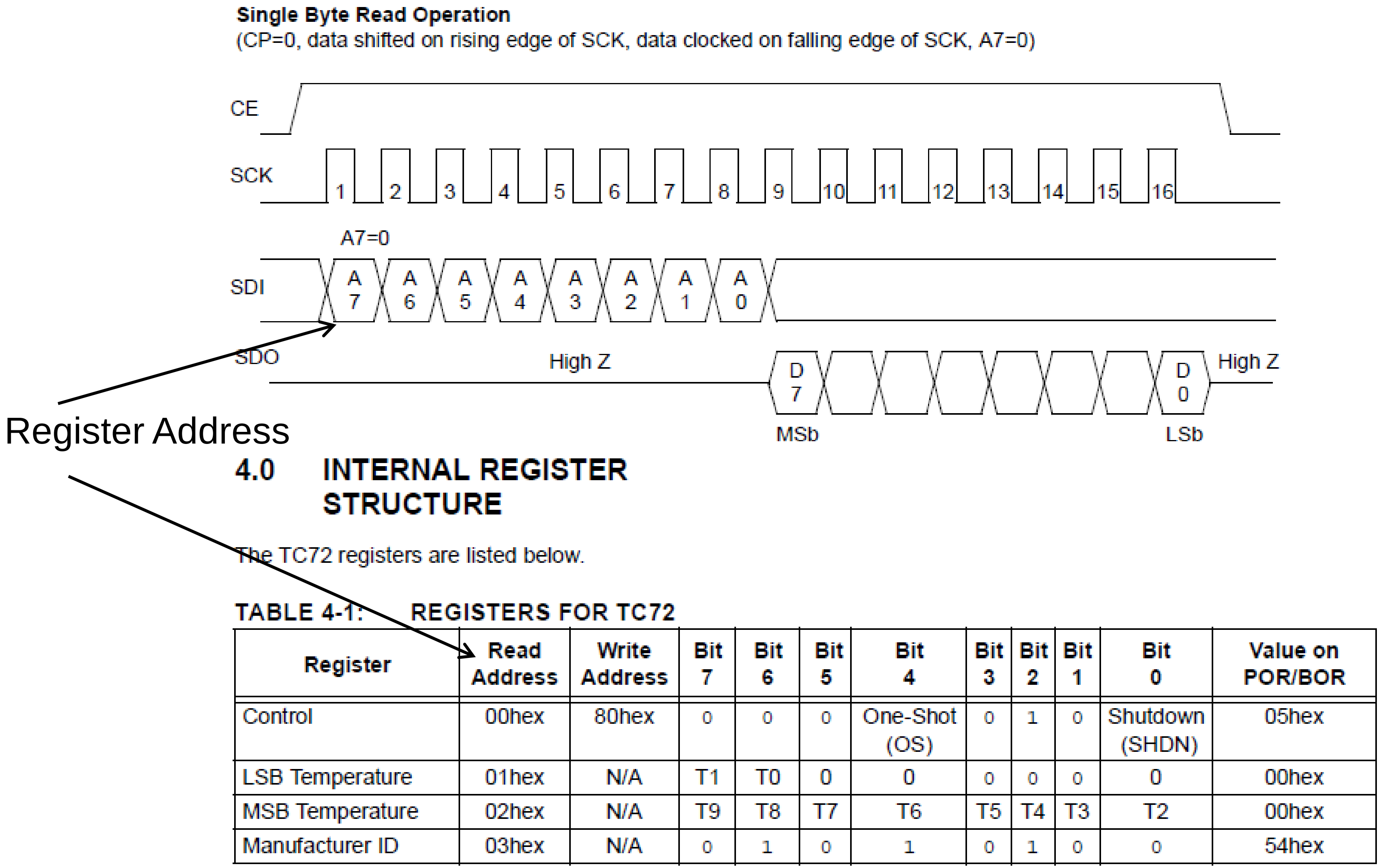
TABLE 3-1: TC72 TEMPERATURE OUTPUT DATA

Temperature	Binary MSB / LSB	Hex
+125°C	0111 1101/0000 0000	7D00
+25°C	0001 1001/0000 0000	1900
+0.5°C	0000 0000/1000 0000	0080
+0.25°C	0000 0000/0100 0000	0040
0°C	0000 0000/0000 0000	0000
-0.25°C	1111 1111/1100 0000	FFC0
-25°C	1110 0111/0000 0000	E700
-55°C	1100 1001/0000 0000	C900

TABLE 3-2: TEMPERATURE REGISTER

D7	D6	D5	D4	D3	D2	D1	D0	Address/ Register
Sign	2^6	2^5	2^4	2^3	2^3	2^1	2^0	02H Temp. MSB
2^{-1}	2^{-2}	0	0	0	0	0	0	01H Temp. LSB

TC72 datasheet (cont.)



Single Byte Read Operation
(CP=0, data shifted on rising edge of SCK, data clocked on falling edge of SCK, A7=0)

CE
SCK
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
A7=0
SDI
A 7
A 6
A 5
A 4
A 3
A 2
A 1
A 0
SDO
High Z
D 7
D 0
High Z
MSb
LSb

Register Address

4.0 INTERNAL REGISTER STRUCTURE

The TC72 registers are listed below.

TABLE 4-1: REGISTERS FOR TC72

Register	Read Address	Write Address	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR/BOR
Control	00hex	80hex	0	0	0	One-Shot (OS)	0	1	0	Shutdown (SHDN)	05hex
LSB Temperature	01hex	N/A	T1	T0	0	0	0	0	0	0	00hex
MSB Temperature	02hex	N/A	T9	T8	T7	T6	T5	T4	T3	T2	00hex
Manufacturer ID	03hex	N/A	0	1	0	1	0	1	0	0	54hex

