Chapter 9: Asynchronous Serial Interface

- Parallel data transfer requires many I/O pins; thus the MCU cannot interface with as many devices as desired. Cost of parallel transfer is higher.

- Many I/O devices do not have high data rate to justify using parallel data transfer.

- Data synchronization for parallel transfer is difficult to achieve over a long distance. This is one reason that data communications always uses serial transfer.

Serial Communication Interface (SCI)

- Transfer data only in ***asynchronous*** mode that utilizes the TIA-232 standard.
- Use two wires; therefore, lower cost.

Serial Peripheral Interface (SPI)

- A protocol proposed by Motorola to facilitate the data exchange between the Motorola MCUs and Motorola peripheral devices.
- Uses three wires (SCK, MOSI, MISO) to exchange data.
- The SCK signal is the clock signal that synchronizes the data transfer.

Asynchronous Serial Communications

Often used for data communication between a ***data terminal equipment*** (DTE) (e.g., a computer or a MCU) and a ***data communication equipment*** (DCE) (e.g., a modem).

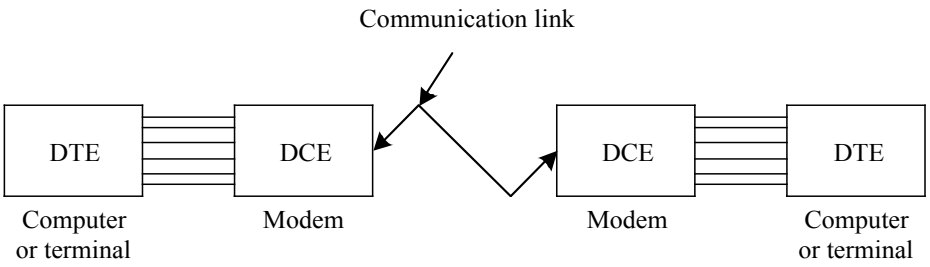


Figure 9.0 A data communication system

Three kinds of data communication links:

- ***simplex*** (one direction transmission only)
- ***half-duplex*** (both directions, but one direction can transmit at one time)
- ***full-duplex*** (both directions can transmit at the same time)

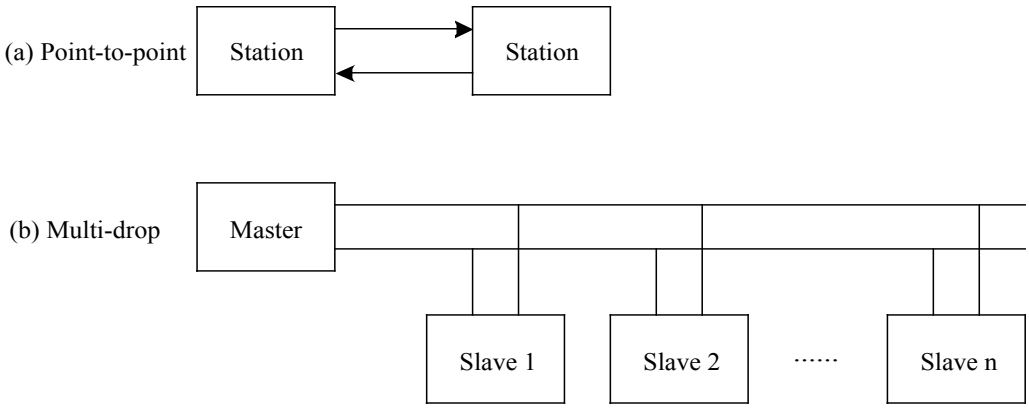


Figure 9P.2 Point-to-point and multi-drop communication links

RS232 Standard

- Most widely used physical level interface for data communications.
- Established in 1960 by Electronics Industry Association (EIA).
- Revised into TIA-232C in 1969, TIA-232D in 1987, TIA-232E in 1992 and renamed as TIA-232E.
- Specifies 25 circuits for DTE/DCE use in data communications.
- Four specifications: ***electrical***, ***functional***, ***procedural***, and ***mechanical.***

1. EIA-232E Electrical Specifications

- Interface is rated at a signal rate ≤ 20 kbps (e.g., 300, 1200, 2400, 9600, 19,200).
- Signal can transfer correctly ≤ 15 meters.
- Maximum driver output voltage is –25V ~ +25V.
- Voltage < –3V is interpreted as logic 1. Voltage > +3V is interpreted as logic 0.
- Between -3V and +3V is the transition region in which a signal state is undefined.
- When there is no transmission, the signal is held in the negative voltage (logic 1).

2. EIA-232E Functional Specifications

- One primary channel (mostly used for data transfer) and one secondary channel (rarely used).
- One data circuit in each direction; therefore ***full-duplex*** is possible.
- When the secondary channel is implemented, it is used for control of the remote modem, requests for transmission when errors occur, and governance over the setup of the primary channel.

Table 9.1 Functions of EIA-232-E signals

Pin No.	Circuit	Description
1	-	Shield
2	BA	Transmitted data
3	BB	Received data
4	CA/CJ	Request to send/ready for receiving[1]
5	CB	Clear to send
6	CC	DCE ready
7	AB	Signal common
8	CF	Received line signal detector
9	-	(reserved for testing)
10	-	(reserved for testing)
11	-	unassigned[3]
12	SCF/CI	Secondary received line signal detection/data rate selector (DCE source[2])
13	SCB	Secondary clear to send
14	SBA	Secondary transmitted data
15	DB	Transmitter signal element timing (DCE source)
16	SBB	Secondary received data
17	DD	Receiver signal element timing
18	LL	Local loopback
19	SCA	Secondary request to send
20	CD	DTE ready
21	RL/CG	Remote loopback/signal quality detector
22	CE	Ring indicator
23	CH/CI	Data signal rate selector (DTE/DCE source)[2]
24	DA	Transmitter signal element timing (DTE source)
25	TM	Test mode

1. When hardware flow control is required, circuit CA may take on the functionality of circuit CJ. This is one change from the former EIA-232.
2. For designs using interchange circuit SCF, interchange circuits CH and CI are assigned to pin 23. If SCF is not used, CI is assigned to pin 12.
3. Pin 11 is unassigned. It will not be assigned in future versions of EIA-232. However, in international standard ISO 2110, this pin is assigned to select transmit frequency.

- One ground lead (pin 1) is for protective isolation.
- Another ground lead (pin 7) serves as the return circuit for both data leads. Hence, transmission is unbalanced, with only one active wire.

- Pin 2 carries the transmit data (TxD) signal from the DTE device to the DCE device. Pin 3 carries the received data (RxD) from the DCE device to the DTE device.

- Pin 4 carries the request to send (RTS) signal, which is asserted to logic 0 (positive voltage) to prepare the DCE device for accepting transmitted data from the DTE device.
- Pin 5 carries the clear to send (CTS) signal, which is asserted to logic 0 (positive voltage) by the DCE device to inform the DTE device that transmission may begin.

- RTS and CTS are used as **handshaking signals** to moderate the flow of data into the DCE device.

- The secondary channel uses pins 14, 16, 10, and 13, respectively.

- Pin 6 is the DCE ready (DSR) signal. When originating from a **modem**, this signal is asserted to logic 0 when the modem is connected to an active telephone line that is off-hook, is in data mode, and has completed dialing or call setup functions and is generating an answer tone.
- Pin 20 is the DTE ready (DTR) signal. This signal is asserted to logic 0 by the DTE device when it wishes to open a communications channel.

- Pin 8 (and pin 12 for the secondary channel) is the **carrier detect** (CD) signal, useful only when the DCE device is a modem. This signal is asserted to logic 0 by the modem when the telephone line is off-hook, a connection has been established, and an answer tone is being received from the remote modem.
- Pin 22 is the ringer indicator (RI) signal, useful only when the DCE device is a modem. This signal is asserted to logic 0 when a ringing signal is being received from the telephone line.

- Pin 23 is the data signal rate selector (CH/CI) signal, which may originate in either the DTE or DEC device (but not both) and is used to select one of two prearranged bauds. Logic 0 = higher baud (symbol rate).

- Pin 15 is the transmitter clock (TC) signal, only useful when the DCE device is a modem, operating with a synchronous protocol. The modem generates this clock signal to control exactly the rate at which data is sent on TxD (pin 2) from the DTE to the DCE. The logic 1-to-0 (negative to positive voltage) transition on this line causes a corresponding transition to the next data element on the TxD line.
- Pin 17 is the receiver clock (RC) signal, similar to TC, except that it provides timing information for the DTE receiver.
- Pin 24 is the external transmitter clock (ETC) signal, with timing signals provided by the DTE device for use by a modem. This signal is used only when TC and RC (pins 15 and 17) are not in use. The logic 1-to-0 transition indicates the time center of the data element.

