[redacted]

Final Report on the Project for Development of New Protocol Hardware and Software for LSI-11 to Accommodate AUTODIN II ADCCP-HDLC, and X.25

Appendix C

Final Report on the Project
for Development of New
Protocol Hardware and Software
for LSI-11 to Accommodate
AUTODIN II ADCCP-HDLC, and X.25

Appendix C

Sponsored by:

Defense Advanced Research Projects Agency (DOD)
Order No. 4024
Under Contract No.: MDA903-81-C-0038
Issued by Dept. of Army, Defense Supply Service
Washington, Washington, D.C. 20310

Submitted to: Defense Advanced Research Projects Agency
Information Processing Techniques Office
1400 Wilson Boulevard, Arlington, VA 22209

Submitted by: Associated Computer Consultants
720 Santa Barbara Street, Santa Barbara, CA 93101
(805)963-9431

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

IF-11Q/X.25.UM.V001
September 1982

IF-11Q/X.25

USER'S MANUAL

A-1

Associated Computer Consultants
228 East Cota Street
Santa Barbara, CA 93101
(805) 963-9431

TABLE OF CONTENTS

Page

1.0 INTRODUCTION 1-1

1.1 Manual Contents 1-1
1.2 IF-11Q/X.25 System Overview 1-1
1.3 Hardware 1-2
1.4 Software 1-2

Figure 1-1 IF-11Q/X.25 System Overview 1-3

2.0 REFERENCES 2-1

2.1 Reference Documents 2-1

3.0 HARDWARE INSTALLATION 3-1

3.1 Shipping Checklist 3-1
3.2 Drawing Reference 3-1
3.3 Installation Considerations 3-2
3.4 Switch & Jumper Options 3-2
3.5 Attachment to LSI-11 System 3-3
3.6 Serial Interface Pinout 3-4

4.0 HOST DEVICE DRIVER PROGRAMMING 4-1

4.1 Communication Registers 4-1
4.2 Hardware Data Transfer 4-2

Table 4-1 IF-11Q/X.25 RCSR (Receive Control and Status Register) 4-3
Table 4-2 IF-11Q/X.25 TCSR (Transmit Control and Status Register) 4-6

4.3 Software Data Transfer 4-9

Figure 4-1 IF-11Q/X.25 Multiplexing Protocol Input Finite State Automation 4-10
Figure 4-2 IF-11Q/X.25 Multiplexing Protocol Output Finite State Automation 4-11

5.0 X.25 PROTOCOL PROGRAMMING 5-1

5.1 Implementation Notes 5-1
5.2 Message Formats 5-2
5.3 Message Contents 5-3
5.4 IF-11Q/X.25 Subsystem Queries and Responses . 5-4

Table 5-1 Correlation between ACC IF-11Q/X.25 and CCITT Packet Types 5-5
Table 5-2a ACC IF-11Q/X.25 Command Codes 5-6
Table 5-2a ACC IF-11Q/X.25 Diagnostic Query/Response Codes 5-6

Table 5-3 ACC IF-11Q/X.25 Response Message Contents 5-7
Table 5-4 ACC IF-11Q/X.25 Error Response Error Codes 5-10

6.0 MICRODIAGNOSTICS 6-1

6.1 Introduction 6-1
6.2 Display LEDs 6-2
6.3 Error Defeat Switches 6-2

APPENDICES

A.0 DRAWINGS A-1

Top Assembly, IF-11Q/X.25
ACC Drawing Number 8600111 .... A-2

Cable Assembly, Null Modem
ACC Drawing Number 8300222 .... A-5

B.0 SAMPLE DEVICE DRIVER LISTING B-1

IF-11Q/X.25

USER'S MANUAL

CHAPTER 1 - INTRODUCTION

CHAPTER 1

1.0 INTRODUCTION

1.1 Manual Contents - This manual contains all of the information needed to successfully use an IF-11Q/X.25 in an LSI-11 based computer system for X.25 protocol communication. Installation considerations and procedures are detailed. Host device driver programming is explained, and an example device driver listing is included as an appendix. The X.25 protocol data and control programming is described with all message formats and contents presented in tabular form. Information explaining the power-up microdiagnostics completes this manual.

1.2 IF-11Q/X.25 System Overview - The IF-11Q/X.25 is a microprocessor based communications front-end developed by Associated Computer Consultants (ACC). The IF-11Q/X.25 provides DMA support for LSI-11 applications which require X.25 capability. The protocol conforms to ISO HDLC specifications for a combined station operating in Asynchronous Balanced Mode (ABM), implementing options 2 (reject), and 8 (command I-frames only). X.25 frames are assembled and verified independent of host activity.

1.3 Hardware - The IF-11Q/X.25 hardware consists of an MDMA controller and an XQ/CP subsystem. The MDMA is a microcoded bit-slice DMA controller which implements a Subsystem Interface Bus which connects the XQ/CP to the LSI-11 bus. All MDMA functions are packaged on a single LSI-11 dual wide circuit card. The XQ/CP is a Zilog Z-80 based communication subsystem which has been used to implement the X.25 protocol by means of ROM-based software. The XQ/CP consists of three LSI-11 dual width circuit cards. The Interface Board (I-Board) connects the XQ/CP to the MDMA. The Memory Board (M-Board) contains the RAM and ROMs for the X.25 protocol. The Processor Board (P-Board) contains the Z-80 CPU and the serial I/O connections.

1.4 Software - The application program can send and receive "Data Frames" as well as control and sense the status of the physical link by means of "Supervisory Command Messages" (See Chapter 5). Separate "logical channels" are used for these two different kinds of information. The host device driver and the XQ/CP I/O Executive (IOX), together, use the LSI-11 bus interface hardware to implement multiple logical channels (See Chapter 4). The IF-11Q/X.25 firmware thus implements two levels of software protocols. The lower level being a multiplexing protocol and the higher level being the X.25 protocol. (See Figure 1-1.)

IF-11Q/X.25

USER'S MANUAL

CHAPTER 2 - REFERENCES

CHAPTER 2

2.0 REFERENCES

2.1 Reference Documents - The following documents will assist the user in understanding the operation of the IF-11Q/HDLC:

1. Data Communications Standards
   (International Organization for Standards)

2. XQ/CP Maintenance Manual(XQCP.MM.V001)
   Associated Computer Consultants, Santa Barbara, CA
   93101

3. Multichannel DMA Controller for LSI-11(MDMA.MM.V002)
   Associated Computer Consultants, Santa Barbara, CA
   93101

4. XQ/CP Communications Processor Software Support Monitor Manual (XQCP.SSMM.V001)
   Associated Computer Consultants, Santa Barbara, CA
   93101

IF-11Q/X.25

USER'S MANUAL

CHAPTER 3 - HARDWARE INSTALLATION

CHAPTER 3

3.0 HARDWARE INSTALLATION

3.1 Shipping Checklist - The IF-11Q/X.25 distribution package consists of the following items:

A. One MDMA Board.

B. One XQ/CP I-Board.

C. One XQ/CP M-Board.

D. One XQ/CP P-Board.

E. One Distribution Panel (RS-232C, RS-449, or MIL-STD-188-114).

F. Three (short) flat ribbon cables to connect the boards together.

G. Two (long) ribbon cables to connect the XQ/CP P-Board to the distribution panel.

H. One IF-11Q/X.25 User's Manual.

3.2 Drawing Reference - When performing hardware installation, refer to the following drawings which are included in Appendix A of this manual:

ACC Drawing No.	Title
8600108	Top Assembly, IF-11Q/X.25
8300222	Cable Configuration, Null Modem

3.3 Installation Considerations - The IF-11Q/X.25 is installed in an LSI-11 processor box or expansion chassis and requires:

A. Four contiguous dual-height LSI-1] bus slots.

B. The following DC currents (Absolute MAX):

DC Voltage	MDMA	I-board	M-board	P-board	TOTAL
+ 5V	3.0A	2.72A	3.26A	2.26A	11.24
+12V	0.0A	0.0A	0.96A	0.27A	1.23

3.4 Switch and Jumper Options - All switch and jumper options have been pre-set by the factory. The default CSR address is 0776200, and the default interrupt vectors are 0140 for input and 0144 for output. Refer to the maintenance manuals or contact the factory for other configurations.

