December 2001

HDLC SUPPORT FOR SANGOMA CARDS

Hardware Interface Manual

LIMITED USE LICENSE AGREEMENT

Sangoma Technologies Inc. provides the computer software program contained on the medium in this package (hereinafter called the Program) and licenses its use.

THE LICENSEE SHOULD CAREFULLY READ THE FOLLOWING TERMS AND CONDITIONS BEFORE ATTEMPTING TO USE THIS PRODUCT. INSERTION OF ANY OF THE DISKETTES IN THIS PACKAGE INTO ANY MACHINE INDICATES THE LICENSEE'S ACCEPTANCE OF THESE TERMS AND CONDITIONS. IF THE LICENSEE DOES NOT AGREE WITH THE TERMS AND CONDITIONS, THE LICENSEE SHOULD PROMPTLY RETURN THE PACKAGE WITHIN 15 DAYS UNUSED AND UNCOPIED IN ANY WAY SHAPE OR FORM, AND MONIES WILL BE REFUNDED.

LICENSE:

a. The purchaser of this license (hereinafter called the Licensee) is granted a personal, non-exclusive license to use the Program in accordance with the terms and conditions set out in this agreement.

b. The Program may be used only on a single computer per license granted.

c. The Licensee and the Licensee's agents and employees shall protect the confidentiality of the Program and shall not distribute or make available the Program or documentation to any third party.

d. The Licensee may copy the programs into machine readable or printed form for backup or modification purposes only in support of the Licensee's use on a single machine. The Licensee must reproduce and include the copyright notice on any copy, modification or portion merged into another program.

e. Any portion of the Program merged into or used in conjunction with another program will continue to be subject to the terms and conditions of this agreement.

f. The Licensee may not assign or transfer the license or the program to any third party without the express prior consent of Sangoma Technologies Inc.

g. The licensee acknowledges that this license is only a limited license to use the Program and documentation, and that Sangoma Technologies Inc. retains full title to the program and documentation.

h. The Licensee shall not use, copy, modify or transfer the Program or documentation or any copy, modification or merged portion, in whole or in part, except as expressly provided for in this license. If the Licensee transfers possession of any copy, modification or merged portion of the program to a third party, the license is automatically terminated under this agreement.

TERM:

The license is effective until terminated. The licensee may terminate the license at any time by destroying the Program together with all copies, modifications and merged portion in any form. The Licensee agrees upon such termination to destroy the Program together with all copies, modifications and merged portion in any form.

LIMITED WARRANTY:

The Program is provided "as is" without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The entire risk as to the performance of the Program is with the Licensee. Should the Program prove defective, the Licensee (and not Sangoma Technologies Inc. or an authorized dealer) shall assume the entire cost of all necessary servicing, or correction. However, Sangoma Technologies Inc. warrants the diskettes on which the Program is furnished will be free of defects in materials or workmanship under normal use for a period of 90 days from the date of delivery to the Licensee. In no event will Sangoma Technologies Inc. be liable for any damages, including incidental or consequential damages arising out of the use or inability to use the Program, even if Sangoma Technologies Inc. or an authorized dealer have been advised of the possibility of such damages, or for any claim by any other party.

The Licensee acknowledges that the Licensee has read this agreement, understands it, and agrees to be bound by its terms and conditions. The Licensee further agrees that it is the complete and exclusive statement of the agreement between the parties and supersedes any proposal or prior agreement, oral or written, and any other communications between the parties relating to the subject matter of this agreement.

1. Introduction

The Sangoma cards are general co-processor communication adapters capable of supporting any V.35 and RS232 based communication protocol autonomously, and providing information transfer into PC work space.

This document describes the support of the HDLC protocol on the cards, together with special enhancements implemented in the Sangoma HDLC code. The hardware/software solution handles the link level autonomously, without PC intervention. The PC accesses the system as required to send or receive Information frames and to recover status and statistical data..

The implementation of HDLC on the S508 adapter corresponds to the specifications of the ISO document ISO 7776: "Information processing systems - Data Communication - High-level data link control procedures- Description of the X.25 LAPB-compatible DTE data link procedures", (first edition 1986-12-15).

The implementation of HDLC on the S508 adapter includes the following features:

- Modulo 8 and modulo 128 operation.
- A maximum information field length of 4099 bytes is supported.
- Configurable as both a DTE and DCE.
- Complete HDLC statistics package.
- A built in datascope with frame time stamping.
- Line speeds up to 2666 kbps.

Conventions Used in this Manual

Programming conventions used are as follows:

Variables described with an **0x** prefix or an **h** suffix are hexadecimal values. All other variables are decimal.

For bit mapping, the **least significant** (**low**) bit is denoted as bit **0**.

2. Hardware

S503

This is a short 4 layer card, compatible with the ISA bus and it supports hardware interrupts as well as operating in a passive polled mode. The RS232 or V.35/X.21 interface is jumper selectable.

Clock speed:

This is factory set by Jumper **JP1**. Do not change without consulting your Sangoma dealer.

I/O port address:

This is set by Jumper **JP3**.

Pins 5-6	Pins 3-4	Pins 1-2	I/O Address Selection
Not Jumpered	Jumpered	Jumpered	250-252 (Hex)
Jumpered	Jumpered	Jumpered	254-256 (Hex)
Not Jumpered	Jumpered	Not Jumpered	300-302 (Hex)
Jumpered	Jumpered	Not Jumpered	304-306 (Hex)
Not Jumpered	Not Jumpered	Jumpered	350-352 (Hex)
Jumpered	Not Jumpered	Jumpered	354-356 (Hex)
Not Jumpered	Not Jumpered	Not Jumpered	360-362 (Hex)☎
Jumpered	Not Jumpered	Not Jumpered	364-366 (Hex)

☎Factory default.

IRQ Selection

The optional IRQ is set using **JP2**.

Pins 1-2	Pins 3-4	Pins 5-6	Pins 7-8	Pins 9-10	Selection
In	Out	Out	Out	Out	IRQ 2
Out	In	Out	Out	Out	IRQ 3
Out	Out	In	Out	Out	IRQ 4
Out	Out	Out	In	Out	IRQ 5
Out	Out	Out	Out	In	IRQ 7☎

☎Factory default.

Interface Level Selection

This is set by Jumper **JP3**.

Pins 9-10	Interface Level
Jumpered	RS-232
Not Jumpered	V.35

S514 PCI card

No jumpers need to be set on this card as it is configured by the PC BIOS.

S508 ISA Card

Jumpers **JP1** on the S508 define the card I/O address range as specified in Table 3-1. The specified card I/O addresses must not conflict with I/O addresses in use by any other hardware installed on the server. Use the **SNOOPER** utility if you are in any doubt as to hardware settings.

Note that **JP1-1** on the S508 is furthest to the left if the board is held such that the connectors are to

the right. **JP1-4** is reserved.

The 8k (2000 Hex) byte shared memory address and the IRQ level are set in software for the S508.

Internal Line Clocking

For back-to-back connections, the cards can provide their own Transmit and Receive clock signals, which, with the appropriate cable, can also provide the clock for third party devices.

All cards are capable of generating the transmit and receive clocks as long as the appropriate back-to-back cable is used. The generated line speed is set by software.

However, the cards have a very large configurable range and therefore cannot easily be tabulated. When asked for the line speed during setup, you may specify any value in kbps from 1 to 2600. The actual generated line speed will be reasonably close the specified value, but will deviate more as the line speed increases.

S514 Port Pinouts

NB: Port PA is the Primary 4Mbps port
Port PB is the Secondary 512Kbps port.

PIN #	PA:RS232	PA:V.35	PB:RS232	PB:V.35
1	RTS		RTS	
2	CTS		CTS	
3	GND	GND	GND	GND
4	DCD		DCD	
5	DTR		DTR (V.10)	
6	TXD			
7	RXD			
8	TXC			
9	RXC			
10		RTS		RTS
11		CTS		CTS
12		DCD		DCD
13		DTR		DTR (V.10)
14		TXD		

PIN #	PA:RS232	PA:V.35	PB:RS232	PB:V.35
15		RXD		
16		TXC		
17		RXC		
18			TXA	
19			TXB	
20			RXA	
21			RXB	
22			TX Clock A	
23			TX Clock B	
24			RX Clock A	
25			RX Clock B	
26			DTR A (V.11)	
27			DTR B (V.11)	
28				TXA
29				TXB
30				RXA
31				RXB
32				TX Clock A
33				TX Clock B
34				RX Clock A
35				RX Clock B
36				DTR A (V.11)
37				DTR B (V.11)

S503/S508 Port Pinouts

RS232

Pin #	**Function**
2	TxD
3	RxD
7	GND
4	RTS
5	CTS
20	DTR
6	DSR
8	DCD
15	TxC
17	RxC
24	BxC

V.35/X.21

Pin #	**Function**
4	RTS
5	CTS
6	DSR
7	GND
8	DCD
10	TxA
9	TxB
12	RxA
11	RxB
19	Tx Clock A
20	DTR (V10 signal)
13	DTRA (V11 signal)
14	DTRB (V11 signal)
21	Tx Clock B
22	RI
23	Rx Clock A
25	Rx Clock B
18	Aux. Clock A (On board clock source)
16	Aux. Clock B (On board clock source)

3. COMMAND CODES

The valid interface commands are:

GLOBAL COMMANDS

0x01 READ_GLOBAL_EXCEPTION_CONDITION
0x02 SET_GLOBAL_CONFIGURATION
0x03 READ_GLOBAL_CONFIGURATION
0x04 READ_GLOBAL_STATISTICS
0x05 FLUSH_GLOBAL_STATISTICS
0x06 SET_MODEM_STATUS
0x07 READ_MODEM_STATUS
0x08 READ_COMMS_ERROR_STATS
0x09 FLUSH_COMMS_ERROR_STATS
0x0A SET_TRACE_CONFIGURATION
0x0B READ_TRACE_CONFIGURATION
0x0C FLUSH_TRACE_BUFFERS
0x0D READ_TRACE_STATISTICS
0x0E FLUSH_TRACE_STATISTICS
0x1E FT1_MONITOR_STATUS_CTRL
0x1F SET_FT1_MODE

HDLC-LEVEL COMMANDS

0x20 READ_HDLC_CODE_VERSION
0x21 READ_HDLC_EXCEPTION_CONDITION
0x22 SET_HDLC_CONFIGURATION
0x23 READ_HDLC_CONFIGURATION
0x24 ENABLE_HDLC_COMMUNICATIONS
0x25 DISABLE_HDLC_COMMUNICATIONS
0x26 HDLC_CONNECT

0x27 HDLC_DISCONNECT

0x28 READ_HDLC_LINK_STATUS

0x29 READ_HDLC_OPERATIONAL_STATS

0x2A FLUSH_HDLC_OPERATIONAL_STATS

0x2B SET_HDLC_BUSY_CONDITION

0x2C SEND_UI_FRAME

0x30 SET_HDLC_INTERRUPT_TRIGGERS

0x31 READ_HDLC_INTERRUPT_TRIGGERS

READ_GLOBAL_EXCEPTION_CONDITION (0x01)

Check to see if a global exception condition has occurred. If so, return the details of that exception condition.

