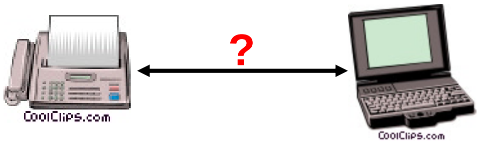
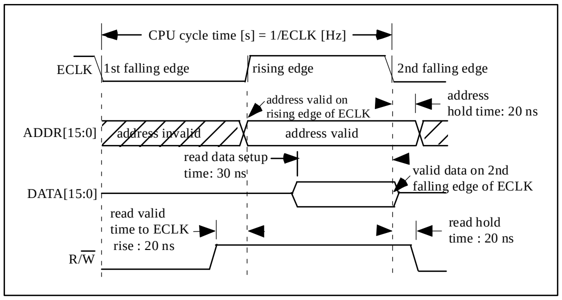
# Serial Communication (Ch. 10)

- How do digital components exchange information?



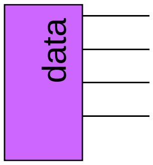
- Standard digital communication interfaces establish the *function* and *protocol* of signals used to exchange data between two (or more) digital components in a system
  - Function:
  - Protocol:

**timing protocol for memory expansion interface**

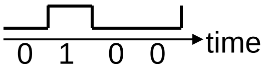

CPU cycle time [s] = 1/ECLK [Hz]
ECLK
1st falling edge
rising edge
2nd falling edge
address valid on rising edge of ECLK
address hold time: 20 ns
ADDR[15:0]
address invalid
address valid
read data setup time: 30 ns
valid data on 2nd falling edge of ECLK
DATA[15:0]
read valid time to ECLK rise : 20 ns
read hold time : 20 ns
R/W

# Digital Communication Interface Basics

- Two primary types of digital communication
  - Parallel
    - multiple data lines (generally one byte worth)
    - multiple data values transmitted simultaneously
  - Serial
    - single data line
    - bits sent in series, one bit at a time, sequentially



parallel or serial?

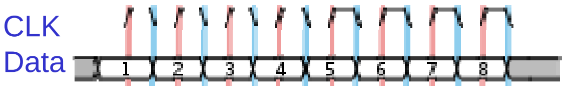


parallel or serial?

  - Terminology
    - bit rate - bits per second
    - bit cell - time to transmit a single bit
    - BAUD rate - bits per second
    - NRZ line code - transmit value for entire bit cell

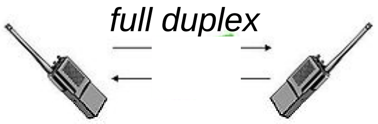
# Features of Serial Interfaces

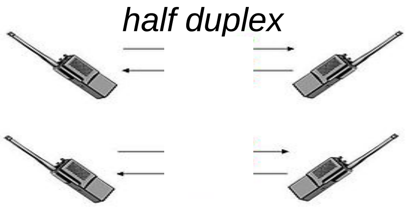
- Synchronous vs. Asynchronous
  - synchronous = data bits read at clock edge
  - asynchronous = no clock; data read at preset interval



Synchronous or asynchronous?

- To/from (out/in, transmit/receive) separate or shared
  - separate: data can be transmitted & received simultaneously
    - similar to "full duplex"
    - for hardwired communication, requires multiple data lines
  - shared: data either incoming or outgoing (not both) at any given time
    - similar to "half duplex"
    - for hardwired communication, requires only one data line



