University of Puerto Rico at Mayagüez
Department of Electrical and Computer Engineering
Microprocessor Interfacing Laboratory

Experiment 5: Introduction to Serial Communications

Objectives

- Become familiar with standard formats for serial communication.
- Understand the basic anatomy and operation of an USART interface.
- Become familiar with USART baud rates and flow control modes.
- Understand how to program and operate interfaces for asynchronous and synchronous serial communication.

Duration

- 2 Hours in laboratory and extra time for homework.

Materials

- Your Microcontroller IDE application
- Your Microcontroller development board
- USB to UART cable
- List of parts attached to this Laboratory

Introduction

In a serial communication channel data is transmitted sequentially, this is, one bit at a time. Data transmission and reception is synchronized using a clock signal, which depending on the synchronization mode could or could not be transmitted. Different to parallel communication that needs one data line for each bit to be transmitted, a serial link only needs one line for transmission (plus ground). This allows reducing communication lines between systems.

Serial communication requires a protocol to reliably transmit and receive information. It also requires hardware components to convert parallel data into serial and vise versa at the transmitting and receiving ends. The most basic requirements in a wired, full duplex serial channel call for a transmit signal (TxD), a receive signal (RxD), and a ground reference, as illustrated in Figure 1. Depending on the standard, additional signals might also be necessary for handshaking, synchronization, or regulating the data flow. Common serial protocols and standards include: RS-232, RS-485, SPI, I2C, 1-wired, etc.

Synchronous Vs. Asynchronous Serial Communication:

Depending on how the transmission clock is handled in a serial channel, the communication can be synchronous or asynchronous. In synchronous communication the signal clock is transmitted from the transmitter to the receiver, as illustrated in Figure 1a. In asynchronous communications, individual clock generators are used on both ends of the channel, as shown in Figure 1b.

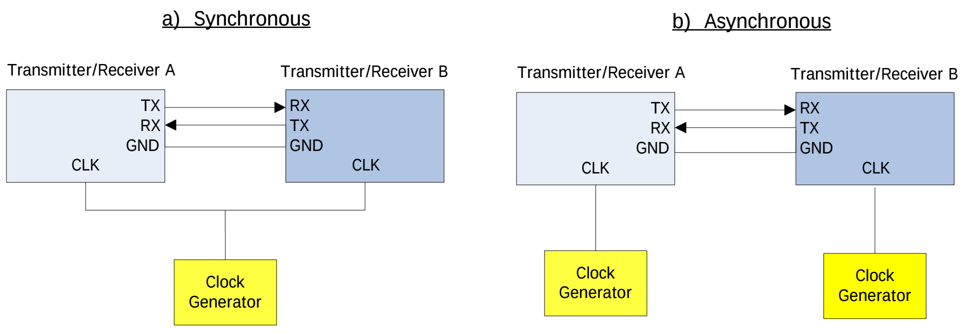


Figure 1: Synchronous vs. Asynchronous Serial Communication.

Serial interfaces:

A Universal Synchronous/Asynchronous Receiver/Transmitter (USART) controller is a fundamental part of the serial communications subsystem of an embedded device. An embedded USART as it is implied by its name can generate the necessary signals for synchronous or asynchronous communication.

typical frame for asynchronous serial communication contains a start bit, multiple data bits, an optional parity bit, and a stop bit, as illustrated in Figure 2. The start bit is followed by the data to be transferred. An optional parity or error check bit can follow. The frame ends with a stop bit, to indicate the channel is idle. These data bits within a frame are serialized (transmit side) or parallelized (receive side) under the control of a clock source. The clock source is supplied by a baud rate generator.

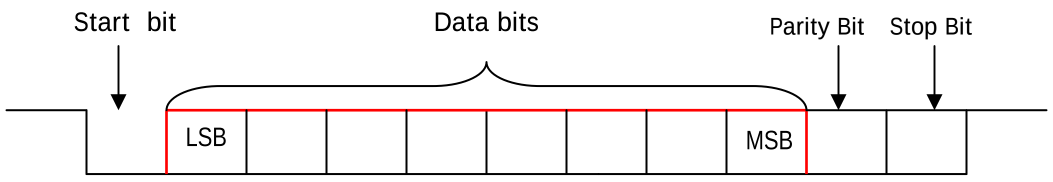


Figure 2: Typical asynchronous serial transmission frame.