3.5 Attachment to LSI-11 System - When attaching the IF-11Q/X.25 to the LSI-11 System, the following sequence should be followed:

A. Remove power from the entire LSI-11 system before performing the subsequent steps.

B. Install the distribution panel at the back of the LSI-11 cabinet.

C. Select 4 contiguous LSI-11 bus slots. All of the IF-11Q/X.25 boards provide LSI-11 bus DMA grant continuity and interrupt grant continuity. Care should be exercised to assure that grant continuities exist between the IF-11Q/X.25 and the LSI-11 processor module due to other boards and perhaps unoccupied LSI-11 Bus slots.

D. The 4-board IF-11Q/X.25 system comes already cabled together. DO NOT DISCONNECT THE BOARDS FROM EACH OTHER.

E. Position the connected boards over the LSI-11 Bus slots and insert each board into its bus slot.

F. Install the Serial I/O cables between the distribution panel and the XQ/CP P-board as per the top level assembly drawing in Appendix A. Special attention should be given to Pin 1 positioning as well as Port A/Port B cable orientation.

G. Power up the system and check all voltages.

H. Close the processor/expansion box.

I. Start up system.

3.6 Serial Interface Pinout - ACC drawing #8300222 (Appendix A) shows the pin layout for a null modem cable used to link together two IF-11Q/X.25 systems. Note that each IF-11Q/X.25 transmits a 9600 bps clock signal on pin 24, which can be used if an external clock is not provided. Also note that the IF-11Q/X.25 does not respond to input on pins 8 and 22 (data carrier detect and ring).

IF-11Q/X.25

USER'S MANUAL

CHAPTER 4 - HOST DEVICE DRIVER PROGRAMMING

CHAPTER 4

4.0 HOST DEVICE DRIVER PROGRAMMING

4.1 Communication Registers - The IF-11Q/X.25 and the Host Device Driver interact via a set of Hardware Communication Registers (see Tables 4-1 and 4-2). Eight registers are available to the Host Device Driver:

Receive Control and Status Register	(RCSR)
Receive Data Register	(RDR)
Receive Address Register	(RAR)
Receive Word Count Register	(RWCR)
Transmit Control and Status Register	(TCSR)
Transmit Data Register	(TDR)
Transmit Address Register	(TAR)
Transmit Word Count Register	(TWCR)

The registers of the Hardware Communication Register occupy a contiguous block of addresses on the LSI-11 BUS, starting at an address determined by switch settings on the MDMA circuit board. The register descriptions in Tables 4-1 and 4-2 are written from the point of view of the Host Device Driver. For example, the sense of read and write of the IF-11Q/X.25 RCSR and TCSR is from the LSI-11. A sample Device Driver listing is included in this manual as Appendix B.

4.2 Hardware Data Transfer - A DMA transfer is started when the device driver loads the data buffer starting address into the xAR (i.e., RAR or WAR), loads the 2's complement of the desired transfer word count into the xWCR, and sets the GO bit in the xCSR. If a matching request has been issued by the IF-11Q/X.25 software, the DMA hardware is activated and the transfer takes place.

Table 4-1 IF-11Q/X.25 RCSR (Receive Control and Status Register)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ERR	NXM	ZS3	ZS2	ZS1	ZS0	Z80 HALT	DBF	RDY	IEN	ADR 17	ADR 16		REC SIG	REC RES	GO
R	R/W	R	R	R	R	R	R	R/W	R/W	R/W	R/W	R/W	R	W	W

BIT 0 GO

The GO bit. Setting this bit clears RDY and initiates a DMA transfer from the IF-11Q/X.25 to the LSI-11 processor. Clearing this bit has no effect. GO will always read as zero.

BIT 1 REC RESET

The RECEIVE RESET bit. Setting this bit resets the receive DMA hardware. The entire IF-11Q/X.25 to LSI-11 channel is reset. Clearing this bit has no effect. This bit will always read as zero.

BIT 2 REC SIGNAL

The RECEIVE SIGNAL bit. This bit is set to one upon completion of a receive DMA data transfer. If EIN is presently set, the LSI-11 is interrupted. RECEIVE SIGNAL is cleared upon reading the RCSR by the LSI-11. This bit is read-only, and writing to this bit has no effect.

BIT 3 UNUSED

This bit is undefined and is reserved for future use. It is cleared upon system startup or reset.

BIT 4 ADR16

This is ADDRESS BIT 16 for extended addressing operation. This bit is never modified by the IF-11Q/X.25. It is cleared upon system startup or reset.

BIT 5 ADR17

This is ADDRESS BIT 17 for extended addressing operation. This bit is never modified by the IF-11Q/X.25. It is cleared upon system startup or reset.

Table 4-1, continued

BIT 6 IEN	The INTERRUPT ENABLE bit. The LSI-11 program sets this bit to allow the channel to request program interrupts and clears it to disallow interrupts. This channel only responds to interrupt polling cycles when this bit is on. It is cleared upon system startup or reset.
BIT 7 RDY	The READY bit. This bit is on when the channel is not in DMA mode. When the channel is ready, the Memory Address Register, the Word Count Register, and the Control and Status Register may be modified. This bit goes off when DMA mode is started by setting the GO bit. When DMA mode is active, setting this bit causes DMA to be stopped. Clearing this bit has no effect. This bit is set upon system startup or reset.
BIT 8 DBF	The DATA BUFFER FLAG bit. This bit indicates that the Receive Data Buffer contains a word and is ready to be on read. This bit is allowed to be on only when the channel is in DMA mode (RDY = 0). DATA BUFFER FLAG is cleared when the Receive Data Buffer is read.
BIT 9 Z80 HALT	The Z80-CPU HALT bit. This bit indicates the halt state of the IF-11Q/X.25 microprocessor. If the IF-11Q/X.25 is halted, this bit will read as one. Otherwise, this bit will read as zero. This is a read-only bit and writing to this bit has no effect.
BIT 10 ZS0	This is Z80 STATUS BIT 0. This bit and the following Z80 status bits are user defined status bits passed from the IF-11Q/X.25 to the LSI-11. These bits are read-only and writing to these bits has no effect.
BIT 11 ZS1	This is Z80 STATUS BIT 1. See ZS0.
BIT 12 ZS2	This is Z80 STATUS BIT 2. See ZS0.
BIT 13 ZS3	This is Z80 STATUS BIT 3. See ZS0.

Table 4-1, continued

BIT 14 NXM	The NONEXISTENT MEMORY ERROR bit. This bit being set indicates that DMA was attempted to an LSI-11 address which did not respond. This bit is set by the MDMA and is tested and reset by the LSI-11.
BIT 15 ERR	The COMPOSITE ERROR bit. This bit is the logical OR of all error bits. Currently the only defined error is NONEXISTENT MEMORY ERROR. The ERROR bit is reset by resetting all other error bits. This bit is read-only. It is cleared upon system startup or reset.

Table 4-2 IF-11Q/X.25 TCSR (Transmit Control and Status Register)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ERR	NXM	PS3	PS2	PS1	PS0		DBF	RDY	IEN	ADR 17	ADR 16	Z80 RES	XMT SIG	XMT RES	GO
R	R/W	R/W	R/W	R/W	R/W	R	R	R/W	R/W	R/W	R/W	R/W	R	W	W

BIT 0 GO

The GO bit. Setting this bit causes RDY to be reset to zero and initiates a DMA transfer from the LSI-11 processor to the IF-11Q/X.25. Clearing this bit has no effect. GO will always read as zero.

BIT 1 TRANS RESET

The TRANSMIT RESET bit. Setting this bit resets the transmit DMA hardware. The entire LSI-11 tc IF-11Q/X.25 channel is reset. Clearing this bit has no effect. This bit will always read as zero.