- Pin 18 is the local loopback (LL) signal, generated by the DTE and used to place the modem into a test state. When LL is asserted to logic 0 (positive voltage), the modem redirects its modulated output signal back into its receive circuitry. This enables data generated by the DTE echoing back to check the condition of the modem circuitry.
- Pin 21 is the remote loopback (RL) signal, generated by the DTE and is used to place the remote modem into a test state to exercise the transmission channel and confirm its integrity. When RL is asserted to logic 0 (positive voltage), the

remote modem redirects its received signal back into its transmitted data input, thus remodulating the received data and returning it to its source.

- Pin 25 is the test mode (TM) signal, useful only when the DCE is a modem. When asserted to login 0, it indicates that the modem is in an LL or RL state. Other internal self-test conditions may also cause the TM signal to be asserted.

3. EIA-232-E Mechanical Specification

- Specifies a 25-pin D-type connector.
- Usually a 9-pin connector (DB9) is used in most PCs, but it is not part of the TIA-232E standard

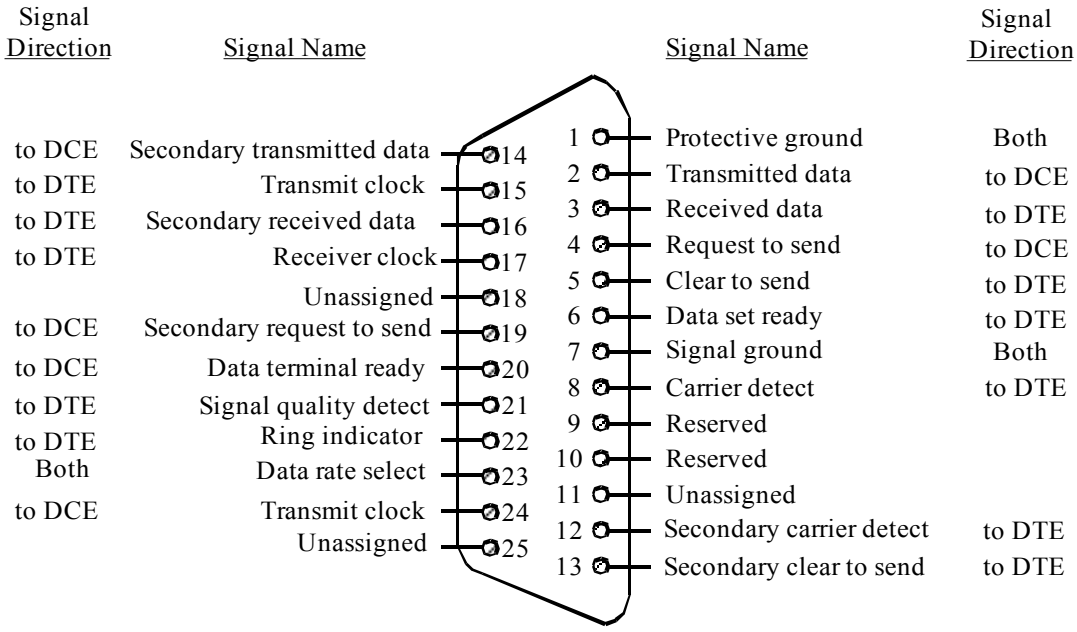


Figure 9.1a TIA-232F DB25 connector and pin assignment

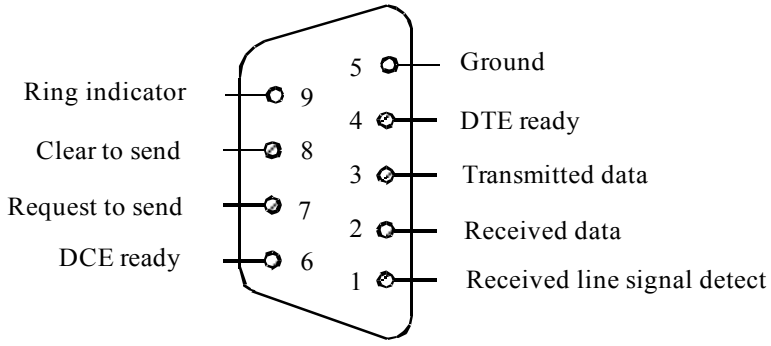


Figure 9.1b TIA-232F DB9 connector and signal assignment

4. EIA-232-E Procedural Specification

- Define the sequence of events that occurs during data transmission.

Case Study 1

Two DTEs connected via a point-to-point link using a modem.

The modem requires only the following circuits to operate:

- signal ground (GND) = AB
- transmitted data (TxD) = BA
- received data (RxD) = BB
- request to send (RTS) = CA
- clear to send (CTS) = CB
- data set ready (DSR) = CC
- carrier detect (CD) = CF

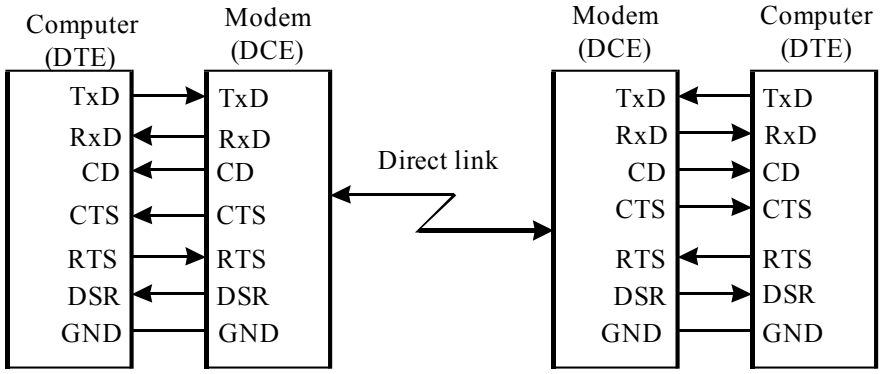
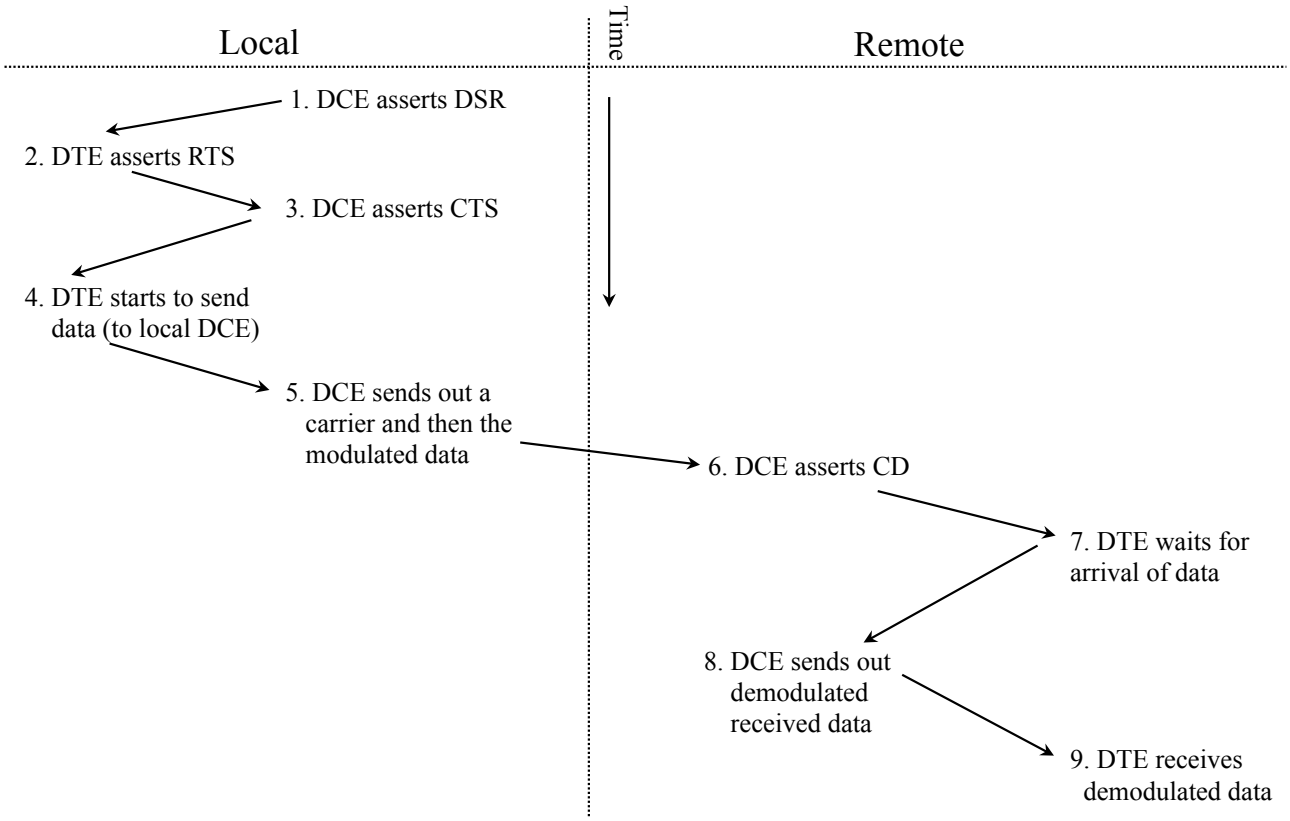