DS1306
Alarm Real Time Clock

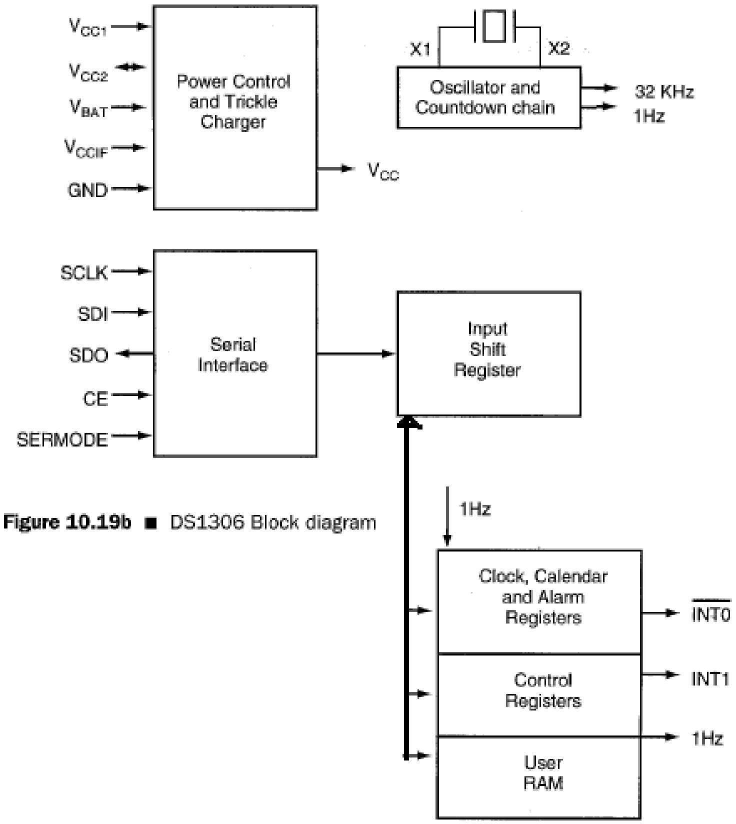
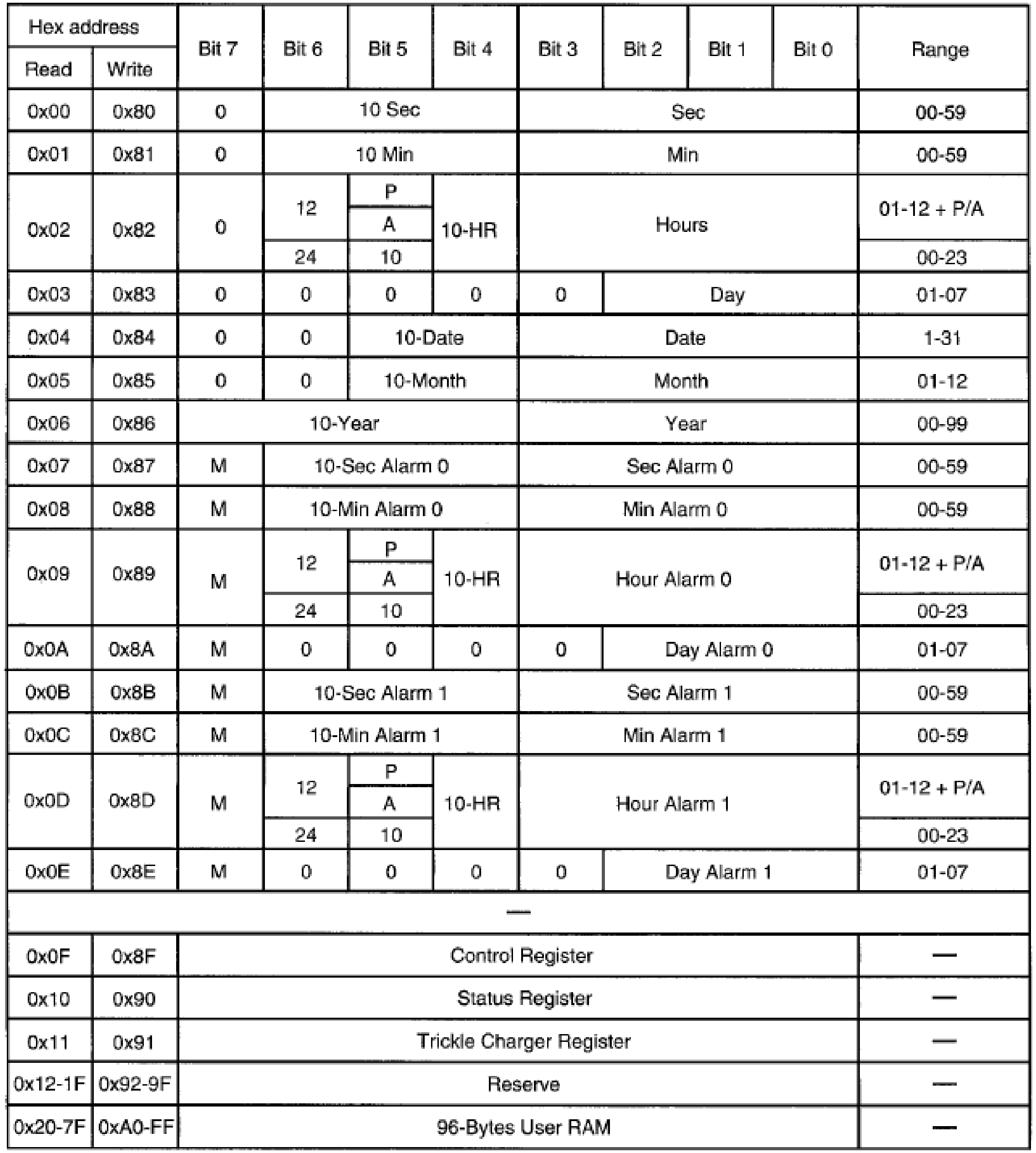


