SR X.25 v1.3.18 User Manual

Revision: 2.9
Date: 26 February 2020

1 Introduction

SR X.25 is a software library in C (supplied with full source code) which implements ITU-T recommendation X.25 - Interface between Data Terminal Equipment (DTE) and Data Circuit-terminating Equipment (DCE) for terminals operating in the packet mode and connected to public data networks by dedicated circuit.

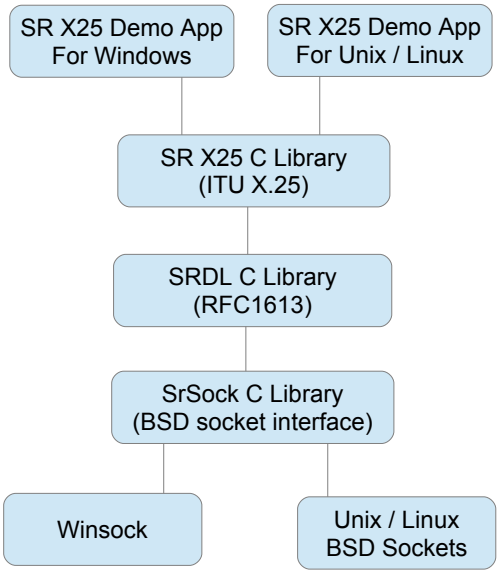
All source code components included in SR X.25 library support the following target operating systems:

- Windows (XP, Vista, 7, 8, 10 and server OSes)
- Linux, Solaris, HP-UX and other Unix like OSes

Individual source code components can also be ported to run in an embedded environment.

SR X.25 operates over XOT in two modes: RFC1613 Cisco Systems X.25 over TCP (XOT) or Proprietary XOT mode. XOT is part of SrDL sub-module included with the SR X.25 package.

Architecture and list of supported features:



SR X.25 internally at a lower layer interfaces to SrDL library. HsDL library is Data Link layer abstraction. When SR X.25 is ported into environment with LAPB or LAPD as data link layer, only HsDL module needs to change.

HsDL links directly to HsSock module (also included in this package) - Winsock interface component that provides reliable TCP transport services similar to LAPB / LAPD / HDLC.

SR X.25 as provided to customers may be used immediately in X.25 over TCP (XOT) solution or it may be used with traditional LAPB or LAPD in which case only HsDL will need to be modified.

When user application initialises SR X.25 library, it provides interface callbacks for the services used by SR X.25 protocol module: timer management, and event callbacks. The application then calls SR X.25 functions to establish virtual circuits, send and receive data, enforce flow control and clear calls.

DCE Operation	Yes
DTE Operation	Yes
RFC1613 XOT	Yes
Maximum Number of VCs	4095
Facilities Support	Yes
Call User Data supported	Yes
SVC Support	Yes
PVC Support	Yes, over RFC1613 XOT
X.25 Version	ITU-T (formerly CCITT) 10/96
Outgoing Calls	Yes
Incoming Calls	Yes
Incoming Call processing	Configurable between: Match on local DTE address or Accept All Calls
Packet Format	Basic format, modulo 8
Packet Size	Configurable in range 128,256,512,1024 with 128 default
Window Sizes	Configurable from 1 to 7 with default of 2
A Bit	Basic format only (non TOA/NPI) addresses supported
Q Bit procedure	Yes
M Bit procedure	Yes
D Bit procedure	Yes
Logical Channel Assignment	Configurable Outgoing and Incoming range
Timers and Counters Supported	T10 T11 T12 T13 T20 T20 Retry T21 T22 T23 T23 Retry
Interrupt packets	Yes
Trace Function Supported	Yes
Flow Control (RNR)	Yes
Per VC statistics	Yes

2 SR X.25 API

2.1 SrX25Init

extern int SrX25Init (sr_x25_init_t *init);		
D E S C R I P T I O N		
Initializes SR X.25 Library - this function must be called first before any other functions are called. Init structure contains function pointers which must be initialised with function addresses in user space. SR X.25 module will call these functions when it needs to manage timers or feed events back to user		
P A R A M E T E R S		
Type	Name	Description
sr_x25_init_t	*init	Pointer to initialisation structure: see next section for description
R E T U R N S		

- *SR_X25_RC_OK – success. SR X.25 initialised.*
- *SR_X25_RC_ALR_INIT – SR X.25 library is already initialized*
- *SR_X25_RC_INV_PAR – invalid parameters*
- *SR_X25_RC_INV_PKTSIZE – invalid packet size*
- *SR_X25_RC_INV_WSIZE – invalid window size*
- *SR_X25_RC_DL_INIT_FAIL – Data Link Layer initialisation failed*