BIT 2 TRANS SIGNAL

The TRANSMIT SIGNAL bit. This bit will always read as zero. Clearing this bit has no effect. NOTE: The host software must never set this bit.

BIT 3 Z80 RESET

The Z80 RESET bit. Setting this bit resets the Z80 hardware in the IF-11Q/X.25 and causes the Z80 CPU to restart at location zero. Clearing this bit has no effect. This bit will always read as zero.

BIT 4 ADR16

This is ADDRESS BIT 16 for extended addressing operation. It is cleared upon system startup or reset.

BIT 5 ADR17

This is ADDRESS BIT 17 for extended addressing operation. It is cleared upon system startup or reset.

Table 4-2, continued

BIT 6 IEN	The INTERRUPT ENABLE bit. The LSI-11 program sets this bit to allow the channel to request program interrupts and clears it to disallow interrupts. This channel only responds to interrupt polling cycles when this bit is on. It is cleared upon system startup or reset.
BIT 7 RDY	The READY bit. This bit has a value of one when the channel is not in DMA mode. When the channel is ready, the Memory Address Register, the Word Count Register, and the Control and Status Register may be modified. This bit is cleared when DMA mode is started by setting the GO bit. When DMA mode is active, setting RDY causes DMA to be stopped. Clearing this bit has no effect. This bit is set upon system startup or reset.
BIT 8 DBF	The DATA BUFFER FLAG bit. This bit indicates that the Transmit Data Buffer is empty and is ready to accept a new word. This bit has a value of one only when the channel is in DMA mode RRDY = 0). DATA BUFFER FLAG is cleared when the Transmit Data Buffer is written.
BIT 9 UNUSED	This bit is always read as a zero.
BIT 10 PS0	This is PROCESSOR STATUS BIT 0. This and the following processor status bits are user defined status bits passed from the LSI-11 program to the IF-11Q/X.25 program. These bits may be set or cleared as required by the LSI-11 program. It is cleared upon system startup or reset.
BIT 11 PS1	This is PROCESSOR STATUS BIT 1. See PS0.
BIT 12 PS2	This is PROCESSOR STATUS BIT 2. See PS0.
BIT 13 PS3	This is PROCESSOR STATUS BIT 3. See PS0.

Table 4-2, continued

BIT 14 NXM — The NONEXISTENT MEMORY ERROR bit. This bit has a value of one DMA was attempted to an LSI-11 address which did not respond. This bit is set by the MDMA and is tested and reset by the LSI-11 program.

BIT 15 ERR — The COMPOSITE ERROR bit. This bit is the logical OR of all error bits. Currently the only defined error is NONEXISTENT MEMORY ERROR. The ERROR bit is reset by resetting all other error bits. This bit is read-only. It is cleared upon system startup or reset.

4.3 Software Data Transfer - The IF-11Q/X.25 interface to the LSI-11 looks like two DMA 'pipes' - one to carry data into the IF-11Q/X.25 and one to carry data away from it. One of the tasks of the IF-11Q/X.25 and the host device driver in the LSI-11 is to make each physical data pipe look like several virtual half duplex paths to higher level programs. This is done by enforcing a multiplexing protocol through a series of supervisory messages associated with each transfer of user data. Thus, two types of information being passed through the pipes are distinguished: the four byte supervisory messages (which are created and read only at the executive driver level), and packets of user data. This 'user data' will be called 'data' while the virtual half-duplex data path will be referred to as a 'channel' in this chapter. See Figures 4-1 and 4-2.

INPUT STATE	READ from user	REQUEST TO SEND from remote	LOCAL ABORT from user	REMOTE ABORT from remote	ABORT ACKNOWLEDGE from remote	IO COMPLETE from hardware
0 IDLE	NO ACTION 2	SIGNAL REQUEST TO SEND 1	SEND 'IO ABORT' 5	SEND 'ABORT ACKNOWLEDGE' 0	SEND 'IO ABORT' 4	SEND 'IO ABORT' 4
1 WAIT FOR READ	CALCULATE NB SEND 'OK TO SEND' 3	SEND 'IO ABORT' 4	SEND 'IO ABORT' 5	SEND 'ABORT ACKNOWLEDGE' 0	SEND 'IO ABORT' 4	SEND 'IO ABORT' 4
2 WAIT FOR REQUEST TO SEND	RETURN ERROR 2	CALCULATE NB SEND 'OK TO SEND' 3	SEND 'IO ABORT' 5	SEND 'ABORT ACKNOWLEDGE' SIGNAL IO COMPLETE 0	SEND 'IO ABORT' 5	SEND 'IO ABORT' 5

3 IO PENDING	RETURN ERROR
	3
		SEND 'IO ABORT'
		5
			SEND 'IO ABORT'
			5
				SEND 'ABORT ACKNOWLEDGE' SIGNAL IO COMPLETE
				0
					SEND 'IO ABORT'
					5
						IF EOS OR DONE SIGNAL IO COMPLETE
						0
						ELSE UPDATE NR AND ADDRESS
						2

4 ABORT PENDING (no signal)	NO ACTION 5	NO ACTION 4	NO ACTION 5	SEND 'ABORT ACKNOWLEDGE' 4	NO ACTION 0	NO ACTION 4
5 ABORT PENDING (signal)	RETURN ERROR 5	NO ACTION 5	NO ACTION 5	SEND 'ABORT ACKNOWLEDGE' 5	SIGNAL IO COMPLETE 0	NO ACTION 5

◺ : NEXT STATE

Figure 4-1 IF-11Q/X.25 Multiplexing Protocol Input Finite State Automation

INPUT / STATE	WRITE from user	OK-TO-SEND from remote	LOCAL ABORT from user	REMOTE ABORT from remote	ABORT ACKNOWLEDGE from remote	IO COMPLETE from hardware
0 WAIT FOR WRITE	SEND 'USF/ REQUEST TO SEND' 1	SEND 'IO ABORT' 3	SEND 'IO ABORT' 4	SEND 'ABORT ACKNOWLEDGE' 0	SEND 'IO ABORT' 3	SEND 'IO ABORT' 3
1 WAIT FOR OK TO SEND	RETURN ERROR 1	CHECK FOR EOS SEND 'DATA FOLLOWS' OR 'DATA FOLLOWS EOS' 2	SEND 'IO ABORT' 4	SEND 'ABORT ACKNOWLEDGE' SIGNAL IO COMPLETE 0	SEND 'IO ABORT' 4	SEND 'IO ABORT' 4

2 IO PENDING	RETURN ERROR
	2
		SEND 'IO ABORT'
		4
			SEND 'IO ABORT'
			4
				SEND 'ABORT ACKNOWLEDGE' SIGNAL IO COMPLETE
				0
					SEND 'IO ABORT'
					4
						UPDATE NW. IF DONE, SIGNAL IO COMPLETE
						0
						ELSE UPDATE ADDR. SEND 'USF/REQUEST TO SEND'
						1

3 ABORT PENDING (no signal)	NO ACTION 4	NO ACTION 3	NO ACTION 4	SEND 'ABORT ACKNOWLEDGE' 3	NO ACTION 0	NO ACTION 3
4 ABORT PENDING (signal)	RETURN ERROR 4	NO ACTION 4	NO ACTION 4	SEND 'ABORT ACKNOWLEDGE' 4	SIGNAL IO COMPLETE 0	NO ACTION 4

◺ =NEXT STATE

Figure 4-2 IF-11Q/X.25 Multiplexing Protocol Input Finite State Automation

4.3.1 Requirements Supported by the Protocol - The protocol insures that no data is transferred on a channel until symmetric requests are outstanding in the LSI-11 and the IF-11Q/X.25. It allows simultaneous requests on multiple channels, but only one request to be outstanding on an individual channel at a time. The supervisory messages encode End-of-Stream and User Subfunction information independent of the data streams.

4.3.2 Description of Message Use - Each message type and its use is described below. The messages and responses are all channel specific.

4.3.2.1 Request-to-Send Message - Whenever a write is activated, a Request-to-Send NW bytes is sent, where NW = the requested byte count. The four bit User Subfunction associated with the write request is put in the high order four bits of the message byte.