Figure 10.19b ■ DS1306 Block diagram



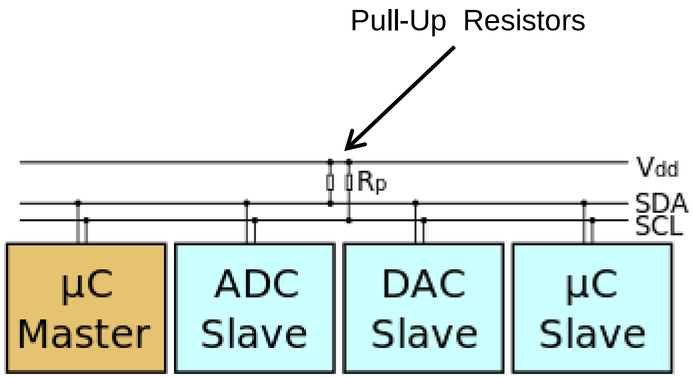
Hex address		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Range
Read	Write									
0x00	0x80	0	10 Sec			Sec				00-59
0x01	0x81	0	10 Min			Min				00-59
0x02	0x82	0	12 24	P A 10	10-HR	Hours				01-12 + P/A 00-23
0x03	0x83	0	0	0	0	0	Day			01-07
0x04	0x84	0	0	10-Date		Date				1-31
0x05	0x85	0	0	10-Month		Month				01-12
0x06	0x86	10-Year				Year				00-99
0x07	0x87	M	10-Sec Alarm 0			Sec Alarm 0				00-59
0x08	0x88	M	10-Min Alarm 0			Min Alarm 0				00-59
0x09	0x89	M	12 24	P A 10	10-HR	Hour Alarm 0				01-12 + P/A 00-23
0x0A	0x8A	M	0	0	0	0	Day Alarm 0			01-07
0x0B	0x8B	M	10-Sec Alarm 1			Sec Alarm 1				00-59
0x0C	0x8C	M	10-Min Alarm 1			Min Alarm 1				00-59
0x0D	0x8D	M	12 24	P A 10	10-HR	Hour Alarm 1				01-12 + P/A 00-23
0x0E	0x8E	M	0	0	0	0	Day Alarm 1			01-07
—										
0x0F	0x8F	Control Register								—
0x10	0x90	Status Register								—
0x11	0x91	Trickle Charger Register								—
0x12-1F	0x92-9F	Reserve								—
0x20-7F	0xA0-FF	96-Bytes User RAM								—

Note. Range for alarm registers does not include mask 'm' bits

Figure 10.20 ■ RTC registers and address map

Synchronous serial interface I2C

I2C, I^{2}C ***Inter-Integrated Circuit Bus***



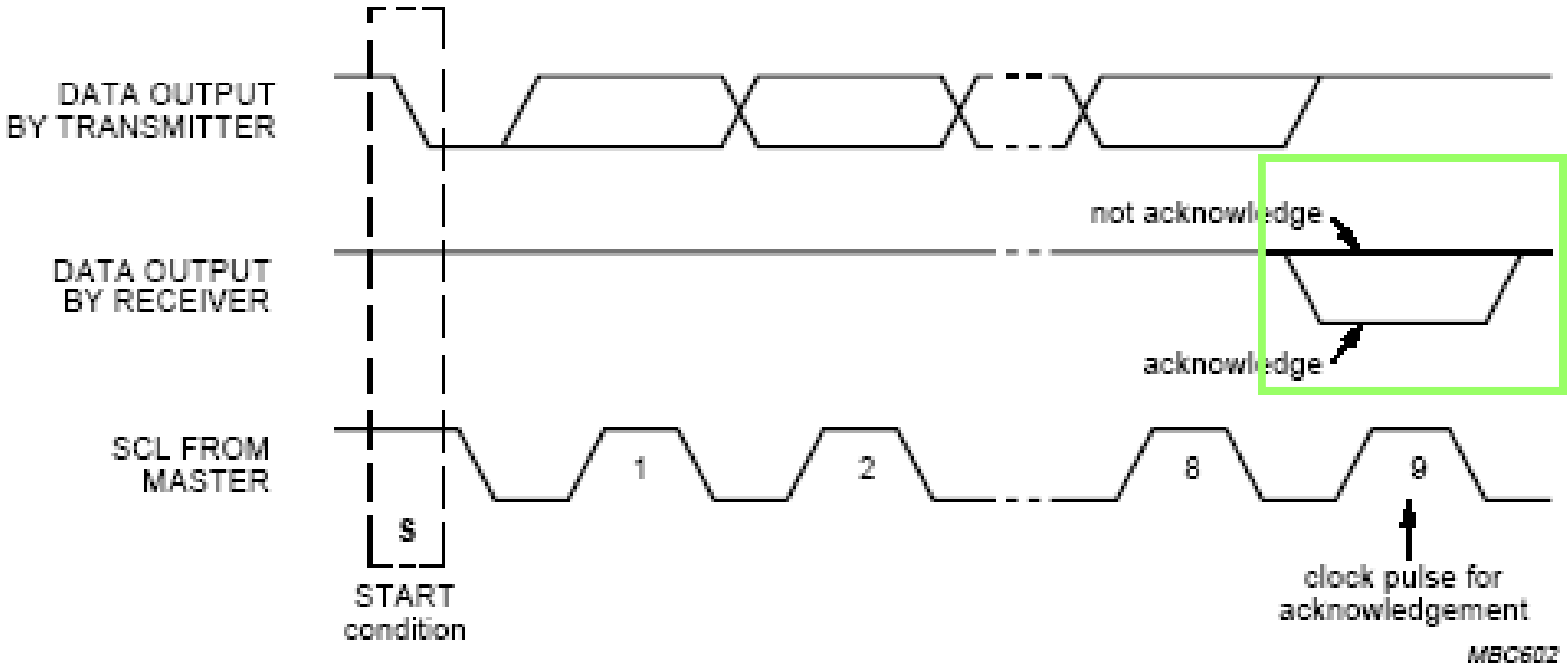
Pull-Up Resistors

Serial synchronous half-duplex bus (1992)

Only two signal lines SDA and SCL plus supply voltage and ground are required to be connected.

Common I^{2}C bus speeds are the 400kbit/s fast mode, the100 kbit/s *standard mode* and the 10 kbit/s *low-speed mode*, but arbitrarily low clock frequencies are also allowed.