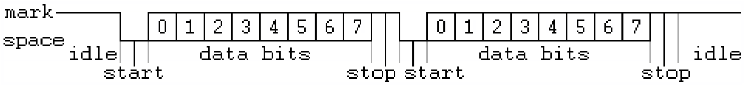


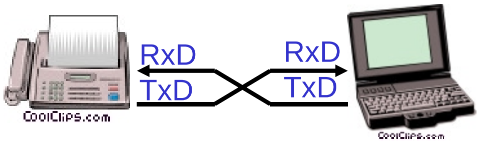
# Common Serial Interfaces

- Morse code telegraphy
- **RS-232** (low-speed, implemented by serial ports) = SCI = UART
- RS-422
- RS-423
- RS-485
- $I^2C$
- **SPI**
- ARINC 818 Avionics Digital Video Bus
- Universal Serial Bus (moderate-speed, for connecting peripherals to computers)
- FireWire
- **Ethernet**
- Fibre Channel (high-speed, for connecting computers to mass storage devices)
- InfiniBand (very high speed, broadly comparable in scope to PCI)
- MIDI control of electronic musical instruments
- DMX512 control of theatrical lighting
- SDI-12 industrial sensor protocol
- Serial Attached SCSI
- Serial ATA
- SpaceWire Spacecraft communication network
- HyperTransport
- **PCI Express** (high speed serial replacement for PCI/PCI-X (parallel))

# Common Microcontroller Serial Ports

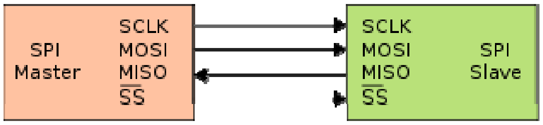
- Universal asynchronous receiver/transmitter (UART)
  - serially transmit/receive a byte of data written to a parallel port
  - also called Serial Communications Interface (SCI)
  - RS-232 (PC serial port) is a UART "standard"
    - standard = defined voltage and timing characteristics
  - asynchronous
    - no clock signal; data read at predetermined BAUD rate
      - baud rate = the number of distinct symbol changes (signaling events) made to the transmission medium per second
    - BAUD rate must match on both sides of the interface
    - typically use start/stop bits to mark bytes
  - full duplex = separate receive and transmit lines





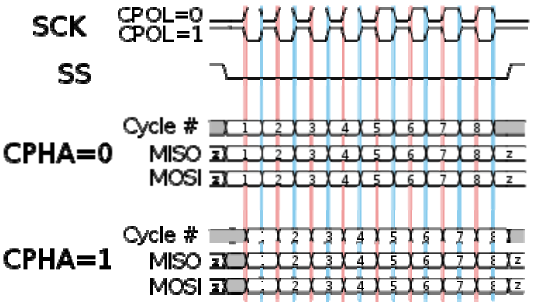
# Common Microcontroller Serial Ports

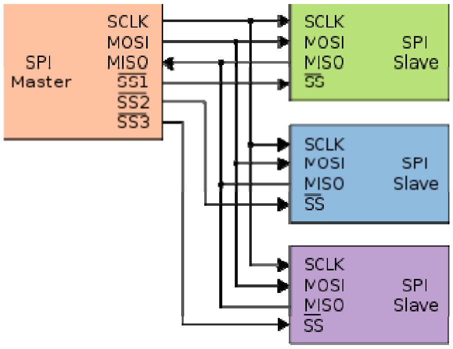
- Serial Peripheral Interface (SPI)
  - synchronous serial data link standard introduced by Motorola
  - operates in full duplex mode using separate in/out data signals
  - devices communicate in master/slave mode
  - multiple slave devices are allowed with individual slave select (chip select) lines
  - signals
    - SCLK —
    - MOSI/SIMO —
    - MISO/SOMI —
    - SS —
      - multiple select lines for multiple slave devices

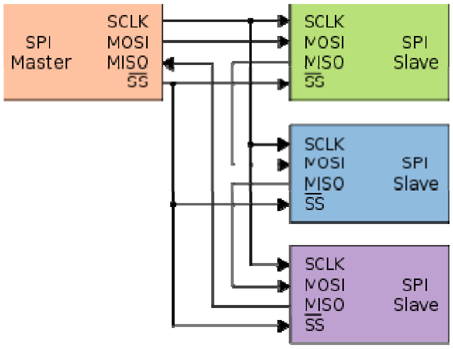


# SPI Variations and Controls

- Clock Polarity and Phase
  - assign rising/falling edge of clock to reading/changing of data
  - Freescale standard: CPOL = polarity, CPHA = phase
- Independent or Daisy Chain Slaves