NOTE

A write is 'activated' when a Write or Write End-of-Stream request is made by an applications program, or whenever a data transfer completes which partially, but not completely, fulfills the original request. In the latter case, NW is updated to be the number of bytes from the original request which have not yet been written.

4.3.2.2 OK-to-Send Message - Whenever a read is outstanding for NR bytes (NR = the number of bytes to be read) and a Request-to-Send NW bytes message is received, an OK-to-Send NB bytes message is sent (NB = minimum {NR,NW}). The reading side always calculates NB.

NOTE

> A read is 'outstanding' when a read request is made by an applications program, or if a data transfer has completed which partially, but not completely, fulfills the original request and the preceeding Data Follows message was NOT End-of-Stream. In the latter case, NR is updated to reflect the number of bytes of the original request which have not yet been read.

4.3.2.3 Data Follows and Data Follows End-of-Stream Messages - Whenever an OK-to-Send NB bytes message has been received, a Data Follows or Data Follows End-of-Stream message is sent, followed immediately by NB bytes of data. NO MESSAGE TRAFFIC OR DATA FROM ANOTHER CHANNEL MAY COME BETWEEN THE MESSAGE AND THE DATA. The Data Follows End-of-Stream message is sent if, and only if, the original Write request was End-of-Stream and NB = NW (i.e., this will be the last physical transfer to fulfill an End-of-Stream Write request).

4.3.2.4 IO Abort Message - An IO Abort message is sent under the following conditions:

1. an applications program closes a channel,
2. an applications program issues an Abort request,
3. a protocol breakdown is identified with respect to a channel.

The only exception is when an IO Abort message has previously been sent and no Abort Acknowledge message has been received.

4.3.2.5 Abort Acknowledge Message - Whenever an IO Abort message is received, an Abort Acknowledge message is returned.

4.3.3 Supervisory Message Format -

Byte 0 : (Bits 0-2) - message code
: (Bit 3) - must be zero
: (Bits 4-7) - user subfunction

Byte 1 : virtual channel number

Byte 2 : requested byte count (low byte)

Byte 3 : requested byte count (high byte)

4.3.4 Message Codes -

0	=	OK-to-Send
1	=	Request-to-Send
2	=	Data Follows
3	=	Data Follows End-of-Stream
4	=	IO Abort
5	=	Abort Acknowledge

IF-11Q/X.25

USER'S MANUAL

CHAPTER 5 - X.25 PROTOCOL PROGRAMMING

CHAPTER 5

5.0 X.25 PROTOCOL PROGRAMMING

5.1 Implementation Notes - Several design decisions were made in the implementation of the IF-11Q/X.25 which must be taken into account in application programming and network usage.

5.1.1 Frame Level Window Size - The X.25 frame level window has the value of seven and cannot be changed by the application program.

5.1.2 Packet Level Window Size - The X.25 packet level window has the value of two and cannot be changed by the application program.

5.1.3 Loop-Back Mode - The IF-11Q/X.25 Loop-Back switch can be set to any value by the application program and initially has a value of zero. A value of zero disables X.25 loop-back mode while a non-zero value enables X.25 loop-back mode. (See "Restart with diagnostic parameters", Table 5-5)

5.1.4 T1 Timer - The X.25 T1 timer can be set to any value by the application program and initially has the value of three seconds. Legal values range from 0 to 63 seconds, with a resolution of one second. Please note that a value of zero specifies no timer activity and, thus, no timer recovery will occur. (See "Restart with diagnostic parameters", Table 5-5)

5.1.5 N2 Counter - The X.25 N2 counter can be set to any value by the application program and initially has the value of twenty. Legal values range from 2 to 255. The current value remains unchanged if an illegal value is specified. (See "Restart with diagnostic parameters", Table 5-5)

5.1.6 Packet Size - The X.25 data packet size has a maximum value of one hundred twenty-eight bytes and cannot be changed by the application program.

5.2 Message Formats - The format of messages between the application program and the IF-11Q/X.25 is not the format of messages specified by CCITT for X.25 networks. The IF-11Q/X.25 system converts application program X.25 protocol requests into the format specified by CCITT. In addition, any message over 128 bytes in length will automatically be converted into multiple M-Bit packets of 128 bytes.

5.2.1 Supervisory Message Formats - All supervisory messages consist of a fixed-length header optionally followed by a variable-length data field. The header is four bytes in length. The optional data field may range from 0 to 128 bytes in length. Supervisory messages are always sent and received via logical channel zero.

5.2.1.1 Fixed Length Messages - Many supervisory messages do not require additional data beyond the header. For these messages the fourth header byte contains zero.

5.2.1.2 Variable Length Messages - Some supervisory messages require additional data beyond the header. For these messages the fourth header byte contains the count of the data bytes which immediately follow the header.

5.2.2 Data Message Formats - User data messages may range from 0 to 65535 bytes in length. Data messages are always sent and received via the logical channel number assigned when the call is established. Logical records longer than 65535 bytes are sent as multiple messages using Write Stream for all but the last message and Write Stream and End for the last message (see Sections 6.6.4.1.1 and 6.6.4.1.2).

5.2.2.1 M-Bit Packets - Any message over 128 bytes in length will automatically be converted into multiple M-Bit packets of 128 bytes. Only the last packet will reflect the M-Bit value specified by the application program, while the others will have the M-Bit set.

5.2.2.2 Q-Bit Data - User data is normally sent as non-Q-Bit data. The user can also optionally cause packets to be sent with the Q-Bit set to one. Q-Bit data is usually used at higher protocol levels to denote control data.

5.3 Message Contents - The contents of messages between the application program and the IF-11Q/X.25 are not the contents specified by CCITT recommendation X.25. The IF-11Q/X.25 system converts application program X.25 protocol request contents into those specified by CCITT. Table 3.1 details the correlation between ACC IF-11Q/X.25 commands and CCITT supervisory packet types.

5.3.1 Supervisory Message Contents - All supervisory messsages have a four byte header. The first byte holds the command code from Table 5-2a. The second byte contains the full-duplex logical channel number (LCN) TIMES TWO. The third byte is used by different commands for different purposes. For several message types the third byte contains the virtual circuit number. The fourth byte contains the count of optional data bytes which follow the header and may be zero.

5.3.1.1 Network Commands - Table 5-1 describes those commands which map directly into CCITT equivalents and initiate or result from actual network activity.

5.4 IF-11Q/X.25 Subsystem Queries and Responses - The IF-11/X.25 maintains internal information pertaining to X.25 network operation. Response messages which return partial contents of this internal information may be elicited by means of query commands from Table 5-2b. Table 5-3 details the contents of the different responses. Virtual circuit or logical channel information can be obtained by means of two related commands. One (Virtual Circuit Query) takes the virtual circuit number as an index while the other (Logical Channel Query) takes the logical channel number as an index. Both commands return the same information. Frame level information can be obtained via the Frame Level Query command. IF-11Q/X.25 network errors and internal error conditions can be obtained by means of the Error Query command. Table 5-4 contains the error codes returned by the error query.

5.4.1 Data Message Contents - User data messages are transmitted and received in complete transparency. No headers are required.

5.4.1.1 Q-Bit Value Specification - The value of the X.25 Q-Bit for a message is typically controlled by a device driver subfunction value. See section 5.4.

5.4.1.2 M-Bit Stream Generation - The generation of M-Bit Packet streams is typically controlled by device driver subfunction values.