A USART contains multiple functional units and registers which may vary from one architecture to another. However, there a few basic components, as illustrated in Figure 3, which are fundamental for the USART operation. These include:

– **Baud Rate Generator**: Generates the clock frequency necessary for the transmission and reception speed or baud rate. The clock signal may be transmitted in a separate line depending if it is synchronous or asynchronous communication.

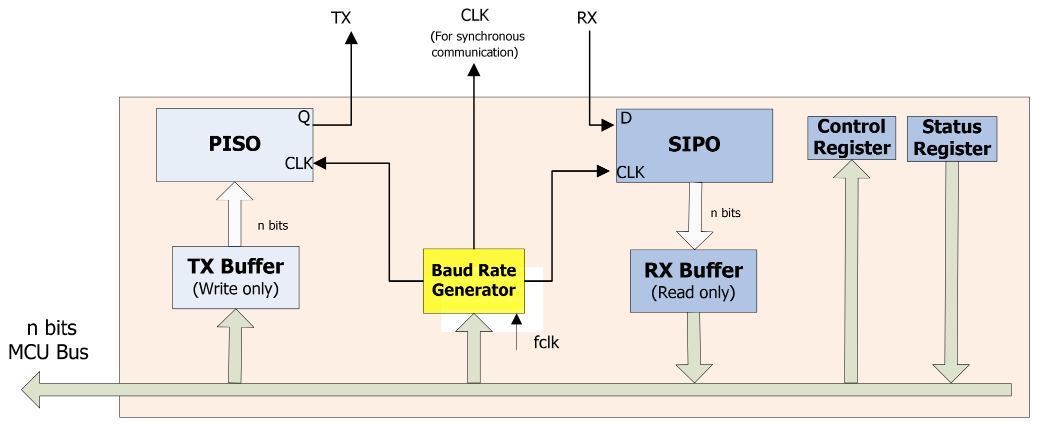


Figure 3: Minimum USART components.

– **Parallel Input Serial Output Register (PISO):** This unit is a shift register that converts n-bit parallel data from the CPU into serial data. The data is converted to serial at a clock rate given by the baud rate generator.

– **Transmit Buffer (TX Buffer):** This register holds the data to be transmitted. Also called “Data-out register”, the TX buffer is written by the CPU to begin a serial transmission.

– **Serial Input Parallel Output Register (SIPO):** This shift register converts the serial input stream into parallel data. The baud rate generator provides the clock signal to this shift register for establishing the reception rate.

– **Receive Buffer (RX Buffer):** When a new character received through the SIPO is completed, it is transferred to the RX buffer (also called data-in register) and made available for the CPU to be read.

Two additional registers necessary to operate an USART include a **control register** and a **status register.** The control register allows configuring the USART in the desired operating mode. For example, it allows choosing either synchronous or asynchronous mode, enable the transmitter or receiver; enable USART interrupts; number of bits to be transmitted; select error check, etc.

The **status registers** contains information to indicate the current USART status. Indicators such as when the TX Buffer can be written, when the RX Buffer can be read, or when an error has been are located in this register.

The baud rate is configured using the system clock frequency and a prescaler value. The prescaler value depends on the MCU.

$$BaudRate = \frac{fclk}{PrescalerValue * (N+1)}$$

Sequence of stepts to configure an USART:

1. Configure the baud rate. This is, specify the clock source and prescaler value. Depending on the particular MCU used, the clock source can come from timers, from the system clock, or from an external source.

2. Configure the baud rate value register to specify the actual rate.

3. In the control register(s) specify the USART mode: synchronous or asynchronous. Also indicate whether error checking will be enabled or not. In some MCU it is possible to even specify a protocol to be used.

4. In the interrupt enable registers, enable the transmission and/or reception interrupt depending on the type of communication (Tx or Rx or both for duplex operation).