Figure 9.2 Point-to-point asynchronous connection

1. Before the DTE can transmit data or make a request to send data, the data-set-ready (DSR) signal must be asserted (logic 0) to indicate that the modem is ready.
2. When a DTE is ready to send data, it asserts the request-to-send (RTS) signal.
3. When ready, the modem responses with clear-to-send (CTS), indicating that data may be transmitted over circuit BA.
4. The DTE sends data to the local modem bit serially.
5. The local modem modulates the data into the carrier signal and transmits the resultant signal over the dedicated communication lines.
6. Before sending out modulated data, the local modem sends out a carrier signal to the remote modem so that the remote modem is ready to receive the data.
7. The remote modem detects the carrier and asserts the data-carrier-detect (CD) signal, telling the remote DTE that the local modem is transmitting.
8. The remote modem receives the modulated signal, demodulates it to recover the data, and sends it to the remote DTE over the received-data pin.

Sequence of events occurred during data transmission over dedicated link



Case Study 2

Two computers (DTEs) exchange data through a public telephone line. One of the computers (initiator) must dial the phone to establish the connection, just like people talking over the phone. Two additional pins are needed.

- data terminal ready (DTR) = CD (used by the DTE to indicate its intention to make a call or accept a call)
- ring indicator (RI) = CE (used by the DCE to indicate if incoming call exits)

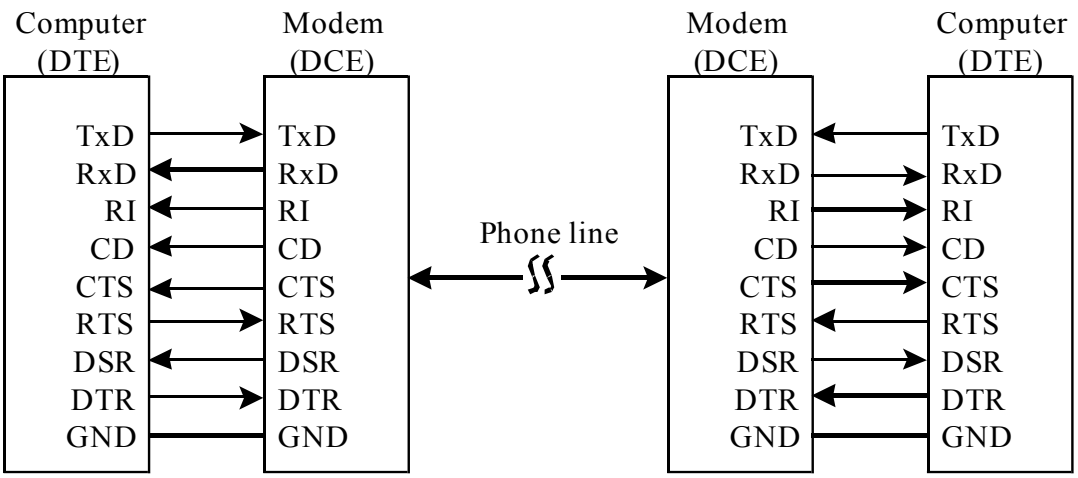


Figure 9.3 Asynchronous connection over public phone line

Phase 1: Establishing the Connection

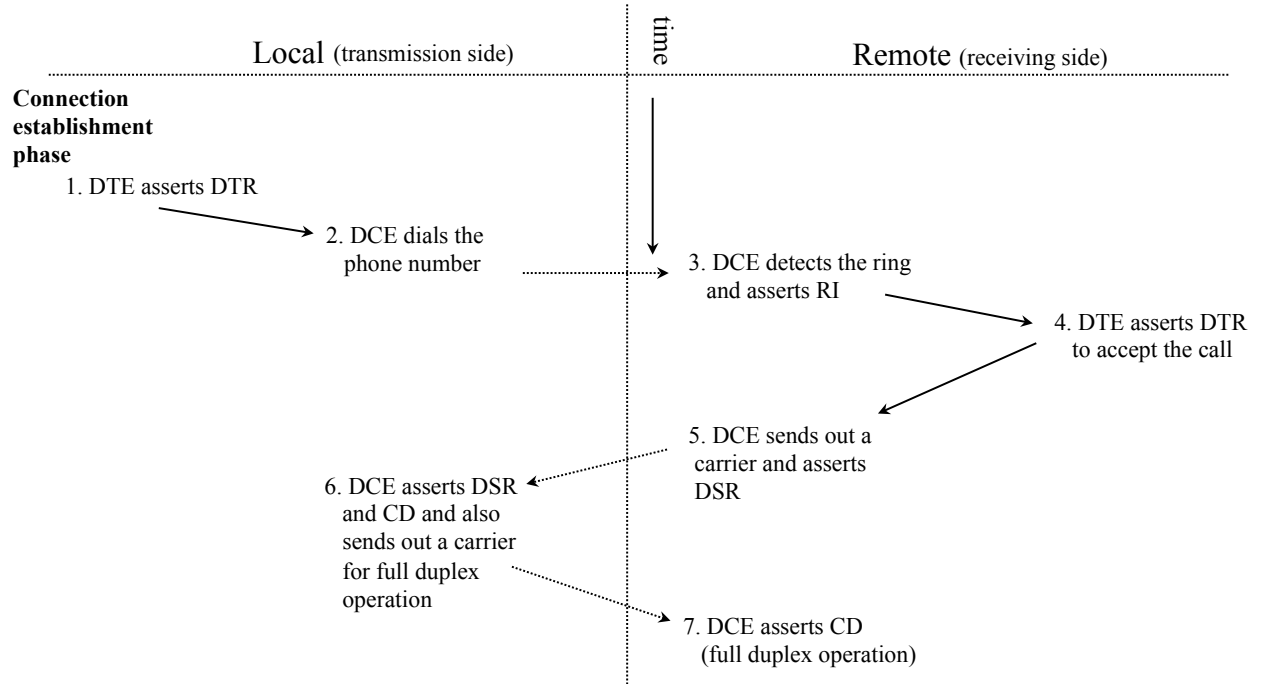
1. The transmitting computer asserts the data terminal ready (DTR) signal to indicate to the local modem that it is ready to make a call.
2. The local modem opens the phone line and dials the destination phone number. (The number can be stored in the modem or transmitted to the modem by the computer via the transmit-data pin.)

3. The remote modem detects a ring on the phone line and asserts the ring indicator (RI) signal to inform the remote computer that a call has arrived.
4. The remote computer asserts the DTR signal to accept the call.
5. The remote modem answers the call by sending carrier signal to the local modem via the phone line. It also asserts the DSR signal to inform the remote computer that it is ready for data transmission.

6. The local modem asserts both DSR and CD signals to the local computer to indicate that the connection is established and it is ready for data communication.

7. For full-duplex data communication, the local modem also sends a carrier signal to the remote modem. The remote modem then asserts the CD signal.

