Freescale Semiconductor
Application Note

AN3092
Rev. 0, 09/2006

CCSRL, SRP, NSRP, and WNSRP Protocols for 3G-324M on PowerQUICC™ Processors

by David Turvey
NCSG
Freescale Semiconductor, Inc.
East Kilbride, UK

This document provides an overview of the following protocols and describes how they can be implemented on a Freescale PowerQUICC™ processor. It also describes the functions to manipulate data within these protocols:

- Control channel segmentation and reassembly layer (CCSRL). A protocol for segmenting and reassembling control messages.
- Simple retransmission protocol (SRP). A protocol that ensures message delivery through a system of acknowledgement messages and retransmission
- Numbered simple retransmission protocol (NSRP). A protocol that enhances SRP functionality by adding sequence numbers to response messages.
- Windowed numbered simple retransmission protocol (WNSRP). A development of NSRP that adds a windowing system so multiple messages can be transmitted simultaneously.

This application note also discusses the following ITU-T recommendations:

- H.324. Defines a terminal for low bit-rate multimedia communication.
- H.245. Defines a control protocol for multimedia communication.

Contents

1 3G-324M Protocol Stack 2
2 Protocol Functionality 3
2.1 Control Channel Segmentation and Reassembly (CCSRL) 3
2.2 Simple Retransmission Protocol (SRP) 4
2.3 Numbered Simple Retransmission Protocol (NSRP) 5
2.4 Windowed Numbered Simple Retransmission Protocol (WNSRP) 5
2.5 SRP, NSRP, WNSRP Protocol Switching 5
2.6 SRP Design Structure 6
3 Using the Protocols 8
3.1 CCSRL Functions 9
3.2 SRP Functions 9
3.3 H.245 and Adaption Layer Interface 10
3.4 Mode Functions 10
3.5 Data Structures 10
3.6 File Structure 11
3.7 Retransmission Configuration 11
4 Testing 11
4.1 Test Environments 12
4.2 Example Test 12
5 References 13

- H.223. Defines a multiplexing protocol for low bit rate multimedia communications. A sublayer of the H.223 protocol is the adaption layer, which interfaces with the SRP, NSRP, and WNSRP protocols.
- 3G-324M. A version of the H.324 recommendation for use in wireless environments. Specifically, the use of annex A and C of H.324.

The platform development kit (PDK) used for testing features an MPC8260 processor and an Ethernet interface. It runs on the Linux operating system.

1 3G-324M Protocol Stack

The 3G-324M protocol stack specifies a range of protocols for delivering multimedia content to 3G-enabled mobile devices. This content can come from one of a variety of applications, such as video conferencing or video-on-demand entertainment services. The 3G-324M stack uses H.324 Annex A and C specifications to provide functionality that targets the issues inherent in communication with mobile devices. Specifically, a higher than normal bit error ratio (BER). Before content can be delivered, communicating terminals must establish the procedures for communication, addressing issues such as the exchange of terminal capabilities and the control of logical channels. 3G-324M uses H.245 to perform these control tasks.

The SRP, NSRP, and WNSRP protocols ensure the delivery of the H.245 messages to the communicating terminal. The accurate transmission of data is not generally guaranteed with 3G-324M. While certain levels of lost data can be accommodated in video and audio transmission, no data loss can be tolerated with the control messages. SRP, NSRP, and WNSRP use a system of response messages to acknowledge the receipt of messages and retransmission of messages, where no receipt is forthcoming.

SRP and NSRP provide basic functionality for this purpose while WNSRP incorporates a windowing technique designed to reduce the time required for call setup. Call setup times, typically 8–10 seconds with SRP, can be reduced to 1–2 seconds. This obviously varies with factors such as round trip delay and error rates. 3G-324M terminals should support all three of the SRP, NSRP, and WNSRP protocols and implement the mechanisms to switch between. This allows them to remain compatible with terminals that do not support one or more of the protocols. CCSRL is used to split H.245 messages into smaller segments for transmission to try to reduce the performance hit due to the retransmission of messages. Because this process generates more messages to be transmitted, CCSRL should ideally be used in conjunction with WNSRP to avoid extending call setup time. Figure 1 shows how the CCSRL, SRP, NSRP, and WNSRP fit into the 3G-324M protocol stack.

