Configuring the Serial Interface

This chapter describes configuring serial interface management.

Configuring the Serial Interface

The Cisco 819 Integrated Services Router (ISR) supports synchronous by default and asynchronous serial interface protocols.

Configuring the serial interface in the Cisco 819 ISR allows you to enable applications such as WAN access, legacy protocol transport, console server, and dial access server. It also allows remote network management, external dial-modem access, low-density WAN aggregation, legacy protocol transport, and high port-density support.

Serial interfaces enables the following features:

- WAN access and aggregation
- Legacy protocol transport
- Dial access server

Serial interfaces can be used to provide WAN access for remote sites. With support for serial speeds up to 8 Mbps, it is ideal for low- and medium-density WAN aggregation.

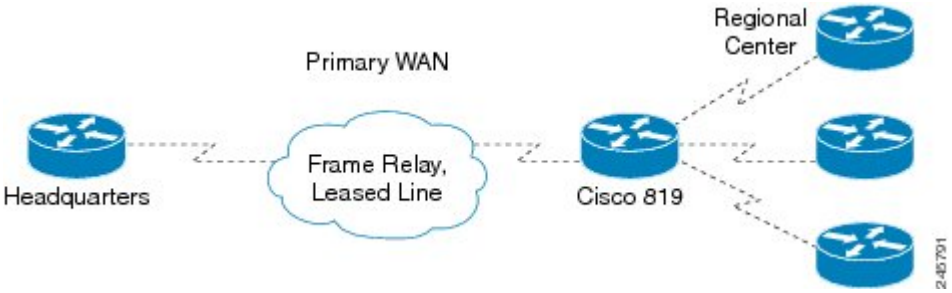
Figure 1: WAN Concentration

Legacy Protocol Transport

Serial and synchronous/asynchronous ports are ideally suited to transport legacy traffic across a TCP/IP network, facilitating network convergence. Legacy protocols supported by Cisco IOSR Software include:

- Synchronous Data Link Control (SDLC) Protocol
- Binary Synchronous Communications Protocol (Bisync)
- X.25 Protocol

Figure 2: Network Convergence



The Cisco 819 series ISRs use Cisco Smart Serial connectors. The supported cables are noted in the table below.

Table 1: Smart Serial Cabling for Cisco 819 ISRs

Product Number	Cable Type	Length	Connector Type
CAB-SS-V35MT	V.35 DTE	10 ft (3m)	Male
CAB-SS-V35FC 10 ft (3m) Female	V.35 DCE	10 ft (3m)	Female
CAB-SS-232MT	EIA/TIA-232 DTE	10 ft (3m)	Male
CAB-SS-232FC	EIA/TIA-232 DTE	10 ft (3m)	Female
CAB-SS-449MT	EIA/TIA-449 DTE	10 ft (3m)	Male
CAB-SS-449FC	EIA/TIA-449 DTE	10 ft (3m)	Female
CAB-SS-X21MT	X.21 DTE	10 ft (3m)	Male
CAB-SS-X21FC	X.21 DTE	10 ft (3m)	Female
CAB-SS-530MT	EIA/TIA-530 DTE	10 ft (3m)	Male
CAB-SS-530AMT	EIA/TIA-232 DTE	10 ft (3m)	Male

Configuring Serial Interfaces

When the router receives an indication that the primary interface is down, the backup interface becomes enabled. After the primary connection has been restored for a specified period, the backup interface is disabled.

Even if the backup interface comes out of standby mode, the router does not enable the backup interface unless the router receives the traffic specified for that backup interface.

To configure serial interfaces, you must understand the following concept:

Cisco HDLC Encapsulation

Cisco High-Level Data Link Controller (HDLC) is the Cisco proprietary protocol for sending data over synchronous serial links using HDLC. Cisco HDLC also provides a simple control protocol called Serial Line Address Resolution Protocol (SLARP) to maintain serial link keepalives. Cisco HDLC is the default for data encapsulation at Layer 2 (data link) of the Open System Interconnection (OSI) stack for efficient packet delineation and error control.

Note Cisco HDLC is the default encapsulation type for the serial interfaces.

When the encapsulation on a serial interface is changed from HDLC to any other encapsulation type, the configured serial subinterfaces on the main interface inherit the newly changed encapsulation and they do not get deleted.

Cisco HDLC uses keepalives to monitor the link state, as described in the Keepalive Timer, on page 5.

PPP Encapsulation

PPP is a standard protocol used to send data over synchronous serial links. PPP also provides a Link Control Protocol (LCP) for negotiating properties of the link. LCP uses echo requests and responses to monitor the continuing availability of the link.

Note When an interface is configured with PPP encapsulation, a link is declared down and full LCP negotiation is re-initiated after five echo request (ECHOREQ) packets are sent without receiving an echo response (ECHOREP).

PPP provides the following Network Control Protocols (NCPs) for negotiating properties of data protocols that will run on the link:

- IP Control Protocol (IPCP) to negotiate IP properties
- Multiprotocol Label Switching control processor (MPLSCP) to negotiate MPLS properties
- Cisco Discovery Protocol control processor (CDPCP) to negotiate CDP properties
- IPv6CP to negotiate IP Version 6 (IPv6) properties
- Open Systems Interconnection control processor (OSICP) to negotiate OSI properties

PPP uses keepalives to monitor the link state, as described in the Keepalive Timer, on page 5.