5. In the control register enable the transmitter and/or receiver and global USART enable. The receiver-ready or transmitter-ready flags will signal when new data has arrived or when a new character can be sent. If interrupts were enabled the corresponding ISRs will be activated.

To use the USART is necessary to take into account the following considerations:

1. If the transmission is by polling, before writing any value to the TX-Buffer, verify the status register determine when the Tx-ready s asserted. Writing an non ready Tx-buffer will cause data loss.

2. If transmitting via interrupts, write the TX Buffer Register with the Tx-ready interrupt, one character at a time per ISR activation.

3. To receive data by polling, prior reading the Rx-buffer, check the status register to verify that new data has arrived (Rx-ready flag). If a received data is not read from the RX Buffer Register and a new one is received in SIPO the old will be overwritten (overrun error).

4. To receive data using interrupts, read the new data when a Rx-ready interrupt is generated. The interrupt will activate once per each new character received.

Procedure

Part I: Asynchronous serial communication (UART)

1. Locate the TX and RX signal pins in the microcontroller and connect the MCU with the computer using a USB to UART cable. Connect the VCC and GND to the USB to UART cable.
2. Open a HyperTerminal on your computer or any RS-232 communication program and configure it for a baud rate of 9600 bauds, no parity bit, and no flow control.
3. Configure the baud rate on your MCU. The baud rates and configuration in the MCU must match those of the other device we wish to communicate to.
   * *Configure the baud register configuration registers; this is, specify the prescaler value and specify the clock source to the baud rate.*
   * *Configure the baud rate generator number necessary for the desired baud rate. If your MCU uses a timer for generating the baud rate, program that timer.*
4. Configure the USART for asynchronous transmission:
   * *In the control registers specify the mode to be asynchronous.*
   * *Also enable the transmitter and global USART enable.*
5. Write a short program to write a character over the UART. Figure 4 illustrates a flowchart with the suggested sequence of steps.

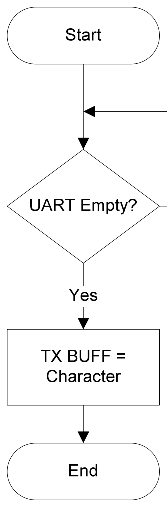


Figure 4: Flowchart to write a character over the UART.

6. Using "character write" function created in step 5, write a program that sends over the UART the string: **"Hello World!"**

Part II: Receiving and sending characters via UART

1. Use connection and code from part I and connect an LCD to your MCU as you did in previous experiments.
2. Modify the configuration procedure outlined in Part I to also activate, in the control register, the receiving end of the UART via interrupts.
3. Write a program that receives a character using interrupts and display it on the LCD. For this task it only takes to write the received character directly to the LCD upon reception. Another way would be via a memory buffer where received charactes are stored and having a function to dump the buffer contents into the LCD.

Exercise 1:

Modify your program to receive 16-character lower case messages from the PC and display them on the LCD. The message must be returned to the PC via de UART in upper case.

Turn-in:

- Email the TA the program source code and give a demonstration its operation in the lab.

Part III: Synchronous Serial Communication (USART)

1. Connect an RTC DS1305 to the SPI pin of your MCU, as illustrated in Figure 5. The DS1305 is a real-time clock-calendar chip that communicates with the MCU via SPI (Serial Peripheral Interface), a form of synchronous serial protocol. Data sheets for the DS1305 are available at http://datasheets.maxim-ic.com/en/ds/DS1305.pdf