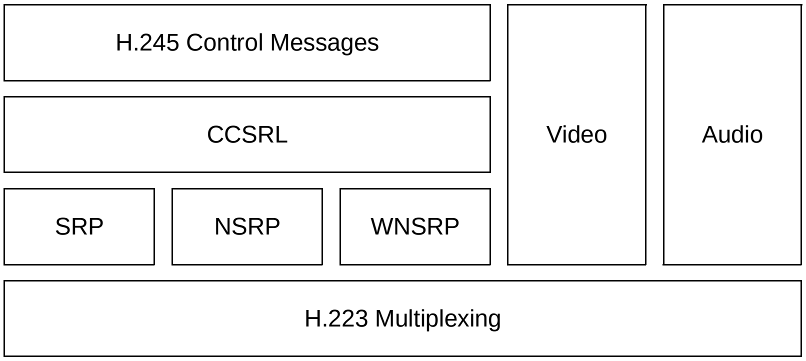

Figure 1. 3G-324M Protocol Stack

2 Protocol Functionality

This section provides details on the functionality of the CCSRL, SRP, NSRP, and WNSRP protocols.

2.1 Control Channel Segmentation and Reassembly (CCSRL)

CCSRL provides services to the H.245 layer, which generates control messages for transmission to the communicating terminal. These services are necessary for the correct transmission of the multimedia components of the system. CCSRL allows H.245 messages up to 256 octets long to be split into as many as 256 segments. The data received from H.245 may be one or more control messages, which the CCSRL treats as a single block of data. The segment length is an arbitrary length determined by the terminal. It adds a header to each segment to facilitate reassembly of the message. The headers are 0xFF if the segment is the final segment of the message and 0x00 otherwise. Figure 2 illustrates this process with a segment size of 15. Note that the final segment can be smaller than the segment size, if necessary.

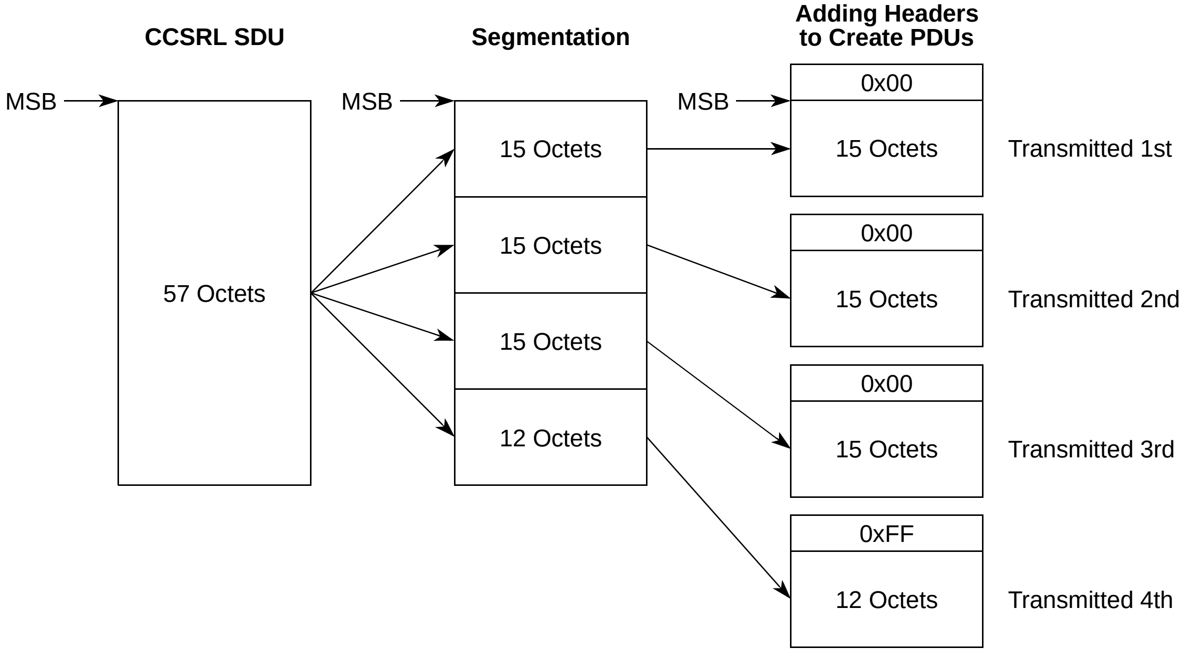

Figure 2. CCSRL Segmentation Process

The CCSRL protocol reassembles these segments into complete H.245 messages as they are received. Figure 3 shows the structure of the implemented CCSRL layer. CCSRL_Request() uses a transmitting buffer to store segments that cannot be passed to SRP when the SRP buffer is full. This prevents loss of data due to full SRP buffers. SRP_Indication uses the reassembly buffer to hold received segments until the entire message is received and can be reconstructed.