If there is an exception condition to report, a **return_code** other than 0x01 is passed to the application. It is important to note that each pending exception condition is reported only once via a **READ_GLOBAL_EXCEPTION_CONDITION** command, and is then cleared on completion of this command.

Control Block values to be set on entry	
command	Set to 0x01.
buffer_length	Set to 0x00.

Control Block values set on exit	
return_code	0x01 - There is no current global exception condition to report. 0x10 - A change has occurred in the status of DCD and/or CTS. The details of this modem status change is returned in the mailbox data area. 0x11 - The line trace has been disabled due to the application not processing trace frames in a timely manor.
buffer_length	Set to the number of bytes of data in the mailbox data area .
data	For a Return_code of 0x10 (modem status change), the details of modem status is shown at offset 0x00 of the mailbox data area as follows: If bit 2 is set, then DCD has changed. Bit 3 is set if DCD is now high and is reset if DCD is low. If bit 4 is set, then CTS has changed. Bit 5 is set if CTS is now high and is reset if CTS is low.

SET_GLOBAL_CONFIGURATION (0x02)

Set the global configuration.

This command is generally not used, as the default global configuration automatically set when the adapter is loaded is adequate for most operating environments. If this command is used, it must be the first command issued after loading the adapter.

The global configuration structure is defined in the file HDLCAPI.H as **GLOBAL_CONFIGURATION_STRUCT**.

Control Block values to be set on entry	
command	Set to 0x02.
buffer_length	Set to the size of the **GLOBAL_CONFIGURATION_STRUCT**.
data	The **GLOBAL_CONFIGURATION_STRUCT** should be completed and copied to offset 0x00 in the mailbox data area. Configuration parameters are as follows: **adapter_config_options** - miscellaneous adapter configuration options as follows: Bits 0 to 15 - reserved. **app_IRQ_timeout** - the time permitted by the application to respond to an IRQ triggered by the adapter. If the application does not respond within this defined time limit, then the adapter will re-issue the IRQ. Valid values are from 0 to 5000 milliseconds, with 0 disabling the IRQ timeout mechanism. **adapter_operating_frequency** - the adapter operating frequency. Valid values are from 0 to 33000000 Hz, with 0 using the default frequency selected on loading.

Control Block values set on exit	
return_code	0x00 - The command was performed successfully. 0x01 - The **buffer_length** set on entry was not set to the size of the **GLOBAL_CONFIGURATION_STRUCT.** 0x02 - The selected **app_IRQ_timeout** is invalid. 0x03 - The selected **adapter_operating_frequency** is invalid.

READ_GLOBAL_CONFIGURATION (0x03)

Read the current global configuration values.

The global configuration structure is defined in the file HDLCAPI.H as **GLOBAL_CONFIGURATION_STRUCT**.

Control Block values to be set on entry	
command	Set to 0x03.
buffer_length	Set to 0x00.

Control Block values set on exit	
return_code	0x00 - The command was performed successfully.
buffer_length	Set to the size of the **GLOBAL_CONFIGURATION_STRUCT**.
data	The **GLOBAL_CONFIGURATION_STRUCT** at offset 0x00 in the mailbox data area. Read configuration parameters are as for the **SET_GLOBAL_CONFIGURATION** command.

READ_GLOBAL_STATISTICS (0x04)

Retrieve the global statistics for the adapter.

The structure for returning the communication error information is defined in the file HDLCAPI.H as the **GLOBAL_STATS_STRUCT**.

Control Block values to be set on entry	
command	Set to 0x04.
buffer_length	Set to 0x00.

Control Block values set on exit	
return_code	0x00 - The command was performed successfully.
buffer_length	Set to the size of the **GLOBAL_STATS_STRUCT**.
data	The **GLOBAL_STATS_STRUCT** is at offset 0x00 of the mailbox data area. The structure members are as follows: **app_IRQ_timeout_count -** the number of times that the application failed to respond to a triggered IRQ within the defined time limit.

FLUSH_GLOBAL_STATISTICS (0x05)

Reset all the global statistics to zero.

Control Block values to be set on entry	
command	Set to 0x05.
buffer_length	Set to 0x00.

Control Block values set on exit	
return_code	0x00 - The command was performed successfully.

SET_MODEM_STATUS (0x06)

Set the status of the DTR and RTS output leads.

Control Block values to be set on entry	
command	Set to 0x06.
buffer_length	Set to 0x01.
data	The status of the DTR and RTS output leads is indicated by the byte at offset 0x00 of the mailbox data area as follows: If bit 0 is reset, then DTR is to be set low. If bit 0 is set, then DTR is to be set high. If bit 1 is reset, then RTS is to be set low. If bit 1 is set, then RTS is to be set high.

Control Block values set on exit	
return_code	0x00 - The command was performed successfully.

READ_MODEM_STATUS (0x07)

Read the status of the DCD and CTS input leads.

Control Block values to be set on entry	
command	Set to 0x07.
buffer_length	Set to 0x00.

Control Block values set on exit	
return_code	0x00 - The command was performed successfully.
buffer_length	Set to 0x01.
data	The status of the DCD and CTS input leads is indicated by the byte at offset 0x00 of the mailbox data area as follows: If bit 3 is reset, then DCD is set low. If bit 3 is set, then DCD is set high. If bit 5 is reset, then CTS is set low. If bit 5 is set, then CTS is set high.

READ_COMMS_ERROR_STATS (0x08)

Retrieve the communications error statistics for the link.

The structure for returning the communication error information is defined in the file HDLCAPI.H as the **COMMS_ERROR_STATS_STRUCT**.

Control Block values to be set on entry	
command	Set to 0x08.
buffer_length	Set to 0x00.

Control Block values set on exit	
return_code	0x00 - The command was performed successfully.
buffer_length	Set to the size of the **COMMS_ERROR_STATS_STRUCT**.
data	The **COMMS_ERROR_STATS_STRUCT** is at offset 0x00 of the mailbox data area. Most of the included statistics are self-explanatory. Additional details pertaining to selected structure members are as follows: **CRC_err_count**, **Rx_abort_count** - these two statistics are valuable in detecting cable and line configuration problems. Please contact your Sangoma Technologies representative if any one of these values increase steadily. **missed_Tx_und_int_count** - the number of frames which were not completely transmitted within a predetermined timeout limit. If this statistic is non-zero, then there is very likely a problem in an external device supplying a transmit clock to the S508 adapter.

FLUSH_COMMS_ERROR_STATS (0x09)

Reset all the communications error statistics to zero.

Control Block values to be set on entry	
command	Set to 0x09.
buffer_length	Set to 0x00.

Control Block values set on exit	
return_code	0x00 - The command was performed successfully.

SET_TRACE_CONFIGURATION (0x0A)

Set the line trace configuration.

The line trace configuration structure is defined in the file HDLCAPI.H as **LINE_TRACE_CONFIG_STRUCT**.

Control Block values to be set on entry	
command	Set to 0x0A.
buffer_length	Set to the size of the **LINE_TRACE_CONFIG_STRUCT**.
data	The **LINE_TRACE_CONFIG_STRUCT** is at offset 0x00 of the mailbox data area. The structure members are as follows: **trace_config** - the line trace configuration byte as follows: Bit 0 - if reset, then the line trace is deactivated. If set, then the line trace is activated. Bit 1 is reserved for later use. Bit 2 - if set, then the line trace delay mode is activated. This mode serves to slow down the actual line protocol if the application is not reading trace data from the adapter at a sufficient rate. If this bit is reset, then trace data will be discarded if no trace buffering is available. Bit 3 - if set, then Information frames will be traced. Bit 4 - if set, then Supervisory frames will be traced. Bit 5 - if set, then Unnumbered frames will be traced. Bits 6 and 7 are reserved for later use. Note that if the line trace is being enabled, then bits 3, 4 and 5 may all be set if all types of frames are to be traced. **trace_deactivation_timer** - this timer is used in conjunction with the trace delay mode and serves to automatically deactivate the trace in the case where the application does not read trace data from the adapter at least once per defined deactivation period. Valid values are from 0 to 8000 milliseconds, with a recommended default of 2000 milliseconds. **ptr_trace_stat_el_cfg_struct** - not used for the **SET_TRACE_CONFIGURATION** command.

Control Block values set on exit	
return_code	0x00 - The command was performed successfully. 0x01 - The **buffer_length** set on entry was not set to the size of the **LINE_TRACE_CONFIG_STRUCT.** 0x02 - The **trace_config** byte is invalid. The trace has been enabled but bits 3,4 and 5 are all reset. 0x03 - The defined **trace_deactivation_timer** is invalid.

READ_TRACE_CONFIGURATION (0x0B)

Read the current line trace settings. Also, establish the location on the adapter of the **TRACE_STATUS_EL_CFG_STRUCT** defined in the file HDLCAPI.H.