2.1.1 Definition of `sr_x25_init_t` structure

Function pointer	Prototype and Description
sr_x25_api_t api	This structure contains X.25 API to user function pointers (these must be initialised by user to point to functions in user code): • sr_x25_start_timer_t *start_timer – X.25 calls this function to start a timer. ○ **Prototype:** long sr_x25_start_timer_t(long handle, unsigned long secs, sr_x25_timer_callback_t *callback, int tid); ○ **Parameters:** handle – SR X.25 module handle secs – number of seconds to timeout after callback – pointer to function in X.25 that the user code calls when this timer expires. It has prototype: *void sr_x25_timer_callback_t(long handle);* where handle is X.25 module handle tid – timer id ○ **Returns:** timer handle • sr_x25_stop_timer_t *stop_timer – X.25 calls this function to stop a timer. ○ **Prototype:** void sr_x25_stop_timer_t(long handle, int tid); ○ **Parameters:** handle – timer handle ○ tid – timer id • sr_x25_event_callback_t *event_cb – X.25 calls this function to feed events back to user application. This function and all events are fully specified in the following section (2.1.2 SR X.25 Event Callback) • logging_enabled – 1 = internal event logging is enabled; 0=internal event logging is disabled. Default setting is 0; Internal event reporting into application supplied event log can be enabled for debugging purposes • evlog_cb – if logging_enabled=1, this is a function pointer of event log function in user code.

	SR X.25 would call this function to log an event. ○ Prototype: void sr_x25_log_event_fn_t(unsigned char *str);
int opt	**X.25 protocol startup options:** SR_X25_OPT_NORMAL =0 – normal operation SR_X25_OPT_L2_LOOP =1 – operation in loopback mode between 2 local level 2 links.
sr_trace_cb_t *trc_fn;	**Trace function pointer in user code** If tracing_enabled is 1, this function shall be called by SR X.25 for every X.25 packet passing through X.25 stack to present raw X.25 packets to application for further decoding, logging and / or display. • Prototype: void sr_trace_cb_t(int is_rx, unsigned char *buf, int len, long arg); • Parameters: ○ Is_rx – indicates packet direction 1=received; 0=transmitted ○ Buf – raw X.25 packet data ○ Len - length of raw data ○ Arg – currently unused, reserved for future extension
int tracing_enabled;	**Enable or Disable trace function:** 1=enable; 0=disable
int is_rfc1613;	**Specifies if RFC1613 Cisco mode XOT is to be used at Data Link layer** 1 – Use RFC1613 Cisco Mode XOT 0 – Use proprietary mode XOT
int is_server;	if proprietary XOT is used (not RFC1613), then this parameter configures data link module (HsDl) to operate as a TCP client (is_server=0) or TCP server (is_server=1). if RFC1613 mode is used, this parameter is ignored
unsigned char *peer_ip;	If parameter is_rfc1613 is set, peer_ip is a null terminated string containing remote IP address of XOT peer where outgoing connections are made to. The IP address string is in a dotted format, for example “192.168.1.2”. SR X.25 can make calls over XOT using two methods. The default mode of operation is that all outgoing calls using RFC1613 XOT go to the same destination IP address, but using separate TCP sessions. In this mode parameter *peer_ip identifies remote IP address for all X.25 calls. The second method is with XOT routing table. If this mode of operation is enabled with a call to API function SrX25RouteTableEnable, parameter *peer_ip is ignored and XOT routes outgoing calls by looking up destination IP address by X.25 called NUA for each call. Each route consisting of destination IP address and remote NUA, must be added individually with function calls to SrX25RouteAdd. In proprietary XOT mode this parameter is unused
int is_dce;	**Specifies DCE / DTE role of SR X.25.** 1 - DCE 0 – DTE
int pktsize	**Maximum default size of data payload of X.25 DATA packets. Valid values:** 0 – use internal default (128), 128, 256, 512, 1024
int wsize;	**Default window size. Valid values:** 0 – use internal default (2), 1, 2 , 3, 4, 5, 6, 7.
int hi_incoming_lcn;	If RFC1613 XOT mode is used, this is highest logical channel number (LCN) SR X.25 will use for incoming calls. Valid values from 1 to 4095
int lo_incoming_lcn;	If RFC1613 XOT mode is used, this is lowest logical channel number (LCN) SR X.25 will use for incoming calls. Valid values from 1 to 4095
int support_dbit;	0 – D bit procedure not supported, 1 – D bit procedure supported
void *hwnd	Handle of main application window which receives all Windows messages.
char local_ip[16];	local IP address to use (Applies to XOT) If local_ip[0] == 0, default local IP address is used Cisco mode XOT endpoint, (RFC1613) will bind to port 1998 and this IP address,

	The IP address is specified as a dotted IP ascii null terminated string, such as “192.168.1.1”
int tcp_ka_enabled	1=Enable TCP keepalives, 2=Disable TCP keepalives (supported on Linux only in current version)
int tcp_ka_interval	TCP keep alive probe send interval in seconds (supported on Linux only in current version)
int tcp_ka_timeout	TCP keep alive timeout in seconds - time to wait for response to probe (supported on Linux only in current version)
int tcp_ka_count	Maximum number of TCP keep alive probes before connection closed (supported on Linux only in current version)

2.1.2 SR X.25 Event Callback

This function is called by SR X.25 to pass events about the state of calls virtual circuits, states of links, transmission and reception of data and other events.