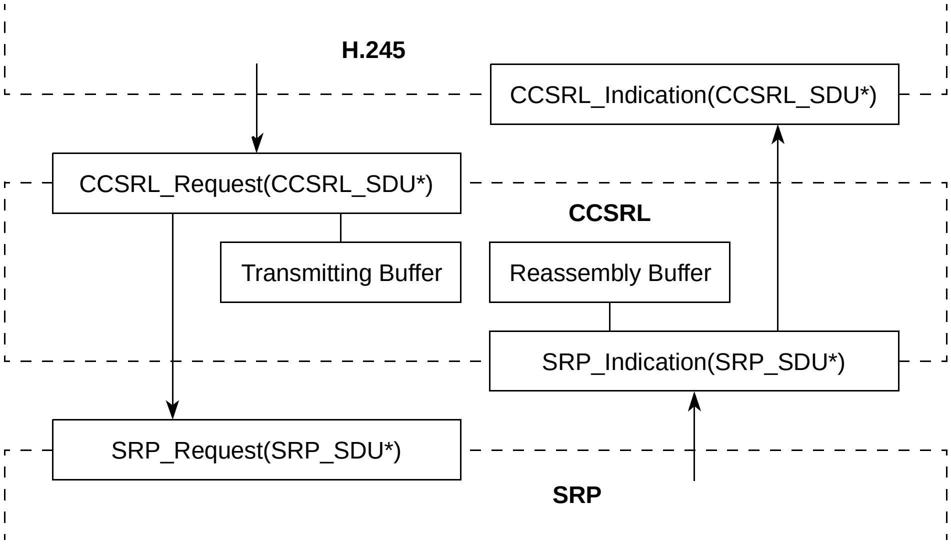

Figure 3. CCSRL Structure

2.2 Simple Retransmission Protocol (SRP)

SRP, the most basic transmission protocol, allows a single message to be transmitted at one time. The message is given a 1-octet sequence number and a 1-octet header to define it as an SRP command frame (0xF9). A 2-octet CRC value is also generated on the entire frame as specified in V.42/8.1.1.6.1. It then logs the time of the transmission and passes the message on to the multiplex adaption layer.

The transmission should be periodically checked to ensure that the time since transmission has not exceeded the timeout value (T401). If timeout occurs, the message should be retransmitted up to N400 times. If the message times out more than N400, the SRP layer should indicate this error to the terminal so appropriate action can be taken. When it correctly receives an SRP frame, the SRP layer should send a response frame to indicate that it has received the command frame. If the sequence number is the next expected number, it should pass the information field of this frame up to CCSRL. Figure 4 shows the SRP frame structure.

Command Frame Format

1 Octet	1 Octet	Up to 2048 Octets	2 Octets
Header	Sequence Number	Information Field	FCS

Response Frame Format

1 Octet	2 Octets
Header	FCS

Figure 4. SRP Frame Format

2.3 Numbered Simple Retransmission Protocol (NSRP)

NSRP is very similar to SRP, except that response frames include the sequence number of the frame to which they are responding, eliminating any ambiguity in the intended purpose of the response frame. Ambiguity can arise when messages start getting lost or delayed in an error-prone channel. Preventing ambiguity reduces the timeout value T401, thus improving performance. Figure 5 shows the format of an NSRP response frame.

Response Frame Format

1 Octet	1 Octet	2 Octets
Header	Sequence Number	FCS

Figure 5. NSRP Response Frame Format

2.4 Windowed Numbered Simple Retransmission Protocol (WNSRP)

WNSRP allows up to 256 messages to transmit simultaneously. Each message has its own timeout and retransmission counter. The frame structure is the same as for SRP, except that the header value is 0xF1. As messages are sent simultaneously, the WNSRP receiver must have a mechanism for ensuring that the received messages are correctly reordered before they are passed up to CCSRL and that retransmissions of already received messages are not reprocessed. These two goals are accomplished through a jitter buffer and frame history table.