Sequence of events occur during data transmission over public phone line



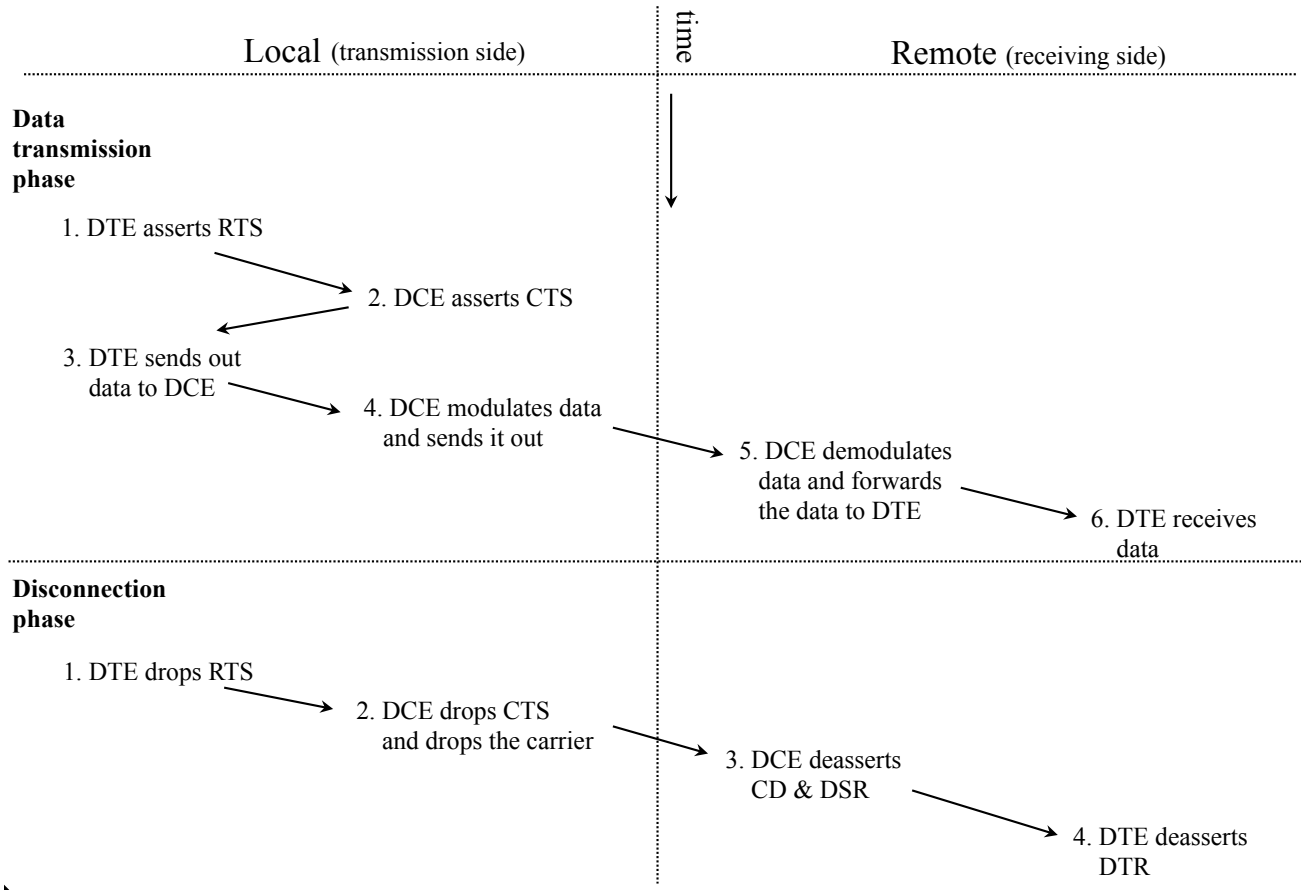
Phase 2: Data Transmission

1. The local computer asserts the RTS signal when it is ready to send data.
2. The local modem responds by asserting the CTS signal.
3. The local computer sends data bit serially to the local modem over the TxD pin. The local modem then modulates its carrier signal to transmit the data to the remote modem.
4. The remote modem receives the modulated signal from the local modem, demodulates it to recover the data, and sends it to the remote computer over the received-data pin.

Phase 3: Disconnect

- When the local computer has finished the data transmission, it drops the RTS signal.
- The local modem then de-asserts the CTS signal and drops the carrier (equivalent to hanging up the phone).

Sequence of events occur during data transmission (continued)



Data Format for Asynchronous Data Communication

- Data is transmitted character by character (one character per frame) and bit by bit.
- Each character consists of
  - 1 start bit (logic 0).
  - 8 to 9 data bits (the 9th bit is the optional parity bit.)
  - 1 stop bit (logic 1).
  - LSB is transmitted first; MSB is transmitted last.

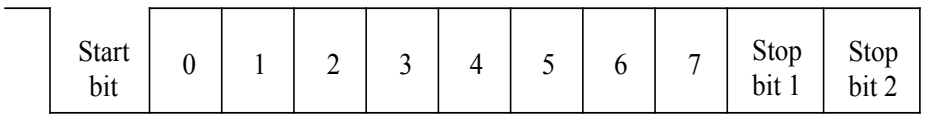


Figure 9.4 The format of a character

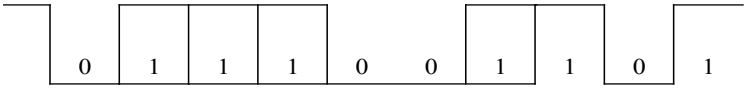
Detecting Arrival of Start Bit

- Since there is no clock information in the asynchronous format, the receiver uses a clock signal with a frequency that is 16 times of the data rate to sample the incoming data signal in order to detect the arrival of the start bit.
- To detect the arrival of a start bit, the SCI waits for the falling edge after the RxD pin has been idle (in logic 1) for at least three sampling times. The SCI circuit will then check the 3rd, 5th, and 7th samples after the first low (logic 0) sample. If the majority of them are low, then a valid start bit is detected. Otherwise, the SCI circuit will restart the process.
- After detecting a valid start bit, the SCI circuit will start to shift in the data bits.

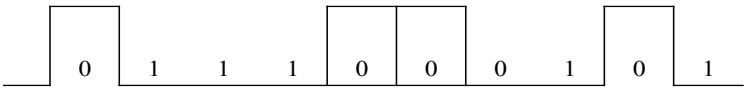
Determining Logic Value of a Data Bit

- Use a clock signal with frequency that is at least 16 times of the data rate to sample the incoming data signal.
- Take the majority function of the 8th, 9th, and 10th samples. If the majority of them are 1s, then the logic value is determined to be 1.

Example: Sketch the output of the letter g when it is transmitted using the format of one start bit, 8 data bit, and 1 stop bit. The ASCII code of letter g is $67 or %01100111.



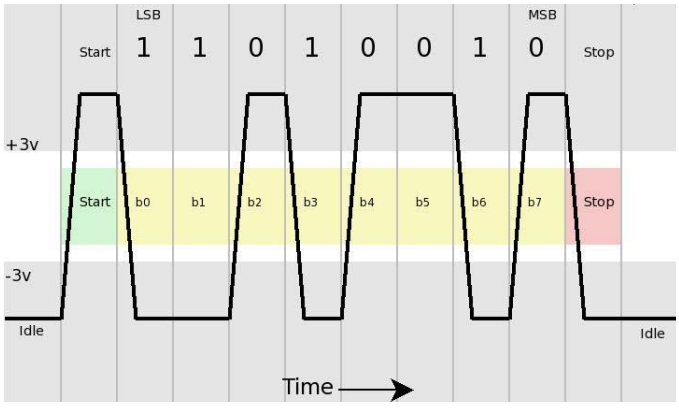
(a) output waveform on microcontroller interface



(b) output waveform on EIA-232-E interface

Figure 9.6 Data format for letter **g**

The output waveform on the TIA-232 interface is <-3V = logic 1 but >+3V = logic 0.



Data Transmission Errors

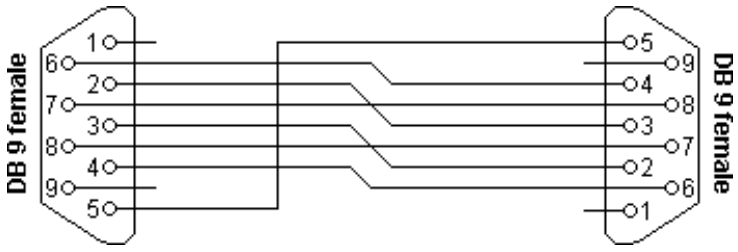
- A framing error occurs when a received character is improperly framed by the start and stop bits. This error indicates a synchronization problem or faulty transmission.
- Receiver overrun occurs when one or more characters received but not read by the CPU.
- Parity error occurs when an odd number of bits change value, due to noise.

Null Modem Connection

- One most popular applications of the TIA-232 interface is to connect the PC to a single-board computer (or called demo board), both sitting side-by-side.
- Both the PC and the single-board computer are DTEs.
- The null modem interconnects fool both DTEs into thinking that they are connected to modems.
- The transmitter and receiver timing signals are not needed in asynchronous data transmission. The ring indicator (RI) signal is not needed neither because the null modem mode does not operate on a public phone line.