2.1.2.1 Prototype

```
/*
 * SR X.25 event callback function
 */
typedef void sr_x25_event_callback_t
(
        // these 2 parameters below (l2addr, port) identify X.25 link
        long                    l2addr, // in case data link XOT, IP address
        unsigned short int      port,   //in case data link XOT, IP port

        unsigned short int      lci,    // VC identifier (channel number 0 to 4095)
        int                     ev,     // event id
        long                    arg1,   // first argument
        long                    arg2    // second argument
);
```

2.1.2.2 Parameters

Parameter	Description
long l2addr	32 bit integer identifying data link, together with port. This is set to IP address. Data links are uniquely identified by IP address and port
unsigned short int port	16 bit integer identifying datalink together with l2addr. This is set to IP port
unsigned short int lci	Logical channel identifier 0 to 4095
int ev	Event id. All events are specified in section the next section
long arg1	Event argument 1
long arg2	Event argument 2

2.1.2.3 Events

Event	Arg1 and Arg2	Remarks
SR_X25_EV_CONN_FAIL_PKTLR	Arg1=0; arg2=0	X.25 call failed, packet layer not ready
SR_X25_EV_PKTLR_DOWN	Arg1=0; arg2=0	VC cleared: Packet layer down as result of physical layer down. This event is generated for every connected VC on the affected X.25 link
SR_X25_EV_PKTLR_RESTARTED	Arg1=0; arg2=0	VC cleared: Packet layer restarted (result of X.25 RESTART packet received from the line). This event is generated for every connected VC on the affected X.25 link
SR_X25_EV_CALL_CONNECTED	Arg1=0; arg2 = points to conn structure (sr_x25_conn_t) which holds X.25 called and calling addresses	X.25 SVC connected.

SR_X25_EV_CALL_CLEAR_COMPLETE	Arg1=0; arg2=0	X.25 SVC clearing completed
SR_X25_EV_CALL_CLEARED	Arg1=0; arg2=0	X.25 SVC cleared locally
SR_X25_EV_DATA	Arg1= pointer to sr_x25data_t structure, which is defined as follows: **unsigned char *bufp** – pointer to data buffer received **int len** – length of data buffer arg2=0	X.25 data sequence received. X.25 data sequence does not always correspond to a single X.25 DATA packet. If data transmitted by remote end is larger that used packet size the data is sent by transmitter in several data packets with M bit set in all packets but last. SR X.25 at the receive end will re-assemble complete data sequence before passing it to application with this event.
SR_X25_EV_DATA_EXP	arg1 – Pointer to start of data, arg2 – Integer length of data	X.25 Expedited data received
SR_X25_EV_EXP_DATA_TX_FAIL	Arg1=0; arg2=0	Cannot send expedited data, previous exp data not acknowledged
SR_X25_EV_EXP_DATA_DELIVERED	Arg1=0; arg2=0	X.25 Expedited data received
SR_X25_EV_CALL_CLEARED_T13	Arg1=0; arg2=0	X.25 SVC cleared - no answer to call indication
SR_X25_EV_BUF_TX_FAIL	Arg1=buffer pointer Arg2=0	buffer transmission failed
SR_X25_EV_INCOMING_CALL_CONNECTED	Arg1=0; arg2 = points to conn structure (sr_x25_conn_t) which holds X.25 called and calling addresses	incoming call connected (SrX25Listen completed) sr_x25_conn_t definition: called – called X.25 address of the incoming call calling – calling X.25 address of the incoming call int dbit – D bit state of the incoming call 0- D bit clear, 1- D bit set
SR_X25_EV_LINK_RELEASED	Arg1=0; arg2=0	X.25 link released. This event is generated when data link has released (meaning that either client or server TCP session has failed) In proprietary XOT mode, this event can be used by application to initiate data link re-establishment.
SR_X25_EV_LINK_ESTABLISHED	Arg1=0; arg2=0	X.25 link established This event is generated when data link has connected and is ready to carry X.25 VCs
SR_X25_EV_PVC_CONNECTED	Arg1=0; arg2=pointer to PVC setup structure - sr_x25_pvc_setup_t	PVC Setup complete
SR_X25_EV_PVC_SETUP_FAILED	Arg1=0; arg2=0	PVC Setup failed
SR_X25_EV_PVC_DISCONNECTED	Arg1=0; arg2=0	PVC disconnected

2.2 SrX25Connect

int SrX25Connect (sr_x25_conn_t *conn, int *lci);

DESCRIPTION

This function is called to initiate an outgoing X.25 SVC establishment

PARAMETERS

Type	Name	Description
sr_x25_conn_t	*conn	Pointer to connect structure as follows:

Data members	Description
long l2_addr;	4 octet level 2 address. RFC1613 XOT mode: - Set to unique identifier, for example 0xffff. Listens must also be submits on the same link identifier Proprietary XOT mode: - remote IP address
long arg;	RFC1613 XOT mode: - Not used Proprietary XOT mode: - Set to remote TCP port number
unsigned char called[SR_MAX_NUA+1];	Called X.25 DTE address (null terminated)
unsigned char calling[SR_MAX_NUA+1];	Calling X.25 DTE address (null terminated)
unsigned char facility_buf[SR_MAX_FAC];	Optional X.25 facilities buffer to be included with the call request. Facilities must be encoded by application according to X.25 standard.
unsigned char facility_len;	Size of facility buffer
unsigned char call_user_date[SR_MAX_CUD];	Call user data to be included with the call
int cud_len;	Length of call user data
int dbit;	D Bit setting for the call

Example setting level 2 address and argument for Proprietary XOT mode:

```
conn.l2_addr  = (long)inet_addr("10.1.73.164");
conn.arg      = 1261;
```

unsigned short int	*lci	Pointer to integer variable to receive selected logical channel number for outgoing virtual circuit

RETURNS

- *SR_X25_RC_OK – success. Outgoing Call initiated. The user application is notified of call completion or call failure asynchronously via event callback depending on call procedure result*
- *SR_X25_RC_INV_PAR – invalid parameters*
- *SR_X25_RC_TOO_LONG – the sum of X.25 addresses, user facilities and call user data exceeds global maximum X.25 packet size*
- *SR_X25_RC_NOLINK - could not allocate new Layer 2 link*
- *SR_X25_RC_NO_VC - no free VC available*
- *SR_X25_RC_NOT_INIT – SR X.25 Library not initialized*

2.3 SrX25Listen

int SrX25Listen(sr_x25_listen_t *listen)

DESCRIPTION

This function is called to listen for an incoming X.25 call

PARAMETERS

Type	Name	Description
sr_x25_listen_t	*listen	Pointer to listen structure as follows:

Long l2_addr	RFC1613 XOT mode: Set to unique link identifier, for example 0xffff. Outgoing calls must also be done on this link identifier Proprietary XOT mode: set to local IP address which remote end will be connecting to. If layer 2 address not found, a new layer 2 link will be allocated.
Long arg	RFC1613 XOT mode: Not used Proprietary XOT mode: Set to local TCP port to listen on
int take_any_call	1- accept all incoming calls, regardless of X.25 called address, 0 – check X.25 called and calling addresses of incoming call against calling and called parameters of sr_x25_listen_t structure
unsigned char called[SR_MAX_NUA];	When processing incoming call, compare the called X.25 address of the incoming call packet with this NUA.
int called_len;	Number of called address digits
unsigned char calling[SR_MAX_NUA];	When processing incoming call, compare the calling X.25 address of the incoming call packet with this NUA.
int calling_len;	Number of calling address digits.

RETURNS

- *SR_X25_RC_OK- success. Listen request accepted. Incoming call establishment is notified asynchronously via event callback*
- *SR_X25_RC_INV_PAR – invalid parameters*
- *SR_X25_RC_DL_LIS_FAIL – Listen failed due to Data Link layer error*
- *SR_X25_RC_DL_LIS_QFULL– Listen queue is full*
- *SR_X25_RC_NOLINK – no free link available*
- *SR_X25_RC_NOT_INIT – SR X.25 Library not initialize*

2.4 SrX25Clear

int SrX25Clear(long l2_addr, unsigned short int port, unsigned short int lci)		
D E S C R I P T I O N		
This function is called to Clear an X.25 SVC		
P A R A M E T E R S		
Type	Name	Description
long	l2_addr	IP address of X.25 link (4 octets)
unsigned short int	port	IP port of X.25 link (2 octets)
unsigned short int	lci	Logical channel number of the VC to clear
R E T U R N S		

- *SR_X25_RC_OK – success. Clearing procedure initiated*
- *SR_X25_RC_NOT_INIT - X.25 Library not initialized*
- *SR_X25_RC_NOLINK – X.25 link not found*
- *SR_X25_RC_NO_VC – X.25 VC is not allocated*
- *SR_X25_RC_INV_PAR – LCI is out of range*

2.5 SrX25Data

int SrX25Data(long l2_addr, unsigned short int port, unsigned short int lci, unsigned char *buf, int len, int qbit);		
D E S C R I P T I O N		
This function is called to send data over established X.25 VC		
P A R A M E T E R S		
Type	Name	Description
long	l2_addr	IP address of X.25 link (4 octets)
unsigned short int	port	IP port of X.25 link (2 octets)
unsigned short int	lci	Logical channel number of the VC
unsigned char	*buf	Pointer to data buffer to send
int	len	Length of data to send
int	qbit	Q Bit setting to use for the DATA packets carrying the specified data buffer.
R E T U R N S		