TABLE 5-1

Correlation between ACC IF-11Q/X.25 and CCITT Packet Types

ACC X.25 Commands	CCITT Packet Types
Answer	Call Accepted/Call Connected
Call	Call Request
Clear Logical Channel and Clear Virtual Circuit	Clear Confirmation or Clear Request/Clear Indication
Interrupt	Interrupt Request
Interrupt Acknowledge	Interrupt Confirmation
Ready	Receiver Ready or Receiver Not Ready
Reset	Reset Indication/Reset Request
Reset Acknowledge	Reset Confirmation
Restart	Restart Indication/Restart Request
Restart Acknowledge	Restart Confirmation
Ring	Incoming Call

TABLE 5-2a

ACC IF-11Q/X.25 Command Codes
(All values are octal)

Code	Command
003	Answer
000	Call
004	Clear Logical Channel to X.25 Network
002	Clear Virtual Circuit from X.25 Network
042	Interrupt
045	Interrupt Acknowledge (Response Only)
043	Ready
040	Reset
041	Reset Acknowledge
100	Restart
101	Restart Acknowledge
001	Ring from X.25 Network

TABLE 5-2b

ACC IF-11Q/X.25 Diagnostic Query/Response Codes
(All values are octal)

Code	Command/Response
207	Error Query to IF-11Q/X.25
206	Error Response from IF-11Q/X.25
213	Frame Query to IF-11Q/X.25
212	Frame Response from IF-11Q/X.25
203	Logical Cnannel Query to IF-11Q/X.25
202	Logical Cnannel Response from IF-11Q/X.25
201	Virtual Circuit Query to IF-11Q/X.25
200	Virtual Circuit Response from IF-11Q/X.25

TABLE 5-3

ACC IF-11Q/X.25 Response Message Contents
(All values are octal)

VIRTUAL CIRCUIT TABLE RESPONSE:

Offset	Contents
0:	response type (200)
1:	virtual circuit number
2:	zero
3:	response length (32)
4:	if non-zero, logical channel is active
5:	logical channel number
6:	p(s) for receive side
7:	p(r) for receive side
10:	receive window
11:	receive flags
12:	p(s) for transmit side
13:	p(r) for transmit side
14:	transmit window
15:	transmit flags
16-17:	addr of first packet on queue to frame level
20-21:	addr of last packet on queue to frame level
22:	state of virtual circuit
23:	state of output side of logical channel (from host)
24:	state of input side of logical channel (to host)
25:	virtual circuit number
26:	flags for virtual circuit
27-30:	address of first buffer on queue to host
31-32:	address of last buffer on queue to host
33-34:	address of current input buffer (to host)
35-36:	address of current output buffer (from host)

LOGICAL CHANNEL TABLE RESPONSE:

Offset	Contents
0:	response type (202)
1:	logical channel number
2:	zero
3-36:	same as virtual circuit table response

TABLE 5-3 (continued)

ERROR RESPONSE:

Offset	Contents
0:	response type (206)
1:	zero
2:	zero
3:	response length
4:	index of next entry in error table
5:	index of last used entry in error table
6:	count of errors encountered
7-n:	error table: 4 bytes per entry. Each entry has the form:
	0: error number - see Table 5-4
	1: -Reserved-
	2-3: PC at error

LAP FRAME RESPONSE:

(NOTE: These offsets are guaranteed to change from one release to the next)

Offset	Contents
0:	Response type (212)
1:	Zero
2:	Zero
3:	Response length
4-5:	-Reserved-
6:	Frame Level State
	0 - initial state
	1 - UA/SARM wait
	2 - UA wait
	3 - Ready
7-41:	-Reserved-
42:	SARM sent count
43:	SARM received count
44:	DISC sent count
45:	DISC received count
46:	CMDR sent count
47:	CMDR received count
50:	REJ sent count
51:	REJ received count
52:	RNR sent count
53:	RNR received count

TABLE 5-5 (continued)

LAPB FRAME RESPONSE:

Offset	Contents
0:	response type (212)
1-2:	zero
3:	response length
4:	Carrier Detect flag (non-zero if carrier detect is on)
10:	Frame Level State: 0 - initial state 1 - UA/SABM wait 2 - UA/DISC wait 3 - Ready
14:	T1 timer value
16:	N2 counter value
17:	SABM sent count
20:	SABM received count
21:	DISC sent count
22:	DISC received count
23:	CMDR sent count
24:	CMDR received count
25:	REJ sent count
26:	REJ received count
27:	RNR sent count
30:	RNR received count
31:	count of frames received with bad CRCs
32:	count of badly formed frames received

TABLE 5-4

ACC IF-11Q/X.25 Error Response Error Codes
(all values are octal)

NOTE

The error codes are the lower six bits of each entry. The high order bits are used internally and should be masked off to yield the error codes listed below.

Code	Description
4	Frame level attempted to retransmit a non-existent frame
5	CMDR received by frame level
6	Frame level found itself in an undefined state
7	Attempt to send call-accepted packet from the wrong state
10	Attempt to send call-request packet on an active logical channel
11	Attempt to send clear-confirm packet from the wrong state
12	Attempt to send data packet on virtual circuit zero
13	Attempt to send data packet from wrong state
14	Attempt to send interrupt packet from wrong state
15	Attempt to send interrupt-confirm packet from wrong state
16	Attempt to send reset-confirm packet from wrong state
17	Attempt to send RR or RNR packet from wrong state
20	Packet level was requested to transmit a poorly-formed packet
21	Packet level was given a packet for an invalid logical channel number
22	A packet from the DCE had an invalid p(s)
23	Invalid packet received while in state p1
25	Invalid packet received while in state p2
26	Invalid packet received while in state p3
27	Invalid packet received while in state p5
30	Invalid packet received while in state p6
31	Invalid packet received while in state p7
32	Invalid packet received while in state d1
33	Invalid packet received while in state d2
34	Invalid packet received while in state d3
35	Illegible packet received from DCE
36	A virtual circuit was found to be in an undefined state

TABLE 5-4 (continued)

Code Description

41 - A packet from the DCE contained an invalid p(r)
43 - Free list exhausted
44 - The buffer monitor was unable to account for all buffers
45 - A transfer to or from the host failed
46 - A write completed on a logical unit assigned for reading
47 - The output side of a logical channel was in an undefined state
50 - A read completed on a logical unit assigned for writing
51 - The input side of a logical channel was in an undefined state
52 - I/O was attempted to an inactive logical unit
53 - Invalid supervisory command received while in data xfer state
54 - Invalid supervisory command received while in answer-wait state
55 - Invalid supervisory command received while in call-wait state
56 - Invalid supervisory command received while in idle state
57 - A virtual circuit was found to be in an undefined state
60 - Attempt to reassign an active logical unit number
61 - NCP received an undefined command
62 - A supervisory command specified an invalid virtual circuit number