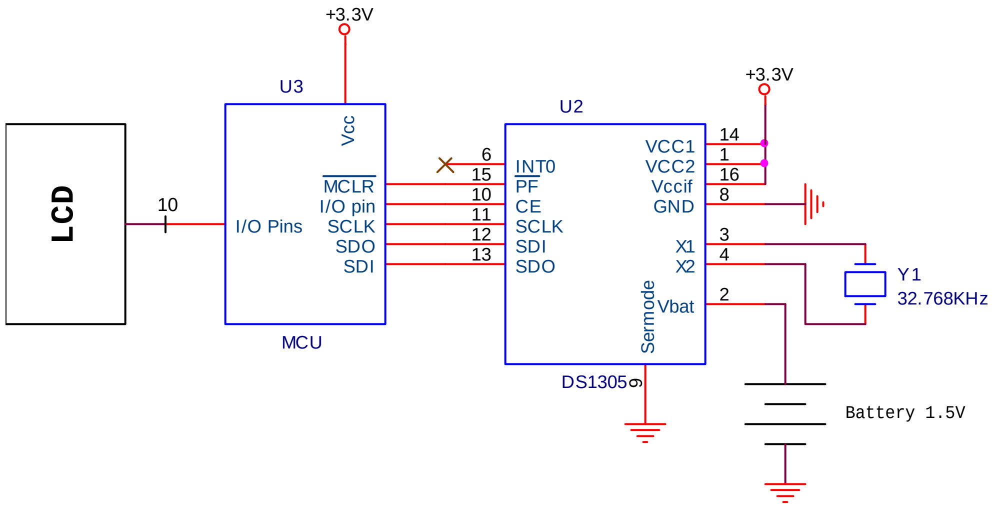


Figure 5: DS1305 Connection.

2. Configure the baud rate for 16kbps :

   * *Configure the baud control registers with the prescaler value and baud rate clock source.*

   * *Configure the baud rate generator number for the desired baud rate. If your MCU uses a timer for setting the baud rate generator, program that timer.*

3. Configure the USART for synchronous SPI communication:

   * *In the USART control registers choose synchronous mode.*

   * *Configure the USRT to write data with high edges and read with low edges.*

   * *In the control register, enable the transmitter, the receiver, and the global USART enable.*

   * *Note that to read data from the DS1305 it is necessary to send an address byte with bit 7 in 0. In response, the RTC will send the time/date data. Figure 6 shows a timing diagram with the required sequence of signals.*

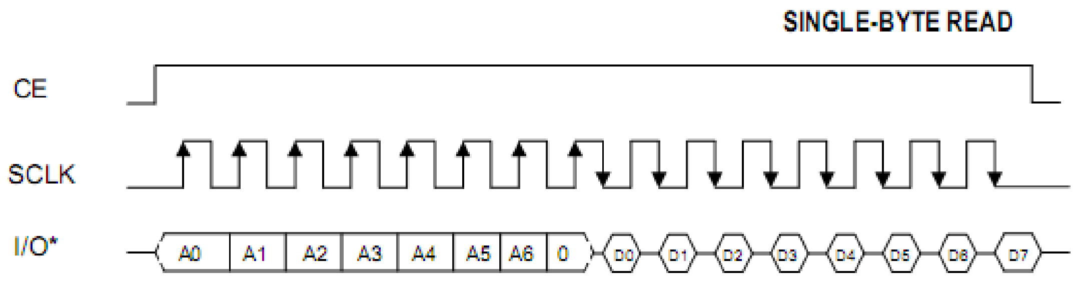


Figure 6: Timing diagram to read a data from DS1305.

* *Note that to write data to the DS130 5it is necessary send a byte with the address and bit 7 in 1.After that, send the data you want to write. See figure 7.*

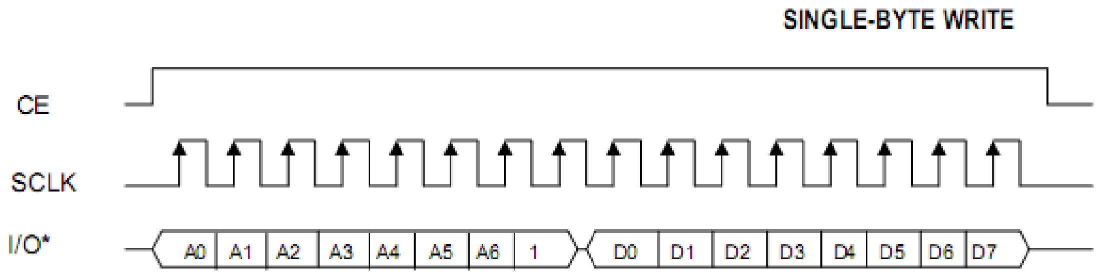


Figure 7: Timing diagram to write a data from DS1305.

