SCI
Serial Communication Interface

Gerrit Becker
James McClearen
Charlie Hagadorn

October 21, 2004
Learning Objectives of the Overview

- Knowledge of the general differences between serial and parallel communication
- Knowledge of the differences between synchronous and asynchronous serial communication
- More detailed knowledge of the asynchronous serial communication (Examples)
- Knowledge of the difference between baud rate and bit rate (Example)
Types of Data Communication

- 2 general types of data transmission
  - Parallel Data Communication
  - Serial Data Communication
Parallel Data Communication

- Simultaneous 8-bit transmission
- Requires 8 separate data lines
- Bits must stay synchronized
- Restricted distance to avoid synchronization problems
- Faster than Serial transmission
- Expensive
- Example: Printer connections

Diagram:
- Transmitter
- Receiver
- First word
- Second word
Serial Data Communication

- Transfers one bit at a time
- Requires only one data line
- Slow compared to parallel transmission
- Less expensive
- Example: modem
Serial Data Communication

- **Full duplex**
  - If we want to send and receive at the same time
  - Therefore you need 2 wires, one to send, one to receive (and one extra as common ground)

- **Number of Data bits**
  - Both, transmitter and receiver must agree on the number of send data bits
  - Usually you use 7 or 8 bit
  - Remark: If you use only 7 bit you send only ASCII values not greater that 127
Serial Data Communication

- Types of Serial Data Communication
  - Synchronous Communication
  - Asynchronous Communication
Synchronous Communication

- Transmitter and receiver have their clocks synchronized
- Data rates are dependent on clock rates
- Continuously transmitting characters to remain in sync.
Asynchronous Communication

- NO synchronization
  - No need to send idle characters
- Transmitter and receiver operate independently
  - Transmitter can send data at any time
  - Receiver is always ready to accept data
- Requires a start and stop bit to identify each byte of data
- How does receiver know that data is arriving?
  - If the line is idle, it is sending a constant '1' (mark state)
  - The receiver is able to recognize a jump from '1' to '0' with the start bit and is alerted that data is about to be sent.
Comparison of synchronous and asynchronous communication

- Synchronous communication is faster but more complicated due to the clock synchronization.
- Asynchronous communication is slower due to the additional bits but easier to accomplish.
Asynchronous Transmission Format

- Bit Types
  - Start bit
  - Data bits
  - Parity bit
  - Stop bits
Definitions

- **Start Bit**
  - Signals the beginning of a word
  - Is normally a '0' and is detected as a transition from high to low

- **Data Bits**
  - The actual data, which should be transmitted
  - Sender and receiver have to agree on the number of data bits (usually 8)
  - Always the least significant bit will be sent first
Parity Bit

- An error check
- Odd or even parity
  - Odd parity means the sum of the 1's will be odd
  - Even parity means the sum of the 1's will be even
  - You count all bits including the parity bit
- Disadvantage: If two bytes altered by noise, an error will not be detected by the parity check
Definitions cont

- **Stop Bits**
  - These bits mark the end of a data word
  - Is usually high (1)
Asynchronous Data Transmission

- Example 1:
  - Hex# $4A_{16}$ is to be sent with one start bit, even parity, 8-bit data length and two stop bits
  - $4A_{16} = 0100\ 1010_2$
Example 2:

- Hex# $B4_{16}$ is to be sent with one start bit, even parity, 8-bit data length and two stop bits
- $B4_{16} = 1011 \ 0100_2$
Asynchronous Data Transmission

**Example 3:**
- Hex# B4₁₆ is to be sent with one start bit, **odd** parity, 8-bit data length and two stop bits
- B₄₁₆ = 1011 0100₂
Baud Rate vs. Bit Rate

- **Definition Baud Rate:**
  - Number of changing states per second
  - Includes start, data, parity and stop bits

- **Definition Bit Rate:**
  - Number of data bits transmitted per second

Baud Rate > Bit Rate
Example:

Consider baud rate: 4800 baud

12 bits/word = 1 start bit + 8 data bits + 1 parity bit + 2 stop bits

Bit time = 1/(baud rate) = 1/4800 baud = 0.208 ms/bit
Word time = (12 bits)*(bit time) = 2.5 ms
Word rate = 1/(word time) = 400 words/s
Bit rate = (word rate)*(8 data bits) = 3200 bits/s
SCI Registers

- 5 Main Registers
  - BAUD – Sets the bit rate for the SCI system
  - SCCR1 – Sets control bits for the 9-bit character format and the receiver wake up feature
  - SCCR2 – Main control register for the SCI sub-system
  - SCSR – Status register for the SCI system
  - SCDR – Main data register for the SCI system

- 3 Ancillary Registers
  - PORTD – Input/Output Port D
  - DDRD – Data direction register for Port D
  - SPCR – SPI control register
BAUD Register

- Used to set the bit rate of the SCI system
  - TCLR – Clear baud rate timing chain bit
  - SCP1 - SCP0 – Baud rate pre-scale select bits
  - RCKB – SCI baud rate clock test bit
  - SCR2 - SCR0 – SCI baud rate select bits
SCCR1 Register

- Contains control bits related to the 9-bit data character format and the receiver wake up feature
  - R8 – Receive data bit 8
  - T8 – Transmit data bit 8
  - M – SCI character length bit
  - WAKE – Wakeup method select bit
  - Bits 0 - 2 & 5 are not used (always 0)
SCCR2 Register

Address: $102D

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Read:  
Write:  
Reset:  

U = Unaffected

- **Main control register for SCI sub-system**
  - TIE – Transmit interrupt enable bit
  - TCIE – Transmit complete interrupt enable bit
  - RIE – Receive interrupt enable bit
  - ILIE – Idle-line interrupt enable bit
  - TE – Transmit enable bit
  - RE – Receive enable bit
  - RWU – Receiver wakeup bit
  - SBK – Send break bit
**SCSR Register**