The structure for reading the line trace configuration is defined in the file HDLCAPI.H as the **LINE_TRACE_CONFIG_STRUCT**.

Control Block values to be set on entry	
command	Set to 0x0B.
buffer_length	Set to 0x00.

Control Block values set on exit	
return_code	0x00 - The command was performed successfully.
buffer_length	Set to the size of the **LINE_TRACE_CONFIG_STRUCT**.
data	The **LINE_TRACE_CONFIG_STRUCT** is at offset 0x00 in the mailbox data area. Read configuration parameters are as for the **SET_TRACE_CONFIGURATION** command with the following additional definition: **ptr_trace_stat_el_cfg_struct** - the physical address of the **TRACE_STATUS_EL_CFG_STRUCT** on the adapter (see the section "Using the Line Trace" for further details).

FLUSH_TRACE_BUFFERS (0x0C)

Discard all data currently in the line trace buffers and reset all trace pointers.
All elements of the **TRACE_STATUS_EL_CFG_STRUCT** are reinitialized.

Control Block values to be set on entry	
command	Set to 0x0C.
buffer_length	Set to 0x00.

Control Block values set on exit	
return_code	0x00 - The command was performed successfully.

READ_TRACE_STATISTICS (0x0D)

Retrieve the line trace statistics for the link.

The structure for returning the link status information is defined in the file HDLCAPI.H as the **LINE_TRACE_STATS_STRUCT**.

Control Block values to be set on entry	
command	Set to 0x0D.
buffer_length	Set to 0x00.

Control Block values set on exit	
return_code	0x00 - The command was performed successfully.
buffer_length	Set to the size of the **LINE_TRACE_STATS_STRUCT**.
data	The **LINE_TRACE_STATS_STRUCT** is at offset 0x00 of the mailbox data area. The structure members are as follows: **frames_traced_count** - the total number of frames traced and made available for the application. **trc_frames_not_recorded_count** - the total number of frames not traced due to trace buffer limitations. **trc_disabled_internally_count** - the number of times that the trace had to be disabled internally (trace delay mode active).

FLUSH_TRACE_STATISTICS (0x0E)

Reset all the line trace statistics to zero.

Control Block values to be set on entry	
command	Set to 0x0E.
buffer_length	Set to 0x00.

Control Block values set on exit	
return_code	0x00 - The command was performed successfully.

FT1_MONITOR_STATUS_CTRL (0x1E)

Reserved for internal use.

SET_FT1_MODE (0x1F)

Reserved for internal use.

READ_HDLC_CODE_VERSION (0x20)

Return the HDLC code version.

.

Control Block values to be set on entry	
command	Set to 0x20.
buffer_length	Set to 0x00.

Control Block values set on exit	
return_code	0x00 - The command was performed successfully.
buffer_length	Set to the length of the ASCII code version string
data	The HDLC code version string (ASCII format) is at offset 0x00 of the mailbox data area. This string is of the format main version.sub-version For example, the version string may be “2.03".

READ_HDLC_EXCEPTION_CONDITION (0x21)

Check to see if an HDLC exception condition has occurred. If so, return the details of that exception condition.

If there is an exception condition to report, a **return_code** other than 0x21 is passed to the application. It is important to note that each occurring exception condition is reported only once via a **READ_HDLC_EXCEPTION_CONDITION** command, and is then cleared.

A number of the exception conditions listed below indicate that the link is no longer in the ABM and I-frames may not be transferred. The link state may be established by examining the **HDLC_status** byte in the **HDLC_INFORMATION_STRUCT** or by using the **READ_HDLC_LINK_STATUS** command.

Control Block values to be set on entry	
command	Set to 0x21.
buffer_length	Set to 0x00.

Control Block values set on exit	
return_code	0x21 - There is no current HDLC exception condition to report. 0x30 - A SABM frame was received from the remote station while the link was in the ABM. The station issued a UA frame to re-enter the ABM. 0x31 - A DISC frame was received from the remote station while the link was in the ABM. The link entered the disconnected state. 0x32 - A DM frame was received from the remote station while the link was in the ABM. The link is no longer in the data transfer mode and SABM frames are issued in an attempt to re-enter the ABM. 0x33 - A UA frame was received from the remote station while the link was in the ABM. The link is no longer in the data transfer mode and SABM frames are issued in an attempt to re-enter the ABM. 0x34 - A FRMR frame was received from the remote station while the link was in the ABM. The contents of the received FRMR frame is returned in the mailbox data area. The link is no longer in the data transfer mode and SABM frames are issued in an attempt to re-enter the ABM. 0x37 - A UI frame was received from the remote station. The contents of the received UI frame is returned in the mailbox data area. 0x38 - A SABM frame was transmitted in an attempt to enter the ABM due to the reception of a DM frame. 0x39 - A SABM frame was transmitted due to the reception of a UA frame while in the link was in the ABM. 0x3A - A SABM frame was transmitted due to the reception of a FRMR frame while in the link was in the ABM. 0x3B - A SABM frame was transmitted due to the reception of an unsolicited response frame with the F-bit set while in the link was in the ABM. The link is no longer in the data transfer mode and SABM frames are issued in an attempt to re-enter the ABM. 0x3C - A SABM frame was transmitted due the expiry of the N2 counter. The link is no longer in the data transfer mode and SABM frames are issued in an attempt to re-enter the ABM. 0x3F - A FRMR frame was transmitted. The contents of the outgoing FRMR frame is returned in the mailbox data area. The link is no longer in the data transfer mode and the station waits to receive a SABM frame from the remote device in an attempt to re-enter the ABM. 0x40 - A SABM retry count was exceeded, i.e., N2 successive SABM link-setup frames were transmitted without receiving a UA response. 0x41- A DISC retry count was exceeded, i.e., N2 successive DISC frames were transmitted without receiving a UA response. 0x42- A FRMR retry count was exceeded, i.e., N2 successive FRMR frames were transmitted without receiving a SABM command from the remote station. The HDLC station has attempted link re-initialization by transmitting a SABM frame. 0x45 - The T3 timeout limit has been exceeded. 0x47 - The HDLC link as entered the ABM (data transfer mode).
buffer_length	Set to the number of bytes of data in the mailbox data area .
data	For a return code of 0x34 or 0x3F (FRMR frame received or transmitted), the FRMR Information field is shown from offset 0x00 in the mailbox data area. For a return code of 0x37 (UI frame received), the details of the incoming UI frame may be read from the **UI_FRAME_STRUCT** (see file HDLCAPI.H) at offset 0x00 of the mailbox data area.

SET_HDLC_CONFIGURATION (0x22)

Set the HDLC code configuration.

This command is generally the first command used after downloading the HDLC code to the adapter and it also serves to reinitialize the HDLC state,

frame numbering and Information frame buffering . Note that communications must be disabled (**DISABLE_HDLC_COMMUNICATIONS** command) before a **SET_HDLC_CONFIGURATION** command is used.

The HDLC configuration structure is defined in the file HDLCAPI.H as **HDLC_CONFIGURATION_STRUCT**.

Control Block values to be set on entry	
command	Set to 0x22.
buffer_length	Set to the size of the **HDLC_CONFIGURATION_STRUCT**.
data	The **HDLC_CONFIGURATION_STRUCT** should be completed and copied to offset 0x00 in the mailbox data area. Configuration parameters are as follows: **baud_rate** - The baud rate of the access line (bps). If external clocking is used, then this value must be set to zero. Valid values are from 0 (external clocking) to 2666000 bps. **line_config_options** If bit 0 is reset, then the electrical interface is V.35. If bit 0 is set, then the electrical interface is RS232. Bits 1 to 15 - reserved. **modem_config_options** If bit 0 is reset, then DTR and RTS will automatically be raised when an **ENABLE_HDLC_COMMUNICATIONS** command is performed. If bit 0 is set, then the setting of DTR and RTS is the responsibility of the application (use the **SET_MODEM_STATUS** command). If bit 1 is reset, then modem status changes will be reported by the **READ_GLOBAL_EXCEPTION_CONDITION** command. If bit 1 is set, then the **READ_GLOBAL_EXCEPTION_CONDITION** command will not return any changes in the modem status. If bit 2 is reset, then the DCD and CTS transitions will trigger interrupts on the adapter. If bit 2 is set, then DCD and CTS transitions will be ignored. Bits 3 to 15 - reserved.

Control Block values to be set on entry (continued)	
data	**HDLC_API_options** If bit 0 is reset, then the application will not be permitted to force the HDLC code to issue a SABM frame via an **HDLC_CONNECT** command while the link is in the ABM. If bit 0 is set, then the application is able to issue a SABM frame via a **HDLC_CONNECT** command while the link is in the ABM. Bits 1 to 15 - reserved. **HDLC_protocol_options** If bit 0 is reset, then modulo 8 operation is used. If bit 0 is set, then modulo 128 (extended) operation is used. If bit 1 is set, then automatic modulus detection is performed. The HDLC station will wait for an incomming SABM or SABME link setup command and then assume the appropriate modulo operation. In this case, the **READ_HDLC_STATUS** command may be used to establish the automatically configured operational link modulus. If bit 2 is reset, then the HDLC station will issue SABM/SABME commands once an **HDLC_CONNECT** command has been performed. If bit 2 is set, then the HDLC station will wait for an incoming SABM/SABME link setup command. If bit 3 is reset, then the station will enter the initialized state after a **HDLC_DISCONNECT** command has been issued. This implies that the application must issue a **HDLC_CONNECT** command to once again enter the ABM (connected) state. If bit 3 is set, then the station will enter the disconnected state after a **HDLC_DISCONNECT** command has been issued. This implies that the station will positively reply to an incoming SABM/SABME link setup command to once again enter the ABM (connected) state. Bits 4 to 15 - reserved. **HDLC_buffer_config_options** Bits 0 to 3 - the Information receive buffering hysteresis. All bits should be set to 0 unless advised by your Sangoma representative. Bits 4 to 5 - define the on-board Information frame buffer as follows: If bits 4 and 5 are reset, then the transmit/receive buffer ratio is 50/50 (recommended). If bit 4 is set and bit 5 is reset , then the transmit/receive buffer ratio is 70/30. If bit 4 is reset and bit 5 is set, then the transmit/receive buffer ratio is 30/70. Bits 6 to 15 - reserved.