- *SR_X25_RC_OK – success, data queued by SR X.25 for sending over specified VC*
- *SR_X25_INV_PAR – invalid parameters*
- *SR_X25_BUF_TOO_BIG – buffer is too big*
- *SR_X25_TXQ_FULL – transmit queue is full, try again later.*
- *SR_X25_RC_NOT_INIT - X.25 Library not initialized*
- *SR_X25_RC_NOLINK – X.25 link not found*
- *SR_X25_RC_NO_VC – X.25 VC is not allocated*

2.6 SrX25DataExp

```
extern int SrX25DataExp(
        long l2_addr,                   // layer 2 address (IP address)
        unsigned short int port,        // IP port
        unsigned short int lci,         // VC channel number
        unsigned char *buf, int len);   // buffer to send, length to send
```

DESCRIPTION

This function is called to send expedited data over established X.25 VC. This function sends X.25 interrupt packet. An interrupt packet bypasses normal flow control mechanisms. Data is sent regardless of the state of the transmit window and no protocol variables are updated.

PARAMETERS

Type	Name	Description
long	l2_addr	IP address of X.25 link (4 octets)
unsigned short int	port	IP port of X.25 link (2 octets)
unsigned short int	lci	Logical channel number of the VC
unsigned char	*buf	Pointer to data buffer. The user must leave a space at the start of the buffer for X.25 header (recommended 10 bytes), example call to SrX25Data: #define HDR_SPACE 10 static unsigned char testbuf2[HDR_SPACE + 8]; SrX25DataExp(x25vc_han, &testbuf2[HDR_SPACE], 8);
int	len	Length of data to send

RETURNS

- *SR_X25_RC_OK – success. Expedited data request accepted.*
- *SR_X25_RC_NOT_INIT - X.25 Library not initialized*
- *SR_X25_RC_NOLINK – X.25 link not found*
- *SR_X25_RC_NO_VC – X.25 VC is not allocated*
- *SR_X25_INV_PAR – invalid parameters*

2.7 SrX25Tick

```
int SrX25Tick(void)
```

DESCRIPTION

This function must be called by user code as often as possible (from main loop or based on a fast timer). This function drives internal SR X.25 timers and services transmit queue.

PARAMETERS

Type	Name	Description
		No parameters

RETURNS

- *SR_X25_RC_OK – success*
- *SR_X25_RC_NOT_INIT - X.25 Library not initialized*

2.8 SrX25StutDown

void SrX25StutDown(sr_x25_shutdown_callback_fn_t *fn)		
D E S C R I P T I O N		
This function is used to close all X.25 links and release all resources used by SR X.25		
P A R A M E T E R S		
Type	Name	Description
sr_x25_shutdown_callback_fn_t void sr_x25_shutdown_callback_fn_t(void);	*fn	Function callback for SR X.25 to call for every cleared VC during shutdown procedure
R E T U R N S		
• *None*		

2.9 SrX25GetStats

void SrX25GetStats(long l2_addr, unsigned short int port, unsigned short int lci, sr_x25_stats_t *stats)		
D E S C R I P T I O N		
This function is used to get X.25 VC statistics		
P A R A M E T E R S		
Type	Name	Description
long	l2_addr	IP address of X.25 link (4 octets)
unsigned short int	port	IP port of X.25 link (2 octets)
unsigned short int	lci	Logical channel number of the VC
sr_x25_stats_t	*stats	Pointer to stats structure where X.25 will copy statistics information sr_x25_stats_t format: Int lci — Logical channel number tx_rx_t tx — Transmit statistics tx_rx_t rx — Receive statistics unsigned char *st — Name of current state int w — Current Window Tx_rx_t definition: long data – DATA packet count long rr; - RR packet count long rnr – RNR packet count long reset – RESET packet count long intr – INTERRUPT packet count
R E T U R N S		
• *None*		

2.10 SrX25GetLinkStats

extern int SrX25GetLinkStats(link_stats_t *pStats);

DESCRIPTION

This function is used to get X.25 Link statistics

PARAMETERS

Type	Name	Description
link_stats_t	*pStats	Pointer to stats structure where X.25 will copy statistics information link_stats_t format:

long l2addr	Link IP address
unsigned short int port	Link IP port
int num_listens	Number of queued listen requests
long dl_handle	Lower layer handle
int dl_state	Data link connection state
char pkt_lyr_state_str[40]	Packet layer state name (foe example “R0”, “R1” etc)
int num_alloc_vcs	Number of allocated VCs

RETURNS

- *SR_X25_RC_OK – success*
- *SR_X25_RC_INV_PAR – invalid parameters specified*
- *SR_X25_RC_NOT_INIT - X.25 Library not initialized*
- *SR_X25_RC_NOLINK – X.25 link not found*

2.11 SrX25Rnr

extern int SrX25Rnr(long l2_addr, unsigned short int port, unsigned short int lci, int on)

DESCRIPTION

This function is used set flow control condition which shall result in X.25 stack sending RNR or RR. Flow control condition can be set to recover temporarily low receive buffer resources.