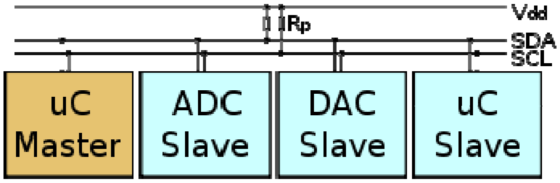


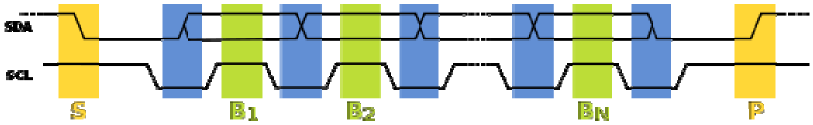
# SPI Pros and Cons

- Advantages
  - Full duplex communication
  - Higher throughput than $I^2C$ or SMBus
  - Complete protocol flexibility for bits transferred (no limit to word or message size)
  - Extremely simple hardware interfacing
    - Typically lower power requirements than $I^2C$ or SMBus due to less circuitry
    - No arbitration or associated failure modes
    - Slaves use the master's clock, and don't need precision oscillators
    - Slaves don't need a unique address -- unlike $I^2C$ or GPIB or SCSI
    - Transceivers are not needed
  - Uses only four pins; much less than parallel interfaces
  - At most one "unique" bus signal per device (chip select); all others are shared
  - Signals are unidirectional allowing for easy Galvanic isolation
- Disadvantages
  - No hardware slave acknowledgment (the master could be "talking" to nothing and not know it)
  - No hardware flow control (but master can delay clock edge to slow transfer rate)
  - Requires more pins on IC packages than $I^2C$
  - Supports only one master device
  - Generally prone to noise spikes causing faulty communication
  - Without a formal standard, validating conformance is not possible
  - Only handles short distances compared to RS-232, RS-485, or CAN-bus

# Inter-Integrated Circuit Bus

- I²C is a multi-master serial single-ended computer bus
  - invented by Philips to attach low-speed peripherals to an embedded system
  - uses only two bidirectional open-drain lines
    - Serial Data Line (SDA)
    - Serial Clock (SCL)
    - both pulled up with resistors, typical to +5 V or +3.3 V
  - 7-bit address (with 16 reserved addresses) →
  - timing



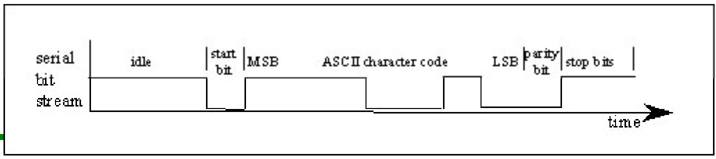


Data transfer is initiated with the START bit (**S**) when SDA is pulled low while SCL stays high. Then, SDA sets the transferred bit while SCL is low (blue) and the data is sampled (received) when SCL rises (green). When the transfer is complete, a STOP bit (**P**) is sent by releasing the data line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