The frame history table is used to determine whether a received frame contains new information or if it is a retransmission of a previously processed frame. The table contains a copy of the last frame received for each of the 256 sequence numbers. When a frame is received, the payload of the frame is compared with that of the last received frame with the same sequence number. If these are identical, the frame is assumed to be a retransmission and is not passed up to CCSRL. If they are different, this is a new frame that can be processed normally. After it is processed, this frame replaces the old entry in the frame history table. This system provides reliable determination of frame validity because frames have only a 1 in $8.7*10^{40}$ chance of having exactly the same payload given random data content and an information field length of 16 bytes.

The jitter buffer is used to reorder frames that arrive out of the order implied by the sequence numbers. This buffer contains up to 256 PDUs received from the adaption layer. If a PDU arrives without the next expected sequence number and is verified as new data by comparison with the frame history table, it is stored in the jitter buffer until its sequence number is the next to be processed. Frames are stored using their sequence number as the buffer index. For example, frame number 146 would be stored in jitterBuffer[146]. This method of indexing implicitly reorders the frames.When a frame with the expected sequence number does arrive, it is processed as normal. Then the jitter buffer is checked to verify that the frame with the next expected number is present. If so, the frame is passed up to CCSRL and the next number is checked. This process repeats until there is no frame with the expected number in the buffer.

2.5 SRP, NSRP, WNSRP Protocol Switching

Figure 1 shows that the SRP, NSRP, and WNSRP protocols all exist on the same stack layer. However, these protocols are not all used simultaneously. H.234 specifies the mechanism for switching between the protocols in use. The possible modes of operation are as follows:

- SRP and WNSRP mode. The initial mode of operation for the system. It allows WNSRP functionality to be used from the start of the call while maintaining backward compatibility for communication with non-WNSRP capable terminals. SRP frames are transmitted using multiplex channel 0, which is reserved for control messages. WNSRP frames are transmitted on channel 15. Channel 15 messages is accepted and processed on WNSRP-enabled systems but ignored on systems lacking this capability.
- NSRP and WNSRP mode. Used if NSRP capability is identified before WNSRP capability can be confirmed. It works in the same way as mode 1 except that it uses NSRP on channel 0 instead of SRP.
- SRP-only mode. Used after the communication terminal verifies that it does not support WNSRP by sending more than N401 SRP frames without sending a WNSRP frame. The system sends only SRP frames in this mode.
- NSRP-only mode. Similar to SRP-only mode, this mode is used to transmit NSRP frames after NSRP capability is identified.
- WNSRP mode. If WNSRP capability can be confirmed, the system sends all messages with only WNSRP. These messages are transmitted on channel 0.

NSRP capability is identified with a H.245 capability exchange message. This identification can occur at any point during program operation. WNSRP capability is confirmed by the reception of a WNSRP command or response frame. If the system receives N401 SRP frames (N401 is an arbitrary number greater than 1), then WNSRP mode is denied. After it is confirmed or denied, the system cannot switch out of or into WNSRP mode, respectively. Table 1 shows the mode switching based on the current mode and the event requiring the switch

Table 1. Protocol Switching

Event	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
NSRP Confirmed	2	2	4	4	5
WNSRP Confirmed	5	5	3	4	5
WNSRP Denied	3	4	3	4	5

2.6 SRP Design Structure

Figure 6 and Figure 7show how the SRP, NSRP, and WNSRP protocols are implemented. They show the functions and structures used and how they relate to one another. Figure 6 shows the functions for transmitting a message, and Figure 7 shows the functions for receiving a message.