TABLE 5-5

ACC IF-11/X.25 Command Formats

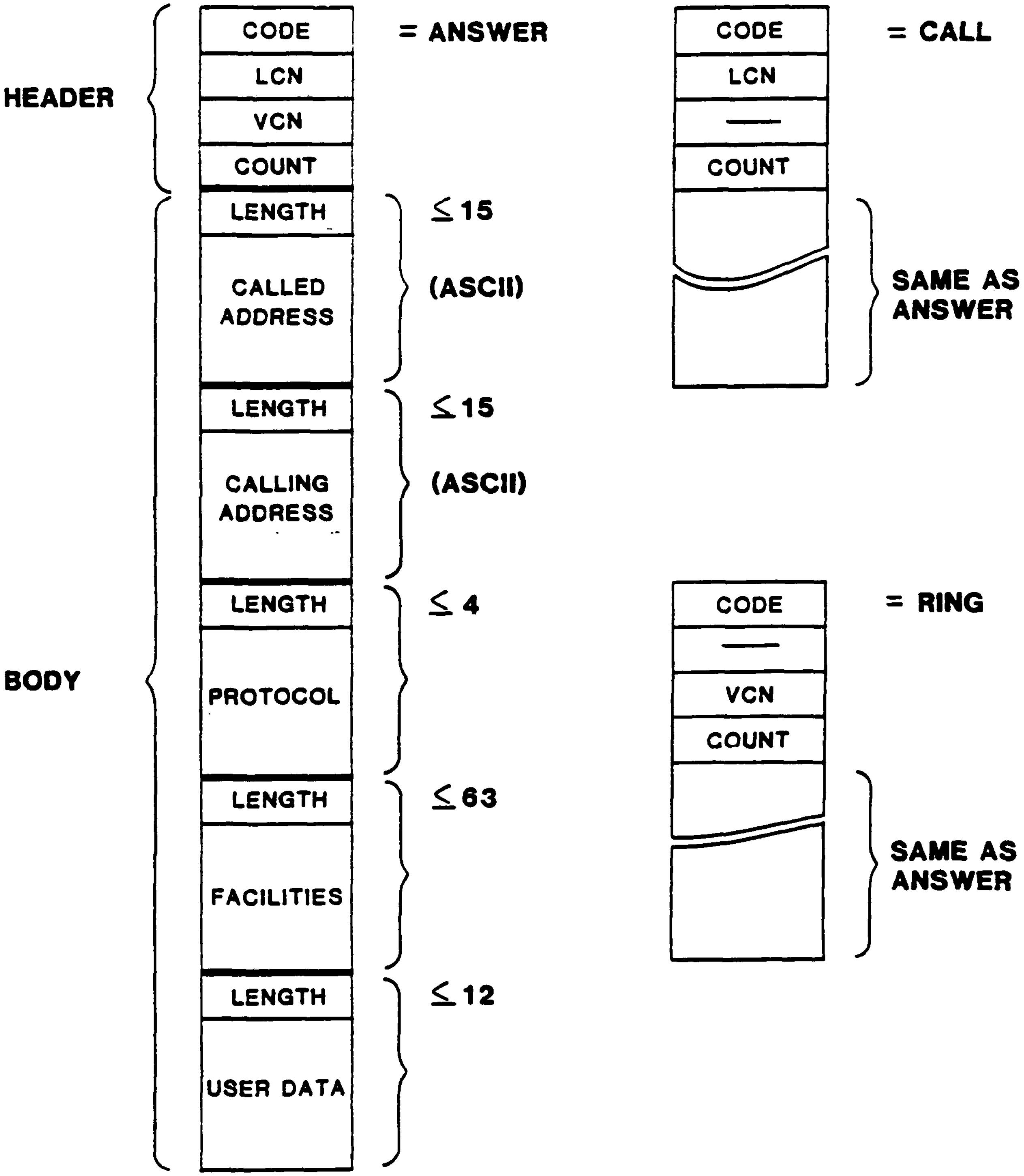


TABLE 5-5 (continued)

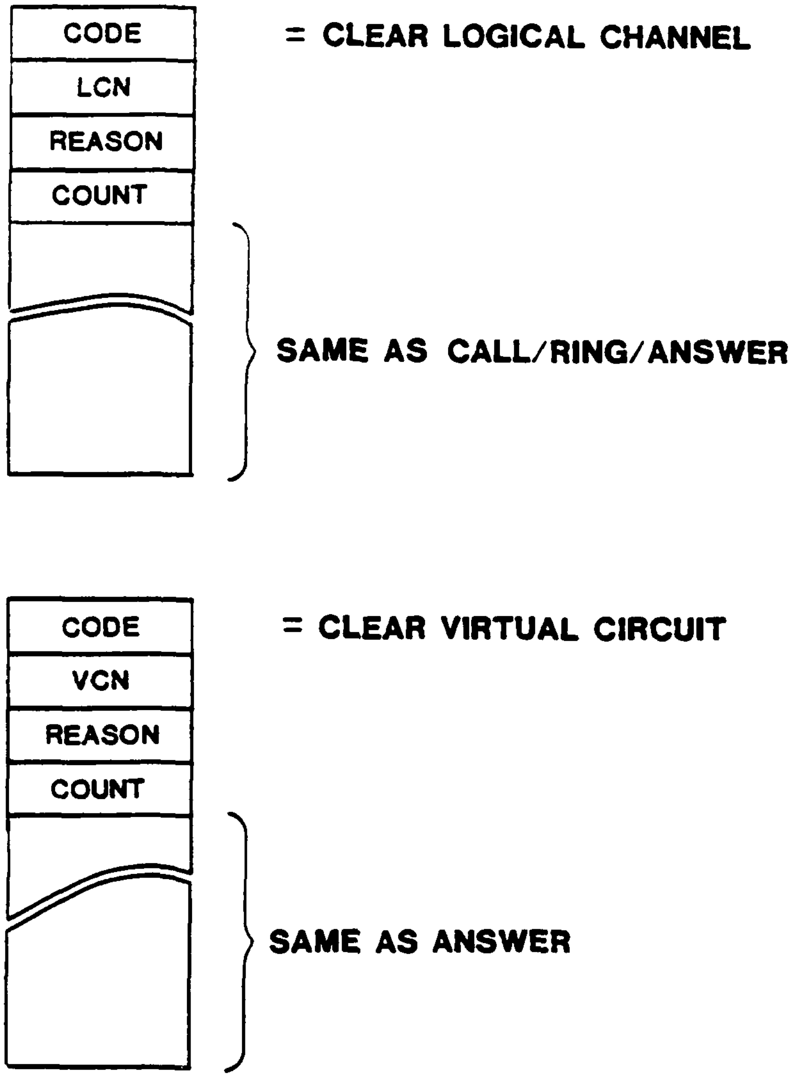


TABLE 5-5 (continued)

CODE	= RESET, INTERRUPT
LCN	
REASON	
0	

CODE	= RESET ACKNOWLEDGE, LOGICAL CHANNEL QUERY
LCN	
——	
0	

CODE	= READY
LCN	
0 → RNR ≠0 → RR	
0	

CODE	= RESTART, RESTART ACKNOWLEDGE, FRAME QUERY, ERROR QUERY
——	
——	
0	

CODE	= RESTART (with DIAGNOSTIC PARAMETERS)
——	
——	
LENGTH	NOTE - MUST BE 3
LOOPBACK	
T1	
N2	

TABLE 5-5 (continued)

CODE	= VIRTUAL CIRCUIT QUERY
VCN	
——	
0	

CODE	= ERROR RESPONSE, FRAME RESPONSE, LOGICAL CHANNEL RESPONSE, VIRTUAL CIRCUIT RESPONSE
——	
——	
COUNT	
DATA	

IF-11Q/X.25

USER'S MANUAL

CHAPTER 6 - MICRODIAGNOSTICS

CHAPTER 6

6.0 MICRODIAGNOSTICS

6.1 Introduction - The IF-11Q/X.25 microdiagnostics perform subsystem integrity tests upon power-on reset and display the results in a bank of eight LEDs which are clearly visible without removing any boards or cables. This display provides quick visual verification of SYSTEM operational readiness. Detected errors, if determined by the operator to be inconsequential, can be defeated by means of hardware switches. If any errors are detected (and have not been defeated by the operator) then processing halts with the error status displayed in the LEDs. The operator must cycle power down and back up to re-run the microdiagnostics after correcting the problem or electing to ignore the error by means of the defeat switches. If no errors are detected (or all which are detected have been defeated) then processing continues in the X.25 protocol code. Note that the X.25 protocol code idle loop "spins the lights" as a system load indication where a heavy traffic load will slow the "spin" rate.

6.2 Display LEDs - The indicator LEDs are located on the XQ/CP P-board. The significance of each LED is as follows:

```
+---+---+---+---+---+---+---+---+
| 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |   |   |   |   |   |   |   |
  |   |   |   |   |   |   |   +-> ROM Failure
  |   |   |   |   |   |   +-----> RAM Failure
  |   |   |   |   |   +---------> SIO Failure
  |   |   |   |   +-------------> CTC Failure
  |   |   |   +-----------------> DMA Register Failure
  |   |   +---------------------> DMA Memory-to-Memory Failure
  |   +-------------------------> Unexpected Interrupt during
  |                               initialization
  +-----------------------------> ALL TESTS COMPLETE
```

Upon power-on reset, all of the display LEDs are turned off and testing begins. All tests are run to completion and then the various error statuses are displayed along with the ALL TESTS COMPLETE indication.