PPP supports the following authentication protocols, which require a remote device to prove its identity before allowing data traffic to flow over a connection:

- Challenge Handshake Authentication Protocol (CHAP)—CHAP authentication sends a challenge message to the remote device. The remote device encrypts the challenge value with a shared secret and returns the encrypted value and its name to the local router in a response message. The local router attempts to match the remote device's name with an associated secret stored in the local username or remote security server database; it uses the stored secret to encrypt the original challenge and verify that the encrypted values match.
- Microsoft Challenge Handshake Authentication Protocol (MS-CHAP)—MS-CHAP is the Microsoft version of CHAP. Like the standard version of CHAP, MS-CHAP is used for PPP authentication; in this case, authentication occurs between a personal computer using Microsoft Windows NT or Microsoft Windows 95 and a Cisco router or access server acting as a network access server.
- Password Authentication Protocol (PAP)—PAP authentication requires the remote device to send a name and a password, which are checked against a matching entry in the local username database or in the remote security server database.

Use the **ppp authentication** command in interface configuration mode to enable CHAP, MS-CHAP, and PAP on a serial interface.

Note Enabling or disabling PPP authentication does not effect the local router's willingness to authenticate itself to the remote device.

Multilink PPP

Multilink Point-to-Point Protocol (MLPPP) is supported on the Cisco 819 ISR serial interface. MLPPP provides a method for combining multiple physical links into one logical link. The implementation of MLPPP combines multiple PPP serial interfaces into one multilink interface. MLPPP performs the fragmenting, reassembling, and sequencing of datagrams across multiple PPP links.

MLPPP provides the same features that are supported on PPP Serial interfaces with the exception of QoS. It also provides the following additional features:

- Fragment sizes of 128, 256, and 512 bytes
- Long sequence numbers (24-bit)
- Lost fragment detection timeout period of 80 ms
- Minimum-active-links configuration option
- LCP echo request/reply support over multilink interface
- Full T1 and E1 framed and unframed links

Keepalive Timer

Cisco keepalives are useful for monitoring the link state. Periodic keepalives are sent to and received from the peer at a frequency determined by the value of the keepalive timer. If an acceptable keepalive response is not received from the peer, the link makes the transition to the down state. As soon as an acceptable keepalive response is obtained from the peer or if keepalives are disabled, the link makes the transition to the up state.

Note The **keepalive** command applies to serial interfaces using HDLC or PPP encapsulation. It does not apply to serial interfaces using Frame Relay encapsulation.

For each encapsulation type, a certain number of keepalives ignored by a peer triggers the serial interface to transition to the down state. For HDLC encapsulation, three ignored keepalives causes the interface to be brought down. For PPP encapsulation, five ignored keepalives causes the interface to be brought down. ECHOREQ packets are sent out only when LCP negotiation is complete (for example, when LCP is open).

Use the **keepalive** command in interface configuration mode to set the frequency at which LCP sends ECHOREQ packets to its peer. To restore the system to the default keepalive interval of 10 seconds, use the **keepalive** command with the **no** keyword. To disable keepalives, use the **keepalive** disable command. For both PPP and Cisco HDLC, a keepalive of 0 disables keepalives and is reported in the **show running-config** command output as **keepalive disable**.

When LCP is running on the peer and receives an ECHOREQ packet, it responds with an ECHOREP packet, regardless of whether keepalives are enabled on the peer.

Keepalives are independent between the two peers. One peer end can have keepalives enabled; the other end can have them disabled. Even if keepalives are disabled locally, LCP still responds with ECHOREP packets to the ECHOREQ packets it receives. Similarly, LCP also works if the period of keepalives at each end is different.

Frame Relay Encapsulation

When Frame Relay encapsulation is enabled on a serial interface, the interface configuration is hierarchical and comprises the following elements:

- The serial main interface comprises the physical interface and port. If you are not using the serial interface to support Cisco HDLC and PPP encapsulated connections, then you must configure subinterfaces with permanent virtual circuits (PVCs) under the serial main interface. Frame Relay connections are supported on PVCs only.
- Serial subinterfaces are configured under the serial main interface. A serial subinterface does not actively carry traffic until you configure a PVC under the serial subinterface. Layer 3 configuration typically takes place on the subinterface.
- When the encapsulation on a serial interface is changed from HDLC to any other encapsulation type, the configured serial subinterfaces on the main interface inherit the newly changed encapsulation and they do not get deleted.
- Point-to-point PVCs are configured under a serial subinterface. You cannot configure a PVC directly under a main interface. A single point-to-point PVC is allowed per subinterface. PVCs use a predefined circuit path and fail if the path is interrupted. PVCs remain active until the circuit is removed from either configuration. Connections on the serial PVC support Frame Relay encapsulation only.

Note The administrative state of a parent interface drives the state of the subinterface and its PVC. When the administrative state of a parent interface or subinterface changes, so does the administrative state of any child PVC configured under that parent interface or subinterface.