Signal Name	DTE 1		DTE 2		Signal Name
	DB25 pin	DB9 pin	DB9 pin	DB25 pin	
FG (frame ground)	1	-	-	1	FG
TD (transmit data)	2	3	2	3	RD
RD (receive data)	3	2	3	2	TD
RTS (request to send)	4	7	8	5	CTS
CTS (clear to send)	5	8	7	4	RTS
SG (signal ground)	7	5	5	7	SG
DSR (data set ready)	6	6	4	20	DTR
CD (carrier detect)	8	1	4	20	DTR
DTR (data terminal ready)	20	4	1	8	CD
DTR (data terminal ready)	20	4	6	6	DSR

Table 9.2 Null Modem connection



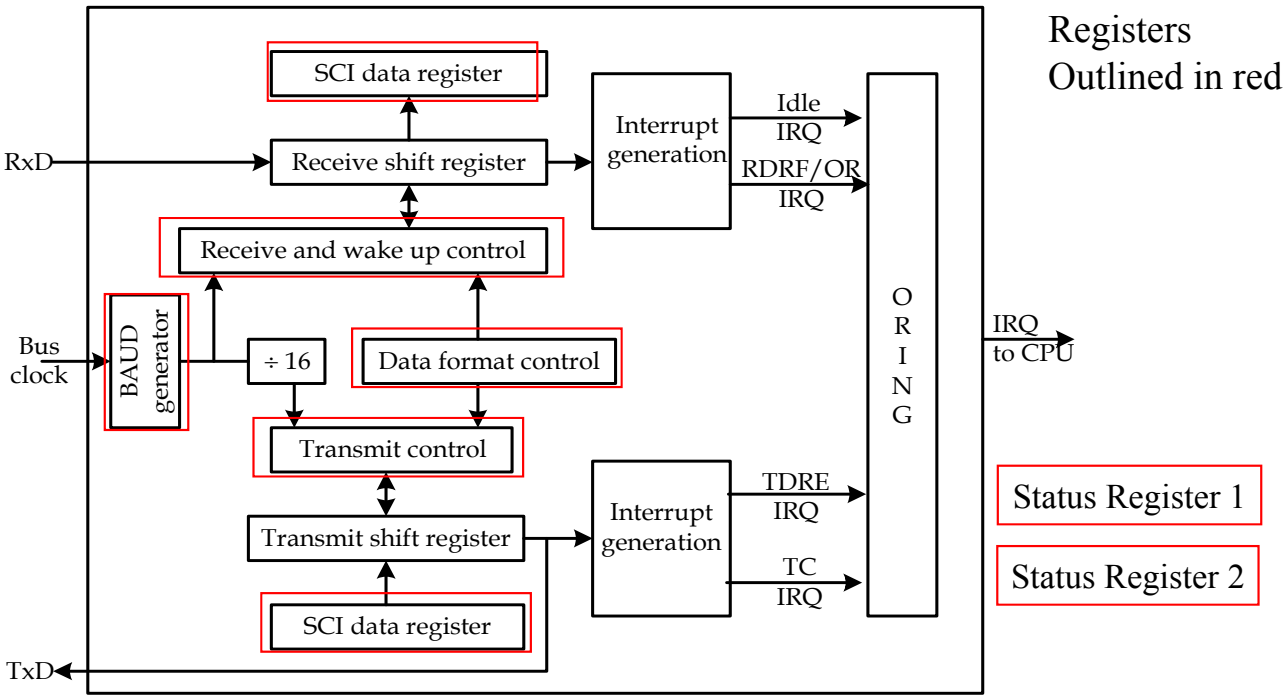
Null modem cable: Computer to Computer

HCS12 SCI Subsystem

- Two identical serial communication interface modules (SCI0 & SCI1).
- Use the data format of a frame, consisting one start, eight or nine data bits, and one stop bit. (The 9th bit can be used as the parity bit.)
- One SCI channel uses two signal pins from the Port S. The channel SCI0 uses pins PS0 (for RxD0) and PS1 (for TxD0), whereas the channel SCI1 uses pins PS2 (for RxD1) and PS3 (for TxD1).

- SCI has the capability to send break to attract the attention of other party of communication. (A **break** is defined as the transmission or reception of logic 0 for a frame or longer time.)
- SCI supports hardware parity for transmission and reception.
- The SCI supports idling line and address mark wakeup, which is useful in **multi-drop** environment to reduce the software overhead.

SCI channel 0	SCI channel 1	
SC0BDH	SC1BDH	; these registers are for
SC0BDL	SC1BDL	; setting the baud (= sampling rate)
SC0CR1	SC1CR1	; these two registers (per channel) are for
SC0CR2	SC1CR2	; controlling the SCI operations
SC0SR1	SC1SR1	; status registers
SC0DRH	SC1DRH	; data register: higher byte
SC0DRL	SC1DRL	; data register: lower byte



Baud (Data-Rate) Generation

- CPU uses a clock signal with frequency equal to 16 times of the data rate to detect the arrival of the start bit and determine the logic value of data bits.
- SCI module uses a 13-bit counter to generate this clock signal. This circuit is called ***baud (data-rate) generator***.
- The user writes an appropriate value into the 16-bit baud control (SCxBD) register pair, SCIxBDH : SCIxBDL, to set the baud rate, where x = 0 or 1.

- The value to be written into the baud rate generator register is the rounding of

Baud (Data-Rate) Divisor = E-clock rate / (16 x data rate)

	7	6	5	4	3	2	1	0
	0	0	0	SBR12	SBR11	SBR10	SBR9	SBR8
reset:	0	0	0	0	0	0	0	0

(a) SCI baud rate control register high (SCI0BDH/SCI1BDH)

	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

(b) SCI baud rate control register low (SCI0BDL/SCI1BDL)

Figure 9.8 SCI baud rate control register

Table 9.3 Baud rate generation

Desired SCI Baud Rate	Baud Rate Divisor for f_E = 16 MHz	Baud Rate Divisor for f_E = 24 MHz
300	3333	5000
600	1667	2500
1200	833	1250
2400	417	625
4800	208	313
9600	104	156
14,400	69	104
19,200	52	78
38,400	26	39

Control Registers Associated with SCI

- Each SCI channel has two control registers that set up other parameters: (SCxCR1 & SCxCR2), where x=0 or 1.
- Character format, parity checking, wake-up method are programmed via the SCxCR1.
- Transmit and receive interrupts enabling, transmission and reception enabling, wakeup enabling and break sending are programmed via the SCxCR2 register.

7	6	5	4	3	2	1	0	
LOOPS	SCISWAI	RSRC	M	WAKE	ILT	PE	PT	Reset value = 0x00

LOOPS: loop select bit
0 = loop operation disabled
1 = loop operation enabled
SCISWAI: SCI stop in wait mode
0 = SCI enabled in wait mode.
1 = SCI disabled in wait mode.
RSRC: receiver source bit
When LOOPS = 1, the RSRC bit determines the source for the receiver shift register
0 = receiver input connected to the transmitter internally(not TxD pin).
1 = receiver input connected externally to the transmitter(TxD pin).
M: data format mode bit
0 = one start bit, eight data bits, one stop bit
1 = one start bit, nine data bits, one stop bit
WAKE: wakeup condition bit
0 = idle line wakeup
1 = address mark wakeup (last data bit set)
ILT: idle line type bit
0 = idle character bit count begins after start bit
1 = idle character bit count begins after stop bit
PE: parity enable bit
0 = parity disabled
1 = parity enabled.
PT: parity type bit(for both transmit and receive)
0 = even parity selected
1 = odd parity selected

Figure 9.9 SCI control register 1 (SCI0CR1/SCI1CR1)

7	6	5	4	3	2	1	0	
TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK	Reset value = 0x00

TIE: transmit interrupt enable bit
0 = TDRE interrupt disabled
1 = TDRE interrupt enabled
TCIE: transmit complete interrupt enable bit
0 = TC interrupt disabled.
1 = TC interrupt enabled.
RIE: receiver full interrupt enable bit
0 = RDRF and OR interrupts disabled
1 = RDRF and OR interrupt enabled
ILIE: idle line interrupt enable bit
0 = IDLE interrupt disabled.
1 = IDLE interrupt enabled.
TE: transmitter enable bit
0 = transmitter disabled.
1 = transmitter enabled.
RE: receiver enable
0 = receiver disabled.
1 = receiver enabled.
RWU: receiver wakeup bit
0 = normal SCI receiver.
1 = enables the wakeup function and inhibits further receiver interrupts. Normally, hardware wakes up the receiver by automatically clearing this bit
SBK: send break bit
0 = no break characters.
1 = generate a break code, at least 10 or 11 contiguous0s. As long as SBK remains set, the transmitter sends 0s.

