

HS X.25 v1.2 User Manual

Revision: 1.2

Date: 18 January 2007

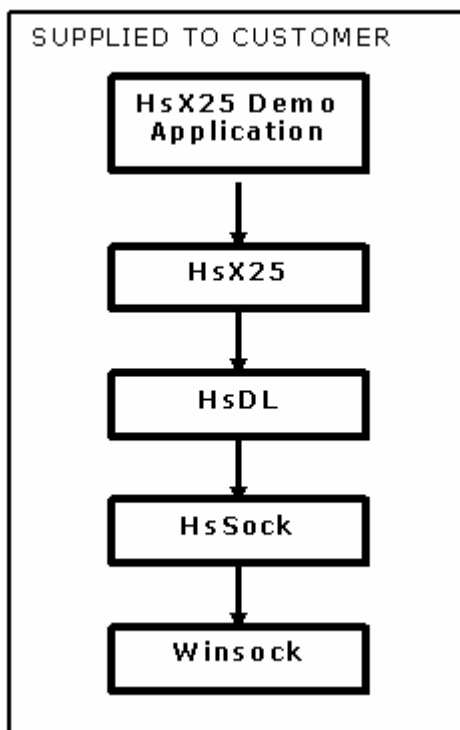
HS X.25 V1.2 USER MANUAL	1
REVISION: 1.2	1
DATE: 18 JANUARY 2007	1
1 INTRODUCTION	2
2 HS X.25 API	3
2.1 HSX25INIT	3
2.1.1 DEFINITION OF HS_X25_INIT_T STRUCTURE	3
2.2 HSX25CONNECT	6
2.3 HSX25LISTEN	8
2.4 HSX25CLEAR	9
2.5 HSX25DATA	9
2.6 HSX25DATAEXP	10
2.7 HSX25TICK	10
2.8 HSX25STUTDOWN	11
2.9 HSX25GETSTATS	11
2.10 HSX25RNR	12
2.11 HSX25DECODEPKT	12

1 Introduction

HS 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.

HS X.25 operates over RFC1613 Cisco Systems X.25 over TCP (XOT) which is part of HsDL sub-module included with the HS X.25 package.

Architecture and list of supported features:



HS X.25 internally at a lower layer interfaces to HsDL library. HsDL library is Data Link layer abstraction. When HS 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.

HsX25 as provided to customer 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 Hs X.25 library, it provides interface callbacks for the services used by HS X.25 protocol module: timer management, and event callbacks. The application then calls HsX25 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	No
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 HS X.25 API

2.1 HsX25Init

```
extern int HsX25Init (hs_x25_init_t *init);
```

DESCRIPTION

Initializes HS 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. HS X.25 module will call these functions when it needs to manage timers or feed events back to user

PARAMETERS

Type	Name	Description
hs_x25_init_t	*init	Pointer to initialisation structure: see next section for description

RETURNS

- *HS_X25_RC_OK* – success. HS X.25 initialised.
- *HS_X25_RC_INV_PAR* – invalid parameters
- *HS_X25_RC_DL_INIT_FAIL* – Data Link Layer initialisation failed

2.1.1 Definition of hs_x25_init_t structure

Function pointer	Prototype and Description				
hs_x25_api_t api	<p>This structure contains X.25 API to user function pointers (these must be initialised by user to point to functions in user code):</p> <ul style="list-style-type: none"> • hs_x25_start_timer_t *start_timer – X.25 calls this function to start a timer. <ul style="list-style-type: none"> ○ Prototype: long hs_x25_start_timer_t(long handle, unsigned long secs, hs_x25_timer_callback_t *callback, int tid); ○ Parameters: handle – HS 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: <i>void hs_x25_timer_callback_t(long handle);</i> where handle is X.25 module handle tid – timer id • hs_x25_stop_timer_t *stop_timer – X.25 calls this function to stop a timer. <ul style="list-style-type: none"> ○ Prototype: void hs_x25_stop_timer_t(long handle, int tid); ○ Parameters: handle – HS X.25 module handle tid – timer id • hs_x25_event_callback_t *event_cb – X.25 calls this function to feed events back to user application <ul style="list-style-type: none"> ○ Prototype: void hs_x25_event_callback_t(long handle, int ev, long arg1, long arg2); ○ Parameters: handle – HS X.25 module handle event - one of the following: <table border="1"> <thead> <tr> <th>Event</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>HS_X25_EV_CONN_FAIL_PKTLR Handle – VC handle,</td> <td>Connect failed, packet layer not ready</td> </tr> </tbody> </table>	Event	Description	HS_X25_EV_CONN_FAIL_PKTLR Handle – VC handle,	Connect failed, packet layer not ready
Event	Description				
HS_X25_EV_CONN_FAIL_PKTLR Handle – VC handle,	Connect failed, packet layer not ready				