To configure Frame Relay encapsulation on serial interfaces, use the **encapsulation (Frame Relay VC-bundle)** command.

Frame Relay interfaces support two types of encapsulated frames:

- Cisco (default)
- IETF

Use the **encap** command in PVC configuration mode to configure Cisco or IETF encapsulation on a PVC. If the encapsulation type is not configured explicitly for a PVC, then that PVC inherits the encapsulation type from the main serial interface.

Note Cisco encapsulation is required on serial main interfaces that are configured for MPLS. IETF encapsulation is not supported for MPLS.

Before you configure Frame Relay encapsulation on an interface, you must verify that all prior Layer 3 configuration is removed from that interface. For example, you must ensure that there is no IP address configured directly under the main interface; otherwise, any Frame Relay configuration done under the main interface will not be viable.

LMI on Frame Relay Interfaces

The Local Management Interface (LMI) protocol monitors the addition, deletion, and status of PVCs. LMI also verifies the integrity of the link that forms a Frame Relay UNI interface. By default, **cisco** LMI is enabled on all PVCs.

If the LMI type is **cisco** (the default LMI type), the maximum number of PVCs that can be supported under a single interface is related to the MTU size of the main interface. Use the following formula to calculate the maximum number of PVCs supported on a card or SPA:

(MTU - 13)/8 = maximum number of PVCs

Note The default setting of the **mtu** command for a serial interface is 1504 bytes. Therefore, the default numbers of PVCs supported on a serial interface configured with **cisco** LMI is 186.

Configuring Serial Interfaces

This section contains the following tasks:

Configuring a Synchronous Serial Interface

Synchronous serial interfaces are supported on various serial network interface cards or systems. This interface supports full-duplex operation at T1 (1.544 Mbps) and E1 (2.048 Mbps) speeds.

To configure a synchronous serial interface, perform the tasks in the following sections. Each task in the list is identified as either required or optional.

See the Examples for Interface Enablement Configuration, on page 20 for examples of configuration tasks described in this chapter.

Specifying a Synchronous Serial Interface

To specify a synchronous serial interface and enter interface configuration mode, use one of the following commands in global configuration mode.

Command	Purpose
`Router(config)#` **interface serial** *0*	Enters interface configuration mode.

Specifying Synchronous Serial Encapsulation

By default, synchronous serial lines use the High-Level Data Link Control (HDLC) serial encapsulation method, which provides the synchronous framing and error detection functions of HDLC without windowing or retransmission. The synchronous serial interfaces support the following serial encapsulation methods:

- HDLC
- Frame Relay
- PPP
- Synchronous Data Link Control (SDLC)
- SMDS
- Cisco Serial Tunnel (STUN)
- Cisco Bisync Serial Tunnel (BSTUN)
- X.25-based encapsulations

To define the encapsulation method, use the following command in interface configuration mode.

Command	Purpose
`Router(config-if)#` **encapsulation** {**hdlc** | **frame-relay** | **ppp** | **sdlc-primary** | **sdlc-secondary** | **smds** | **stun** | **x25** | **bstun**}	Configures synchronous serial encapsulation.

Note You cannot use the **physical-layer async** command for frame-relay encapsulation.

Encapsulation methods are set according to the type of protocol or application you configure in the Cisco IOS software.

- PPP is described in Configuring Media-Independent PPP and Multilink PPP.
- The remaining encapsulation methods are defined in their respective books and chapters describing the protocols or applications. Serial encapsulation methods are also discussed in the Cisco IOS Interface and Hardware Component Command Reference**encapsulation** command.

By default, synchronous interfaces operate in full-duplex mode. To configure an SDLC interface for half-duplex mode, use the following command in interface configuration mode.

Command	Purpose
`Router(config-if)#` **half-duplex**	Configures an SDLC interface for half-duplex mode.

Binary synchronous communication (Bisync) is a half-duplex protocol. Each block of transmission is acknowledged explicitly. To avoid the problem associated with simultaneous transmission, there is an implicit role of primary and secondary stations. The primary sends the last block again if there is no response from the secondary within the period of block receive timeout.

To configure the serial interface for full-duplex mode, use the following command in interface configuration mode.

Command	Purpose
`Router(config-if)#` **full-duplex**	Specifies that the interface can run Bisync using switched RTS signals.

Configuring PPP

To configure PPP, refer to the Configuring Media-Independent PPP and Multilink PPP.

Configuring Bisync

To configure the Bisync feature on the synchronous serial port adapters on Cisco 819 ISRs, refer to the Block Serial Tunneling (BSTUN) Overview. All commands listed in this section apply to the synchronous serial port adapters on Cisco 891 ISRs. Any command syntax that specifies an interface*number* supports the Cisco 891 ISRs **slot/port** syntax.

Configuring Compression of HDLC Data

You can configure point-to-point software compression on serial interfaces that use HDLC encapsulation. Compression reduces the size of a HDLC frame via lossless data compression. The compression algorithm used is a Stacker (LZS) algorithm.