PARAMETERS

Type	Name	Description
long	l2_addr	IP address of X.25 link (4 octets)
unsigned short int	port	IP port of X.25 link (2 octets)
unsigned short int	lci	Logical channel number of the VC
int	on	1= set flow control condition 0= clear flow control condition

RETURNS

- *SR_X25_RC_OK – success*
- *SR_X25_RC_INV_PAR – invalid parameters specified*
- *SR_X25_RC_NOT_INIT - X.25 Library not initialized*
- *SR_X25_RC_NOLINK – X.25 link not found*
- *SR_X25_RC_NO_VC – X.25 VC is not allocated*

2.12 SrX25Reset

int SrX25Reset(long l2_addr, unsigned short port, unsigned short lci)		
D E S C R I P T I O N		
This function is called to Reset an X.25 VC. If VC is in the connected data state (state P4), then X.25 reset procedure will be performed on the VC: X.25RESET request sent and X.25 RESET confirmation received. At that time X.25 VC flow control variables, such as next send sequence number, next expected receive sequence number and send window are reset at both sides of the VC.		
P A R A M E T E R S		
Type	Name	Description
long	l2_addr	IP address of X.25 link (4 octets)
unsigned short	port	IP port of X.25 link (2 octets)
unsigned short	lci	Logical channel number of the VC to clear
R E T U R N S		
• *SR_X25_RC_OK – success. Clearing procedure initiated* • *SR_X25_RC_NOT_INIT - X.25 Library not initialized* • *SR_X25_RC_NOLINK – X.25 link not found* • *SR_X25_RC_NO_VC – X.25 VC is not allocated* • *SR_X25_RC_INV_PAR – LCI is out of range*		

2.13 SrX25DecodePkt

int SrX25DecodePkt(unsigned char *pkt, int len, int *ev, sr_x25_pkt_info_t *info, int need_name);		
D E S C R I P T I O N		
This function may be used for debugging or diagnostics to decode the content of an X.25 packet		
P A R A M E T E R S		
Type	Name	Description
unsigned char	*pkt	Pointer to X.25 packet
int	len	Length of X.25 packet
int	*ev	Pointer to integer that will receive decoded PTI (packet type identifier), one of the following: SR_X25_CALL_ACCEPTED – Call accept SR_X25_CALL_REQ – Call request SR_X25_RESTART_REQ, – Restart request SR_X25_RESTART_CNF – Restart confirm SR_X25_RR – Receiver ready SR_X25_RNR – Receiver not ready SR_X25_DAT – DATA SR_X25_CLEAR_CNF – Clear confirm SR_X25_CLEAR_REQ – Clear request SR_X25_INT_CNF – Interrupt confirm SR_X25_INT_REQ – Interrupt request SR_X25_RESET_REQ – Reset request SR_X25_RESET_CNF – Reset confirm SR_X25_PVCSETUP – PVC Setup SR_X25_PVCKILL – PVC Kill

<table>
<tr><td>sr_x25_pkt_info_t</td><td>*info</td><td>Pointer to structure in user code that receives decoded packet information.

Structure definition:<table><tr><td>Int lci</td><td>Logical channel identifier</td></tr><tr><td>Int pti</td><td>Packet type identifier</td></tr><tr><td>unsigned char *pkt</td><td>Points to start of packet</td></tr><tr><td>Int len</td><td>Packet length</td></tr><tr><td>Int Pr</td><td>PR value</td></tr><tr><td>Int Ps</td><td>PS value</td></tr><tr><td>unsigned char *pktname</td><td>Name of packet</td></tr></table></td></tr>
<tr><td>int</td><td>need_name</td><td>1= fill in packet name
0= do not fill in packet name</td></tr>
<tr><td colspan="3">RETURNS</td></tr>
<tr><td colspan="3">SR_X25_RC_DECODE_SHORT – invalid packet – too short
SR_X25_RC_DECODE_LONG – invalid packet – too long
SR_X25_RC_DECODE_LCI_UNASSIGN – packet on unassigned logical channel
SR_X25_RC_DECODE_UNSPEC – decoded error, unspecified
SR_X25_RC_DECODE_INV_GFI – invalid general format identifier
SR_X25_RC_DECODE_OK – packet is good and is correctly decoded</td></tr>
</table>

2.14 SrX25GetCallList

<table>
<tr><td colspan="3">extern void SrX25GetCallList(call_list_fn_t *fn);</td></tr>
<tr><td colspan="3">DESCRIPTION</td></tr>
<tr><td colspan="3">This function is used to retrieve a list of connected VCs.</td></tr>
<tr><td colspan="3">PARAMETERS</td></tr>
<tr><td>Type</td><td>Name</td><td>Description</td></tr>
<tr><td>call_list_fn_t</td><td>*fn</td><td>Function pointer SR X.25 shall call for every connected VC

Definition:
void call_list_fn_t(unsigned short int lci, sr_x25_conn_t *pConn);

lci – channel number of the VC
pConn points to sr_x25_conn_t structure and contains X.25 called and calling DTE addresses and link identification information (IP address and port number)</td></tr>
<tr><td colspan="3">RETURNS</td></tr>
<tr><td colspan="3">none</td></tr>
</table>