	arg1 – LCI of the virtual circuit, arg2 – 0	
	HS_X25_EV_PKTDR_DOWN Handle – VC handle, arg1 – LCI of the virtual circuit, arg2 – 0	Packet layer down as result of physical layer down
	HS_X25_EV_PKTDR_RESTARTED Handle – VC handle, arg1 – LCI of the virtual circuit, arg2 – 0	Packet layer restarted (result of restart from the line)
	HS_X25_EV_CALL_CONNECTED Handle – VC handle, arg1 – LCI of the virtual circuit, arg2 – 0	X.25 SVC connected
	HS_X25_EV_CALL_CLEAR_COMPLETE Handle – VC handle, arg1 – LCI of the virtual circuit, arg2 – 0	X.25 SVC clearing completed
	HS_X25_EV_CALL_CLEARED Handle – VC handle, arg1 – LCI of the virtual circuit, arg2 – 0	X.25 SVC cleared locally
	HS_X25_EV_DATA Handle – VC handle, arg1 – Pointer to hs_x25data_t structure arg2 – LCI of the virtual circuit	X.25 data received <u>hs_x25data_t structure:</u> unsigned char *bufp – pointer to data buffer received int len – length of data buffer
	HS_X25_EV_DATA_EXP Handle – VC handle, arg1 – Pointer to start of data, arg2 – integer length of data	X.25 Expedited data received
	HS_X25_EV_EXP_DATA_TX_FAIL Handle – VC handle, arg1 – 0, arg2 – 0	Cannot send expedited data, previous exp data not acknowledged
	HS_X25_EV_EXP_DATA_DELIVERED Handle – VC handle, arg1 – 0, arg2 – 0	expedited data delivered to remote end
	HS_X25_EV_CALL_CLEARED_T13 Handle – VC handle, arg1 – LCI of the virtual circuit, arg2 – 0	X.25 SVC cleared - no answer to call indication
	HS_X25_EV_BUF_TX_FAIL Handle – VC handle, arg1 – buffer pointer, arg2 – 0	buffer transmission failed
	HS_X25_EV_INCOMING_CALL_CONNECTED Handle – VC handle, arg1 – LCI of the virtual circuit, arg2 – pointer to hsx25_conn_t structure	incoming call connected (HsX25Listen completed) <u>hs_x25_conn_t definition:</u> 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
int opt;	X.25 protocol startup options: HS_X25_OPT_NORMAL =0 – normal operation HS_X25_OPT_L2_LOOP =1 – operation in loopback mode between 2 local level 2 links.	

hs_trace_cb_t *trc_fn;	Trace function pointer in user code
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
unsigned char *rfc1613_peer_ip;	If parameter is_rfc1613 is set, rfc1613_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"
int is_dce;	Specifies DCE / DTE role of HS 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) HS 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) HS 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

2.2 HsX25Connect

Extern int HsX25Connect (hs_x25_conn_t *conn, long *handle, int *lci);