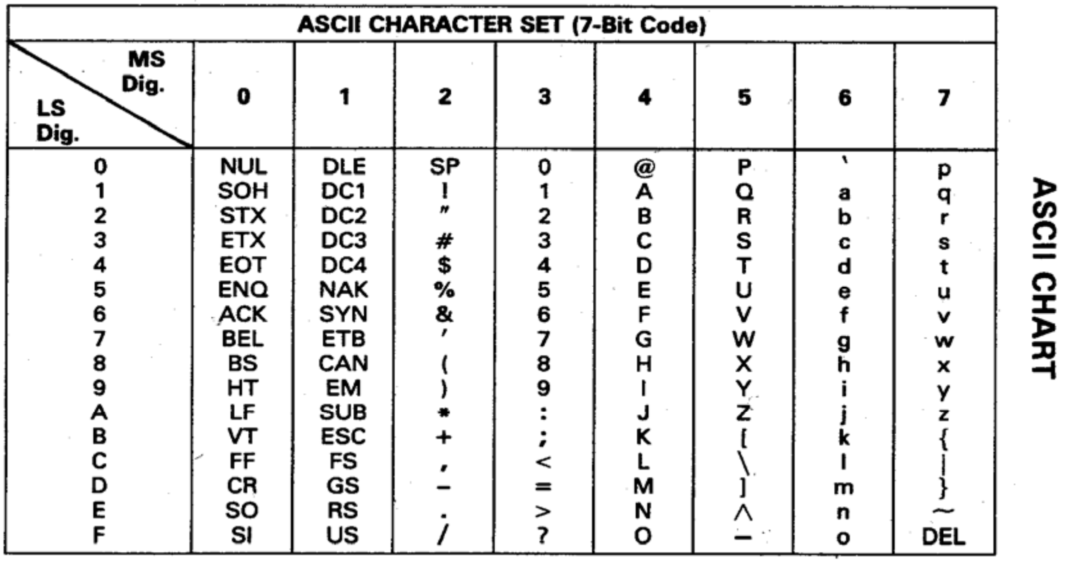
# Error Checking

- A noisy transmission medium can lead to errors in transmission
- Error detection
  - identifying if a transmission error has occurred
- Error correction
  - determine which particular bit is corrupted and fix it
- Parity Check
  - a parity bit is added to ensure that the number of bits with the value one in a set of bits is even (or odd)
  - the simplest form of error detecting code
  - common as 8th bit for 7-bit ASCII codes

| 7 bits of data (number of 1s) | 8 bits including parity | |
|---|---|---|
| | even | odd |
| 0000000 (0) | **0**0000000 | **1**0000000 |
| 1010001 (3) | **1**1010001 | **0**1010001 |
| 1101001 (4) | **0**1101001 | **1**1101001 |
| 1111111 (7) | **1**1111111 | **0**1111111 |



# ASCII - American Standard Code for Information Interchange



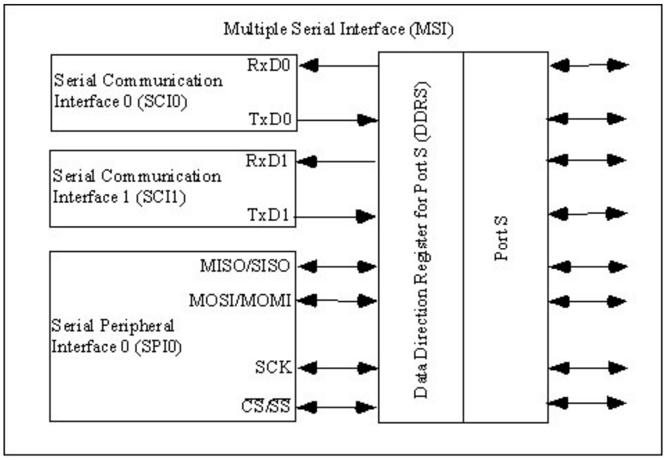
ASCII CHARACTER SET (7-Bit Code)

| LS Dig. \ MS Dig. | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 0 | NUL | DLE | SP | 0 | @ | P | ` | p |
| 1 | SOH | DC1 | ! | 1 | A | Q | a | q |
| 2 | STX | DC2 | " | 2 | B | R | b | r |
| 3 | ETX | DC3 | # | 3 | C | S | c | s |
| 4 | EOT | DC4 | $ | 4 | D | T | d | t |
| 5 | ENQ | NAK | % | 5 | E | U | e | u |
| 6 | ACK | SYN | & | 6 | F | V | f | v |
| 7 | BEL | ETB | ' | 7 | G | W | g | w |
| 8 | BS | CAN | ( | 8 | H | X | h | x |
| 9 | HT | EM | ) | 9 | I | Y | i | y |
| A | LF | SUB | * | : | J | Z | j | z |
| B | VT | ESC | + | ; | K | [ | k | { |
| C | FF | FS | , | < | L | \ | l | \| |
| D | CR | GS | – | = | M | ] | m | } |
| E | SO | RS | . | > | N | ^ | n | ~ |
| F | SI | US | / | ? | O | _ | o | DEL |