2.15 SrX25EnableTracing

extern void SrX25EnableTracing(int enable);		
D E S C R I P T I O N		
This function is used to enable or disable tracing function		
P A R A M E T E R S		
Type	Name	Description
int	enable	1=enable tracing 0=disable tracing
R E T U R N S		
• *none*		

2.16 XOT Routing Table for Outgoing Calls

SR X.25 can make calls over XOT using to methods. The default mode of operation is that all outgoing calls using RFC1613 XOT go to the same destination IP address, but using separate TCP sessions. In this mode parameter *rfc1613_peer_ip in the X.25 stack initialization structure sr_x25_init_t identifies remote IP address for all X.25 calls.

The second method is with XOT routing table. If this mode of operation is enabled with a call to API function SrX25RouteTableEnable, parameter *rfc1613_peer_ip is ignored and XOT routes outgoing calls by looking up destination IP address by X.25 called NUA for each call. Each route consisting of destination IP address and remote NUA, must be added individually with function calls to SrX25RouteAdd.

Routes can be added, modified and removed at any after SR X.25 Library was initialized with a call to SrX25Init.

2.16.1 SrX25RouteTableEnable

void SrX25RouteTableEnable(int enable)		
DESCRIPTION		
*This function is used to enable XOT routing table for making outgoing calls. If XOT routing table is enabled parameter *rfc1613_peer_ip is ignored and XOT routes outgoing calls by looking up destination IP address by X.25 called NUA for each call. Each route consisting of destination IP address and remote NUA, must be added individually with function calls to SrX25RouteAdd*		
PARAMETERS		
Type	Name	Description
int	enable	1=enable; 0=disable
RETURNS		
• *None*		

2.16.1 SrX25RouteAdd

int SrX25RouteAdd(unsigned char *remote_nua, unsigned char *remote_ip, int *route_index);		
DESCRIPTION		
This function is used to add an outgoing XOT route to XOT routing table. A route consists of destination X.25 NUA and remote IP address. This function may be called at any time. Maximum supported number of routes is 4095 in the current version of SR X25 Library.		
PARAMETERS		
Type	Name	Description
unsigned char	*remote_nua	Pointer to null terminated string – X.25 called DTE address, also called remote NUA. Maximum length of string not including terminating zero is 15 digits.
unsigned char	*remote_ip	Pointer to null terminated string – remote IP address where the XOT will connect TCP session to before sending X.25 call request. Remote IP address is in dotted IP format. “n.n.n.n”. Maximum length of string not including zero terminating character is 15 bytes.
int	*route_index	Pointer to integer variable that will receive route index on completion of the function. This route index can then by used to modify or delete the route using functions SrX25RouteSet and SrX25RouteDelete
RETURNS		
• *SR_X25_RC_FAIL – adding route failed* • *SR_X25_RC_OK – success, route added*		

2.16.2 SrX25RouteSet

int SrX25RouteAdd(int route_index, unsigned char *remote_nua, unsigned char *remote_ip);		
D E S C R I P T I O N		
This function is used to modify an existing XOT route of XOT routing table.		
P A R A M E T E R S		
Type	Name	Description
int	route_index	Index of route to modify. Index is obtained with a call to SrX25RouteAdd
unsigned char	*remote_nua	Pointer to null terminated string – X.25 called DTE address, also called remote NUA. Maximum length of string not including terminating zero is 15 digits.
unsigned char	*remote_ip	Pointer to null terminated string – remote IP address where the XOT will connect TCP session to before sending X.25 call request. Remote IP address is in dotted IP format. “n.n.n.n”. Maximum length of string not including zero terminating character is 15 bytes.
R E T U R N S		
• *SR_X25_RC_FAIL – changing the route failed* • *SR_X25_RC_OK – success, the route changed*		

2.16.1 SrX25RouteDelete

int SrX25RouteDelete(int route_index);		
D E S C R I P T I O N		
This function is used to delete an existing XOT route from XOT routing table.		
P A R A M E T E R S		
Type	Name	Description
int	route_index	Index of route to delete. This index is obtained with a call to SrX25RouteAdd
R E T U R N S		
• *SR_X25_RC_FAIL – deleting the route failed* • *SR_X25_RC_OK – success, route deleted*		

3 Support for Permanent Virtual Circuits (PVC)

In the current version of SR X.25 PVCs are supported over XOT RFC1613 only

3.1.1 SrX25CreateLink

int SrX25CreateLink (sr_x25_link_cfg_t *pLinkCfg);

DESCRIPTION

This function creates an X.25 link and configures it.
This is an optional function which may be called before calling SrX25Connect or SrX25Listen.
This function must be called if the user intends to use PVC functionality of SR X.25 library

PARAMETERS

Type	Name	Description

sr_x25_link_cfg_t	pLinkCfg	Pointer to link configuration structure, defined as follows:
		
		// link configuration
		typedef struct
		{
		long l2_addr; // unique link identifier
		unsigned short int arg;
		int num_pvcs; // number of VCs reserved for PVCs
		int pvc_base_lcn; // 1st LCN number reserved for PVCs
		} sr_x25_link_cfg_t;
		
		l2addr – link identifier, use RFC1613_LINK_ADDR constant
		arg – second part of link identifier, use RFC1613_LINK_PORT
		
		num_pvcs – number of VCs to be allocated for PVCs
		pvc_base_lcn – LCN of first PVC.