Control Block values to be set on entry (continued)	
data	**HDLC_statistics_option** If bit 0 is set, then the number of bytes of data in outgoing Information frames will be tallied. If bit 1 is set, then the number of bytes of data in received Information frames will be tallied. If bit 2 is set, then the Information frame transmit throughput statistic will be made available for the application via the **READ_HDLC_OPERATIONAL_STATS** command. If bit 3 is set, then the Information frame receive throughput statistic will be made available for the application. Bits 4 to 15 - reserved. **configured_as_DTE** - Defines the station's link level configuration. If bit 0 is set, then the station is a DTE. If bit 0 is reset, then the station is a DCE. **max_HDLC I_field_length** - The maximum number of data bytes in a transmitted or received Information frame. Valid values are from 300 to 4099 bytes. **HDLC_I_frame_window** - The size of the frame level window, i.e. the maximum number of sequentially numbered Information frames that may be unacknowledged at any given time. Valid values are 1 to 7 for modulo 8 operation and 1 to 127 for modulo 128 operation. Note that a frame level window value of 0 may also be used under special circumstances. This setting implies that the transmit window will not be closed by this station, i.e., the station will transmit I-frames without regards to the number of currently unacknowledged I-frames. **HDLC_T1_timer** - The period timer which is used for various link level re-transmission and recovery procedures. Valid values are from 1 to 60000 milliseconds and a typical value is 3000 milliseconds. **HDLC_T2_timer** - The amount of time available at the station before an acknowledging frame must be initiated in order to ensure its receipt by the remote station, prior to timer T1 running out at that station. Valid values are from 0 to 60000 milliseconds and is generally set to 0 to indicate an immediate response. This value must always be less than T1. **HDLC_T3_timer** - The amount of time in which no HDLC traffic is received from the remote station to consider the link to be in the disconnected state. Valid values are from 0 to 60000 milliseconds and is generally set to 0 so as to ignore the T3 timer. If not set to 0, then T3 must be much greater than the T4 timer.

Control Block values to be set on entry (continued)	
data	**HDLC_T4_timer** - The maximum time that the station will allow without frames being on the data link. Once the HDLC link is in the asynchronous balanced mode (ABM) information transfer phase, there is no need for any HDLC Supervisory frame traffic to occur between the two devices on the link if no Information frames are being transmitted. The T4 parameter is used to force the HDLC code to issue a link level Supervisory frame (and to elicit a response from the remote device) periodically during this 'quiescent' ABM link phase. Valid values are from 0 to 60000 milliseconds. A setting of 0 implies that no unnecessary Supervisory frames will be generated. If not set to 0, then T4 must be greater that the T1 timer. **HDLC_N2_counter** - This is the maximum number of transmissions and retransmissions of an HDLC frame at T1 intervals before a state change occurs. Valid values are 1 to 30 and a typical value is 10. **ptr_SDLA_app_info_area** **ptr_HDLC_Tx_stat_el_cfg_struct** **ptr_HDLC_Rx_stat_el_cfg_struct** - not used for the **SET_HDLC_CONFIGURATION** command.

Control Block values set on exit	
return_code	0x00 - The command was performed successfully. 0x21 - Disable communications (**DISABLE_HDLC_COMMUNICATIONS** command) before setting the HDLC configuration. 0x22 - The **buffer_length** set on entry was not set to the size of the **HDLC_CONFIGURATION_STRUCT.** 0x23 - The passed configuration data was invalid. The **buffer_length** is set to the offset within the **HDLC_CONFIGURATION_STRUCT** of the invalid configuration parameter. 0x24 - The selected T3 timer is less than the T1 timer. 0x25 - The selected T4 timer is less than the T1 timer. 0x26 - The selected **max_HDLC_I_field_length** is excessive. Adjust the **HDLC_buffer_config_options** to increase the size of Information frame receive buffer. The **buffer_length** is set to the maximum I-frame size that is supported with the current receive buffer size.
buffer_length	Valid for a **return_code** of 0x23. Set to the offset within the **HDLC_CONFIGURATION_STRUCT** of the invalid configuration parameter.

READ_HDLC_CONFIGURATION (0x23)

Read the current HDLC code configuration.

The HDLC configuration structure is defined in the file HDLCAPI.H as **HDLC_CONFIGURATION_STRUCT**.

Control Block values to be set on entry	
command	Set to 0x23.
buffer_length	Set to 0x00.

Control Block values set on exit	
return_code	0x00 - The command was performed successfully.
buffer_length	Set to the size of the **HDLC_CONFIGURATION_STRUCT**.
data	The **HDLC_CONFIGURATION_STRUCT** at offset 0x00 in the mailbox data area. Read configuration parameters are as for the **SET_HDLC_CONFIGURATION** command with the following additional definitions : **ptr_shared_mem_info_struct** - this is a pointer to the location on the adapter of the **SHARED_MEMORY_INFO_STRUCT** structure as defined in the file HDLCAPI.H. **ptr_HDLC_Tx_stat_el_cfg_struct** - this is a pointer to the location on the adapter of the **HDLC_TX_STATUS_EL_CFG_STRUCT** (Transmit Status Element Configuration structure) as defined in the file HDLCAPI.H. **ptr_HDLC_Rx_stat_el_cfg_struct** - this is a pointer to the location on the adapter of the **HDLC_RX_STATUS_EL_CFG_STRUCT** (Receive Status Element Configuration structure) as defined in the file HDLCAPI.H.

ENABLE_HDLC_COMMUNICATIONS (0x24)

This command allows the HDLC code to transmit frames and to respond to incoming frames.

DTR and RTS will be automatically raised once the **ENABLE_HDLC_COMMUNICATIONS** command has been issued, unless otherwise configured in the **modem_config_options** (see the **SET_HDLC_CONFIGURATION** command).

Note that after down loading the code to the adapter, the communications may not be enabled until the initial **SET_HDLC_CONFIGURATION** command has been issued.

Control Block values to be set on entry	
command	Set to 0x24.
buffer_length	Set to 0x00.

Control Block values set on exit	
return_code	0x00 - The command was performed successfully. 0x21 - Communications are already enabled. 0x22 - The HDLC configuration must be set by using the **SET_HDLC_CONFIGURATION** command before enabling communications.

DISABLE_HDLC_COMMUNICATIONS (0x25)

This command causes the HDLC code to no longer send frames or to respond to incoming frames. No HDLC state change is performed.

Note that this command must be executed before using the **SET_HDLC_CONFIGURATION** command.

Control Block values to be set on entry	
command	Set to 0x25.
buffer_length	Set to 0x00.

Control Block values set on exit	
return_code	0x00 - The command was performed successfully. 0x21 - Communications are already disabled.

HDLC_CONNECT (0x26)

The HDLC code will attempt to enter the Asynchronous Balanced Mode (ABM), thereby permitting transfer of Information frames.

Incoming SABM frames will illicit a UA response and the HDLC code will also attempt to initialize the link by issuing SABM/SABME frames of its own (unless otherwise configured in the **HDLC_protocol_options** by the **SET_HDLC_CONFIGURATION** command).

Control Block values to be set on entry	
command	Set to 0x26.
buffer_length	Set to 0x00.

Control Block values set on exit	
return_code	0x00 - The command was performed successfully. 0x21 - Communications must be enabled before connecting. 0x22 - The link is already in the ABM. Note that the HDLC code may be configured to permit a SABM command to be issued while in the ABM (see the **HDLC_API_options** in the **SET_HDLC_CONFIGURATION** command).

HDLC_DISCONNECT (0x27)

This command causes link disconnection by the issuing of a DISC frame.

Once this command has been issued, a **HDLC_CONNECT** command must be issued to permit the link to re-enter the ABM. However, the HDLC code may be configured so as to automatically re-enter the ABM on reception of a SABM command (see the **HDLC_protocol_options** in the **SET_HDLC_CONFIGURATION** command).

Control Block values to be set on entry	
command	Set to 0x27.
buffer_length	Set to 0x00.

Control Block values set on exit	
return_code	0x00 - The command was performed successfully. 0x21 - Communications must be enabled before disconnecting. 0x22 - The link is not in the ABM and so can not be disconnected.

READ_HDLC_LINK_STATUS (0x28)

This command return the HDLC link status as well as other useful operational information.

The structure for returning the link status information is defined in the file HDLCAPI.H as the **HDLC_LINK_STATUS_STRUCT**.

Control Block values to be set on entry	
command	Set to 0x28.
buffer_length	Set to 0x00.

Control Block values set on exit	
return_code	0x00 - The command was performed successfully.
buffer_length	Set to the size of the **HDLC_LINK_STATUS_STRUCT**.
data	The **HDLC_LINK_STATUS_STRUCT** is at offset 0x00 of the mailbox data area. The structure members are as follows: **HDLC_link_status** - set to 0x00 if the link if disconnected. set to 0x01 if the link is in the ABM. **modulus_type** - set to the modulus type configured (8 or 128). Note that if the HDLC code is configured for automatic modulus detection (see the **HDLC_protocol_options** in the **SET_HDLC_CONFIGURATION** command), then the **modulus_type** will be displayed as 0x00 until a SNRM or SNRME command is received from the remote station. **no_I_frms_for_app** - the number of Information frames currently available for reception by the application. **receiver_status** - set to 0x01 if the receiver is currently disabled due to flow control restrictions. **LAPB_state** - reserved. **rotating_SUP_frm_count** - this value is incremented on the reception of a Supervisory frame and is useful for monitoring link activity. The **rotating_SUP_frm_count** rotates between a value of 0x00 and 0xFF.

READ_HDLC_OPERATIONAL_STATS (0x29)

Retrieve the operational statistics for this HDLC link.

The structure for returning the operational statistics information is defined in the file HDLCAPI.H as the **HDLC_OPERATIONAL_STATS_STRUCT**.