DESCRIPTION

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

PARAMETERS

Type	Name	Description																				
hs_x25_conn_t	*conn	<p>Pointer to connect structure as follows:</p> <table border="1"> <thead> <tr> <th>Data members</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>long l2_addr;</td> <td>4 octet level 2 address. <u>RFC1613 XOT mode:</u> - Set to unique identifier, for example 0xffff. Listens must also be submits on the same link identifier <u>Proprietary XOT mode:</u> - remote IP address</td> </tr> <tr> <td>long arg;</td> <td><u>RFC1613 XOT mode:</u> - Not used <u>Proprietary XOT mode:</u> - Set to remote TCP port number</td> </tr> <tr> <td>unsigned char called[HS_MAX_NUA+1];</td> <td>Called X.25 DTE address (null terminated)</td> </tr> <tr> <td>unsigned char calling[HS_MAX_NUA+1];</td> <td>Calling X.25 DTE address (null terminated)</td> </tr> <tr> <td>unsigned char facility_buf[HS_MAX_FAC];</td> <td>Optional X.25 facilities buffer to be included with the call request. Facilities must be encoded by application according to X.25 standard.</td> </tr> <tr> <td>unsigned char facility_len;</td> <td>Size of facility buffer</td> </tr> <tr> <td>unsigned char call_user_data[HS_MAX_CUD];</td> <td>Call user data to be included with the call</td> </tr> <tr> <td>int cud_len;</td> <td>Length of call user data</td> </tr> <tr> <td>int dbit;</td> <td>D Bit setting for the call</td> </tr> </tbody> </table> <p>Example setting level 2 address and argument for Proprietary XOT mode: conn.l2_addr = (long)inet_addr("10.1.73.164"); conn.arg = 1261;</p>	Data members	Description	long l2_addr;	4 octet level 2 address. <u>RFC1613 XOT mode:</u> - Set to unique identifier, for example 0xffff. Listens must also be submits on the same link identifier <u>Proprietary XOT mode:</u> - remote IP address	long arg;	<u>RFC1613 XOT mode:</u> - Not used <u>Proprietary XOT mode:</u> - Set to remote TCP port number	unsigned char called[HS_MAX_NUA+1];	Called X.25 DTE address (null terminated)	unsigned char calling[HS_MAX_NUA+1];	Calling X.25 DTE address (null terminated)	unsigned char facility_buf[HS_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_data[HS_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
Data members	Description																					
long l2_addr;	4 octet level 2 address. <u>RFC1613 XOT mode:</u> - Set to unique identifier, for example 0xffff. Listens must also be submits on the same link identifier <u>Proprietary XOT mode:</u> - remote IP address																					
long arg;	<u>RFC1613 XOT mode:</u> - Not used <u>Proprietary XOT mode:</u> - Set to remote TCP port number																					
unsigned char called[HS_MAX_NUA+1];	Called X.25 DTE address (null terminated)																					
unsigned char calling[HS_MAX_NUA+1];	Calling X.25 DTE address (null terminated)																					
unsigned char facility_buf[HS_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_data[HS_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																					
long	*handle	Pointer to variable in user code where X.25 shall store session handle – i.e. VC handle																				
int	*lci	Pointer to integer variable to receive selected logical channel number for outgoing virtual circuit																				

RETURNS

- *HS_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*

- *HS_X25_RC_INV_PAR* – invalid parameters
- *HS_X25_RC_TOO_LONG* – the sum of X.25 addresses, user facilities and call user data exceeds global maximum X.25 packet size
- *HS_X25_RC_NOLINK* - could not allocate new Layer 2 link
- *HS_X25_RC_NO_VC* - no free VC available

2.3 HsX25Listen

int HsX25Listen(hs_x25_listen_t *listen)

DESCRIPTION

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

PARAMETERS

Type	Name	Description														
hs_x25_listen_t	*listen	<p>Pointer to listen structure as follows:</p> <table border="1"> <tr> <td>Long l2_addr</td> <td> <p>RFC1613 XOT mode: Set to unique link identifier, for example 0xffff. Outgoing calls must also be done on this link identifier</p> <p>Proprietary XOT mode: set to local IP address which remote end will be connecting to.</p> <p>If layer 2 address not found, a new layer 2 link will be allocated.</p> </td> </tr> <tr> <td>Long arg</td> <td> <p>RFC1613 XOT mode: Not used</p> <p>Proprietary XOT mode: Set to local TCP port to listen on</p> </td> </tr> <tr> <td>int take_any_call</td> <td> <p>1- accept all incoming calls, regardless of X.25 called address,</p> <p>0 – check X.25 called and calling addresses of incoming call against calling and called parameters of hs_x25_listen_t structure</p> </td> </tr> <tr> <td>unsigned char called[HS_MAX_NUA];</td> <td>When processing incoming call, compare the called X.25 address of the incoming call packet with this NUA.</td> </tr> <tr> <td>int called_len;</td> <td>Number of called address digits</td> </tr> <tr> <td>unsigned char calling[HS_MAX_NUA];</td> <td>When processing incoming call, compare the calling X.25 address of the incoming call packet with this NUA.</td> </tr> <tr> <td>int calling_len;</td> <td>Number of calling address digits.</td> </tr> </table>	Long l2_addr	<p>RFC1613 XOT mode: Set to unique link identifier, for example 0xffff. Outgoing calls must also be done on this link identifier</p> <p>Proprietary XOT mode: set to local IP address which remote end will be connecting to.</p> <p>If layer 2 address not found, a new layer 2 link will be allocated.</p>	Long arg	<p>RFC1613 XOT mode: Not used</p> <p>Proprietary XOT mode: Set to local TCP port to listen on</p>	int take_any_call	<p>1- accept all incoming calls, regardless of X.25 called address,</p> <p>0 – check X.25 called and calling addresses of incoming call against calling and called parameters of hs_x25_listen_t structure</p>	unsigned char called[HS_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[HS_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.
Long l2_addr	<p>RFC1613 XOT mode: Set to unique link identifier, for example 0xffff. Outgoing calls must also be done on this link identifier</p> <p>Proprietary XOT mode: set to local IP address which remote end will be connecting to.</p> <p>If layer 2 address not found, a new layer 2 link will be allocated.</p>															
Long arg	<p>RFC1613 XOT mode: Not used</p> <p>Proprietary XOT mode: Set to local TCP port to listen on</p>															
int take_any_call	<p>1- accept all incoming calls, regardless of X.25 called address,</p> <p>0 – check X.25 called and calling addresses of incoming call against calling and called parameters of hs_x25_listen_t structure</p>															
unsigned char called[HS_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[HS_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