4. Write a program to ask the DS1305 for its current time and display it on the LCD in the format HH:MM:SS. See Figure 8 suggest a flowchart for completing this task. Note that this exercise did not set the time (and date) in the chip and therefore the read time will correspond to the time elapsed after the last power-up.

Exercise:

Modify your program to read the current time and date from the DS1305 and display them on the LCD in the form a clock calendar. Refer to Figure 9 for the address of date registers.

Turn-in:

- Code for this exercise and produce a demo for the TA.

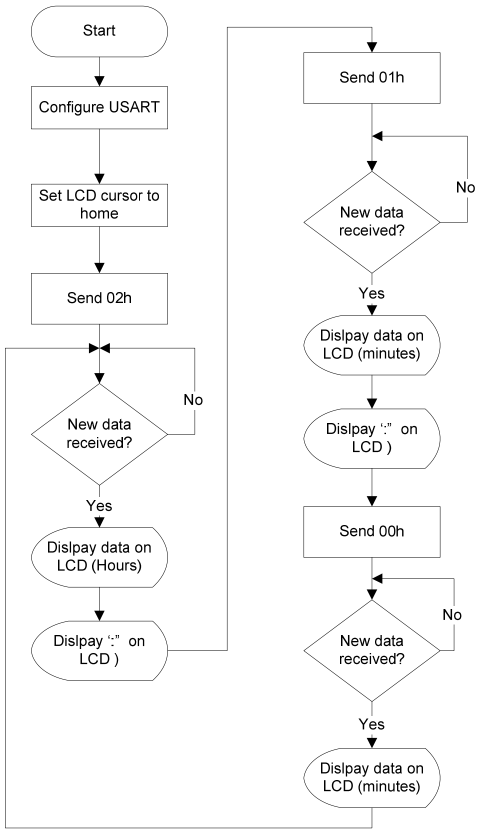
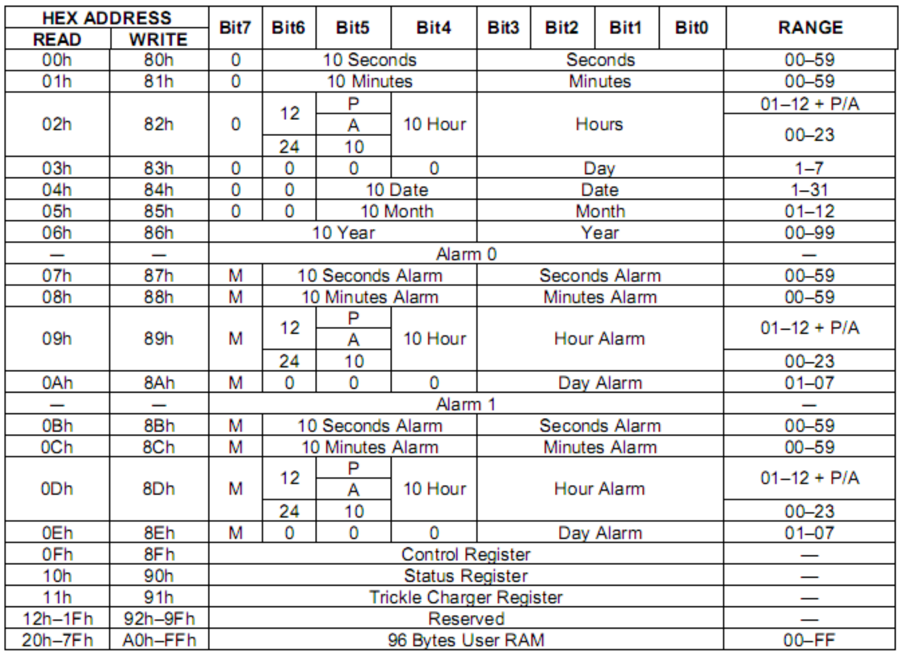


Figure 8: Flowchart to read the time from the DS1305.