Control Block values to be set on entry	
command	Set to 0x29.
buffer_length	Set to 0x00.

Control Block values set on exit	
return_code	0x00 - The command was performed successfully.
buffer_length	Set to the size of the **HDLC_OPERATIONAL_STATS_STRUCT**.
data	The **HDLC_OPERATIONAL_STATS_STRUCT** is at offset 0x00 of the mailbox data area. Most of the included statistics are self-explanatory. Additional details pertaining to selected structure members are as follows: **I_frames_Tx_ack_count** - the total number of Information frames transmitted and acknowledged by the remote station. **I_frm_Tx_throughput** - the current transmit throughput (bits per second). This throughput computation is started on transmission of the first frame and may be restarted by performing a **FLUSH_HDLC_OPERATIONAL_STATS** command. The throughput value will only be valid if the transmit throughput statistics has been enabled (see the **HDLC_statistics_option** in the **SET_HDLC_CONFIGURATION** command). **no_ms_for_HDLC_Tx_thruput_comp** - the number of milliseconds used in calculating the transmit throughput. **I_frames_retransmitted_count** - the number of Information frames which had to be re-transmitted due to the original frame not being acknowledged by the remote station.

Control Block values set on exit (continued)	
data	**I_frms_not_Tx_lgth_err_count** - the number of Information frames which were not transmitted (and discarded) due to the outgoing frame length being incorrectly set to zero or to a value greater than the defined **max_HDLC_I_field_length** (see the **SET_HDLC_CONFIGURATION** command). **Tx_I_frms_disc_st_chg_count** - the number of Information frames not transmitted (and discarded) due to the link being disconnected while outgoing I-frames were queued and awaiting transmission. **I_frm_Rx_throughput** - the current receive throughput (bits per second). This throughput computation is started on reception of the first frame and may be restarted by performing a **FLUSH_HDLC_OPERATIONAL_STATS** command. The throughput value will only be valid if the receive throughput statistics has been enabled (see the **HDLC_statistics_option** in the **SET_HDLC_CONFIGURATION** command). **I_frms_Rx_too_long_count** - the number of Information frames which were received but discarded due to the frame length being greater than the defined **max_HDLC_I_field_length** (see the **SET_HDLC_CONFIGURATION** command). Note that this error will result in the occurrence of a Frame Reject condition if the link is in the ABM. **I_frms_Rx_seq_err_count** - the number of Information frames received which were considered to be out of sequence. This sequence error will result in a Reject frame being transmitted. **Rx_frm_shorter_32_bits_count** - the number of incoming frames discarded as the frame length was less than 32 bits in length. **Rx_I_fld_Sup_Unnum_frm_count** - the number of incoming frames discarded as the frame contained an information field which was not permitted. Note that this error will result in the occurrence of a Frame Reject condition if the link is in the ABM. **Rx_frms_invld_HDLC_addr_count** - the number of incoming frames discarded as the frame contained an invalid HDLC address.
data	**Rx_invld_HDLC_ctrl_fld_count** - the number of incoming frames discarded as the frame contained a control field which is invalid or not supported. Note that this error will result in the occurrence of a Frame Reject condition if the link is in the ABM. **Rx_FRMR_frms_discard_count** - the number of incoming FRMR frames not stored due to buffering limitations. **Rx_UI_frms_discard_count** - the number of incoming UI frames not stored due to buffering limitations. **UI_frms_Rx_invld_lgth_count** - the number of incoming UI frames not stored as the frame was greater than **MAX_LGTH_UI_DATA** bytes (see the file HDLCAPI.H).

FLUSH_HDLC_OPERATIONAL_STATS (0x2A)

Reset all the HDLC operational statistics to zero.

Control Block values to be set on entry	
command	Set to 0x2A.
buffer_length	Set to 0x00.

Control Block values set on exit	
return_code	0x00 - The command was performed successfully.

SET_HDLC_BUSY_CONDITION (0x2B)

The HDLC code has built-in flow control mechanisms in that an RNR frame will be sent to the remote station if all the Information frame receive buffers are occupied. This command allows the application to manually intervene in this process by forcing the HDLC code to issue an RNR frame even though the I-frame receive buffers are not full.

Control Block values to be set on entry	
command	Set to 0x2B.
buffer_length	Set to 0x01.
data	The required 'busy' status is indicated the byte at offset 0x00 of the mailbox data area as follows: If the byte is set to 0x00, then the HDLC code may exit the 'busy' status and the remote station should be notified that it may once again transmit I-frames. If the byte is set to 0x01, then the HDLC code must enter the 'busy' status and the remote station should be notified that it may not transmit I-frames.

Control Block values set on exit	
return_code	0x00 - The command was performed successfully. 0x22 - The link is not in the ABM.

SEND_UI_FRAME (0x2C)

Transmit an HDLC Unnumbered Information (UI) frame.

The structure for formatting outgoing UI frames is defined in the file HDLCAPI.H as the **UI_FRAME_STRUCT**.

Control Block values to be set on entry	
command	Set to 0x2C.
buffer_length	Set to the number of bytes in the UI frame structure and is dependant on the number of bytes of UI data to be sent. The minimum **buffer_length** that may be defined is 3 bytes - one byte of data plus two bytes for the frame header. The maximum **buffer_length** that may be defined is equal to the **MAX_LGTH_UI_DATA** plus two bytes for the frame header.
PF_bit	Set to 0x00 if the P/F bit is to be reset in the outgoing frame. Set to 0x01 if the P/F bit is to be set in the outgoing frame.
data	The **UI_FRAME_STRUCT** is at offset 0x00 of the mailbox data area. The structure members are as follows: **HDLC_address** - the HDLC address to be used in the outgoing UI frame. **data** - the UI data to be transmitted. The maximum number of bytes of UI data is defined as the **MAX_LENGTH_UI_DATA** in the file HDLCAPI.H.

Control Block values set on exit	
return_code	0x00 - The command was performed successfully. 0x21 - Communications must be enabled before transmitting a UI frame. 0x23 - The UI frame buffer is currently in use. This command should be repeated. 0x24 - The passed **buffer_length** parameter is invalid.

SET_HDLC_INTERRUPT_TRIGGERS (0x30)

Set the occurrences which will cause the HDLC code to trigger a hardware interrupt on the PC. See the section on "Interrupt Usage" later in this manual for further details.

The structure for setting these interrupt triggers is defined in the file HDLCAPI.H as the **HDLC_INT_TRIGGERS_STRUCT**.

Control Block values to be set on entry	
command	Set to 0x30.
buffer_length	Set to the size of the **HDLC_INT_TRIGGERS_STRUCT**.

Control Block values to be set on entry	
data	The **HDLC_INT_TRIGGERS_STRUCT** is at offset 0x00 of the mailbox data area. The structure members are as follows: **HDLC_interrupt_triggers** - defines the interrupt triggers as follows: bit 0 - the receive interrupt bit. If this bit is set, then an interrupt will be triggered if there is an incoming Information frame available for reception by the application. bit 1 - the transmit interrupt bit. If this bit is set, then an interrupt will be triggered if there is a transmit buffer available in which the application may place an outgoing Information frame. bit 2 - the 'command complete' interrupt bit. If this bit is set, then an interrupt will be triggered on completion of an interface command, i.e. when the **opp_flag** in the mailbox has been reset. bit 3 - the millisecond timer interrupt bit. If this bit is set, then an interrupt will be triggered by the adapter at a specified millisecond interval. bit 4 - the global exception condition interrupt bit. If this bit is set, then an interrupt will be triggered on the occurrence of a global exception condition such as a modem status change or a critical line trace error. The details of this interrupt may be established by using the **READ_GLOBAL_EXCEPTION_CONDITION** command. bit 5 - the HDLC exception condition interrupt bit. If this bit is set, then an interrupt will be triggered on the occurrence of an HDLC exception condition. The details of this interrupt may be established by using the **READ_HDLC_EXCEPTION_CONDITION** command. bit 6 - reserved for later use. bit 7 - the trace interrupt bit. If this bit is set, then an interrupt will be triggered by the adapter when trace data is available for reception by the application.
data	**IRQ** - the IRQ to be used by the adapter. Valid IRQs are 3, 4, 5, 7, 10, 11 and 12. **interrupt_timer** - the specified millisecond timer interrupt interval (valid when the timer interrupt has been enabled). Valid value are from 0 to 60000 milliseconds.
return_code	0x00 - The command was performed successfully. 0x22 - The **buffer_length** set on entry was not set to the size of the **HDLC_INT_TRIGGERS_STRUCT.** 0x23 - The selected IRQ is invalid. 0x24 - The specified millisecond timer interrupt interval is invalid.

READ_HDLC_INTERRUPT_TRIGGERS (0x31)

Read the current HDLC interrupt trigger settings.

The structure for reading these interrupt triggers is defined in the file HDLCAPI.H as the **HDLC_INT_TRIGGERS_STRUCT**.

Control Block values to be set on entry	
command	Set to 0x31.
buffer_length	Set to 0x00.

Control Block values set on exit	
return_code	0x00 - The command was performed successfully.
buffer_length	Set to the size of the **HDLC_INT_TRIGGERS_STRUCT**.
data	The returned **HDLC_INT_TRIGGERS_STRUCT** is at offset 0x00 of the mailbox data area.

4. THE SHARED MEMORY AREA INFORMATION STRUCTURE

There are a number of structures that may be used by the application for establishing the HDLC status, the current modem status, the application interrupt status, etc. without using an interface command. The physical offset on the adapter of the reference structure, the **SHARED_MEMORY_INFO_STRUCT** structure (HDLCAPI.H) is found by reading the **ptr_shared_mem_info_struct** from the **HDLC_CONFIGURATION_STRUCT** after using the **READ_HDLC_CONFIGURATION** command.