- **HS_X25_RC_OK**- success. Listen request accepted. Incoming call establishment is notified asynchronously via event callback
- **HS_X25_RC_INV_PAR** – invalid parameters
- **HS_X25_RC_DL_LIS_FAIL** – Listen failed due to Data Link layer error
- **HS_X25_RC_NOLINK** – no free link available

2.4 HsX25Clear

int HsX25Clear(long handle)		
DESCRIPTION		
<i>This function is called to Clear an X.25 SVC</i>		
PARAMETERS		
Type	Name	Description
Long	handle	X.25 VC handle
RETURNS		
<ul style="list-style-type: none"> • <i>HS_X25_RC_OK – success. Clearing procedure initiated</i> 		

2.5 HsX25Data

int HsX25Data(long handle, unsigned char *buf, int len, int qbit);		
DESCRIPTION		
<i>This function is called to send data over established X.25 VC</i>		
PARAMETERS		
Type	Name	Description
long	handle	X.25 VC handle
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.
RETURNS		
<ul style="list-style-type: none"> • <i>HS_X25_RC_OK – success, data queued by HS X.25 for sending over specified VC</i> • <i>HS_X25_INV_PAR – invalid parameters</i> • <i>HS_X25_BUF_TOO_BIG – buffer is too big</i> • <i>HS_X25_TXQ_FULL – transmit queue is full, try again later.</i> 		

2.6 HsX25DataExp

int HsX25DataExp (long handle, unsigned char *buf,)

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	handle	X.25 VC handle
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 HsX25Data: #define HDR_SPACE 10 static unsigned char testbuf2[HDR_SPACE + 8]; HsX25DataExp(x25vc_han, &testbuf2[HDR_SPACE], 8);
int	len	Length of data to send

RETURNS

- *HS_X25_RC_OK – success. Expedited data request accepted.*

2.7 HsX25Tick

int HsX25Tick(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 HS X.25 timers and services transmit queue.

PARAMETERS

Type	Name	Description
		No parameters

RETURNS

- *HS_X25_RC_OK - success*

2.8 HsX25StutDown

Extern void HsX25StutDown(void)		
DESCRIPTION		
<i>This function is used to close all X.25 links and release all resources used by HS X.25</i>		
PARAMETERS		
Type	Name	Description
		No parameters
RETURNS		
<ul style="list-style-type: none"> • <i>None</i> 		

2.9 HsX25GetStats

void HsX25GetStats(long handle, hs_x25_stats_t *stats)																	
DESCRIPTION																	
<i>This function is used to get X.25 VC statistics</i>																	
PARAMETERS																	
Type	Name	Description															
long	handle	X.25 VC handle															
hs_x25_stats_t	*stats	<p>Pointer to stats structure where X.25 will copy statistics information</p> <p>hs_x25_stats_t format:</p> <table border="1"> <tr> <td>Int</td> <td>lci</td> <td>Logical channel number</td> </tr> <tr> <td>tx_rx_t</td> <td>tx</td> <td>Transmit statistics</td> </tr> <tr> <td>tx_rx_t</td> <td>rx</td> <td>Receive statistics</td> </tr> <tr> <td>unsigned</td> <td>char *st</td> <td>Name of current state</td> </tr> <tr> <td>int</td> <td>w</td> <td>Current Window</td> </tr> </table> <p>Tx_rx_t definition:</p> <p>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</p>	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
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															
RETURNS																	
<ul style="list-style-type: none"> • <i>None</i> 																	