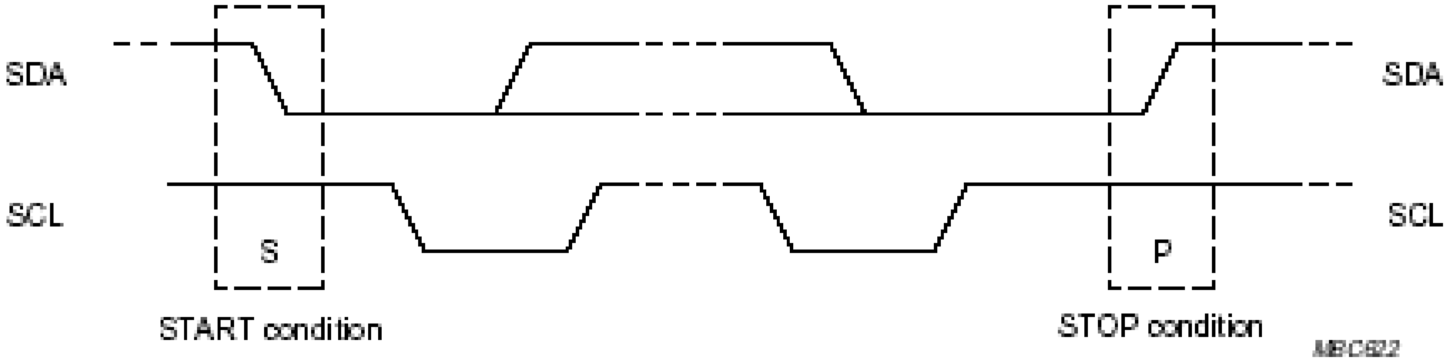
I^{2}C *Inter-Integrated Circuit Bus*



Timing and bit acknowledgment.

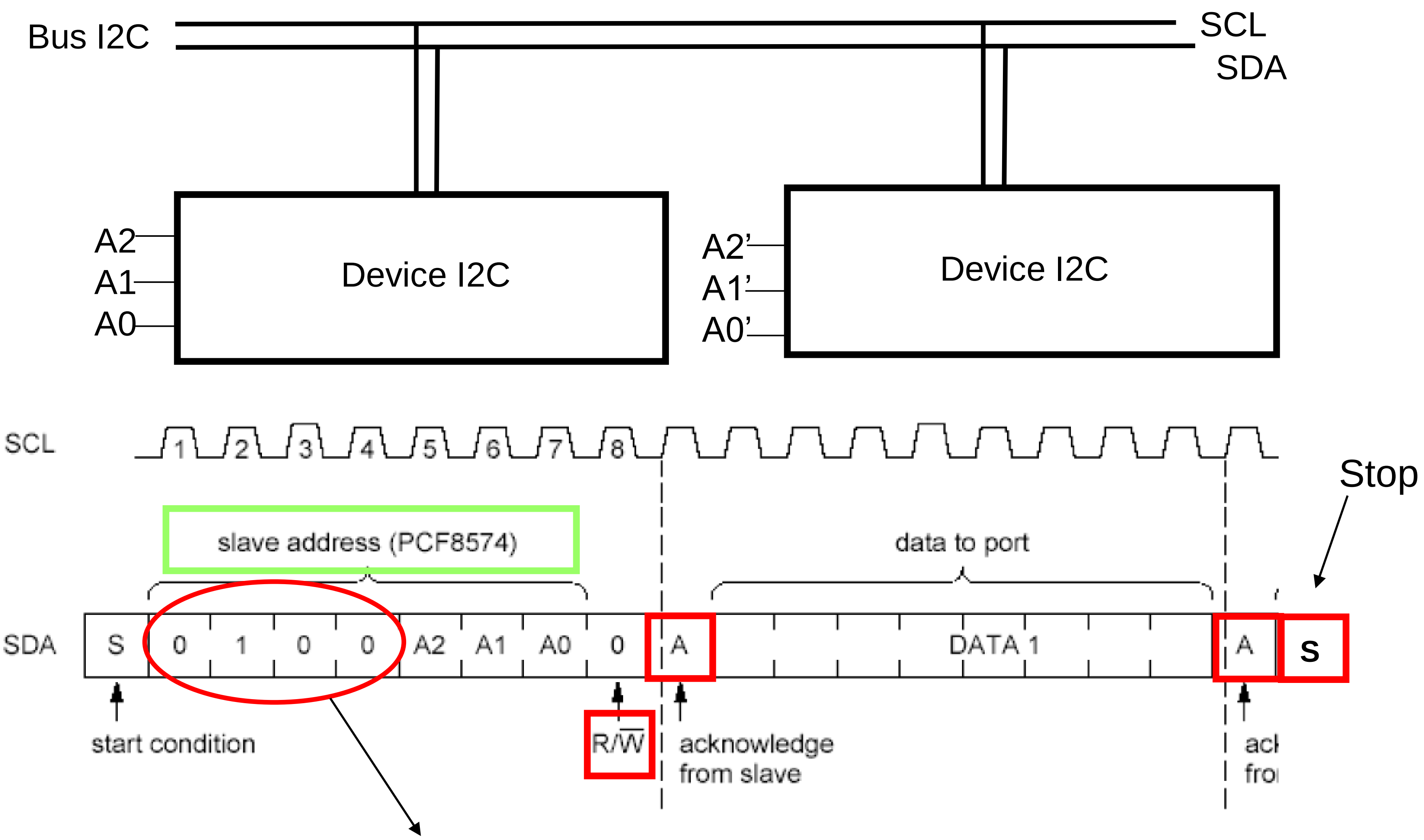
I^2C *Inter-Integrated Circuit Bus*

Data transfer is initiated with the START condition (**S**) when SDA is pulled low while SCL stays high. Then, SDA sets the transferred bit while SCL is low and the data is sampled (received) when SCL rises. When the transfer is complete, a STOP bit (**P**) is sent by releasing the data line to allow it to be pulled up while SCL is constantly high.