6.3 Error Defeat Switches - Hardware initialization errors can be ignored by means of a bank of DIP switches also located on the XQ/CP P-Board. In order to defeat each detected error (as represented by a lit LED), it is necessary to turn on (or close) the corresponding switch. Each and every error must be defeated for processing to continue into the X.25 protocol code. On the other hand, processing will not continue if a switch is set for which there is no corresponding error. In addition, the high order switch, which corresponds with the ALL TESTS DONE condition and is not an error, should be left off (or open) even though the LED is lit.

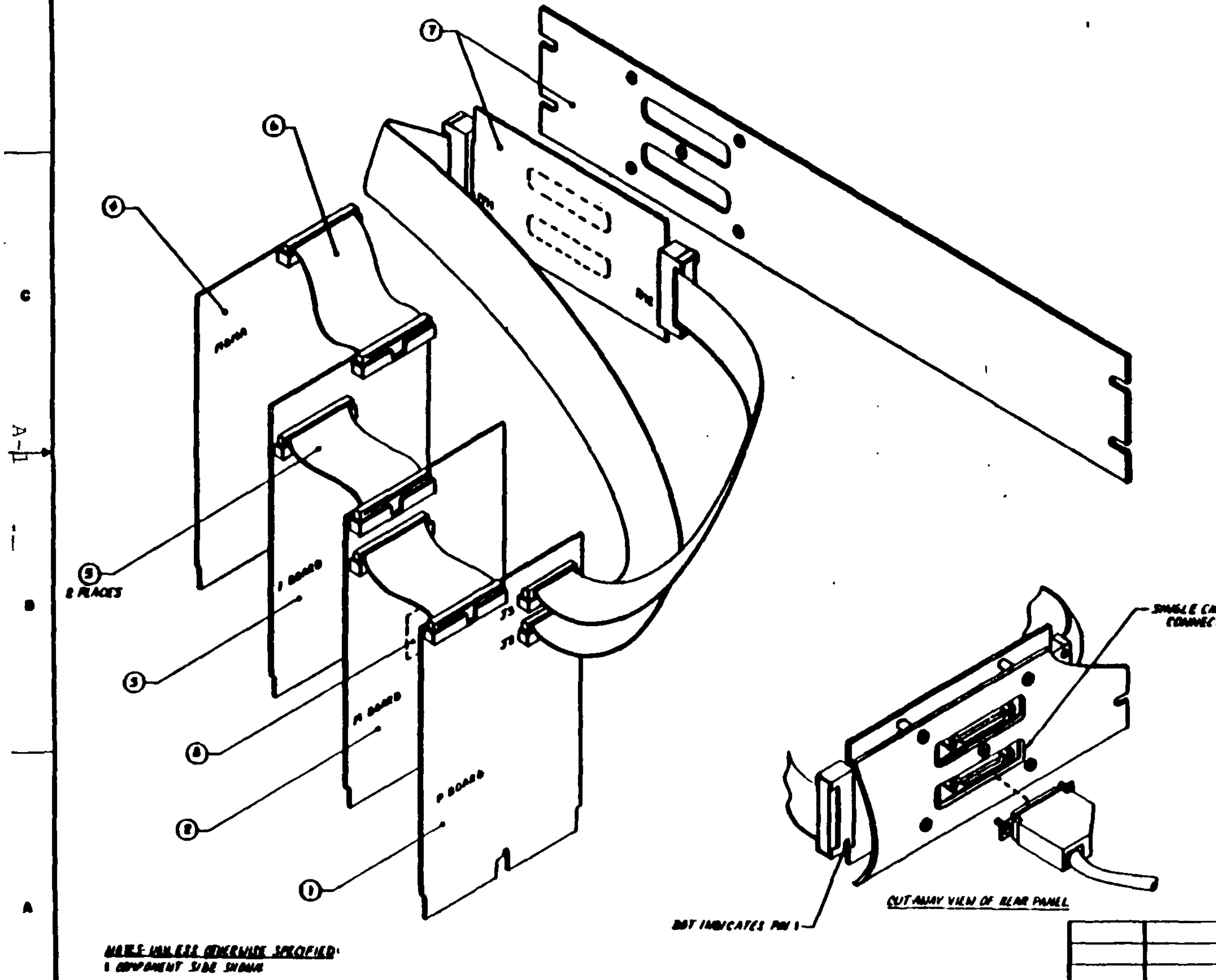
IF-11Q/X.25

USER'S MANUAL

APPENDIX A - DRAWINGS

8600108 A

A	DRAWING ISSUED			



NOTES: UNLESS OTHERWISE SPECIFIED:
1 COMPONENT SIDE SHOWN

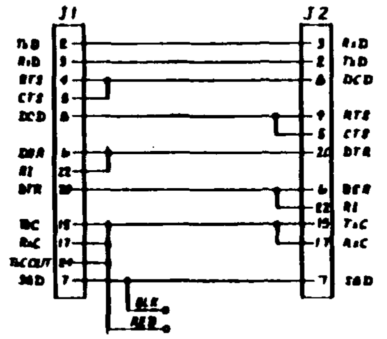
8	1		5200020	EPROM SET, IF-110/X25 (HDLC/LAPB)
7	1		8000063	PANEL ASSEMBLY, XQ/CP, RS232 REAR PANEL
6	1		8300031	CABLE ASSEMBLY, BOARD INTERCONNECTION, 2 CONNECTORS
5	2		8300137	CABLE ASSEMBLY, INTERCONNECT CABLE
4	1		8100093	P.C ASSEMBLY, MDMA
3	1		8100096	P.C ASSEMBLY, XQ/CP, I BOARD
2	1		8100097	P.C. ASSEMBLY, XQ/CP, M BOARD
1	1		8100095	P.C ASSEMBLY, XQ/CP, P BOARD

TOP ASSEMBLY
IF-110/X25
(HDLC/LAPB)

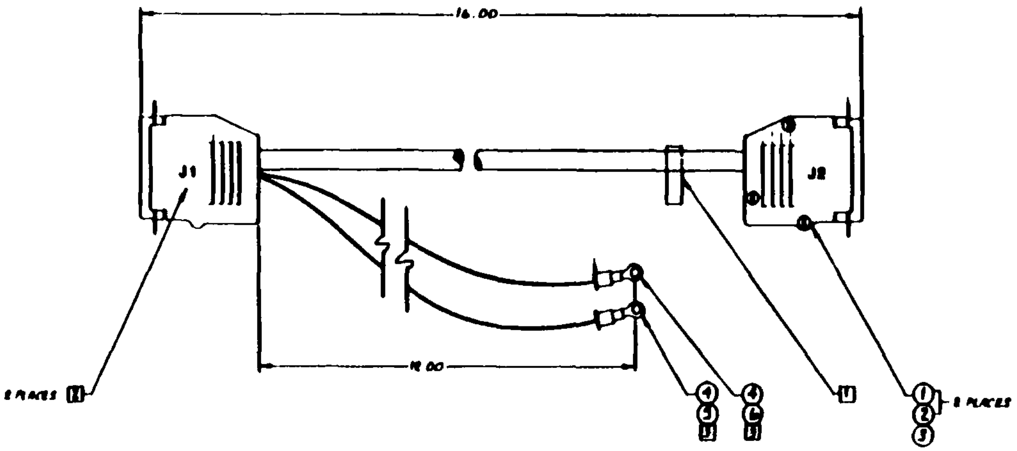
B | TN084 | 8600108 | A

	A	DRAWING ISSUED	5 7 82	

B300222 1 A



WIRING DIAGRAM



NOTES: UNLESS OTHERWISE SPECIFIED
1 MARK PER ACC STANDARD 3101000 APPROX AS SHOWN.
2 LABEL PER ACC STANDARD 3101000 APPROX AS SHOWN WITH .20 HIGH REFERENCE DESIGNATIONS.
3 STRIP WIRE 37 MAX, CRIMP AND SOLDER

6	1		9922-4166	BELDON 22 AWG STRANDED HOOK UP WIRE, RED	12.00
5	1		9922-94162	BELDON 22 AWG STRANDED HOOK UP WIRE, BLACK	12.00
4	2		8-01-404 A	3M RING TERMINAL 22-18 AWG, #6 STUD	
3	1		C6017	COLUMBIA 22 AWG STRANDED 6 PR CABLE	16.00
2	2		8630 96C 25	SOURIAU JUNCTION SHELL	
1	2		DB 25S	ITT CANNON D TYPE CONNECTOR W/ SOLDER CUP CONTACTS	