Compression is performed in software and might significantly affect system performance. We recommend that you disable compression if CPU load exceeds 65 percent. To display the CPU load, use the **show process cpu** EXEC command.

If the majority of your traffic is already compressed files, you should not use compression.

To configure compression over HDLC, use the following commands in interface configuration mode.

SUMMARY STEPS

1. **encapsulation hdlc**
2. **compress stac**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**encapsulation hdlc** **Example:** `Router(config-if)# encapsulation hdlc`	Enables encapsulation of a single protocol on the serial line.
Step 2	**compress stac** **Example:** `Router(config-if)# compress stac`	Enables compression.

Using the NRZI Line-Coding Format

The nonreturn-to-zero (NRZ) and nonreturn-to-zero inverted (NRZI) formats are supported on the Cisco 819 serial ports.

NRZ and NRZI are line-coding formats that are required for serial connections in some environments. NRZ encoding is most common. NRZI encoding is used primarily with EIA/TIA-232 connections in IBM environments.

The default configuration for all serial interfaces is NRZ format. The default is **no nrzi-encoding**.

To enable NRZI format, use one of the following commands in interface configuration mode.

SUMMARY STEPS

1. Do one of the following:
   - **nrzi-encoding**

DETAILED STEPS

	Command or Action	Purpose

Step 1	Do one of the following:
	• **nrzi-encoding**
	Example:
	`Router(config-if)# nrzi-encoding`
	`Router(config-if)# nrzi-encoding [mark]`
		Enables NRZI encoding format.
		Enables NRZI encoding format for router.

Enabling the Internal Clock

When a DTE does not return a transmit clock, use the following interface configuration command on the router to enable the internally generated clock on a serial interface:

SUMMARY STEPS

1. **transmit-clock-internal**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**transmit-clock-internal** **Example:** `Router(config-if)# transmit-clock-internal`	Enables the internally generated clock on a serial interface.

Inverting the Transmit Clock Signal

Systems that use long cables or cables that are not transmitting the TxC signal (transmit echoed clock line, also known as TXCE or SCTE clock) can experience high error rates when operating at the higher transmission speeds. For example, if the interface on the PA-8T and PA-4T+ synchronous serial port adapters is reporting a high number of error packets, a phase shift might be the problem. Inverting the clock signal can correct this shift. To invert the clock signal, use the following commands in interface configuration mode.

SUMMARY STEPS

1. **invert txclock**
2. **invert rxclock**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**invert txclock** **Example:** `Router(config-if)# invert txclock`	Inverts the clock signal on an interface.
Step 2	**invert rxclock** **Example:** `Router(config-if)# invert rxclock`	Inverts the phase of the RX clock on the UIO serial interface, which does not use the T1/E1 interface.

Setting Transmit Delay

It is possible to send back-to-back data packets over serial interfaces faster than some hosts can receive them. You can specify a minimum dead time after transmitting a packet to remove this condition. This setting is available for serial interfaces on the MCI and SCI interface cards and for the HSSI or MIP. Use one of the following commands, as appropriate for your system, in interface configuration mode.

Command	Purpose
`Router(config-if)#` **transmitter-delay** *microseconds*	Sets the transmit delay on the MCI and SCI synchronous serial interfaces.
`Router(config-if)#` **transmitter-delay** *hdlc-flags*	Sets the transmit delay on the HSSI or MIP.

Configuring DTR Signal Pulsing

You can configure pulsing Data Terminal Ready (DTR) signals on all serial interfaces. When the serial line protocol goes down (for example, because of loss of synchronization), the interface hardware is reset and the DTR signal is held inactive for at least the specified interval. This function is useful for handling encrypting or other similar devices that use the toggling of the DTR signal to reset synchronization. To configure DTR signal pulsing, use the following command in interface configuration mode.

Command	Purpose
`Router(config-if)#` **pulse-time** *seconds*	Configures DTR signal pulsing.

Ignoring DCD and Monitoring DSR as Line Up/Down Indicator

By default, when the serial interface is operating in DTE mode, it monitors the Data Carrier Detect (DCD) signal as the line up/down indicator. By default, the attached DCE device sends the DCD signal. When the DTE interface detects the DCD signal, it changes the state of the interface to up.

In some configurations, such as an SDLC multidrop environment, the DCE device sends the Data Set Ready (DSR) signal instead of the DCD signal, which prevents the interface from coming up. To tell the interface to monitor the DSR signal instead of the DCD signal as the line up/down indicator, use the following command in interface configuration mode.

SUMMARY STEPS

1. **ignore-dcd**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**ignore-dcd** **Example:** `Router(config-if)# ignore-dcd`	Configures the serial interface to monitor the DSR signal as the line up/down indicator.

What to Do Next

Caution Unless you know for certain that you really need this feature, be very careful using this command. It will hide the real status of the interface. The interface could actually be down and you will not know just by looking at show displays.

Specifying the Serial Network Interface Module Timing

On Cisco 819 series ISRs, you can specify the serial Network Interface Module timing signal configuration. When the board is operating as a DCE and the DTE provides terminal timing (SCTE or TT), you can configure the DCE to use SCTE from the DTE. When running the line at high speeds and long distances, this strategy prevents phase shifting of the data with respect to the clock.