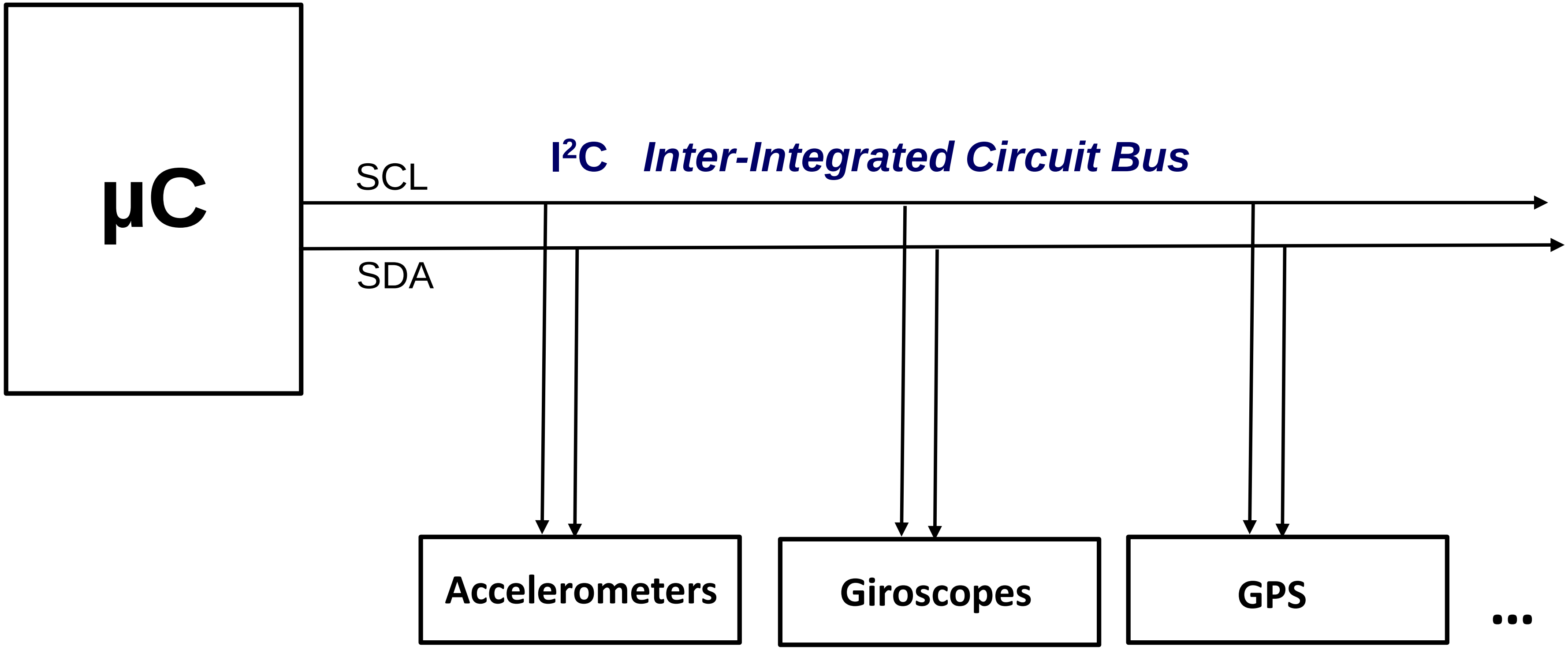


Start and Stop signaling

I^2C *Inter-Integrated Circuit Bus*



4 bits factory written

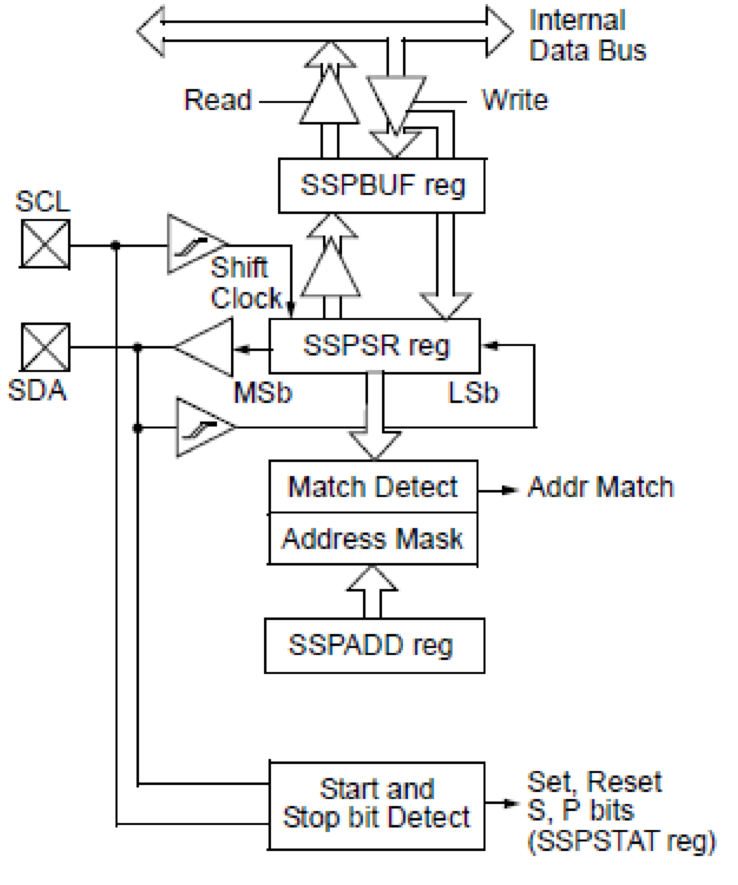


I^{2}C *Inter-Integrated Circuit Bus*

I2C possible applications: car navigation control

I^{2}C (MSSP) Module in the PIC18F

FIGURE 19-7: MSSP BLOCK DIAGRAM (I^{2}C™ MODE)



Note: Only port I/O names are used in this diagram for the sake of brevity. Refer to the text for a full list of multiplexed functions.

Registers in the MSSP in I^{2}C mode:

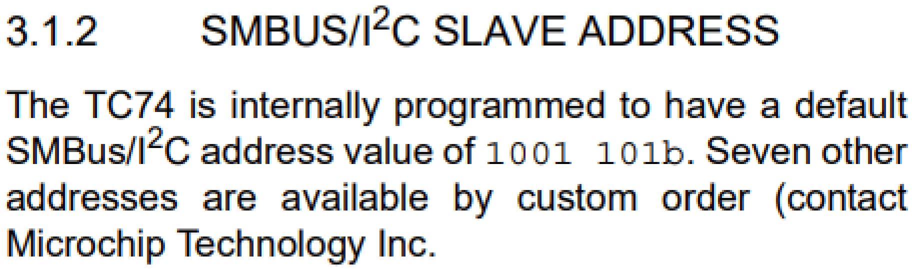
- SSPCON1
- SSPCON2
- SSPSTAT
- SSPBUF
- SSPADD (slave add. or Master Clk rate)

