

Lecture 6

Embedded system design

CS476 - ESD
April 16, 2024

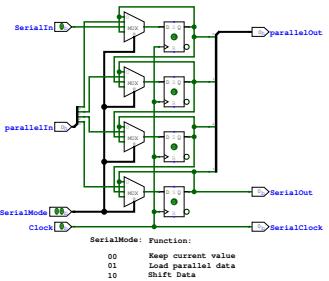
Dr. Theo Kluter
EPFL

Rev. 1.0 - 6.

Notes

Introduction

- ▶ Now that we have seen most of the interior of an embedded system we are going to look into the peripherals.
 - ▶ Many peripherals are based on serial protocols, like I²C, I2S, RS232, CAN,
 - ▶ To be able to convert parallel data to serial and vice versa, a shift register is used.
 - ▶ Note that shifting to the left or shifting to the right is basically the same circuit!
 - ▶ In this lecture we are going to visit some of these protocols.



Rev. 1.0 - 63

Notes

History: RS-232 still a protocol to be found everywhere

- Arguably the “oldest” serial protocol is the RS-232 introduced by the *Electronic Industries Association (EIA)* in 1960.
 - It is an asynchronous point-to-point protocol that still is very “active” today. Note that it was the “enabler” for the internet as we know it today.
 - RS-232 is also known as *Universal Asynchronous Receiver/Transmitter (UART)*.
 - Although the protocol dates from 1960, the latest “update” dates from 2012.
 - It is a “proven protocol” that can be even found in current server systems as “backup” interface in case something went wrong.
 - And you are using it each time for uploading your program to the VP. So how does it work....
 - The original interface was:

Signal:	Function:
TXD	Transmit Data
RXD	Receive Data
DTR	The slave is ready to receive, initiate, or continue a call
DCD	The slave is detecting a carrier from the remote device
DSR	The slave is ready to receive and send data
RI	The slave detected an incoming call
RTS	The slave is ready to receive data
CTS	The master is ready to send data

5851

5/10

design

Introduction

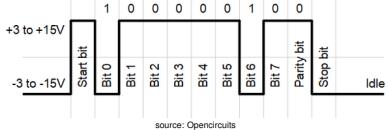
RS-232

SPI

Notes

Today: RS-232 still a protocol to be found everywhere

- ▶ Although most of the signals are "obsolete", even most interfaces found nowadays provide them for back-ward compatibility.
 - ▶ However, most of the time we only use the TxD and RxD signals. These are cross-connected between the two devices allowing for full-duplex communication.
 - ▶ The RS-232 sends the information by frames:



- ▶ There are 8 data-bits shown above, however, the protocol allows for 5 to 9 data bits.
 - ▶ The parity-bit is optional and can be odd, even, mark(1), or space(0).
 - ▶ The frame is ended by one or two stop bit(s).
 - ▶ The speed of the communication is defined by the *baudrate*. Basically this measures the bits transferred per second.
 - ▶ Note that the voltage levels nowadays also be allowed to be GND and VCC.

PRL

Embedded system design

Theo Kluter

Introduction

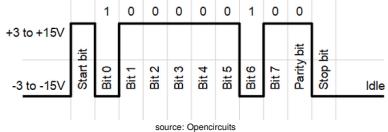
1020

1

Rev. 1.0 - 6.4

RS-232 is robust but slow

- ▶ Although the RS-232 is so wide spread, it is relatively slow.
 - ▶ Let's take a baudrate of 115200 baud and a frame as depicted below



As the frame consists of 11 bits (containing one data byte), we can transfer a maximum of:

$$\frac{115200}{11} \approx 10472.7 \frac{\text{bytes}}{\text{s}} \approx 10.5 \frac{\text{kBytes}}{\text{s}}$$

- Hence we require more performing protocols.

Notes

Embedded system design

design
Theo Kluter

Introduction

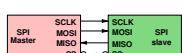
RS-232

SPI

Box 1.0 = 6.5

Serial Peripheral Interface (SPI)

- The arguably “simplest” serial interface to implement is the *Serial Peripheral Interface (SPI)*.
 - The SPI is a synchronous serial communication interface specification.
 - Developed by Motorola in the mid 1980s and nowadays a de facto standard.
 - The SPI has following signals:
 - SCLK: Serial CLoCK (output from the master).
 - MOSI: Master Out Slave In (data output from the master).
 - MISO: Master In Slave Out (data input to the master).
 - SS: Slave Select (often active-low). Output(s) from the master to select the slave to communicate with.
 - Signal/pin names as well as their timing constraints vary among manufacturers. Always check the device data sheet!
 - Most slave devices have tri-state outputs, i.e., their MISO output becomes high impedance if their SS input is not active. This allows that in a multiple-slave system all MISO signals can be connected together.



Notes

Embedded system

design

Introduction

RS-232

SPI

Box 10 = 66

SPI: Extensions

- Although the SPI-interface is relatively "fast", it only transports 1-bit each clock cycle. Especially for devices as Flash and SD-cards, this might be limiting as most of the time we only "read" their contents. This has lead to some extension to the SPI-protocol (note: not all slaves support these modes, you always have to consult the datasheets!).
 - Note that these extensions need to be activated in the slave device; all slave devices start out with the "standard" SPI-configuration!
 - These extensions are:
 - Dual SPI*: In this case the MOSI and MISO wires are transformed to a bi-directional communication channel. This allows to read or write 2 bits each clock cycle, doubling the data-throughput.
 - Quad SPI*: In this case we require two more connection (often the reset and write-protect signal). Here the MOSI, MISO, and the two extra signals are used as bi-directional communication channel. This allows to read or write 4 bits each clock cycle. This is used for example with SD-cards and the SPI-Flash that is on your GECKO4.
 - Octal SPI*: I think that you get the idea. It requires six more connections that are not used in "normal" SPI-mode.



Notes

Notes

Notes