To configure the DCE to use SCTE from the DTE, use the following command in interface configuration mode.

SUMMARY STEPS

1. **dce-terminal-timing enable**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**dce-terminal-timing enable** **Example:** `Router(config-if)# dce-terminal-timing enable`	Configures the DCE to use SCTE from the DTE.

Specifying the Serial Network Interface Module Timing

When the board is operating as a DTE, you can invert the TXC clock signal it gets from the DCE that the DTE uses to transmit data. Invert the clock signal if the DCE cannot receive SCTE from the DTE, the data is running at high speeds, and the transmission line is long. Again, this prevents phase shifting of the data with respect to the clock.

To configure the interface so that the router inverts the TXC clock signal, use the following command in interface configuration mode.

SUMMARY STEPS

1. **dte-invert-txc**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**dte-invert-txc** **Example:** `Router(config-if)# dte-invert-txc`	Specifies timing configuration to invert TXC clock signal.

Configuring Low-Speed Serial Interfaces

This section describes how to configure low-speed serial interfaces and contains the following sections:

For configuration examples, see the Examples for Low-Speed Serial Interface, on page 20.

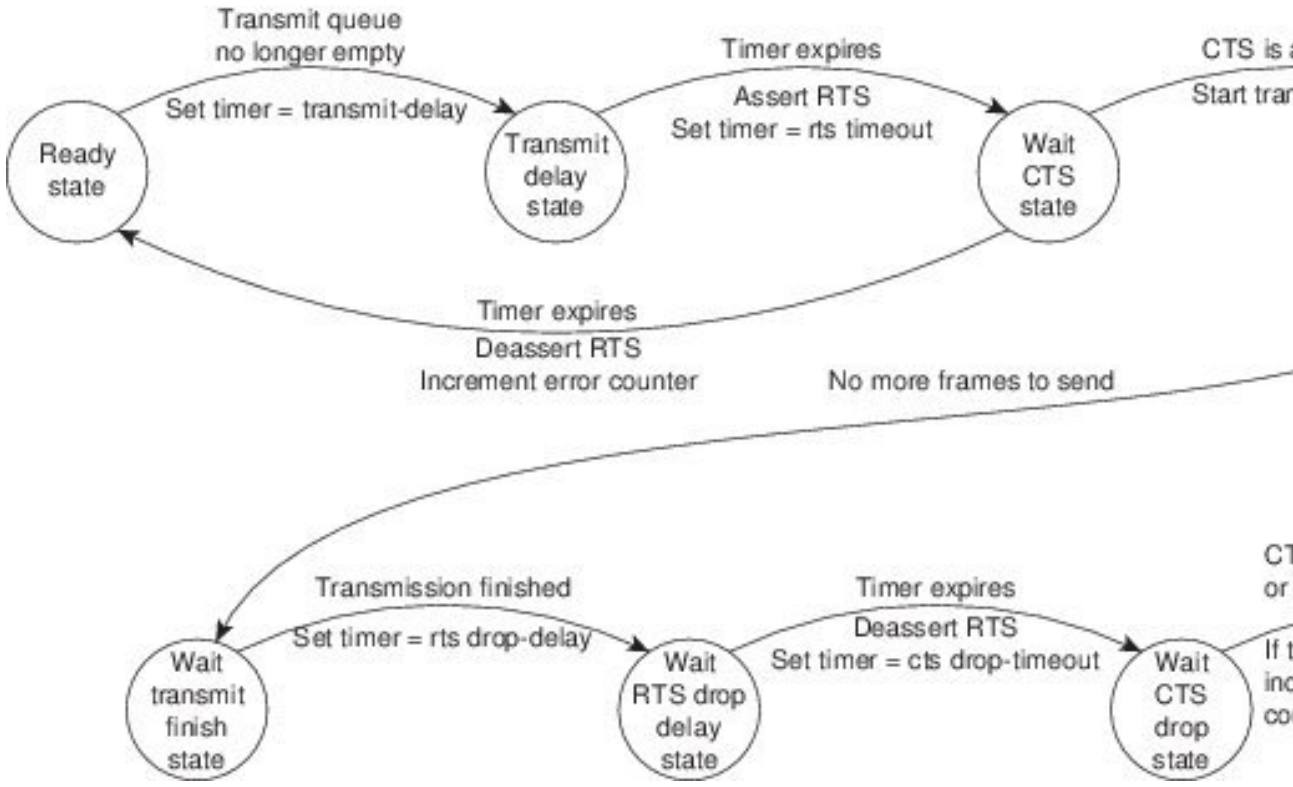
Half-Duplex DTE and DCE State Machines

The following sections describe the communication between half-duplex DTE transmit and receive state machines and half-duplex DCE transmit and receive state machines.

Half-Duplex DTE State Machines

As shown in the figure below, the half-duplex DTE transmit state machine for low-speed interfaces remains in the ready state when it is quiescent. When a frame is available for transmission, the state machine enters the transmit delay state and waits for a time period, which is defined by the **half-duplex timer transmit-delay** command. The default is 0 milliseconds. Transmission delays are used for debugging half-duplex links and assisting lower-speed receivers that cannot process back-to-back frames.