RETURNS

- *SR_X25_RC_NOT_INIT – SR X.25 library not initialized*
- *SR_X25_RC_INV_PAR – invalid parameters*
- *SR_X25_RC_LINK_EXISTS – link already exits (lookup is done on link identifier)*
- *SR_X25_RC_NOLINK – Failed to allocated new link*
- *SR_X25_RC_NO_VC – failed to allocate PVC*
- *SR_X25_RC_OK – success*

3.1.1 SrX25ConnectPvc

int SrX25ConnectPvc (sr_x25_pvc_setup_t *conn);		
DESCRIPTION		
This function initiates PVC setup over XOT RFC1616 *The remote side must be listening for PVC setup*		
PARAMETERS		
Type	Name	Description
sr_x25_pvc_setup_t	conn	Pointer to PVC setup configuration structure, defined as follows: l2_addr - link identifier, use RFC1613_LINK_ADDR constant arg – second part of link identifier, use RFC1613_LINK_PORT local_lcn – local VC logical channel number remote_lcn - remote VC logical channel number local_name - local interface name (null terminated string, max 64 bytes) remote_name - remote interface name (null terminated string, max 64 bytes) int local_incoming_window - local incoming window size valid values are 1 to 7 int local_outgoing_window - local outgoing window size valid values are 1 to 7 int local_incoming_max_pktsize - local incoming max packet size Values from 4 to 10, see note below int local_outgoing_max_pktsize - local outgoing max packet size Values from 4 to 10, see note below The values for the packet sizes are encoded as follows: 4: 16 bytes 5: 32 bytes 6: 64 bytes 7: 128 bytes 8: 256 bytes 9: 512 bytes 10: 1024 bytes
RETURNS		
• *SR_X25_RC_NOT_INIT – SR X.25 library not initialized* • *SR_X25_RC_INV_PAR – invalid parameters* • *SR_X25_RC_NOLINK – Failed to allocated new link* • *SR_X25_RC_NO_VC – PVC not found* • *SR_X25_RC_OK – success, PVC setup initiated*		

3.1.1 SrX25ListenPvc

int SrX25ListenPvc (sr_x25_pvc_setup_t *conn);		
DESCRIPTION		
This function initiates listening for PVC setup over XOT RFC1616		
PARAMETERS		
Type	Name	Description
sr_x25_pvc_setup_t	conn	Pointer to PVC setup configuration structure, defined as follows: l2_addr - link identifier, use RFC1613_LINK_ADDR constant arg – second part of link identifier, use RFC1613_LINK_PORT local_lcn – local VC logical channel number remote_lcn - remote VC logical channel number local_name - local interface name (null terminated string max 64 bytes) remote_name - remote interface name (null terminated string, max 64 bytes) int local_incoming_window - local incoming window size valid values are 1 to 7 int local_outgoing_window - local outgoing window size valid values are 1 to 7 int local_incoming_max_pktsize - local incoming max packet size Values from 4 to 10, see note below int local_outgoing_max_pktsize - local outgoing max packet size Values from 4 to 10, see note below The values for the packet sizes are encoded as follows: 4: 16 bytes 5: 32 bytes 6: 64 bytes 7: 128 bytes 8: 256 bytes 9: 512 bytes 10: 1024 bytes
RETURNS		
• *SR_X25_RC_NOT_INIT – SR X.25 library not initialized* • *SR_X25_RC_INV_PAR – invalid parameters* • *SR_X25_RC_NOLINK – Failed to allocated new link* • *SR_X25_RC_NO_VC – PVC not found* • *SR_X25_RC_OK – success, PVC setup initiated*		

3.1.1 SrX25DisconnectPvc

int SrX25DisconnectPvc (long l2_addr, unsigned short int port, unsigned short int lcn);

DESCRIPTION

This function disconnects an existing PVC

PARAMETERS

Type	Name	Description
long	l2_addr	Link identifier, use RFC1613_LINK_ADDR constant
long	arg	second part of link identifier, use RFC1613_LINK_PORT
short	lcn	PVC logical channel number

RETURNS

- *SR_X25_RC_NOT_INIT – SR X.25 library not initialized*
- *SR_X25_RC_INV_PAR – invalid parameters*
- *SR_X25_RC_NOLINK – Failed to allocated new link*
- *SR_X25_RC_NOT_VC – VC is not a PVC*
- *SR_X25_RC_OK – success*