ASCII CHART

# HC12/S12 Serial Interfaces

- HC12: Multiple Serial Interface (MSI)
  - two SCI channels and one SPI channel

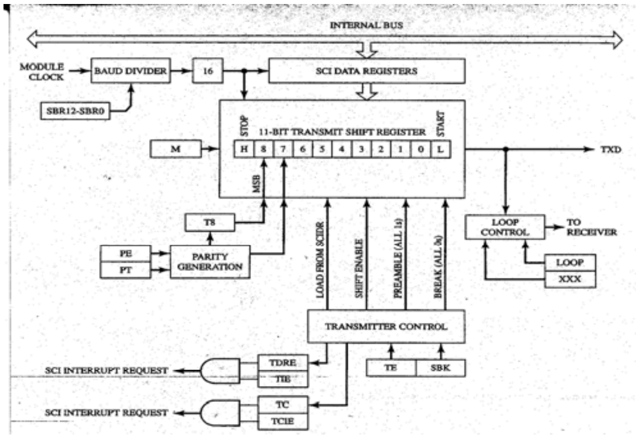

Multiple Serial Interface (MSI)
Serial Communication Interface 0 (SCI0)
RxD0
TxD0
Serial Communication Interface 1 (SCI1)
RxD1
TxD1
Serial Peripheral Interface 0 (SPI0)
MISO/SISO
MOSI/MOMI
SCK
CS/SS
Data Direction Register for Port S (DDRS)
Port S

- HCS12
  - two 2-pin SCI
  - two 4-pin SPI
  - one I$^2$C port
  - several CAN ports
    - standard common in automotive industry

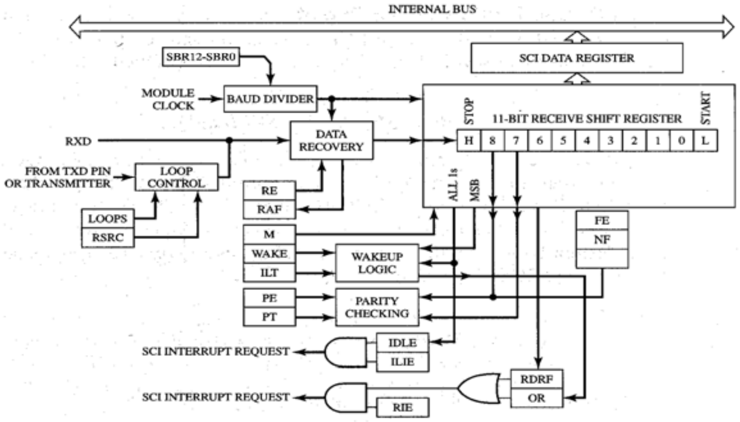
- Parallel ports can be used for additional SPI slave select signals

# HC12 SCI Hardware

- Transmitter

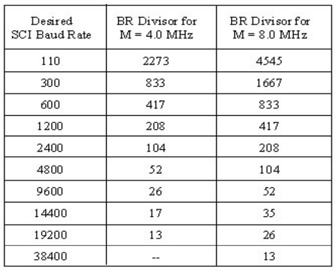


- Receiver

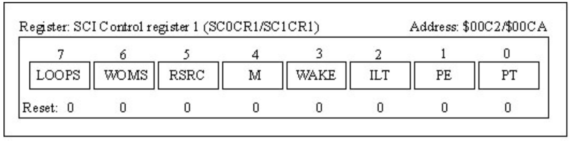


# HC12 SCI Registers

- SCI Baud Rate Control Registers
- SCI Control Registers 1, 2
- SCI Status Registers 1, 2
- SCI Data Registers High/Low

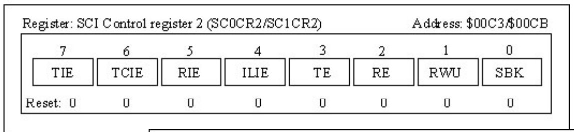


| Desired SCI Baud Rate | BR Divisor for M = 4.0 MHz | BR Divisor for M = 8.0 MHz |
|---|---|---|
| 110 | 2273 | 4545 |
| 300 | 833 | 1667 |
| 600 | 417 | 833 |
| 1200 | 208 | 417 |
| 2400 | 104 | 208 |
| 4800 | 52 | 104 |
| 9600 | 26 | 52 |
| 14400 | 17 | 35 |
| 19200 | 13 | 26 |
| 38400 | -- | 13 |