The details of the **SHARED_MEMORY_INFO_STRUCT** structure and other associated structures are given below:

SHARED_MEMORY_INFO_STRUCT	
global_info_struct	A structure of the format **GLOBAL_INFORMATION_STRUCT** which provides global information such as the current modem input status.
HDLC_info_struct	A structure of the format **HDLC_INFORMATION_STRUCT** which provides HDLC status information.
interrupt_info_struct	A structure of the format **INTERRUPT_INFORMATION_STRUCT** which is used for SDLA/application interrupt handling.
FT1_info_struct	A structure of the format **FT1_INFORMATION_STRUCT** which provides information pertaining to a S508/FT1 adapter.

GLOBAL_INFORMATION_STRUCT	
global_status	The global adapter status. Reserved for Sangoma use.
modem_status	Indicates the status of the DCD and CTS input leads as follows: If bit 3 is reset, then DCD is set low. If bit 3 is set, then DCD is set high. If bit 5 is reset, then CTS is set low. If bit 5 is set, then CTS is set high.
global_exception_conditions	If this byte is not set to 0x00, then a global exception condition has occurred. A **READ_GLOBAL_EXCEPTION_COMMAND** should be used to recover the details of this exception condition and also to clear this exception condition.

HDLC_INFORMATION_STRUCT	
HDLC_status	If bit 0 is set to 0, then the link is disconnected. If bit 0 is set to 1, then the link is connected (ABM).
HDLC_excep_frms_Rx	If this byte is not set to 0x00, then an exception condition has been caused by the reception of an Unnumbered frame. A **READ_HDLC_EXCEPTION_COMMAND** should be used to recover the details of this exception condition and also to clear this exception condition.
HDLC_excep_frms_Tx	If this byte is not set to 0x00, then an exception condition has been caused by the transmission of an Unnumbered frame. A **READ_HDLC_EXCEPTION_COMMAND** should be used to recover the details of this exception condition and also to clear this exception condition.
HDLC_excep_miscellaneous	If this byte is not set to 0x00, then an exception condition has been caused by the HDLC link entering the ABM or the expiry of an HDLC timer or retry counter. A **READ_HDLC_EXCEPTION_COMMAND** should be used to recover the details of this exception condition and also to clear this exception condition.
rotating_SUP_frm_count	This value is incremented on the reception of a Supervisory frame and is useful for monitoring link activity. The **rotating_SUP_frm_count** rotates between a value of 0x00 and 0xFF.
LAPB_status **internal_HDLC_status**	Reserved for internal use.

The format of the **INTERRUPT_INFORMATION_STRUCT** is discussed in the Section "Using Interrupts" below.

5. TRANSMITTING INFORMATION FRAMES

Structures Used for Information Frame Transmission

Information frames are transmitted by using the Transmit Status Elements. The details of the location and number of these elements are found in the

HDLC_TX_STAT_EL_CFG_STRUCT as defined in the file HDLCAPI.H. The physical offset of this structure on the adapter is found by reading the **ptr_HDLC_Tx_stat_el_cfg_struct** from the **HDLC_CONFIGURATION_STRUCT** after using the **READ_HDLC_CONFIGURATION** command.

The **HDLC_TX_STAT_EL_CFG_STRUCT** structure is as follows:

HDLC_TX_STAT_EL_CFG_STRUCT	
number_Tx_status_elements	The number of Transmit Status Elements.
base_addr_Tx_status_elements	The base address of the Transmit Status Element list.
next_Tx_Status_element_to_use	The address of the next Transmit Status Element to be used by the application. In general, this parameter is only read by the application on code on initialization and is reset when the **SET_HDLC_CONFIGURATION** and **ENABLE_HDLC_COMMUNICATIONS** commands are used.

The actual Transmit Status Elements are defined as the **HDLC_I_FRM_TX_STATUS_EL_STRUCT** in the file HDLCAPI.H. The details of the Transmit Status Elements are as follows:

HDLC_I_FRM_TX_STATUS_EL_STRUCT	
opp_flag	A flag set to 0x01 by the application when an Information frame is to be transmitted and set to 0x00 by the HDLC code when the Transmit Status Element is available for use.
I_frame_length	The length of the data in the Information frame to be transmitted.
P_bit	The setting of the P-bit in the I-frame to be transmitted.
reserved_1 reserved_2	Reserved for internal use and **must not be altered by the application**.
ptr_data_bfr	A pointer to the actual Information frame data buffer to be used. The application must copy the I-frame data to this buffer.

Non-interrupt Driven Information Frame Transmissions

The basic methodology used for I-frame transmission is as follows: The Transmit Status Elements are used sequentially by the application as I-frames need to be transmitted. The HDLC code running on the adapter follows the same sequential pattern, and if a Transmit Status Element has been used by the application (**opp_flag** set to 0x01), then an outgoing Information frame is formatted and transmitted. The Transmit Status Element is freed up by the HDLC code (**opp_flag** set to 0x00) once the transmitted I-frame has been acknowledged by the remote station.

The steps used for non-interrupt driven I-frame transmission are as follows:

1) The physical offset of the **HDLC_TX_STAT_EL_CFG_STRUCT** structure on the adapter is found by reading the **ptr_HDLC_Tx_stat_el_cfg_struct** from the **HDLC_CONFIGURATION_STRUCT** after using the **READ_HDLC_CONFIGURATION** command. The location of the first (next) Transmit Status Element is found from this structure.

2) The **opp_flag** of the next Transmit Status Element to be used is examined by the application. Note that on code initialization, the next sequential Transmit Status Element is set to the **base_addr_Tx_status_elements** (see the **HDLC_TX_STATUS_EL_CFG_STRUCT).** If this flag is set to 0x00, then the Transmit Status Element is available for use. It is important to note that if the **opp_flag** is not set to 0x00, then the application may not transmit an I-frame. Also, the application must not attempt to use another Transmit Status Element in a non-sequential manner. So, the application may re-attempt I-frame transmission using the same Transmit Status Element after a short delay.

3) If the **opp_flag** is set to 0x00, then the application fills in the **I_frame_length** and **P-bit** parameters. The **ptr_data_bfr** value is read by the application and the I-frame data to be transmitted is copied into this buffer on the adapter. When writing transmit data to the adapter, the following logic is used:

a) The 8k memory window is set and a physical pointer to the base address of the transmit data buffer is calculated according to the **ptr_data_bfr** .

b) The transmit data is written to the adapter until either:

i) all the data has been written

ii) the upper boundary of the current 8k memory segment has been reached

c) If not all the data has been written, then the 8k memory segment access is adjusted and step b) is repeated.

4) The **opp_flag** is set to 0x01 by the application. The HDLC code will now format and transmit the Information frame.

5) The application sets up to examine the next sequential Transmit Status Element. For example if the **base_addr_Tx_status_elements** and the **number_Tx_status_elements** read from the **HDLC_TX_STATUS_EL_CFG_STRUCT** are 0x0000F140 and 0x0004 respectively, then the consecutive pointers to the Transmit Status Elements would be 0x0000F140, 0x0000F150, 0x0000F160, 0x0000F170, 0x0000F140, 0x0000F150, etc.

Important - the next sequential Transmit Status Element to be used is set to the **base_addr_Tx_status_elements** (see the **HDLC_TX_STATUS_EL_CFG_STRUCT)** whenever the application uses the **SET_HDLC_CONFIGURATION** or the **ENABLE_HDLC_COMMUNICATIONS** commands.

Interrupt Driven Information Frame Transmissions

Most applications use non-interrupt, polling routines for I-frame transmission. However, the adapter may be set to trigger a transmit interrupt when the next sequential Transmit Status Element is available for use by the application. This type of interrupt is generally only used in the following situation:

The application attempts to transmit an I-frame and finds that the **opp_flag** is not set to 0x00 in the next sequential Transmit Status Element. So, instead of looping and waiting for the HDLC code to reset the **opp_flag** so that the frame may be queued on the adapter, the application enables transmit interrupts (see the **SET_HDLC_INTERRUPT_TRIGGERS** command and the 'Interrupt Usage' section of this manual). When this interrupt occurs, then the Transmit Status Element has become available and the frame may be successfully transmitted using the same logic described for non-interrupt driven transmissions. The application then disables transmit interrupts.

6. RECEIVING INFORMATION FRAMES

Structures Used for Receiving Information Frames

Information frames are received by using the Receive Status Elements. The details of the location and number of these elements are found in the

HDLC_RX_STATUS_EL_CFG_STRUCT as defined in the file HDLCAPI.H. The physical offset of this structure on the adapter is found by reading the **ptr_HDLC_Rx_stat_el_cfg_struct** from the **HDLC_CONFIGURATION_STRUCT** after using the **READ_HDLC_CONFIGURATION** command.

The **HDLC_RX_STATUS_EL_CFG_STRUCT** structure is as follows:

HDLC_RX_STATUS_EL_CFG_STRUCT	
number_Rx_status_elements	The number of Receive Status Elements.
base_addr_Rx_status_elements	The base address of the Receive Status Element list.
next_Rx_Status_element_to_use	The address of the next Receive Status Element to be used by the application. In general, this parameter is only read by the application on code on initialization and is reset when the **SET_HDLC_CONFIGURATION** and **ENABLE_HDLC_COMMUNICATIONS** commands are used.
base_addr_Rx_buffer	The base address of the actual I-frame data receive buffer area on the adapter.
end_addr_Rx_buffer	The end address of the actual I-frame data receive buffer area on the adapter.

The actual Receive Status Elements are defined as the **HDLC_I_FRM_RX_STATUS_EL_STRUCT** in the file HDLCAPI.H. The details of the Receive Status Elements are as follows:

HDLC_I_FRM_RX_STATUS_EL_STRUCT	
opp_flag	A flag set to 0x01 by the HDLC code when an Information frame has been received and made available for the application and set to 0x00 by the application when the Information frame has been processed.
I_frame_length	The length of the data in the received Information frame.
P_bit	The setting of the P-bit in the received I-frame.
reserved_1 reserved_2	Reserved for internal use and **must not be altered by the application**.
ptr_data_bfr	A pointer to the actual Information frame data buffer associated with this incoming frame. The application must copy the I-frame data from this buffer.