Figure 3: Half-Duplex DTE Transmit State Machine



After idling for a defined number of milliseconds (ms), the state machine asserts a request to send (RTS) signal and changes to the wait-clear-to-send (CTS) state for the DCE to assert CTS. A timeout timer with a value set by the **half-duplex timer rts-timeout** command starts. The default is 3 ms. If the timeout timer expires before CTS is asserted, the state machine returns to the ready state and deasserts RTS. If CTS is asserted before the timer expires, the state machine enters the transmit state and sends the frames.

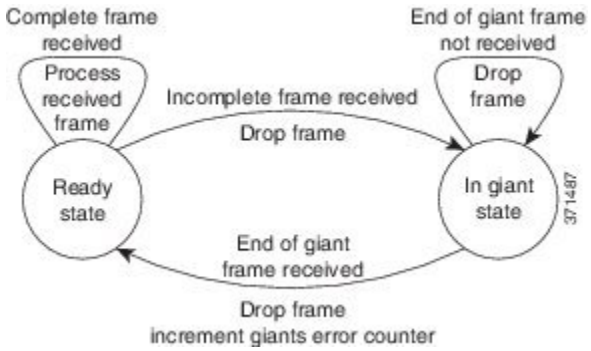
Once there are no more frames to transmit, the state machine transitions to the wait transmit finish state. The machine waits for the transmit FIFO in the serial controller to empty, starts a delay timer with a value defined by the **half-duplex timer rts-drop-delay** interface command, and transitions to the wait RTS drop delay state.

When the timer in the wait RTS drop delay state expires, the state machine deasserts RTS and transitions to the wait CTS drop state. A timeout timer with a value set by the **half-duplex timer cts-drop-timeout** interface command starts, and the state machine waits for the CTS to deassert. The default is 250 ms. Once the CTS

signal is deasserted or the timeout timer expires, the state machine transitions back to the ready state. If the timer expires before CTS is deasserted, an error counter is incremented, which can be displayed by issuing the **show controllers** command for the serial interface in question.

As shown in the figure below, a half-duplex DTE receive state machine for low-speed interfaces idles and receives frames in the ready state. A giant frame is any frame whose size exceeds the maximum transmission unit (MTU). If the beginning of a giant frame is received, the state machine transitions to the in giant state and discards frame fragments until it receives the end of the giant frame. At this point, the state machine transitions back to the ready state and waits for the next frame to arrive.

Figure 4: Half-Duplex DTE Receive State Machine



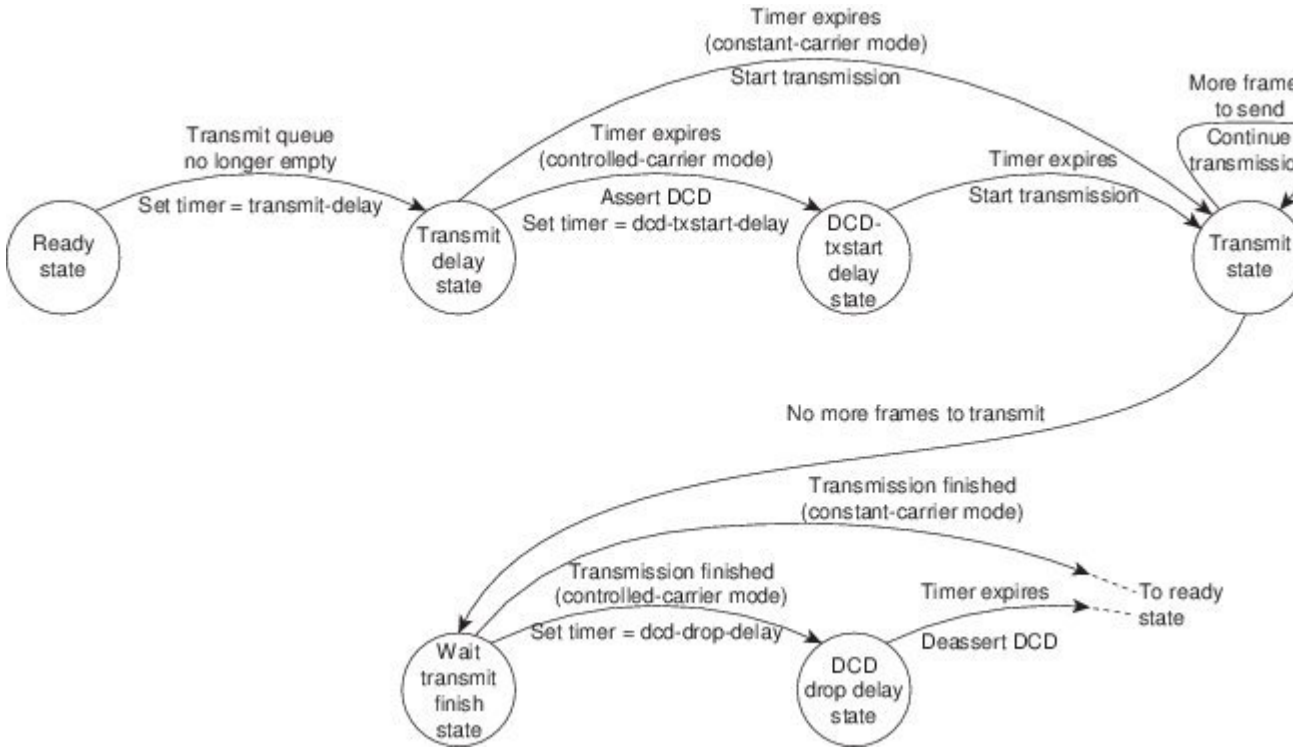
An error counter is incremented upon receipt of the giant frames. To view the error counter, use the **show interfaces** command for the serial interface in question.