Figure 9.10 SCI control register 2 (SCI0CR2/SCI1CR2)

Character Transmission

- To transmit a character, the MCU writes the data bits into the 16-bit SCI data registers SCIxDRH and SCIxDRL, where x = 0 or 1. (When 8 data bit format is used, these 8 bits are stored in the lower byte and we only need to deal with the SCxDRL data register.)

- The data bits in SCIxDRH and SCIxDRL registers will be transferred to the transmit shift register and shifted out serially from the TxD pin in the TIA-232 frame format (after adding a start and stop bit).

- Each time the SCI transfers data from the SCIxDRH and SCIxDRL data registers to the transmit shift register, it also sets the transmit data register empty (TDRE) flag in the SCI status register SCIxSR1.
- The setting of the TDRE flag to logic 1 indicates that the MCU can write new data into the SCI data register.

- When the transmit shift register is not transmitting data, the TxD signal goes to idle state.

- When both the transmit data registers and shift register are empty, the transmit complete (TC) flag in the SCIxSR1 register is set to 1.

7	6	5	4	3	2	1	0	
TDRE	TC	RDRF	IDLE	OR	NF	FE	PF	Reset value = 0xC0

TDRE: transmit data register empty flag
0 = no byte was transferred to the transmit shift register .
1 = transmit data register is empty .
TC: transmit complete flag
0 = transmission in progress
1 = no transmission in progress
RDRF: receiver data register full flag
0 = SCIxDR empty
1 = SCIxDR full
IDLE: idle line detected flag
0 = RxD line active
1 = RxD line becomes idle
OR: overrun error flag
0 = no overrun
1 = overrun detected
NF: noise error flag
Set during the same cycle as the RDRF bit but not set in the case of an overrun (OR).
0 = no noise
1 = noise
FE: framing error flag
Set when a 0 is detected where a stop bit was expected .
0 = no framing error
1 = framing error
PF: Parity error flag
0 = parity correct
1 = incorrect parity detected

Figure 9.12 SCI status register 1 (SCI0SR1/SCI1SR1)

7	6	5	4	3	2	1	0	
0	0	0	0	0	BK13	TXDIR	RAF	Reset value = 0x00

BK13: break transmit character length
0 = break character is 10- or 11-bit long.
1 = break character is 13- or 14-bit long.
TXDIR: transmit pin data direction in single -wire mode
0 = TxD pin to be used as an input in single -wire mode.
1 = TxD pin to be used as an output in single -wire mode.
RAF: receiver active flag
RAF is set when the receiver detects a logic 0 during the RT1 time period of the start bit search . RAF is cleared when the receiver detects an idle character .
0 = no reception in progress
1 = reception in progress

Figure 9.13 SCI status register 2 (SCI0SR2/SCI1SR2)

- Each SCI channel has two **status registers** (SCxSR1 and SCxSR2) to record the status of SCI operation, where x = 0 or 1.

- Data transmission and reception status, idle line status, and reception errors are recorded in the SCxSR1 status register.

- The TDRE flag and TC flag in SCxSR1 represent different situations.
  - When the byte in the SCxDRL **data register** is transferred to the transmit shift register, then TDRE =1 and allows a new byte to be written into the SCxDRL data register.
  - Transmit complete flag TC=1 only when the transmitter is idle.

- If the transmit interrupt enable bit (TIE) in the SCIxCR2 register is also set to 1, the TDRE flag generates a transmit interrupt request.

- The noise error (NF) flag in ScxSR1 will be set to 1 whenever one of the three samples is different from the other two used to detect the start bit, or determine the stop bit or value of the data bit. NF=1 may not be an error condition. However, the setting of any one of the ORE, FE, or PF flags is an error condition.

- The receiver-related flags in SCxSR1 are cleared by a read of the SCxSR1 register followed by a read of the transmit/receive data register low byte. The transmit-related bits in SCxSR1 (TDRE and TC) are cleared by a read of the SCxSR1 register followed by a write to the transmit/receive data register low byte.