Non-interrupt Driven Reception of Information Frames

The steps used for non-interrupt driven I-frame reception are as follows:

1) The physical offset of the **HDLC_RX_STAT_EL_CFG_STRUCT** structure on the adapter is found by reading the **ptr_HDLC_Rx_stat_el_cfg_struct** from the **HDLC_CONFIGURATION_STRUCT** after using the **READ_HDLC_CONFIGURATION** command. The location of the first (next) Receive Status Element is found from this structure.

2) The **opp_flag** of the next Receive Status Element to be used is examined by the application. Note that on code initialization, the next sequential Receive Status Element is set to the **base_addr_Rx_status_elements** (see the **HDLC_RX_STATUS_EL_CFG_STRUCT).** If this flag is set to 0x01, then an I-frame has been received by the HDLC code and the application should process this frame. It is important to note that if the **opp_flag** is not set to 0x01, then no I-frames are currently available. Also, the application must not attempt to use another Receive Status Element in a non-sequential manner. So, the application should again check the **opp_flag** in the same Receive Status Element after a short delay.

3) If the **opp_flag** is set to 0x01, then the application reads the **I_frame_length**, **P_bit** and **ptr_data_bfr** parameters. The application then uses the **I_frame_length** and **ptr_data_bfr** values to read the actual I-frame data from the adapter into a local buffer on the PC. When reading this data from the adapter, the following logic is used:

a) The 8k memory window is set and a physical pointer to the base address of the receive data buffer is calculated according to the **ptr_data_bfr** .

b) The data is read until either:

i) all the data has been read

ii) the upper boundary of the current 8k memory segment has been reached

iii) the end address of the receive buffer has been reached

c) If not all the data has been read, then the 8k memory segment access is adjusted and step b) is repeated.

It is important to note that the receive data buffer is a rotational buffer and so the base and end addresses of this buffer are made available to the application in the **HDLC_RX_STATUS_EL_CFG_STRUCT** to facilitate data reception.

4) The **opp_flag** is set to 0x00 by the application to free up the Receive Status Element.

.

5) The application sets up to examine the next sequential Receive Status Element. For example if the **base_addr_Rx_status_elements** and the **number_Rx_status_elements** read from the **HDLC_RX_STATUS_EL_CFG_STRUCT** are 0x0000F1A0 and 0x0004 respectively, then the consecutive pointers to the Receive Status Elements would be 0x0000F1A0, 0x0000F1B0, 0x0000F1C0, 0x0000F1D0, 0x0000F1A0, 0x0000F1B0, etc.

Important - the next sequential Receive Status Element to be used is set to the **base_addr_Rx_status_elements** (see the **HDLC_RX_STATUS_EL_CFG_STRUCT)** whenever the application uses the **SET_HDLC_CONFIGURATION** or the **ENABLE_HDLC_COMMUNICATIONS** commands.

Interrupt Driven Reception of Information Frames

Interrupt and non-interrupt driven reception of I-frames both use the same processing methodology. However, for interrupt driven frame reception, use the **SET_HDLC_INTERRUPT_TRIGGERS** command to enable receive interrupts (see the **SET_HDLC_INTERRUPT_TRIGGERS** command and the 'Interrupt Usage' section of this manual). This will cause the HDLC code to issue an IRQ to the application when an Information frame has been received and made available for the application in the next sequential Receive Status Element. Once the application has received this IRQ, the I-frame should be processed as described for non-interrupt driven frame reception.

7. INTERRUPT USAGE

Once the interrupt vector and the PIC are initialized on the PC, the S508 adapter must be configured so as to generate interrupts to the PC. This is done by writing 0x16 out to the adapter I/O port base address. Thereafter,

the application informs the HDLC code of the required interrupt configuration by using the **SET_HDLC_INTERRUPT_TRIGGERS** command.

When an interrupt occurs on the PC, the interrupt handler should examine

the **interrupt_type** byte in the **INTERRUPT_INFORMATION_STRUCT** on the adapter. This byte will allow the application to determine the cause of the interrupt and to take appropriate action. Note that only one type of interrupt can occur at a time. For example, the adapter will not trigger both a 'receive' and 'command complete' interrupt at the same time. Details of the **interrupt_type** byte are found in description of the **INTERRUPT_INFORMATION_STRUCT** structure below.

To acknowledge the interrupt and to permit the next interrupt to be triggered, the interrupt handler should:

- reset the PIC on the PC
- reset the the **interrupt_type** byte in the **INTERRUPT_INFORMATION_STRUCT** (i.e., set this byte to 0x00).

No hardware IRQ reset is required on the S508 adapter itself.

Once the interrupts have been enabled by using the **SET_HDLC_INTERRUPT_TRIGGERS** command, the interrupts may be dynamically disabled and enabled by using the **interrupt_permission** byte in the **INTERRUPT_INFORMATION_STRUCT**. This functionality is very useful as can be seen from the following scenario:

> Assume that the **SET_HDLC_INTERRUPT_TRIGGERS** command is used to enable both receive and transmit interrupts. The application then immediately disables transmit interrupts by resetting the appropriate bit in the **interrupt_permission** byte. If this is not done, then the HDLC code will constantly interrupt the application as there are I-frame transmit buffers available for use. Now the application passes I-frames to the adapter for transmission, but eventually the adapter runs out of transmit buffer space (the **opp_flag** of the next Transmit Status Element to be used is not set to 0x00). Instead of the application constantly examining the Transmit Status Element to wait for the **opp_flag** to be reset, transmit interrupts are now enabled by setting the appropriate bit in the **interrupt_permission** byte. The adapter will then cause an interrupt when this transmit buffer becomes available for use. Once this interrupt occurs, then the application once again disables transmit interrupts by using the **interrupt_permission** byte.

The details of the **interrupt_permission** byte are found in the table below.

INTERRUPT_INFORMATION_STRUCT	
interrupt_type	Indicates the type of interrupt which has occurred as follows: 0x01 - there is an incoming Information frame available for reception by the application. 0x02 - there is a transmit buffer available in which the application may place an outgoing Information frame. 0x04 - the interface command has been completed, i.e, the **opp_flag** in the mailbox has been reset. 0x08 - the timer interrupt, i.e, the specified millisecond timer interval has expired. 0x10 - a global exception condition has occurred. The details of this interrupt may be established by using the **READ_GLOBAL_EXCEPTION_CONDITION** command. Note that this exception condition will only be cleared by the use of this interface command. 0x20 - an HDLC exception condition has occurred. The details of this interrupt may be established by using the **READ_HDLC_EXCEPTION_CONDITION** command. Note that this exception condition will only be cleared by the use of this interface command. 0x40 - reserved. 0x80 - trace data is available for reception by the application.
interrupt_permission	The interrupt bit map for the **interrupt_permission** byte is the same as that used for the **SET_HDLC_INTERRUPT_TRIGGERS** command, i.e, bit 0 is used for frame reception bit 1 is used for frame transmission bit 2 is used for command completions bit 3 is used for timer interrupts bit 4 is used for global exception conditions bit 5 is used for HDLC exception conditions bit 7 is used for line tracing If the bit is reset by the application, then an interrupt of that type will **not** occur until the bit is set again.

8. USING THE LINE TRACE

The methodology used for line tracing is very similar to that of receiving Information frames from the adapter, i.e., the application sequentially rotates through the Trace Status Elements and reads trace data from that element if the **opp_flag** is set. The details of the location and number of these trace elements are found in the **TRACE_STATUS_EL_CFG_STRUCT** as defined in the file HDLCAPI.H. The physical offset of this structure on the adapter is found by reading the **ptr_trace_status_el_cfg_struct** from the **LINE_TRACE_CONFIG_STRUCT** after using the **READ_TRACE_CONFIGURATION** command.

The **TRACE_STATUS_EL_CFG_STRUCT** structure is as follows:

TRACE_STATUS_EL_CFG_STRUCT	
number_trace_status_elements	The number of Trace Status Elements.
base_addr_trace_status_elements	The base address of the Trace Status Element list.
next_trace_element_to_use	The address of the next Trace Status Element to be used by the application. In general, this parameter is only read by the application on code on initialization or after a **FLUSH_TRACE_BUFFERS** command.
base_addr_trace_buffer	The base address of the actual trace data buffer area on the adapter.
end_addr_trace_buffer	The end address of the actual trace data buffer area on the adapter.

The actual Trace Status Elements are defined as the **TRACE_STATUS_ELEMENT_STRUCT** in the file HDLCAPI.H. The details of the Trace Status Elements are as follows:

TRACE_STATUS_ELEMENT_STRUCT	
opp_flag	A flag set to 0x01 by the HDLC code when a trace frame is available for the application and set to 0x00 by the application when the trace frame has been processed.
trace_length	The length of the trace data.
trace_type	The type of frame traced as follows: If bit 0 is reset, then this trace is of an incoming frame. If bit 0 is set, then this trace is of an outgoing frame. If bit 4 is set, then this incoming frame was an aborted frame. If bit 5 is set, then this incoming frame was received with a CRC error. If bit 6 is set, then this incoming frame was received with an overrun error.
trace_time_stamp	A millisecond time stamp associated with this traced frame. This time stamp rotates from 0x0000 to 0xFFFF.

TRACE_STATUS_ELEMENT_STRUCT	
ptr_data_bfr	A pointer to the actual trace data buffer. This trace data includes the complete HDLC frame (HDLC address, control field, data and CRC bytes). For an outgoing frame, the two CRC bytes are both shown as 0xFF. **Important** - if this data pointer is set to NULL (0x00), then the trace data could not be stored due to buffer limitations. However, the rest of the parameters in the Trace Status Element are valid.

Non-interrupt Driven Line Tracing

The steps used for line tracing as follows:

1) The physical offset of the **TRACE_STAT_EL_CFG_STRUCT** structure on the adapter is found by reading the **ptr_trace_stat_el_cfg_struct** from the **LINE_TRACE_CONFIG_STRUCT** after using the **READ_TRACE_CONFIGURATION** command. The location of the first (next) Trace Status Element is found from this structure.