Half-Duplex DCE State Machines

As shown in the figure below, for a low-speed serial interface in DCE mode, the half-duplex DCE transmit state machine idles in the ready state when it is quiescent. When a frame is available for transmission on the serial interface, such as when the output queues are no longer empty, the state machine starts a timer (based on the value of the **half-duplex timer transmit-delay** command, in milliseconds) and transitions to the transmit delay state. Similar to the DTE transmit state machine, the transmit delay state gives you the option of setting a delay between the transmission of frames; for example, this feature lets you compensate for a slow receiver that loses data when multiple frames are received in quick succession. The default **transmit-delay**

value is 0 ms; use the **half-duplex timer transmit-delay** interface configuration command to specify a delay value not equal to 0.

Figure 5: Half-Duplex DCE Transmit State Machine



After the transmit delay state, the next state depends on whether the interface is in constant-carrier mode (the default) or controlled-carrier mode.

If the interface is in constant-carrier mode, it passes through the following states:

1. The state machine passes to the transmit state when the **transmit-delay** timer expires. The state machine stays in the transmit state until there are no more frames to transmit.
2. When there are no more frames to transmit, the state machine passes to the wait transmit finish state, where it waits for the transmit FIFO to empty.
3. Once the FIFO empties, the DCE passes back to the ready state and waits for the next frame to appear in the output queue.

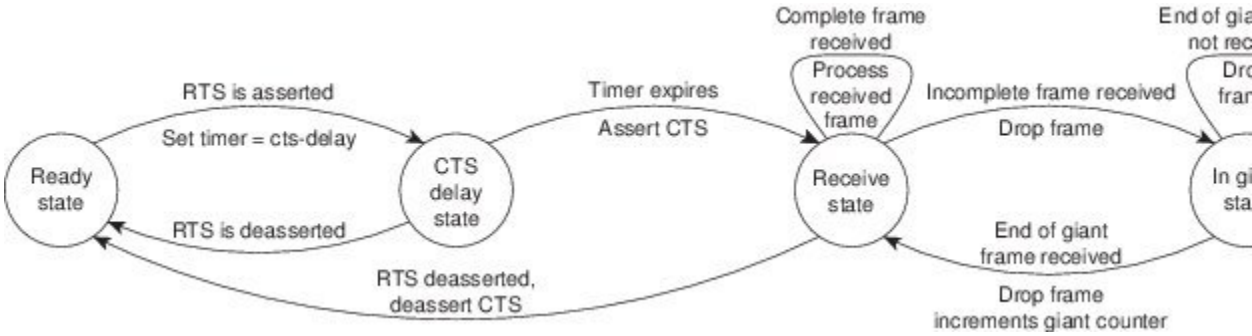
If the interface is in controlled-carrier mode, the interface performs a handshake using the data carrier detect (DCD) signal. In this mode, DCD is deasserted when the interface is idle and has nothing to transmit. The transmit state machine transitions through the states as follows:

1. After the **transmit-delay** timer expires, the DCE asserts DCD and transitions to the DCD-txstart delay state to ensure a time delay between the assertion of DCD and the start of transmission. A timer is started based on the value specified using the **dcd-txstart-delay** command. (This timer has a default value of 100 ms; use the **half-duplex timer dcd-txstart-delay** interface configuration command to specify a delay value.)
2. When this delay timer expires, the state machine transitions to the transmit state and transmits frames until there are no more frames to transmit.

3. After the DCE transmits the last frame, it transitions to the wait transmit finish state, where it waits for transmit FIFO to empty and the last frame to transmit to the wire. Then DCE starts a delay timer by specifying the value using the **dcd-drop-delay** command. (This timer has the default value of 100 ms; use the **half-duplex timer dcd-drop-delay** interface configuration command to specify a delay value.)
4. The DCE transitions to the wait DCD drop delay state. This state causes a time delay between the transmission of the last frame and the deassertion of DCD in the controlled-carrier mode for DCE transmits.
5. When the timer expires, the DCE deasserts DCD and transitions back to the ready state and stays there until there is a frame to transmit on that interface.

As shown in the figure below, the half-duplex DCE receive state machine idles in the ready state when it is quiescent. It transitions out of this state when the DTE asserts RTS. In response, the DCE starts a timer based on the value specified using the **cts-delay** command. This timer delays the assertion of CTS because some DTE interfaces expect this delay. (The default value of this timer is 0 ms; use the **half-duplex timer cts-delay** interface configuration command to specify a delay value.)