Steps in the SCI Transmission

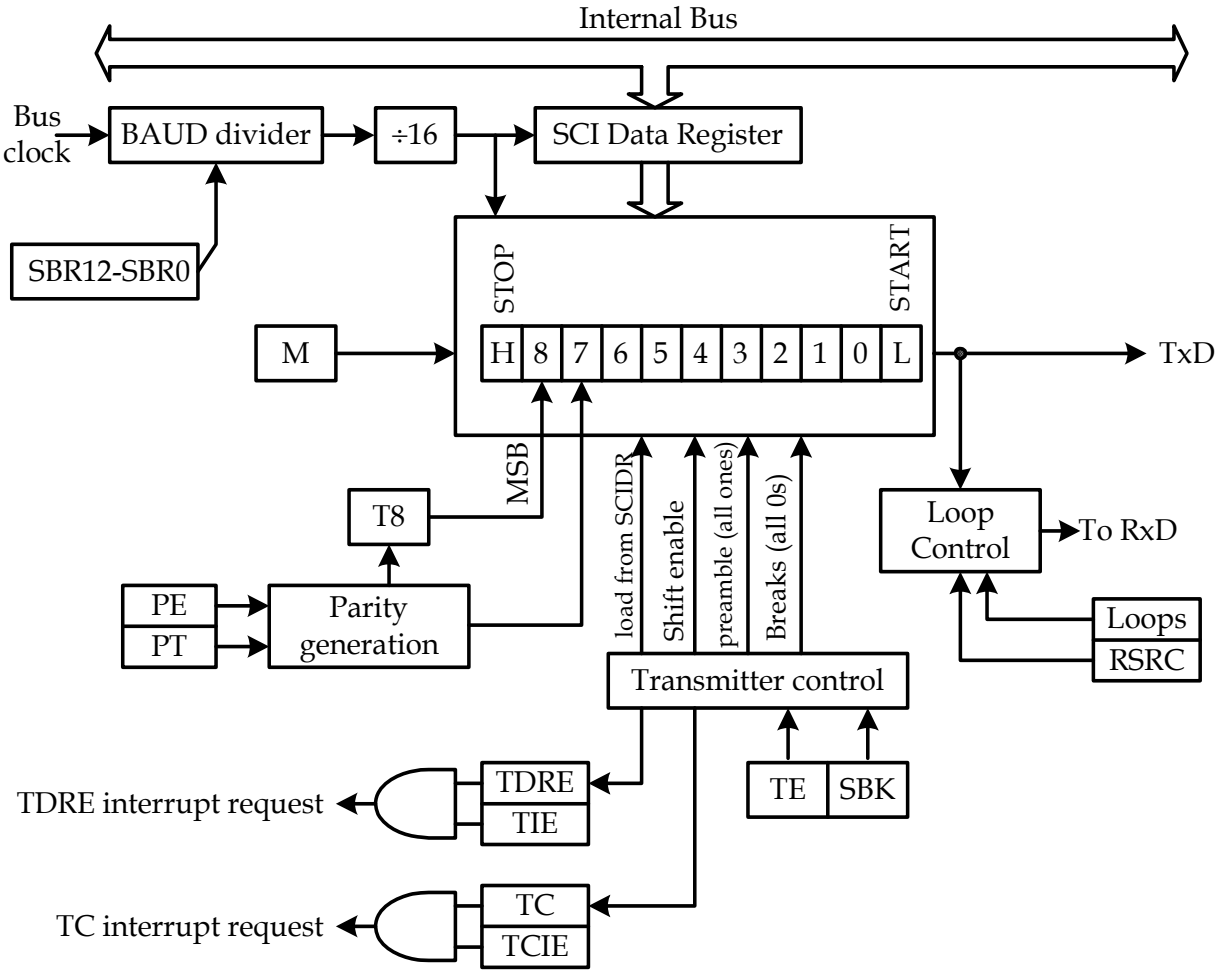


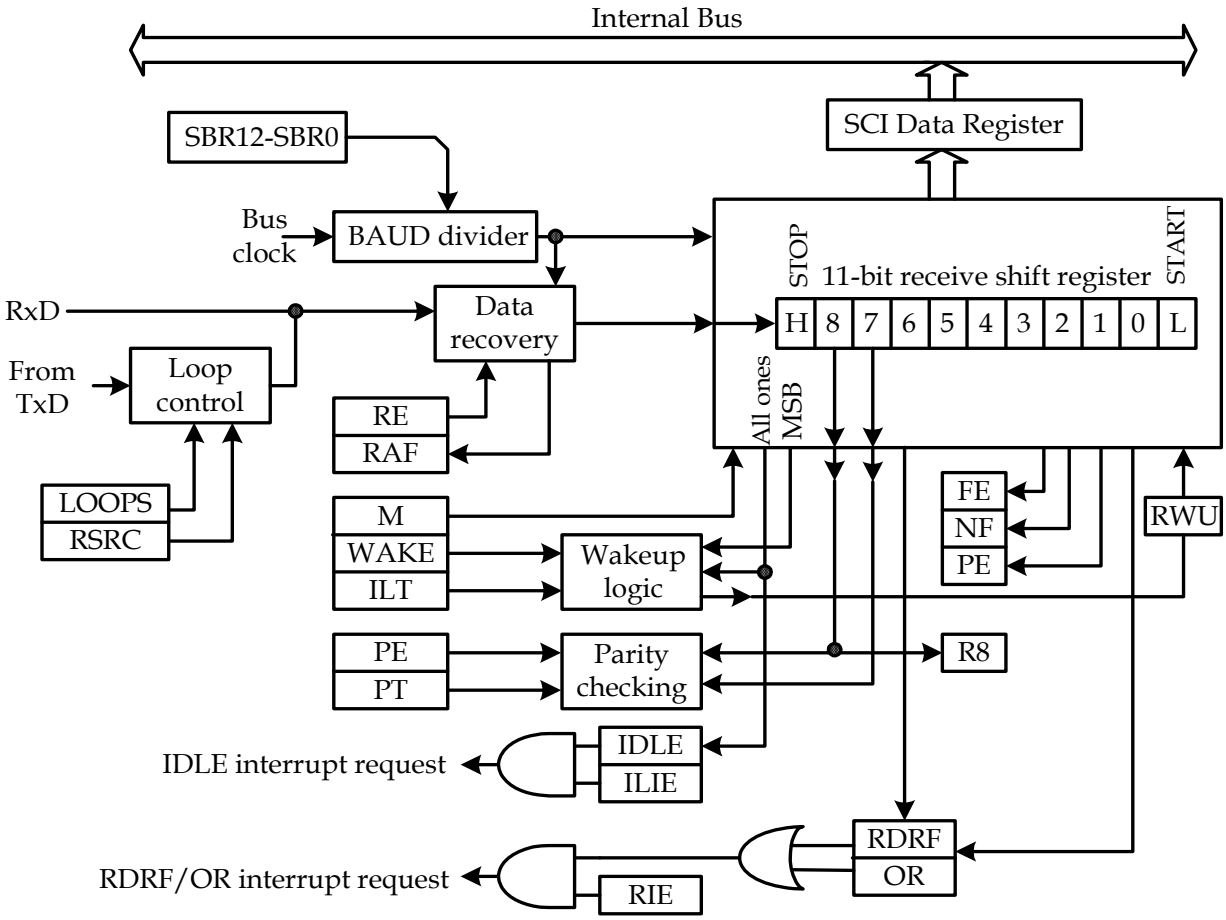
Figure 9.12 SCI transmitter block diagram

1. Configure the SCI Transmission

- Select a baud (= data rate) by writing to the SCIxBDH and SCIxBDL registers.
- Write to the SCIxCR1 register to configure the word length, parity, and other configuration bits.
- Enable the transmitter, interrupt, receive, and wakeup as required by writing to the SCIxCR2 control register bits.

2. Setting the transmission procedure for each character

- Poll the TDRE flag by reading the SCIxSR1 register or responding to the TDRE interrupt.
- If TDRE=1, write the data to be transmitted to SCIxDRH/L data registers. The 9th bit is written to the T8 bit in the SCIxDRH register if in 9-bit format.
- No new transmission until TDRE=0.
- Repeat step 2 for each subsequent character.



0

Figure 9.15 SCI receiver block diagram

Example: Configure SCI0 to operate with the following parameters:

- 9600 baud (E clock is 24 MHz)
- one start bit, 8 data bits, one stop bit
- no interrupt
- address mark wakeup
- disable wakeup initially
- long idle line mode
- no loop back
- disable parity checking
- enable transmit and receive

; SBR = ECLK / (16 x Baud_Rate) = 24,000,000 / (16 x 9600) = 156 (or $9C)

```
        movb   #$00,SC0BDH      ; set up baud rate in SCI0
        movb   #$9C,SC0BDL      ;        "
        movb   #$4C,SC0CR1      ; select 8 data bits, address mark wakeup
        movb   #$0C,SC0CR2      ; enable transmit and receive
```

Interfacing SCI with EIA-232-E

- SCI uses 0V and 5V to represent logic 0 and logic 1.
- The TIA-232 signals Tx and Rx cannot be driven by the SCI TxD and RxD signals without translation.
- Voltage level translation by the TIA-232 transceiver is required for the SCI signals to drive and be driven by the TIA-232 signals.
- Example: National Semiconductor DS14C232 performs the voltage translation.
- This also supports a NULL modem connection so that this connection can talk to a PC directly.