2) The **opp_flag** of the next Trace Status Element to be used is examined by the application. Note that on code initialization, the next sequential Trace Status Element is set to the **base_addr_trace_status_elements** (see the **TRACE_STATUS_EL_CFG_STRUCT).** If this flag is set to 0x01, then an trace data is available for the application. It is important to note that if the **opp_flag** is not set to 0x01, then trace data is currently available. Also, the application must not attempt to use another Trace Status Element in a non-sequential manner. So, the application should again check the **opp_flag** the same Trace Status Element after a short delay.

3) If the **opp_flag** is set to 0x01, then the application reads the **trace_length**, **trace_type**, **trace_time_stamp** and **ptr_data_bfr** parameters. The application then uses the **trace_length** and **ptr_data_bfr** values to read the actual trace data from the adapter into a local buffer on the PC. It is important to note that if the **ptr_data_bfr** is set to NULL (0x00), then the trace data could not be stored due to buffer limitations. However, the rest of the parameters in the Trace Status Element are valid. When reading this data from the adapter, the following logic is used:

a) The 8k memory window is set and a physical pointer to the base address of the trace data buffer is calculated according to the **ptr_data_bfr** .

b) The data is read until either:

i) all the data has been read

ii) the upper boundary of the current 8k memory segment has been reached

iii) the end address of the trace buffer has been reached

c) If not all the trace data has been read, then the 8k memory segment access is adjusted and step b) is repeated.

It is important to note that the trace data buffer is a rotational buffer and so the base and end addresses of this buffer are made available to the application in the **TRACE_STATUS_EL_CFG_STRUCT** to facilitate data reception.

4) The **opp_flag** is set to 0x00 by the application to free up the Trace Status Element.

.

5) The application sets up to examine the next sequential Trace Status Element. For example if the **base_addr_trace_status_elements** and the **number_trace_status_elements** read from the **TRACE_STATUS_EL_CFG_STRUCT** are 0x0000B020 and 0x0004 respectively, then the consecutive pointers to the Trace Status Elements would be 0x0000B020, 0x0000B030, 0x0000B040, 0x0000B060, 0x0000B020, 0x0000B040, etc.

Important - the next sequential Trace Status Element to be used is set to the **base_addr_trace_status_elements** (see the **TRACE_STATUS_EL_CFG_STRUCT)** whenever the application uses the **FLUSH_TRACE_BUFFERS** commands.

Interrupt Driven Line Tracing

Interrupt and non-interrupt driven line tracing both use the same processing methodology. However, for interrupt driven tracing, use the **SET_HDLC_INTERRUPT_TRIGGERS** command to enable the line trace interrupt (see the **SET_HDLC_INTERRUPT_TRIGGERS** command and the 'Interrupt Usage' section of this manual). This will cause the board-level code to issue an IRQ to the application when trace data is available for the application in the next sequential Trace Status Element. Once the application has received this IRQ, the trace element should be processed as described for non-interrupt driven line tracing.

9. MS-DOS Code

General

Software support for HDLC under MS-DOS consists of the following modules:

- **ULOAD.EXE** loads the HDLC microcode onto the S508 adapter, configures it, runs through a self test and starts the program.

- **HDLC.508** is the run time on board HDLC module loaded by ULOAD.EXE.

ULOAD.EXE

ULOAD.EXE is a PC-DOS program and has the following command line syntax:

ULOAD -c< CODE MODULE> -p<I/O PORT> -m<MEMORY>

where:

CODE MODULE — is the run time HDLC module. This argument must be included, and the module **HDLC.508** is the one generally listed.

I/O PORT — is the I/O port base address of the SDLA card. Valid values are 0x250, 0x270, 0x280, 0x300, 0x350, 0x360, 0x380 and 0x390. Note that the selection of this parameter must match the jumper settings on the adapter.

MEMORY — is the location of the start of the 8k shared memory window of the SDLA card. Valid values are Ax, Cx, Dx, Ex, where x = 0, 2, 4, 6, 8, A, C, E. For example CA means that the shared memory area is from CA000 to CBFFF.

For example:

```
ULOAD -cHDLC.508 -mC8 -p300
```

ULOAD will perform a system test, read the CODE file and load and configure the adapter. If ULOAD does not execute successfully, an error message will be displayed and an exit code will be returned.

HDLC.508

This is the HDLC support code which is loaded onto the card, executes and establishes the communications. HDLC.508 is **NOT** a PC-DOS program and is not executable under DOS.

10. The Shared Memory interface to the Sangoma Cards

Accessing Memory

The cards are operated by reading and writing bytes and structures to and from positions within the shared memory window. Usually this will all be taken care of by the operating system and driver, but if writing your own driver, the following information is relevant.

The size of this memory window is 8192 (8k or 0x2000) bytes, and it may be dynamically moved to access any particular 8k segment on the board. In order for the application to interface with the HDLC code, it will be necessary to move the shared memory window, and to calculate the offset of the required data within this window.

Note that all memory addresses referred to in this manual are absolute physical addresses on the board.

The 8k memory window is adjusted by writing a specified value out to the port address

adapter base I/O port + 2

as follows:

Value to be written to I/O base port + 2	Adapter physical memory accessed
0x00	0x00000 - 0x01FFF
0x01	0x02000 - 0x03FFF
0x02	0x04000 - 0x05FFF
0x03	0x06000 - 0x07FFF
0x04	0x08000 - 0x09FFF
0x05	0x0A000 - 0x0BFFF
0x06	0x0C000 - 0x0DFFF
0x07	0x0E000 - 0x0FFFF
0x08	0x10000 - 0x11FFF

0x09 0x12000 - 0x13FFF

0x0A 0x14000 - 0x15FFF

Value to be written to I/O base port + 2	Adapter physical memory accessed
0x0B	0x16000 - 0x17FFF
0x0C	0x18000 - 0x19FFF
0x0D	0x1A000 - 0x1BFFF
0x0E	0x1C000 - 0x1DFFF
0x0F	0x1E000 - 0x1FFFF
0x10	0x20000 - 0x21FFF
0x11	0x22000 - 0x23FFF
0x12	0x24000 - 0x25FFF
0x13	0x26000 - 0x27FFF
0x14	0x28000 - 0x29FFF
0x15	0x2A000 - 0x2BFFF
0x16	0x2C000 - 0x2DFFF
0x17	0x2E000 - 0x2FFFF
0x18	0x30000 - 0x31FFF
0x19	0x32000 - 0x33FFF
0x1A	0x34000 - 0x35FFF
0x1B	0x36000 - 0x37FFF
0x1C	0x38000 - 0x39FFF
0x1D	0x3A000 - 0x3BFFF
0x1E	0x3C000 - 0x3DFFF
0x1F	0x3E000 - 0x3FFFF

Since there is at least 128k of addressable memory on the S508 adapter, and more on the S514 adapter, and the window size is 8k, there are at least 16 windows. To set the window, write the window number out to the port at the base I/O address + 2. To calculate the window number, divide the physical address on the adapter by 8192 (0x2000). To calculate the physical offset within the window, find the remainder of this division.

The following is a set of example 'C' functions which may be used to set the appropriate window and

read a byte from the board. In this case, a function is provided to read a modem status byte on the S508.

```
/* offset from the base I/O port address of the port used */
/* for changing the memory window */
#define PORT_OFFSET_WINDOW      2
/* window size is 8k (0x2000) */
#define WIN_SIZE                0x2000UL
/* address of modem status interface byte */
#define ADDR_CURR_MODEM_STAT_IB 0xF004

unsigned char read_modem_status_byte(void);
unsigned char read_byte(unsigned long address);
unsigned short set_8k_window(unsigned long);

/* the base I/O port address of the adapter */
unsigned short card_port = 0x360;
/* the card was loaded at with memory arguments -mD2 */
unsigned long card_seg = 0xD2000000L;

main()
{
unsigned char modem_status_byte;

   modem_status_byte = read_modem_status_byte();
   printf("\nmodem status byte = 0x%X", modem_status_byte);

}
unsigned char read_modem_status_byte(void)
{
   return(read_byte(ADDR_CURR_MODEM_STAT_IB));
}

unsigned char read_byte(unsigned long address)
{
```

```
unsigned char far *card_ptr;
unsigned short card_off;

   card_off = set_8k_window(address);
   (unsigned long)card_ptr = card_seg | card_off;
   return(*card_ptr);
}

unsigned short set_8k_window(unsigned long address)
{
unsigned char win;

   win = (unsigned char)(address / WIN_SIZE);
   outp(card_port + PORT_OFFSET_WINDOW, win);
   return(address % WIN_SIZE);
}
```

Using the HDLC Shared Memory Control Block Structure

The HDLC interface commands listed in the file HDLCAPI.H are issued to the adapter via a special control block structure, the **HDLC_MAILBOX_STRUCT** (see the file HDLCAPI.H for structure details). The structure members are filled in by the application according to the requirements of the specific command. Once this has been completed, the application sets the **opp_flag** byte to 0x01 to instruct the HDLC code to process the command. The operation has been completed by the adapter when this **opp_flag** byte is reset to 0x00. At this stage, the **HDLC_MAILBOX_STRUCT** members such as the **return_code** have been updated by the HDLC code and may be read by the application.

Other HDLC interface functions such as sending and receiving Information frames and line tracing are performed by using special interface structures that are defined later in this manual.

The HDLC_MAILBOX_STRUCT Control Block Structure

The **HDLC_MAILBOX_STRUCT** command control block structure is found at physical address **0xE000** on the S508 adapter. The details of the elements within this structure are given in the table below.

HDLC_MAILBOX_STRUCT

opp_flag A flag set to 0x01 by the application to inform the adapter processor that an interface command is pending. This flag is in turn reset by the processor when the command has been completed.

command The command code.

buffer_length Length of the data buffer associated with this command.

return_code Result code for the interface command.

PF_bit Poll/Final bit setting in HDLC frames.

reserved Reserved for later use.

data Used for transferring data associated with the selected interface command.