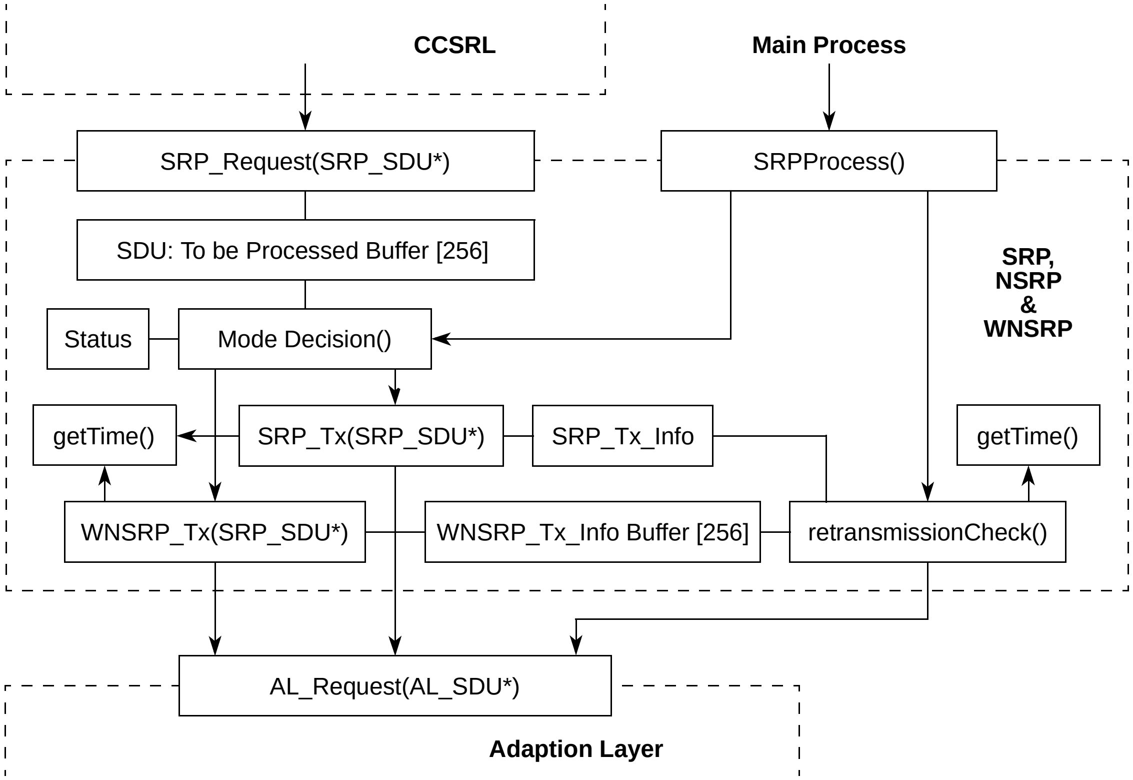

Figure 6. SRP, NSRP, and WNSRP Transmitting Design Structure

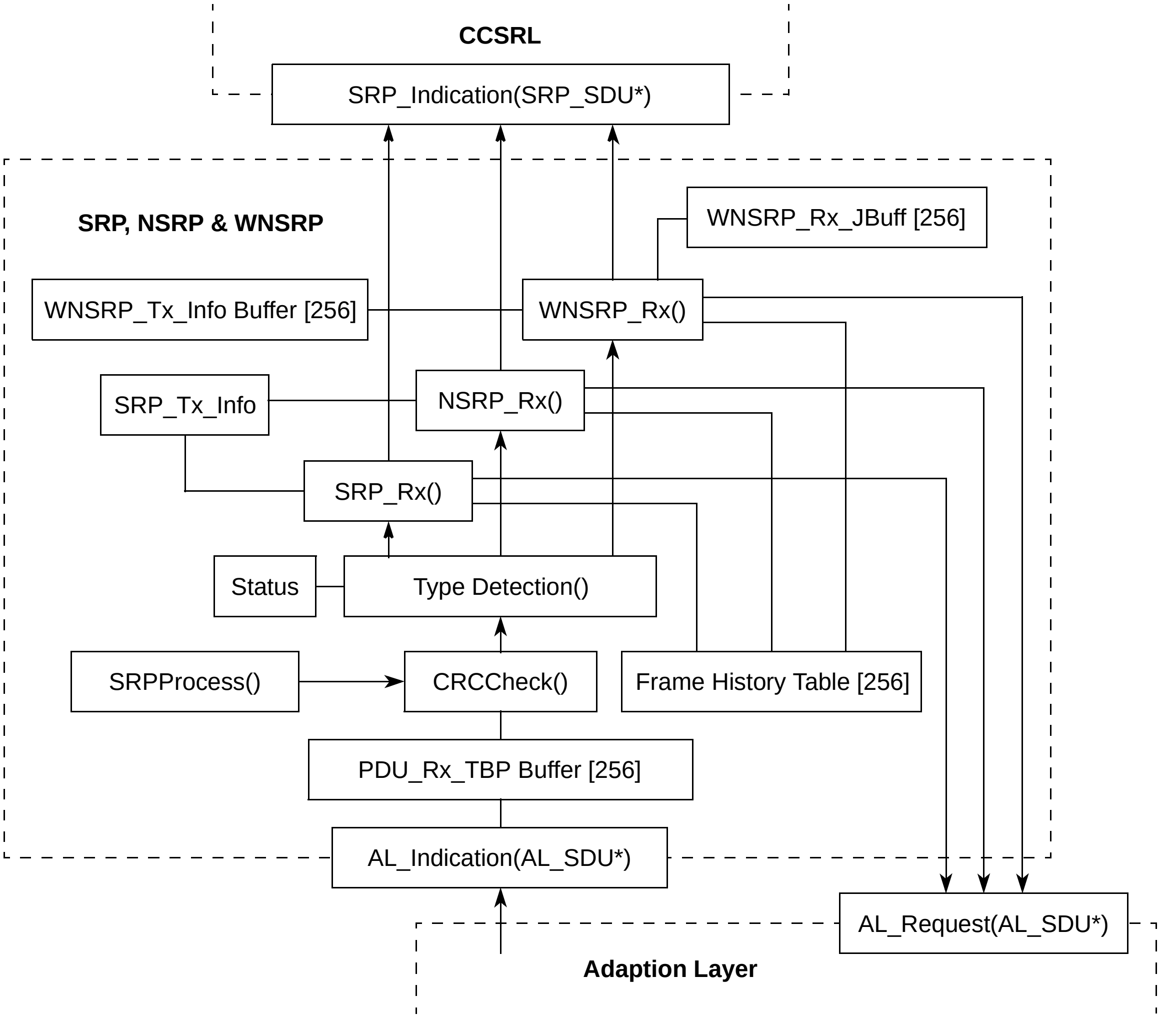

Figure 7. SRP, NSRP, and WNSRP Receiving Design Structure

3 Using the Protocols

These protocols discussed in this document are written in C and follow the ANSI Standard as closely as possible. The only deviation is getTime() in `SRP_misc_functions.c`. This file uses Linux-specific timing functions because standard C does not provide timers with high enough resolution. To run the protocol on non-Linux systems, this function must be redesigned. It should return the current time in milliseconds as a 32-bit number. The value returned by the initial use of this function is not arbitrary, but further use of this function should return a value consistent with the delay between use. The following subsections describe the functions used to set up, control, and pass data to the protocols. The interfaces for these functions, along with relevant data structures, are defined in `headers.h`.

3.1 CCSRL Functions

The CCSRL functions are as follows:

- void CCSRLinitialise(). Prepares the CCSRL data structures for operation. It should be run at the start of each call to initialize the protocol.
- int CCSRL_Request(CCSRL_SDU*). Passes data from H.245 to CCSRL for segmentation. The data is passed in as a pointer to a CCSRL_SDU structure (described in Section 3.5, “Data Structures”). To clear the data that may be stored in buffers, this function should be called with NULL as a parameter.

  Return values:

  — 0 Successful operation.

  — 2 Input larger than maximum allowed size or NULL input.

  — 3 Could not pass a segment to SRP.
- int SRP_Indication(SRP_SDU*). Passes data from the SRP layer up to CSRL for reassembly. The data is passed in as a pointer to a SRP_SDU structure (described in Section 3.5, “Data Structures”).

  Return values:

  — 0 Successful operation.

  — 1 Invalid input. Either incorrect header or no data in after header.

  — 2 Reassembly buffer full.

  — 3 Reassembled message is bigger than the maximum allowed size.

3.2 SRP Functions

The SRP functions are as follows:

- void SRPinitialise(). Prepares the SRP data structures for operation. It should be run at the start of each call to initialize the protocol.
- int SRP_Request(SRP_SDU*). Passes data to the SRP protocols for transmission. It adds the data to a buffer until the protocol can process it. The data input is a pointer to a SRP_SDU structure (defined below).