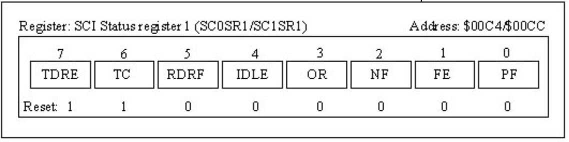
Register: SCI Control register 1 (SC0CR1/SC1CR1) Address: $00C2/$00CA

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | LOOPS | WOMS | RSRC | M | WAKE | ILT | PE | PT |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |



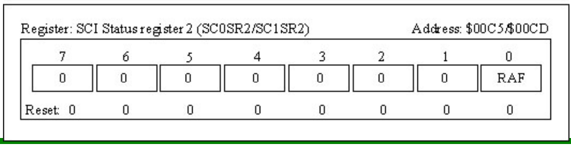
Register: SCI Control register 2 (SC0CR2/SC1CR2) Address: $00C3/$00CB

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TIE | TCIE | RIE | ILIE | TE | RE | RWU | SBK |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |



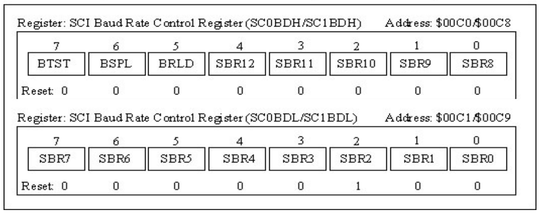
Register: SCI Status register 1 (SC0SR1/SC1SR1) Address: $00C4/$00CC

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TDRE | TC | RDRF | IDLE | OR | NF | FE | PF |
| Reset: | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |



Register: SCI Status register 2 (SC0SR2/SC1SR2) Address: $00C5/$00CD

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | RAF |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

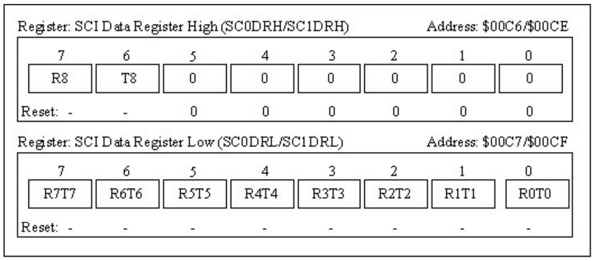


Register: SCI Baud Rate Control Register (SC0BDH/SC1BDH) Address: $00C0/$00C8

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | BTST | BSPL | BRLD | SBR12 | SBR11 | SBR10 | SBR9 | SBR8 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Register: SCI Baud Rate Control Register (SC0BDL/SC1BDL) Address: $00C1/$00C9

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | SBR7 | SBR6 | SBR5 | SBR4 | SBR3 | SBR2 | SBR1 | SBR0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |



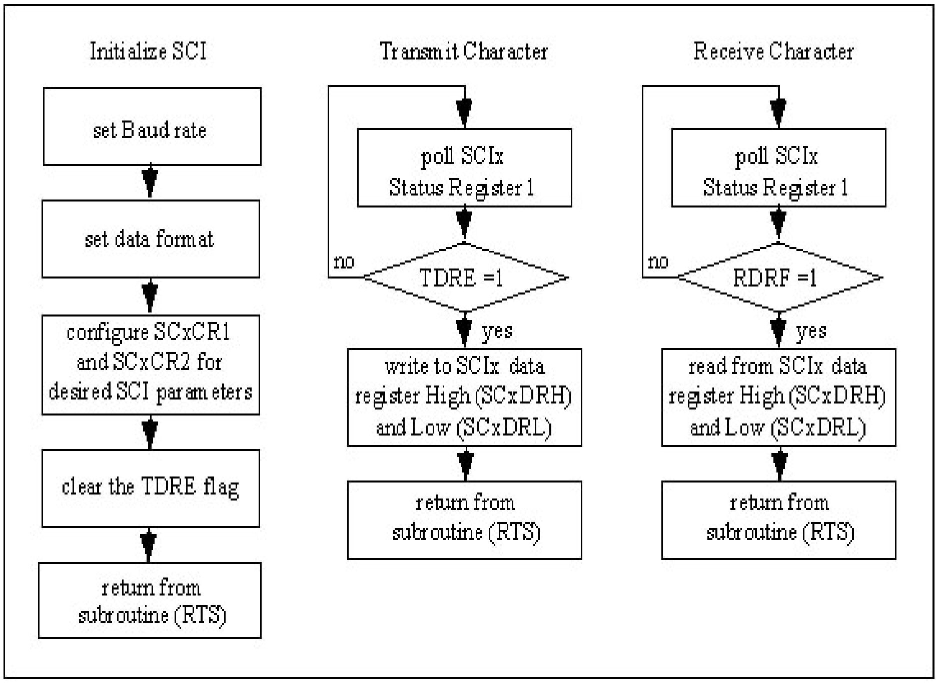
Register: SCI Data Register High (SC0DRH/SC1DRH) Address: $00C6/$00CE

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | R8 | T8 | 0 | 0 | 0 | 0 | 0 | 0 |
| Reset: | - | - | 0 | 0 | 0 | 0 | 0 | 0 |

Register: SCI Data Register Low (SC0DRL/SC1DRL) Address: $00C7/$00CF

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | R7T7 | R6T6 | R5T5 | R4T4 | R3T3 | R2T2 | R1T1 | R0T0 |
| Reset: | - | - | - | - | - | - | - | - |

# SCI Programming



Initialize SCI

set Baud rate

set data format

configure SCxCR1 and SCxCR2 for desired SCI parameters

clear the TDRE flag

return from subroutine (RTS)

Transmit Character

poll SCIx Status Register 1

TDRE =1

no

yes

write to SCIx data register High (SCxDRH) and Low (SCxDRL)

return from subroutine (RTS)

Receive Character

poll SCIx Status Register 1

RDRF =1

no

yes

read from SCIx data register High (SCxDRH) and Low (SCxDRL)

return from subroutine (RTS)

# SCI Example

Write a subroutine to initialize the SCI. Assume the MCLK = 8 MHz and the data rate is 9600 BAUD.

- Configure SC1BDL, BDH
- Configure SC1CR1
- Configure SC1CR2
- Clear the TDRE flag in SC1SR1
  - Read SC1SR1
  - Write to SC1DR

**Solution:**

**SC1BDH, SC1BDL:**

SC1BDH = $00

SC1BDL = $34

**SC1CR1:**

M=0, (1 start, 1 stop, 8 data bits)

**SC1CR2:**

TE = 1

```
;SCI Initialization Example
;Assume M = 8 MHz, 9600 BAUD

;define register locations and masks
SC1BDH = $00C8            ; Baud Register High location
SC1BDL = $00C9            ; Baud Register Low location
BAUD_HI = $00             ; Baud Register High mask
BAUD_LOW = $34            ; Baud Register Low mask
SC1CR1 = $00CA            ; Control Register 1 location
SC1_CONT = $00            ; Control Register 1 mask
SC1CR2 = $00CB            ; Control Register 2 location
SC1_MASK = $08            ; Control Register 2 mask
SC1SR1 = $00CC            ; Status Register 1 location
SC1DRL = $00CF            ; Data Register Low location
_main::
             JSR          sci_init
             :
             :
             swi

;sci_init: initializes SCI system
sci_init::   LDAA         #BAUD_LOW     ;set BAUD rate
             STAA         SC1BDL
             LDAA         #BAUD_HI
             STAA         SC1BDH
             LDAA         #SCI_CONT     ;set MODE
             STAA         SC1CR1
             LDAA         #SC1_MASK     ;enable transmitter
             STAA         SC1CR2
             LDAA         SC1SR1        ;clear TDRE bit - 2 step
   process
             STAA         SC1DRL        ;1: read SC1SR1, write SC1DRL
             RTS                        ;return from subroutine
```