2.10 HsX25Rnr

int HsX25Rnr(long handle, 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
pong	handle	X.25 VC handle
int	on	1= set flow control condition 0= clear flow control condition

RETURNS

- *HS_X25_RC_INV_PAR* – invalid parameters specified
- *HS_X25_RC_OK* - success

2.11 HsX25DecodePkt

int HsX25DecodePkt(unsigned char *pkt, int len, int *ev, hs_x25_pkt_info_t *info, int need_name);

DESCRIPTION

This function may be used for debugging or diagnostics to decode the content of an X.25 packet

PARAMETERS

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: <table border="1" data-bbox="703 1384 1433 1742"> <tbody> <tr><td>HS_X25_CALL_ACCEPTED</td><td>Call accept</td></tr> <tr><td>HS_X25_CALL_REQ</td><td>Call request</td></tr> <tr><td>HS_X25_RESTART_REQ,</td><td>Restart request</td></tr> <tr><td>HS_X25_RESTART_CNF</td><td>Restart confirm</td></tr> <tr><td>HS_X25_RR</td><td>Receiver ready</td></tr> <tr><td>HS_X25_RNR</td><td>Receiver not ready</td></tr> <tr><td>HS_X25_DAT</td><td>DATA</td></tr> <tr><td>HS_X25_CLEAR_CNF</td><td>Clear confirm</td></tr> <tr><td>HS_X25_CLEAR_REQ</td><td>Clear request</td></tr> <tr><td>HS_X25_INT_CNF</td><td>Interrupt confirm</td></tr> <tr><td>HS_X25_INT_REQ</td><td>Interrupt request</td></tr> <tr><td>HS_X25_RESET_REQ</td><td>Reset request</td></tr> <tr><td>HS_X25_RESET_CNF</td><td>Reset confirm</td></tr> </tbody> </table>	HS_X25_CALL_ACCEPTED	Call accept	HS_X25_CALL_REQ	Call request	HS_X25_RESTART_REQ,	Restart request	HS_X25_RESTART_CNF	Restart confirm	HS_X25_RR	Receiver ready	HS_X25_RNR	Receiver not ready	HS_X25_DAT	DATA	HS_X25_CLEAR_CNF	Clear confirm	HS_X25_CLEAR_REQ	Clear request	HS_X25_INT_CNF	Interrupt confirm	HS_X25_INT_REQ	Interrupt request	HS_X25_RESET_REQ	Reset request	HS_X25_RESET_CNF	Reset confirm
HS_X25_CALL_ACCEPTED	Call accept																											
HS_X25_CALL_REQ	Call request																											
HS_X25_RESTART_REQ,	Restart request																											
HS_X25_RESTART_CNF	Restart confirm																											
HS_X25_RR	Receiver ready																											
HS_X25_RNR	Receiver not ready																											
HS_X25_DAT	DATA																											
HS_X25_CLEAR_CNF	Clear confirm																											
HS_X25_CLEAR_REQ	Clear request																											
HS_X25_INT_CNF	Interrupt confirm																											
HS_X25_INT_REQ	Interrupt request																											
HS_X25_RESET_REQ	Reset request																											
HS_X25_RESET_CNF	Reset confirm																											

hs_x25_pkt_info_t	*info	Pointer to structure in user code that receives decoded packet information. Structure definition: <table border="1" data-bbox="703 342 1430 607"> <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>	Int lci	Logical channel identifier	Int pti	Packet type identifier	unsigned char *pkt	Points to start of packet	Int len	Packet length	Int Pr	PR value	Int Ps	PS value	unsigned char *pktname	Name of packet
Int lci	Logical channel identifier															
Int pti	Packet type identifier															
unsigned char *pkt	Points to start of packet															
Int len	Packet length															
Int Pr	PR value															
Int Ps	PS value															
unsigned char *pktname	Name of packet															
int	need_name	1= fill in packet name 0= do not fill in packet name														

R E T U R N S

- *HS_X25_RC_DECODE_SHORT* – invalid packet – too short
- *HS_X25_RC_DECODE_LONG* – invalid packet – too long
- *HS_X25_RC_DECODE_LCI_UNASSIGN* – packet on unassigned logical channel
- *HS_X25_RC_DECODE_UNSPEC* – decoded error, unspecified
- *HS_X25_RC_DECODE_INV_GFI* – invalid general format identifier
- *HS_X25_RC_DECODE_OK* – packet is good and is correctly decoded