  Return values:

  — 0 Successful operation.

  — 1 Buffer is full, not added.
- int AL_Indication(AL_SDU*). The adaption layer uses this function to pass data up to the SRP layer for processing. It takes a pointer to an AL_SDU data structure where the data is held. It adds received SDUs to a buffer for processing.

  Return values:

  — 0 Successful Operation

  — 1 Buffer full, not added.
- int SRPProcess(). Performs tasks necessary to run the protocol properly. It processes the data in the transmit and receive buffers. It also checks all transmitting frames for timeouts and retransmits as

necessary. This process should be run periodically; how often depends on the system load and transmission timeout delay specified.

Return values:

— 0 Successful Operation

— 1 A transmitting frame has exceeded the maximum retransmission count.

3.3 H.245 and Adaption Layer Interface

The following two functions define the interface to communicate with the H.245 and adaption protocols. They should be implemented by the relevant protocol.

- int CCSRL_Indication (CCSRL_SDU*). Allows the CCSRL protocol to pass up a reassembled message to H.245. The message is stored in a CCSRL_SDU structure.

  Return values:

  — 0: Successful Operation.
- int AL_Request (AL_SDU*). Allows the SRP/NSRP/WNSRP protocol to request that a message be transmitted by the adaption layer. The message is stored in an AL_SDU structure.

  Return values:

  — 0 Successful Operation

3.4 Mode Functions

The mode functions can be used for getting and setting the mode of SRP operation currently in use. For details, see Section 2.5 .

- int getMode(). Returns the number corresponding to the current mode of operation. The adaption layer can use it to determine the appropriate multiplex channel.
- void setNSRP(). For use by the H.245 protocol after a capability exchange message confirms that NSRP communication is possible. This function updates the mode of operation as shown in Table 1.

3.5 Data Structures

This section defines the data structures used to pass data between protocols. The length variable holds the length of the payload data in bytes. This should be an integer number. Data that is not a integer number of bytes is not valid. Payload* holds the address in memory where the data starts.

```
typedef struct
{

         int16_t length;            /*length of payload in octets*/
         uint8_t*  payload;         /*pointer to payload*/

}CCSRL_SDU;

typedef struct

{
```

```
         uint16_t length;            /*length of payload in octets*/
         uint8_t* payload;           /*pointer to payload*/

}SRP_SDU;

typedef struct

{

         uint16_t length;            /*length of payload in octets*/
         uint8_t* payload;           /*pointer to payload*/

}AL_SDU;
```

3.6 File Structure

The following files are required in order to run the protocols correctly.

- headers.h
- CCSRL.h
- SRP.h
- ccsrl.c
- srp.c
- SRP_Misc_Functions.c
- SRP_Rx.c
- SRP_Tx.c

Headers.h should be included in the main() function or wherever the protocol functions are used. The following files are required for code testing and demonstration. They simulate the H.245 and adaption layer functionality and provide control of test routines.

- main.c
- testfunctions.h
- testfunctions.c

3.7 Retransmission Configuration

The protocol provides for the arbitrary selection of certain specific elements of the retransmission system. These values are defined in `srp.h`. No mechanism is provided for dynamically changing these parameters during runtime.

- T401. Defines the time in milliseconds before a message is retransmitted. See V.42 appendix IV for information on this value.
- N400. The number of times a message can be retransmitted before communication is assumed to be lost. The minimum value is 5
- N402. The number on SRP frames that can be received without a WNSRP frame before WNSRP communication attempts are abandoned. The minimum number is 1.

4 Testing

This section describes the test environments and an example test.

4.1 Test Environments

The protocols were tested in three different setups:

- On a single Pentium 4 Linux machine, primarily to verify correct code operation for simple test cases
- On two pentium 4 Linux machines communicating over an Ethernet connection, for more complex testing and verification of correct operation over a real-world communication link.
- On two PDKs with MPC8260 processors running a Linux kernel to demonstrate that the system can be used in a real-world embedded environment.

A variety of tests were run on the protocols to ensure correct operation under a variety of test conditions. Acceptance tests were run to ensure that the system correctly handled valid and invalid inputs. These were simple tests that targeted a single subsystem within the protocols.