I^{2}C Example

Using a microchip TC74 temperature sensor

See TC74 datasheet at https://ww1.microchip.com/downloads/aemDocuments/documents/APID/ProductDocuments/DataSheets/21462D.pdf

TC74 datasheet

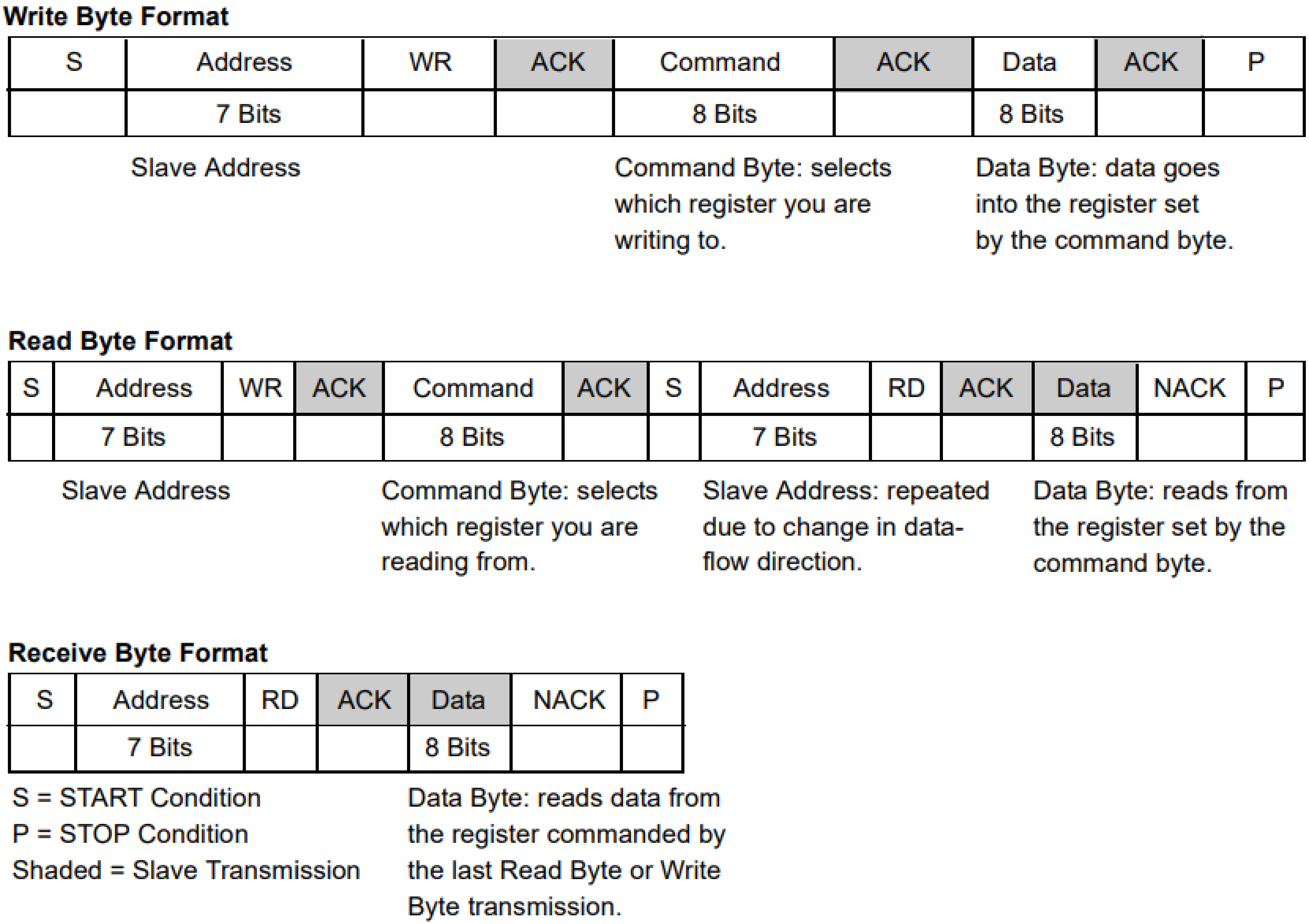


3.1.2 SMBUS/I^{2}C SLAVE ADDRESS

The TC74 is internally programmed to have a default SMBus/I^{2}C address value of `1001 101b`. Seven other addresses are available by custom order (contact Microchip Technology Inc.

TC74 datasheet

I^{2}C Example



Write Byte Format

S	Address	WR	ACK	Command	ACK	Data	ACK	P
	7 Bits			8 Bits		8 Bits		

Slave Address

Command Byte: selects which register you are writing to.

Data Byte: data goes into the register set by the command byte.

Read Byte Format

S	Address	WR	ACK	Command	ACK	S	Address	RD	ACK	Data	NACK	P
	7 Bits			8 Bits			7 Bits			8 Bits		

Slave Address

Command Byte: selects which register you are reading from.

Slave Address: repeated due to change in dataflow direction.

Data Byte: reads from the register set by the command byte.

Receive Byte Format

S	Address	RD	ACK	Data	NACK	P
	7 Bits			8 Bits		

S = START Condition
P = STOP Condition
Shaded = Slave Transmission

Data Byte: reads data from the register commanded by the last Read Byte or Write Byte transmission.

