Serial I/O
Synchronous vs. Asynchronous

- There are two main types of serial systems, synchronous and asynchronous
- Synchronous serial uses a single clock to synchronize all devices on the serial bus
- Asynchronous serial devices each have an independent clock source
- The 68HC11 has both types. The Serial Peripheral Interface (SPI) is synchronous, the Serial Communications Interface (SCI) is asynchronous
SCC

• The SCI interface is commonly used for communicating (with relatively low bandwidth) between computers or as part of a large communications network

• The SCI system is full-duplex (it can transmit and receive at the same time)

• The transmitter and receiver clocks must run at the same rate, but are not synchronised
Asynchronous Communications

TRANSMITTER and RECEIVER typically are remote

TRANSMITTER and RECEIVER use separate clocks

RECEIVER samples bits at same rate that TRANSMITTER sends bits

START and STOP bits used to synchronize data transfer
Framing

- Normally one character (byte) transmitted at a time, followed by a small idle time
- Usually, each character is ‘framed’ by start and stop bits
- Start bit is always 0, stop bit is always 1
- Framing is usually handled in hardware
Framing 2

(a) 8 Bits + 1 Stop Bit

(b) 7 Bits + Parity + 1 Stop Bit

(c) 7 Bits + Parity + 2 Stop Bits

(d) 8 Bits + Parity + 1 Stop Bit
Parity

- Parity is used to detect single-bit transmission errors

- If there are an even number of 1’s in a byte, it is said to have even parity

- If there are an odd number of 1’s in a byte, it is said to have odd parity

- If parity is used, the hardware will force the transmitted data to have a particular parity (usually by changing the last bit in an ascii character)
RS-232

• RS-232 is an industry standard serial communications interface. The serial ports in your PC conform to this standard

• This standard defines signal levels, connectors, and pin assignments among other things

• It does not specify a software protocol. It is mainly an electrical specification

• A logic 1 is represented by a -12 Volt signal. Logic zero is +12V

• The high voltages allow signals to be sent further
RS-232

• The RS-232 standard defines 25 different signals, however in practice only a few are ever used.

• In the EV board, only 3 wires are used. One for Transmit, one for Receive, and Ground (used as a reference).

• Signal levels are usually shifted via a driver chip. See if you can find the RS-232 driver chip on the EVBPlus board schematic.

• If you are interested, standard RS-232 pin configurations, and cable connections are listed on a board at the back of the lab.
Software Protocols

• A protocol is simply a set of rules for making connections and transferring data

• ACK/NAK – acknowledge/not acknowledge. In some protocols, the transmitter waits for a receiver to acknowledge successful transmission before a new block of data is sent

• If the receiver examines the data and determines that it is valid, an ACK is sent, if the data is invalid, a NAK is sent.

• If a NAK is received by the sender, it will retransmit the previous data
CRC’s

• CRC – Cyclic Redundancy Check

• Data integrity can be verified by calculating a code based on some combination of the data bytes. It can be made to be significantly more reliable than a parity check

• The sender and receiver both calculate the CRC from the data. If the sender and receiver both calculate the same value, the data is valid

• CRC-16 is a common CRC. It has a better than 99% chance of detecting bursts of errors up to 16 bits in length
Figure 10.3  Modes of Channel Operation
XON/XOFF

• XON/XOFF is a reasonably common protocol

• If the receiver wants to tell the transmitter to stop sending data, it sends the XOFF character. To restart transmission, it sends XON. These are both part of the ASCII character set
SCI registers

• First, select the baud rate by writing to the BAUD register

• Next, select the word length and wake up. Bits M and WAKE in the SCCR1 register (Serial Communication Control Register)

• Finally, enable the SCI and associated interrupts (if required) by writing to SCCR2

• Monitor the state of the SCI Module by reading SCSR (Serial Communication Status Register)
## SCCR1

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>R8</td>
<td>T8</td>
<td>0</td>
<td>M</td>
<td>WAKE</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

### R8 — Receive Data Bit 8
If the M bit is set, this bit provides a storage location for the ninth bit in the receive data character.

### T8 — Transmit Data Bit 8
If the M bit is set, this bit provides a storage location for the ninth bit in the transmit data character. It is not necessary to write to this bit for every character transmitted, only when the sense is to be different than that for the previous character.

### Bit 5 — Not Implemented
This bit always reads zero.

### M — SCI Character Length
- 0 = 1 start bit, 8 data bits, 1 stop bit
- 1 = 1 start bit, 9 data bits, 1 stop bit

### WAKE — Wake Up Method Select
- 0 = Idle Line
- 1 = Address Mark

### Bits 2-0 — Not Implemented
These bits always read zero.
# Baud Rate Selection

<table>
<thead>
<tr>
<th>$102B$</th>
<th>TCLR</th>
<th>0</th>
<th>SCP1</th>
<th>SCP0</th>
<th>RCKB</th>
<th>SCR2</th>
<th>SCR1</th>
<th>SCR0</th>
<th>BAUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESET</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>U</td>
<td>U</td>
<td>U</td>
<td></td>
</tr>
</tbody>
</table>

## Table 9-3. Baud Rates by Crystal Frequency, SCP1, SCP0, and SCR2–SCR0

<table>
<thead>
<tr>
<th>SCP1</th>
<th>SCP0</th>
<th>SCR2</th>
<th>SCR1</th>
<th>SCR0</th>
<th>Crystal Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2^29 Hz</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

## Bus frequency (F clock)

- 2 MHz
- 1.2288 MHz
- 1 MHz
- 921.6 kHz

---

**Monday, 25 March 13**  
**Serial I/O**
SCSR:

- **TDRE**  —  Transmit Data Register Empty
- **TC**  —  Transmit Complete
- **RDRF**  —  Receive Data Register Full
- **IDLE**  —  Idle Line Detect
- **OR**  —  Overrun Error
- **NF**  —  Noise Flag
- **FE**  —  Framing Error
Double Buffering

Transmitter

Data from MCU

T8

TDR Buffer (SCDR during write)

Shift Register

TDRE

1

Data Out

Tx/Do PD1

Receiver

Data to MCU

Rx/Do PD0

Shift Register

R8

RDR Buffer (SCDR during read)

RDRF

1
Transmitting Data

- Poll the SCSR register (or respond to an interrupt)
- If nine bits are being sent (M=1), write to bit T8 in SCCR1
- If bit TDRE = 1, write data to be sent to register SCDR
Receiving Data

• Poll the SCSR register (or respond to an interrupt)

• If bit RDRF = 1, read sent data from register SCDR

• If there has been a transmission error, handle it.
Synchronous Serial

- The SPI system is a synchronous serial interface

- On the EVBPlus2 board, the SPI is used to interface to an SPI-RS232 chip, and is used by the monitor program

- Synchronous serial is faster than asynchronous – there is no need for framing bits (start and stop bits)

- SPI is typically used for fast communication between chips on a circuit board

- There are a number of standard chips that use this interface
Synchronous Operation

[Diagram showing synchronous operation with labels for MOSI, MISO, CLK, and registers.]