Tests were run to ensure that SRP/NSRP/WNSRP mode switching was invoked by the correct stimuli from the system and did not result in any loss or corruption of data. Packet loss, jitter, and out-of-order delivery were simulated in the transmission channel between two communicating terminals. This tested the SRP protocols for correct retransmission handling and the WNSRP protocol for correct reordering of packets.

All buffers in the system were stress tested for overflow handling, on each buffer individually and as a sustained torture test on the system as a whole. The sustained testing was accomplished by flooding the CCSRL with requests to transmit data from the simulated H.245 for a sustained period of time. The messages were of varying length and content. They were retransmitted back by the communicating terminal as soon as they were received by its H.245 layer. This provided a high level of data flow in both directions. Although this is not typical of the message load for which this system is designed, it does demonstrate that it can withstand high loading. Therefore, it can handle moderate loads comfortably.

4.2 Example Test

One particular test setup was used to test transmission of a large number of messages in a single transmission session so the sequence number would overflow. The test sent a total of 960 messages, enough to overflow the sequence number several times. The setup is also designed to test the handling of a variety of message sizes. This test can be completed with or without simulation of packet loss and jitter. In this instance there was no jitter simulation.

- *Test inputs*. 256 messages of sizes 1-256 bytes
- *Test setup*. There are two communicating terminals in this setup, A and B, which transmit packets over an Ethernet connection using a UDP socket. This test was run on two PDKs, and the results were observed using a serial connection with the PDK (see Figure 8). The test was also run on a Pentium 4 running Linux where terminal A and B were run as separate processes and the IP address used was the loopback address 127.0.0.1.
- *Test method***.** Transmits a 1-byte message to CCSRL. This message is processed using the rules governing WNSRP-to-WNSRP communication and then transmitted to B over the Ethernet link. After all responses are received, it transmits a message one byte longer. This process repeats until all responses are received for a 256-byte message.

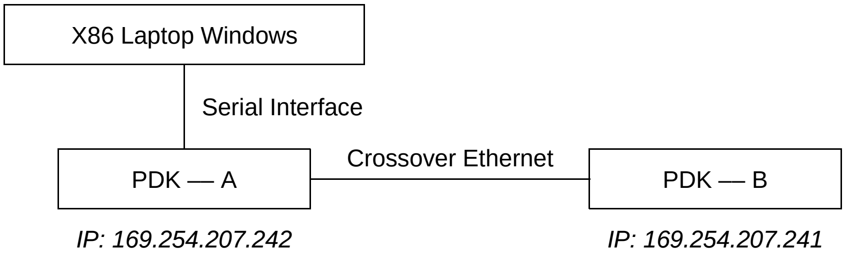

Figure 8. PDK-to-PDK Test Setup

- *Results***.** The test successfully completed without errors. It completed in the following times, measured from the transmission of the first message to the reception of the last response message.
  - — X86 - 2633ms
  - — PDK to PDK -4760ms

A full account of the testing is given in the SDD for the SRP/NSRP/WNSRP layer of the 3G-324M Protocol Stack.

5 References

The following ITU recommendation documents are used as references in this document. The validity of these URLs was checked in September of 2006:

- H.324. `http://www.itu.int/rec/T-REC-H.324/en`
- H.245. `http://www.itu.int/rec/T-REC-H.245/en`
- H.223. `http://www.itu.int/rec/T-REC-H.223/en`
- V.42. `http://www.itu.int/rec/T-REC-V.42/en`

The following documents are related to the development of these protocols.

- *Software Design Document for the CCSRL layer of the 3G-324M Protocol Stack*, Freescale internal use only.
- *Software Design Document for the SRP/NSRP/WNSRP layer of the 3G-324M Protocol Stack*, Freescale internal use only.
- *ICD for the CCSRL/SRP/NSRP/WNSRP layer of the 3G-324M Protocol Stack*, Freescale internal use only.

THIS PAGE INTENTIONALLY LEFT BLANK

THIS PAGE INTENTIONALLY LEFT BLANK

How to Reach Us:

Home Page:
www.freescale.com

email:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
1-800-521-6274
480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku
Tokyo 153-0064, Japan
0120 191014
+81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate,
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor
Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447
303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor
@hibbertgroup.com

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters which may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

Document Number: AN3092
Rev. 0
09/2006