HEX ADDRESS										
READ	WRITE	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	RANGE
00h	80h	0	10 Seconds			Seconds				00–59
01h	81h	0	10 Minutes			Minutes				00–59
02h	82h	0	12	P	10 Hour	Hours				01–12 + P/A
				A						
			24	10						00–23
03h	83h	0	0	0	0	Day				1–7
04h	84h	0	0	10 Date		Date				1–31
05h	85h	0	0	10 Month		Month				01–12
06h	86h	10 Year				Year				00–99
—	—	Alarm 0								—
07h	87h	M	10 Seconds Alarm			Seconds Alarm				00–59
08h	88h	M	10 Minutes Alarm			Minutes Alarm				00–59
09h	89h	M	12	P	10 Hour	Hour Alarm				01–12 + P/A
				A						
			24	10						00–23
0Ah	8Ah	M	0	0	0	Day Alarm				01–07
—	—	Alarm 1								—
0Bh	8Bh	M	10 Seconds Alarm			Seconds Alarm				00–59
0Ch	8Ch	M	10 Minutes Alarm			Minutes Alarm				00–59
0Dh	8Dh	M	12	P	10 Hour	Hour Alarm				01–12 + P/A
				A						
			24	10						00–23
0Eh	8Eh	M	0	0	0	Day Alarm				01–07
0Fh	8Fh	Control Register								—
10h	90h	Status Register								—
11h	91h	Trickle Charger Register								—
12h–1Fh	92h–9Fh	Reserved								—
20h–7Fh	A0h–FFh	96 Bytes User RAM								00–FF

Figure 9: Addresses for the DS1305 read and write registers.

Homework

Using a DS1305 make a programmable alarm clock. The clock shall use the keys UP, DOWN and ENTER to configure the current time, date, and desired alarm upon reset. Upon setup, the current date shall be displayed in the top LCD line and the time on the bottom line. Use the ENTER key to toggle between alarm time and current time and date.

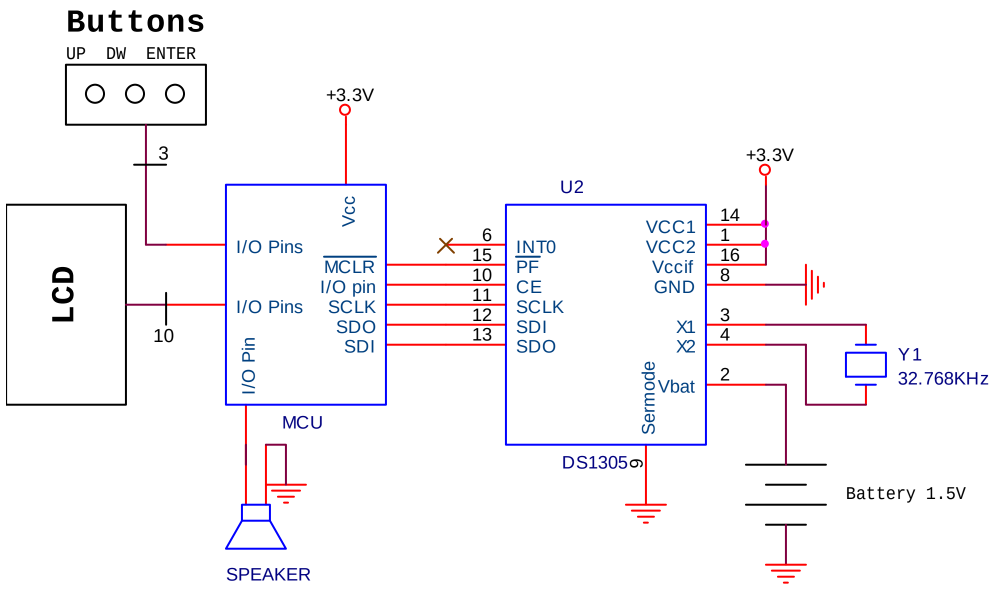


Figure 10: Schematic for homework.

Turn-in:

- Software plan and explanation (pseudo code or flowchart)
- Listing of code via e-mail
- Have your hardware assembled and produce a demo for the TA or professor