TC74 datasheet

I^{2}C Example

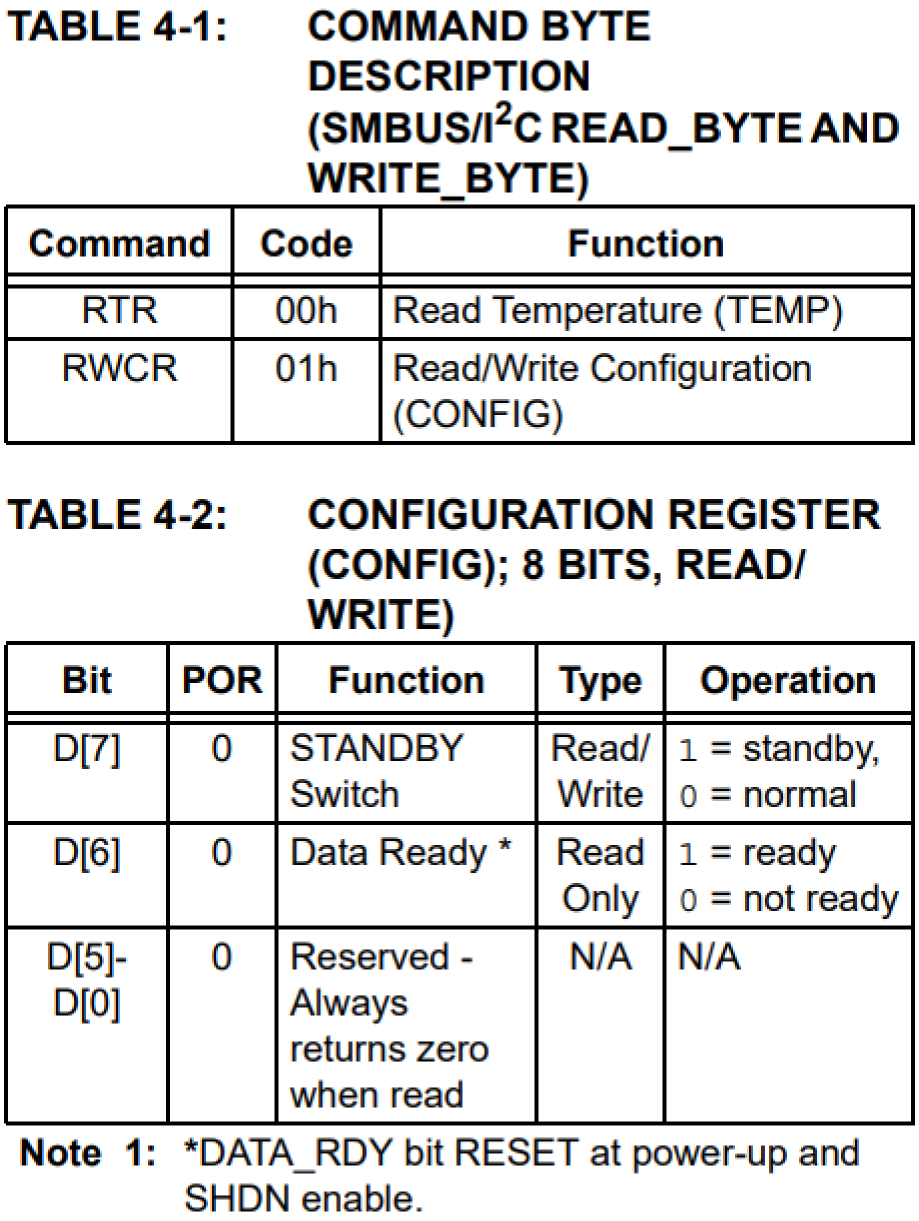


TABLE 4-1: COMMAND BYTE DESCRIPTION (SMBUS/I^{2}C READ_BYTE AND WRITE_BYTE)

Command	Code	Function
RTR	00h	Read Temperature (TEMP)
RWCR	01h	Read/Write Configuration (CONFIG)

TABLE 4-2: CONFIGURATION REGISTER (CONFIG); 8 BITS, READ/ WRITE)

Bit	POR	Function	Type	Operation
D[7]	0	STANDBY Switch	Read/ Write	1 = standby, 0 = normal
D[6]	0	Data Ready *	Read Only	1 = ready 0 = not ready
D[5]-D[0]	0	Reserved - Always returns zero when read	N/A	N/A

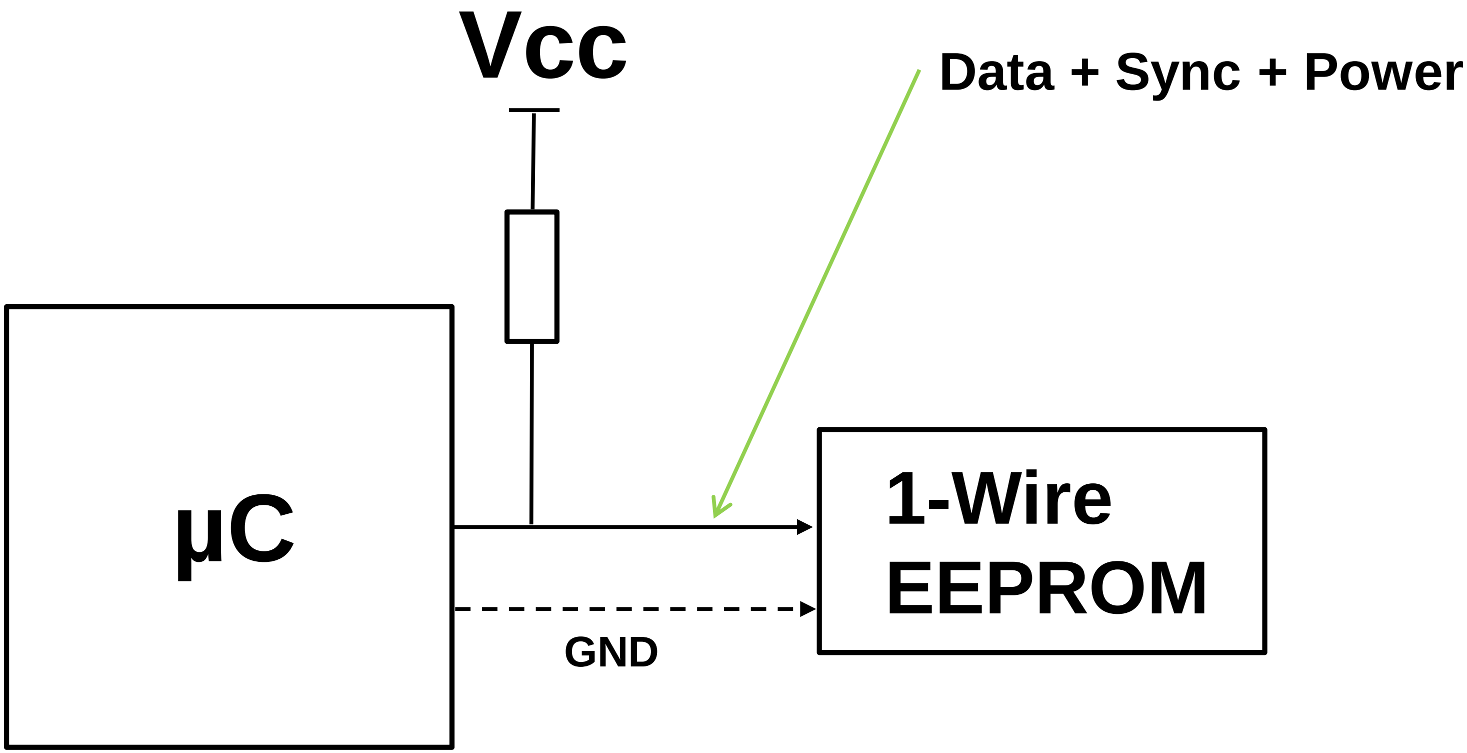
Note 1: *DATA_RDY bit RESET at power-up and SHDN enable.