- **Address:** $102E

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read</td>
<td>TDRE</td>
<td>TC</td>
<td>RDRF</td>
<td>IDLE</td>
<td>OR</td>
<td>NF</td>
<td>FE</td>
</tr>
<tr>
<td>Write</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reset</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- **U = Unaffected**

- **SCI status register**
  - TDRE – Transmit data register empty bit
  - TC – Transmit complete bit
  - RDRF – Receive data register full bit
  - IDLE – Idle-line detect bit
  - OR – Overrun error bit
  - NF – Noise flag
  - FE – Framing Error bit
  - Bit 0 is not used (always 0)
## SCDR Register

<table>
<thead>
<tr>
<th>Address: $102F</th>
<th>Bit 7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read:</td>
<td>R7</td>
<td>R6</td>
<td>R5</td>
<td>R4</td>
<td>R3</td>
<td>R2</td>
<td>R1</td>
<td>R0</td>
</tr>
<tr>
<td>Write:</td>
<td>T7</td>
<td>T6</td>
<td>T5</td>
<td>T4</td>
<td>T3</td>
<td>T2</td>
<td>T1</td>
<td>T0</td>
</tr>
<tr>
<td>Reset:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>U = Unaffected</td>
<td>Unaffected by rest</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**SCI data register**
- Two separate registers
- When SCDR is read, the read-only RDR is accessed
- When SCDR is written, the write-only TDR is accessed
- R7 - R0 – Read bits
- T7 - T0 – Write bits
Ancillary Registers – PORT D

- SCI uses the two least significant bits of Port D.
- These bits are used for receiving and transmitting data.
- Data direction register does not control Port D while SCI is in use but it is important since it will have control when the SCI operation is aborted.
- SPCR register controls the Port D wire-OR mode bit, which controls the driver functions of the Port D pins, even if they are being used by the SCI.
Wake Up

- M68HC11 supports a receiver wake up function, which is intended for systems having more than one receiver.
- The transmitting device directs messages to an individual receiver or group of receivers by passing addressing information in the initial byte.
- Receivers not addressed activate the receiver wakeup function.
  - This makes these receivers dormant for the remainder of the unwanted message.
- Wake up mode is enabled by writing a 1 to the RWU bit in the SCCR2 register.
Wake Up

- Two methods of Wakeup
  - Idle-Line
    - Uninterested receivers are only sent the messaging frame
    - All receivers are awake (RWU = 0) when each message begins
    - When a receiver detects a non-interesting message the software sets RWU = 1
    - This inhibits further flag setting until the RxD line goes idle at then end of the message
    - When the idle line is detected, hardware clears the RWU bit so the first frame of the next message can be read
Wake Up and Send Breaks

- **Two methods of Wakeup**
  - **Address-Mark Wakeup**
    - Most significant bit is used to indicate if the message is data(0) or address(1)
    - All receivers wake up if the bit is 1 and check to see if the message is for them

- **Send Breaks**
  - Break characters are character-length periods where the TxD line goes to 0
  - Character length is influenced by the M bit in the SCCR1
    - M = 0 – All characters are 10 bit times long
    - M = 1 – all characters are 11 bit times long
  - Break characters have no start and stop bits
How to Send and Receive Data

Transmitter

- Set Baud rate of transmitter
  - Must match Receiver

- Set M bit of SCCR1 for 8 or 9 bit data
  - Must match Receiver

- Set TE bit of SCCR2 high to enable transmitter

Receiver

- Set Baud rate of receiver
  - Must match Transmitter

- Set M bit of SCCR1 for 8 or 9 bit data
  - Must match Transmitter

- Set RE bit of SCCR2 high to enable receiver
How to Send and Receive Data

**Transmitter**
- Activate WAKE condition
- Load data character into SCDR
- When TDRE bit of SCSR register goes high, the SCDR register is clear and another character can be loaded

**Receiver**
- Set WAKE bit on SCCR1
- RDRF bit of SCSR set when all data has entered RDR
- Read data from RDR and Store
- Check flags for possible error protocols
How to Send and Receive Data

Transmitter
- When TC bit of SCSR register goes high, transmit buffer clear
- Transmitter resumes Idle

Receiver
- Receiver returns to wake/sleep mode previously set
Noise Flag – HC11

- Noise is detected if three samples, taken near the middle, during the data and stop bit times do not agree.
- During the reception of the start bit, four additional samples are taken during the first half of the bit time:
  - Detects the leading edge of the bit and verification of a start bit.
- If any of these three samples are not zero, the noise flag is set.
- Noise Flag:
  - 0 – No noise detected during reception of the character in the SCDR.
  - 1 – Data recovery logic detected noise during reception of the character in the SCDR.
- Noise flag does not generate interrupt because it is associated with RDRF.

RDRF = Receive Data Register Full
SCDR = SCI Data Register
Noise

- Noise causes start bit to be detected too soon
- RT5 and RT7 are 0, so start will be accepted
- RT3 is 1, so noise flag will be set
Noise

- Start bit is found correctly
- Start accepted because RT3, RT5 and RT7 are 0
- RT8 and RT10 are 1, so noise flag will be set, but bit sense is still 0, because it is start bit
SCI Interrupts

- 2 Interrupts for Transmitter
  - TDRE enabled with TIE bit in SCCR2
  - TCIE enabled with TC bit in SCCR2
- 2 Receiver Interrupts
  - RDRF enabled with RIE bit in SCCR2
  - OR enabled with RIE bit in SCCR2
- One Interrupt Vector for SCI System