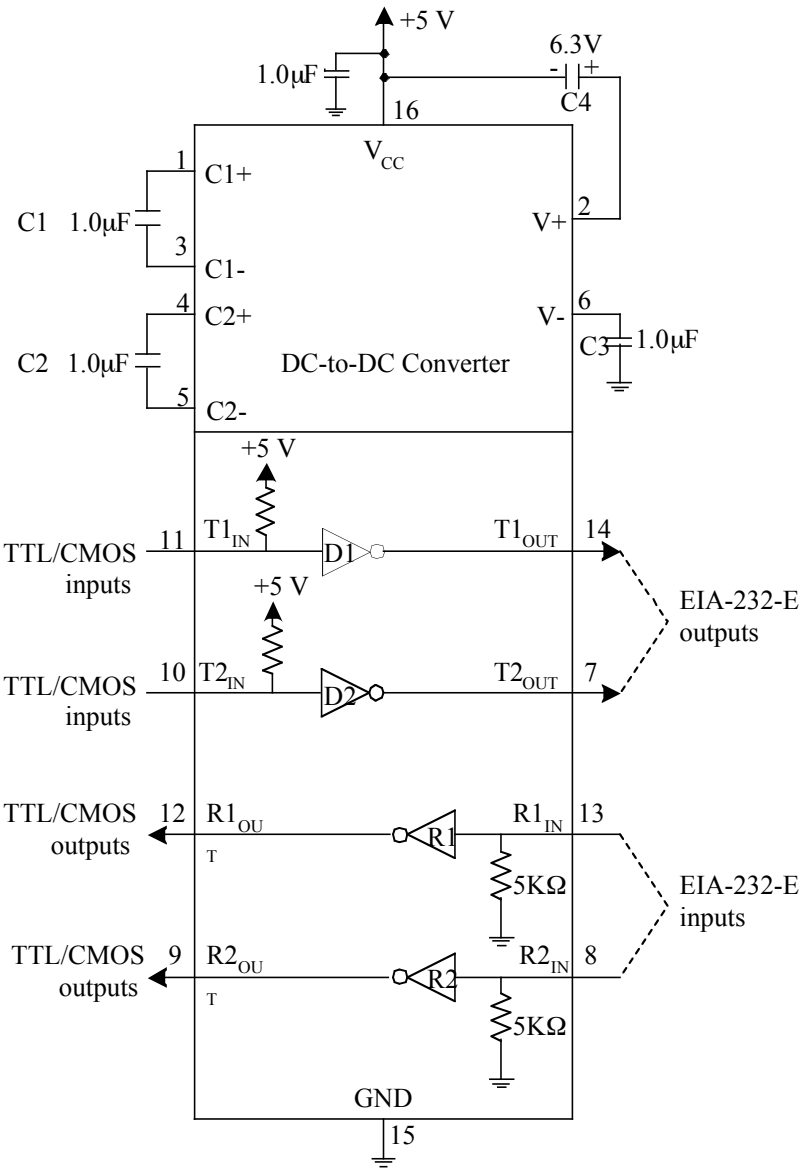


Figure 9.16 Pin assignments and connections of the DS14C232

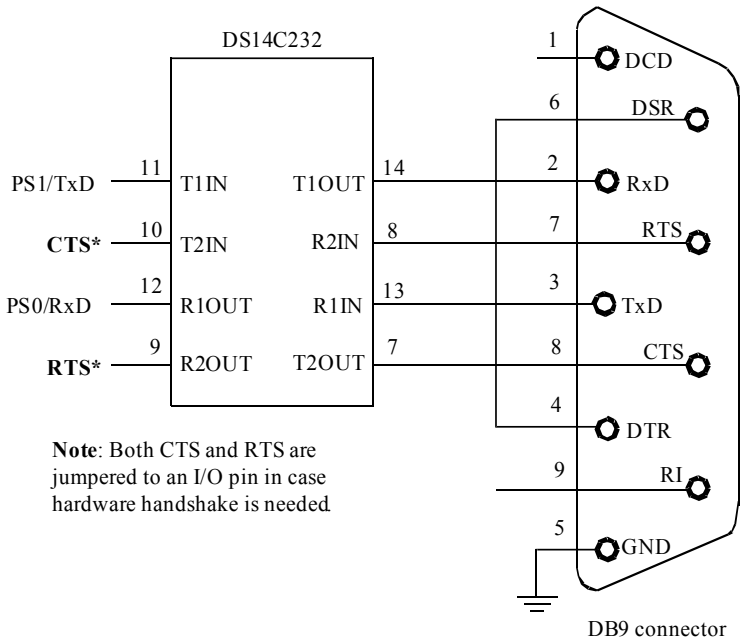


Figure 9.18 Diagram of SCI and TIA-232 DB9 connector wiring in Dragon12 demo board

Example: Send a break (of about 1 ms) to the communication port controlled by the SCI0 interface.

- A break character is represented by 10 or 11 consecutive zeros and can be sent out by setting the bit 0 of the SC0CR2 register.
- As long as bit 0 of the SC0SR2 register remains logic 1, the SCI will keep sending out the break character.

```
sendbrk     bset    SC0CR2,$01   ; turn on send break
            jsr     d1ms         ; see Chapter 8 for the 1ms ISR
            bclr    SC0CR2,$01   ; turn off send break
            rts
```

Example: Write a subroutine to output the character in accumulator A to the SCI0 channel using the **polling** method.

- A new character should be sent out only when the transmit data register is empty.
- The subroutine will wait until the TDRE bit (bit 7) of SC0SR1 register is set to 1 before sending out the character in accumulator A.

```
putc_sc0    brclr   SC0SR1,$80,*    ; wait for TDRE to be set
            staa    SC0DRL          ; output the character to data register
            rts
```

Example: Write a subroutine to read a character from the SCI0 using the polling method. The character will be returned in accumulator A.

- The subroutine will wait until the RDRF bit (bit 5) of the SC0SR1 register is set to 1 and then read the character held in the SC0DRL register.

```
getc_sc0    brclr   SC0SR1,$20,*    ; wait until RDRF bit is set
            ldaa    SC0DRL          ; read the character from data register
            rts
```

Example: Write a subroutine that outputs a string pointed to by index register X to the SCI0 using the polling method.

- Repeat putc_sc0 until all characters have been sent. That is, the NULL character (ASCII 0) is detected.

```
puts_sc0    ldaa    1,x+            ; get a character and move the pointer
            beq     done            ; is this the end of the string (null = $00) ?
            jsr     putc_sc0        ; if not, call the output character subroutine
            bra     puts_sc0        ; repeat
done        rts
```

Example: Write a subroutine that inputs a string from the SCI0 module. The string is terminated by the "enter" character and must be stored in a buffer (memory) pointed to by index register X. Also allow the user to backspace to erase errors.

- Repeat getc_sc0 until the "enter" character is received.

```
gets_sc0    jsr     getc_sc0
            cmpa    #CR             ; is the character a carriage return?
            beq     exit
            staa    0,x             ; save the character and increment the pointer
            jsr     putc_sc0        ; echo it back to SCI0
            cmpa    #BS             ; is it a backspace character? (ASCII of BS = $08)
```

```
                bne     nc              ; no, continue
                dex                     ; decrement the input buffer pointer
                ldaa    #WS             ; wipe out previous character (ASCII of WS = $20)
                jsr     putc_sc0
                ldaa    #BS
                jsr     putc_sc0
                bra     gets_sc0        ; continue
nc              inx
                bra     gets_sc0        ; continue
exit            clr     0,x             ; terminate the string with a NULL character
                rts
```

Example: Display "Hello Class" on the terminal window via SCI0.

#include "reg9s12.h"

```
                org     $2000
                lds     #$2000
                movb    #156,SC0BDL     ; get baud to 9600
                movb    #0,SC0BDH       ;
                movb    #$00,SC0CR1     ; configure SCI0 control registers
                movb    #$0C,SC0CR2     ;  enable transmit and receive
                ldx     #msg1           ; get message string
                jsr     outstrg         ; send it out serial port
                jsr     outcrlf         ; output new line & carriage-return
                jsr     outcrlf         ; output new line & carriage-return
                swi

; Output string of ASCII bytes starting at X until end of text ($04).
outstrg         jsr     outcrlf         ; output carriage-return
outstrg0        psha                    ; save A
outstrg1        ldaa    0,X             ; read char into A
                cmpa    #$04            ; is this end of text?
                beq     outstrg3        ; jump if yes
                jsr     output          ; output character
                inx                     ; increment pointer
                bra     outstrg1        ; loop
outstrg3        pula                    ; restore A
                rts

; Output a Carriage return and a line feed. Returns A = CR
outcrlf         ldaa    #$0A            ; ASCII code of LF (line feed)
                jsr     output          ; send it
                ldaa    #$0D            ; ASCII code of CR (enter)
```

```
                jsr     output          ; send it
                ldaa    #$00            ; null
                jsr     output          ; output padding
                ldaa    #$0D            ;
                rts

output          ldab    SC0SR1          ; read status
                bitb    #$80            ; test Transmit Data Register Empty bit
                beq     output          ; loop if TDRE=1
                anda    #$7F            ; mask parity
                staa    SC0DRL          ; send character
                rts

msg1            fcc     "Hello Class"
                fcb     $04             ; end of text character (EOT)
                end
```

Example: Get input from keyboard and display the message after pressing the "enter" key.

```
#include "reg9s12.h"

RDRF            equ     $20             ; bit position for RDRF flag
TDRE            equ     $80             ; bit position for TDRE flag

                org     $2000
                lds     #$2000
                ldy     #prompt1
                jsr     putstr0         ; output a prompt
                jsr     newline
                ldy     #prompt2
                jsr     putstr0
                ldy     #buff
                jsr     getstr0         ; read a string from the keyboard
                jsr     newline         ; move cursor to the next line
                ldy     #buff
                jsr     putstr0         ; echo the string
                jsr     newline
                jsr     newline
                swi

prompt1         fcc     "This is a test of the SCI0 routine."
                fcb     0
prompt2         fcc     "Please enter a string from the keyboard. Then press enter:"
```

```
                fcb     0
buff            rmb     50

;init_sc0       psha                    ; optional in Dragon12+
;               movb    #156,SC0BDL     ; get baud rate constant
;               movb    #0,SC0BDH
;               movb    #$00,SC0CR1     ; configure SCI0 control registers
;               movb    #$0C,SC0CR2     ; enable transmit and receive
;               pula
;               rts

; returns the received character in A using polling method
getch0          brclr   SC0SR1,RDRF,*   ; wait until RDRF is set to 1
                ldaa    SC0DRl          ; read character in the receive data register
                rts

; uses polling method to transmit the character in A to SC0
putch0          brclr   SC0SR1,TDRE,*   ; wait until the transmit data register empty
                staa    SC0DRL          ; transmit the character in A
                rts

; Inputs a string from SC0 and store in the buffer pointed to by Y
getstr0         jsr     getch0          ; read a character
                cmpa    #$0D            ; is it a carriage return?
                beq     done
                staa    1,Y+            ; store it in the buffer and increment the pointer
                bra     getstr0         ; get the next character
done            clr     0,Y             ; store a NULL character to the end of the string
                rts

putstr0         ldaa    1,Y+            ; get a character and increment the pointer
                beq     exit            ; reach the NULL character?
                jsr     putch0          ; output it
                bra     putstr0
exit            rts

newline         ldaa    #$0D            ; ASCII code of CR (enter)
                jsr     putch0
                ldaa    #$0A            ; ASCII code of LF (line feed)
                jsr     putch0
                rts
                end
```

End of Chapter 9