I2C Read more at

https://www.nxp.com/docs/en/user-guide/UM10204.pdf

Asynchronous serial interfaces (1Wire)

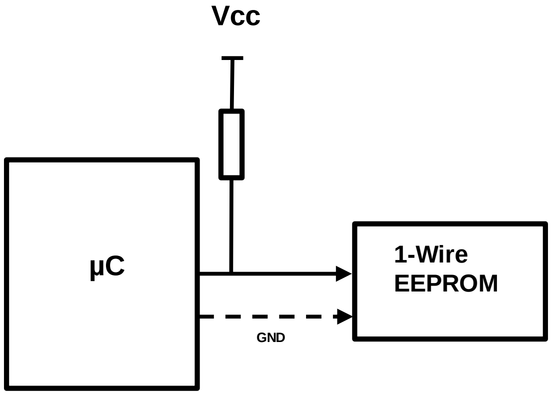
1 Wire: Bidirectional, half-duplex, serial communication that powers over a single connection and ground return. Two serial communication speeds 15Kbps or 125kbps. Unique Unalterable ID in every device !!!



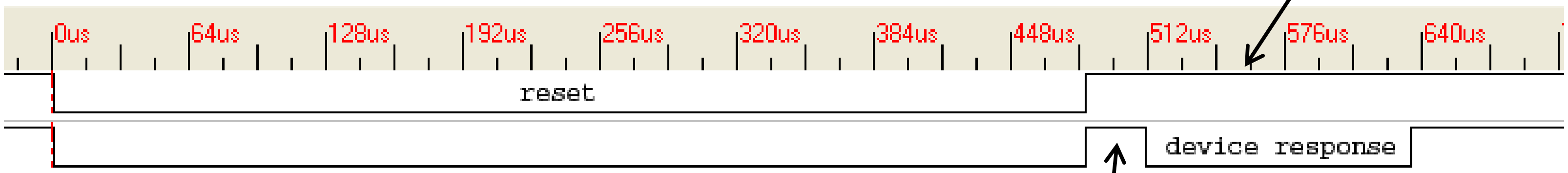
1Wire possible application: door key

1Wire Device Presence Pulse

The master starts a transmission with a "reset" pulse, which pulls the wire to 0 volts for 480 µs. This resets every slave device on the bus, probably by depriving them all of power. After that, any slave device, if present, shows that it exists with a "presence" pulse: it holds the wire to ground for at least 60 µs after the master releases the bus.



No device



Port bit configured as input

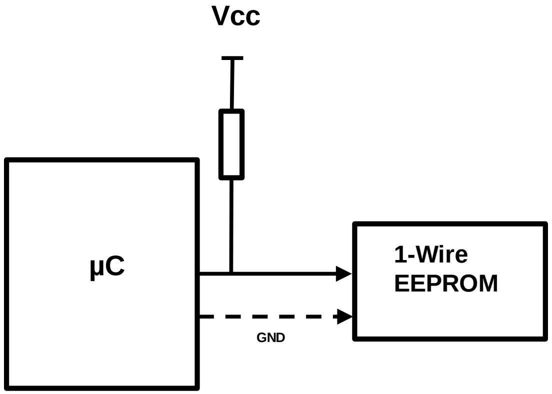
At least, one
device is present

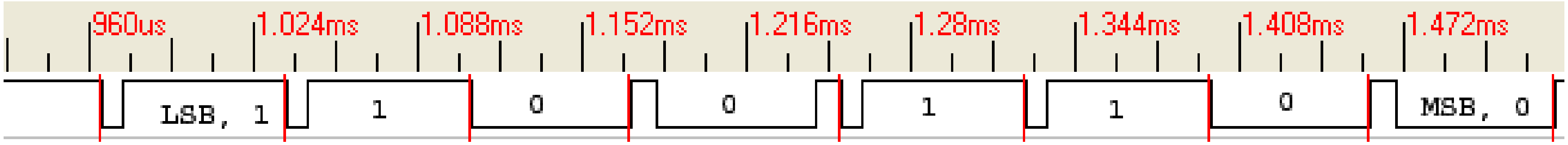
Port bit configured as output
Port bit = 0

The pull-up resistor charges the device

1Wire. Sending bits…

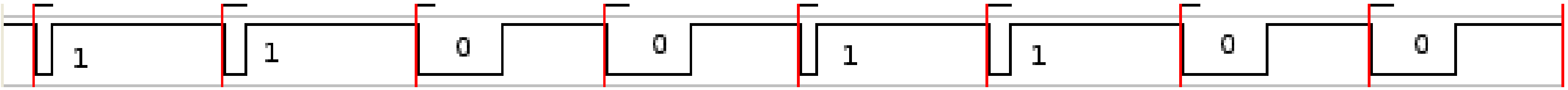
To send a "1", the master sends a very brief (1 - 15 µs) low pulse. To send a "0", the master sends a 60 µs low pulse.





1Wire. Receiving bits…

When receiving data, the master sends a 1-15 µs 0 volt pulse to start each bit. If the transmitting slave unit wants to send a "1", it does nothing, and the wire goes immediately up to the pulled-up voltage. If the transmitting slave wants to send a "0", it pulls the data line to ground for at least 15 µs.



See 1Wire overview video at

https://www.maximintegrated.com/en/products/1-wire/flash/overview/