Figure 6: Half-Duplex DCE Receive State Machine



When the timer expires, the DCE state machine asserts CTS and transitions to the receive state. It stays in the receive state until there is a frame to receive. If the beginning of a giant frame is received, it transitions to the in giant state and keeps discarding all the fragments of the giant frame and transitions back to the receive state.

Transitions back to the ready state occur when RTS is deasserted by the DTE. The response of the DCE to the deassertion of RTS is to deassert CTS and go back to the ready state.

Placing a Low-Speed Serial Interface in Constant-Carrier Mode

To return a low-speed serial interface to constant-carrier mode from controlled-carrier mode, use the following command in interface configuration mode.

SUMMARY STEPS

1. **no half-duplex controlled-carrier**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**no half-duplex controlled-carrier** **Example:** `Router(config-if)# no half-duplex controlled-carrier`	Places a low-speed serial interface in constant-carrier mode.

Tuning Half-Duplex Timers

To optimize the performance of half-duplex timers, use the following command in interface configuration mode.

Command	Purpose
`Router(config-if)#` **half-duplex timer** {**cts-delay** *value* | **cts-drop-timeout** *value* | **dcd-drop-delay** *value* | **dcd-txstart-delay** *value* | **rts-drop-delay** *value* | **rts-timeout** *value* | **transmit-delay** *value* }	Tunes half-duplex timers.

The timer tuning commands permit you to adjust the timing of the half-duplex state machines to suit the particular needs of their half-duplex installation.

Note that the **half-duplex timer** command and its options replaces the following two timer tuning commands that are available only on high-speed serial interfaces:

- **sdlc cts-delay**
- **sdlc rts-timeout**

Changing Between Synchronous and Asynchronous Modes

To specify the mode of a low-speed serial interface as either synchronous or asynchronous, use the following command in interface configuration mode.

SUMMARY STEPS

1. **physical-layer {sync | async}**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**physical-layer {sync | async}** **Example:** `Router(config-if)# physical-layer sync`	Specifies the mode of a low-speed interface as either synchronous or asynchronous.

Changing Between Synchronous and Asynchronous Modes

This command applies only to low-speed serial interfaces available on Cisco 2520 through Cisco 2523 routers.

Note When you make a transition from asynchronous mode to synchronous mode in serial interfaces, the interface state becomes down by default. You should then use the **no shutdown** option to bring the interface up.

In synchronous mode, low-speed serial interfaces support all interface configuration commands available for high-speed serial interfaces, except the following two commands:

- **sdlc cts-delay**
- **sdlc rts-timeout**

When placed in asynchronous mode, low-speed serial interfaces support all commands available for standard asynchronous interfaces. The default is synchronous mode.

Note When you use this command, it does not appear in the output of the **show running-config** and **show startup-config** commands because the command is a physical-layer command.

To return to the default mode (synchronous) of a low-speed serial interface on a Cisco 2520 through Cisco 2523 router, use the following command in interface configuration mode.

SUMMARY STEPS

1. **no physical-layer**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**no physical-layer** **Example:** `Router(config-if)# no physical-layer`	Returns the interface to its default mode, which is synchronous.

Examples for Interface Enablement Configuration

The following example illustrates how to begin interface configuration on a serial interface. It assigns PPP encapsulation to serial interface 0.

```
interface serial 0
 encapsulation ppp
```

The same example on the router, assigning PPP encapsulation to port 0 in slot 1, requires the following commands:

```
interface serial 1/0
 encapsulation ppp
```

The following example shows how to configure the access server so that it will use the default address pool on all interfaces except interface 7, on which it will use an address pool called lass:

```
ip address-pool local
ip local-pool lass 172.30.0.1
 async interface
 interface 7
 peer default ip address lass
```

Examples for Low-Speed Serial Interface

The section includes the following configuration examples for low-speed serial interfaces:

Examples for Synchronous or Asynchronous Mode

The following example shows how to change a low-speed serial interface from synchronous to asynchronous mode:

```
interface serial 2
 physical-layer async
```

The following examples show how to change a low-speed serial interface from asynchronous mode back to its default synchronous mode:

```
interface serial 2
 physical-layer sync
```

or

```
interface serial 2
 no physical-layer
```

The following example shows some typical asynchronous interface configuration commands:

```
interface serial 2
 physical-layer async
 ip address 10.0.0.2 255.0.0.0
 async default ip address 10.0.0.1
 async mode dedicated
 async default routing
```

The following example shows some typical synchronous serial interface configuration commands available when the interface is in synchronous mode:

```
interface serial 2
 physical-layer sync
 ip address 10.0.0.2 255.0.0.0
 no keepalive
 ignore-dcd
 nrzi-encoding
 no shutdown
```

Example for Half-Duplex Timers

The following example shows how to set the cts-delay timer to 1234 ms and the transmit-delay timer to 50 ms:

```
interface serial 2
 half-duplex timer cts-delay 1234
 half-duplex timer transmit-delay 50
```