PARTS LIST

ACC Associated Computer Consultants

CABLE ASSEMBLY
NULL MODEM

ROSENWALD 4 29 82
6.4.82
Carlisle 5/4/82

D 7N084 B300222 A

SCALE NONE SHEET 1 OF 1

DO NOT SCALE DRAWING

IF-11Q/X.25

USER'S MANUAL

APPENDIX B - SAMPLE DEVICE DRIVER LISTING

```
RSXMC   MACRO V04.00  25-AUG-82 17:30:56 PAGE 4


                        .ENABL LC
        000000          .rept 0

                                                          AAAAA
                                                         AAAAAA
                                                     .  AAA AAA
                                                       AAA  AAA
                                                      AAA   AAA
                                                     AAA    AAA
                                                    AAA     AAA
                                                   AAAAAAAAAAAA
                                                  AAA       AAA
                                                 AAA        AACCCCCCCCCC
                                                AAAA       ACCCA       CCC
                                                           CCC          CCC
                                                           CCC
                                                           CCC
                                                           CCC
                                                           CCC
                                                           CCC          CCC
                                                            CCC        CCCCCCCCCC
                                                              CCCCCCCCCC        CCC
                                                                    CCC          CCC
                                                                    CCC
                                                                    CCC
                                                                    CCC
                                                                    CCC
                                                                    CCC          CCC
                                                                     CCC        CCC
                                                                       CCCCCCCCCC

                                Copyright (c) 1981 by Associated Computer Consultants
                                228 East Cota Street, Santa Barbara, California  93101
                                (805) 963-8801

                        File:           UNDRV.MAC for the XQ/CP

                        Project:        X.25, among others.

                        Function:       RSX-11M Multi-Channel Driver (MCD).
                                        This module is the operating system component which interfaces
                                        RSX-11M tasks to an ACC XQ/CP based product.
                                        Typically 33 full-duplex data paths are provided.

                        Components:     RSX-11M User QIO interface
                                        RSX-11M Pool interface
                                        ACC XQ/CP Protocol interface


                        Revision History:
                            19-Oct-81       Convert to XQ/CP Protocol.
                            H. G. Miller

                        Associated Files:
                                debug.def   Select BPT at driver load time to allow for the setting
```

```
 193        56                              of XDT breakpoints for driver debugging.
 194        57
 195
```

```
RSXMC   MACRO V04.00  25-AUG-82 17:30:56 PAGE 4-1


    58                                  dvlp.def    Select the exclusion of hardware reset at driver load
    59                                              time so as to retain RAM-based development code.
    60
    61                              Assembly Command Line:
    62                                      MAC>UNDRV,UNDRV/-SP=[1,1]EXEMC/ML,[1,54]RSXMC,[100,100]UNDRV
    63
    64                              Task Build Command Line:
    65                                      TKB>UNDRV/-HD/-MM,UNDRV/-SP/CR,UNDRV=UNDRV
    66                                      TKB>[1,54]RSX11M.STB/SS,[1,1]EXELIB/LB
    67                                      TKB>/
    68                                      TKB>STACK=0
    69                                      TKB>PAR=DRVPAR:120000:10000
    70                                      TKB>//
    71
    72                              .endr
    73                              .sbttl | Definitions
    74                              .title Macros
    75                              .sbttl |        Macros
```

```
216   MACROS  MACRO V04.00  25-AUG-82 17:30:56 PAGE 5
217   |               ASECT - SET TABLE OFFSET POINTER
218
219        1                        .sbttl |            asect - Set table offset pointer
220        2                        ;
221        3                        ;       Asect is used by other macros which use the Word Macro
222        4                        ;
223        5                        ;       Entry:
224        6                        ;               arg is the address value to be selected.
225        7                        ;
226        8                        .macro asect arg
227        9                        .nlist
228       10                        .. = arg
229       11                        .list
230       12                        .endm
231       13
232       14                        .sbttl |            Word - allocate a word of memory and assign offset to symbo
233       15                        ;
234       16                        ;  Word - allocate a word of memory and assign offset to symbol
235       17                        ;
236       18                        ;       The symbol is not assigned a value if it has already been assigned one else
237       19                        ;
238       20                        ;       Entry:
239       21                        ;               Name - Name of the symbol to be defined.
240       22                        ;               Size - Optional number of words to be allocated
241       23                        .macro word name, size
242       24                        .nlist
243       25                        .list meb
244       26                        .dsabl crf
245       27                        .if nb size
246       28                          ... = size
247       29                        .iff
248       30                          ... = 1
249       31                        .endc
250       32                        .iif ndf name   name = ..
251       33                                .blkw ...
252       34                                .. = .. +  ... + ...
253       35                        .enabl crf
254       36                        .nlist meb
255       37                        .list
256       38                        .endm
257       39
258       40                        .macro dispatch arg,addr
259       41                                cmp     #arg,r0
260       42                                beq     addr
261       43                        .endm
262       44
263       45
264       46
265       47                        .title DEC Device Driver Information
266       48                        .sbttl |        DEC Device Driver Information
267
```

```
267  DEC DEVICE DRIVER INFORMATION   MACRO V04.00  25-AUG-82 17:30:56 PAGE 6
268  |      DEC DEVICE DRIVER INFORMATION
269
270        1
271        2                                ;  DEC Device Driver Macros
272        3                                .mcall abodf$, hwddf$, pktdf$, tcbdf$, devdf$
273        4 000000                         abodf$
274        5 000000                         hwddf$
275        6 000000                         pktdf$
276        7 000000                         tcbdf$
277        8 000000                         devdf$
278        9
279       10
280       11                                ;  UCB address for controller
281       12 000000                         cntbl:  .blkw   1
282       13
283       14
284       15                                ;  defined for loadable driver
285       16         000000                 ld$un = 0;
286       17
287       18
288       19                                ;  drive dispatch table (DDT)
289       20 000002                         $untbl::
290       21 000002  001312'                        .word   unini   ; QIO Request Entry
291       22 000004  001630'                        .word   uncan   ; I/O Kill Entry
292       23 000006  001644'                        .word   untmo   ; Device Time-Out Entry
293       24 000010  001176'                        .word   unpwf   ; Driver Load Entry
294       25
295       26                                .title ACC Device Driver Information
296
```

```
ACC DEVICE DRIVER INFORMATION   MACRO V04.00  25-AUG-82 17:30:56 PAGE 7
|       ACC DEVICE DRIVER INFORMATION

      1                              .sbttl |       ACC Device Driver Information
      2                              ;
      3                              ;  RSX-11M I/O Packet Definitions Required by $GTPKT Circumvention
      4                              ;
      5        000024                i..buf = i.prm;         /* First Word of Buffer Address Double-word */
      6        000030                i..cnt = i.prm+4;       /* Data Buffer Byte Count */
      7        000032                i..chn = i.prm+6;       /* RSX-11M Full-Duplex Channel Number */
      8                                                     ;/* here for xfer requests, .-2 for con/dsc */
      9        000034                i..xfr = i.prm+10;      /* in-process transfer byte count */
     10
     11                              ;
     12                              ;  RSX-11M QIO Subfunction Definitions
     13                              ;
     14        000002                sf.str = 2;      /* Stream Subfunction Flag */
     15        000004                sf.end = 4;      /* End Subfunction Flag */
     16
     17                              ;  RSX-11M QIO Function [illegible] Definitions
     18                              ;
     19        003000                io.con = 3000;  /* Connect to Data Path */
     20        003400                io.dsc = 3400;  /* Disconnect Data Path */
     21        004000                io.dbr = 4000;  /* Debug Read */
     22        004400                io.dbw = 4400;  /* Debug Write */
     23
     24                              .sbttl |
     25                              .title Non-Pool Data Base Definitions
     26                              .sbttl |       ACC Non-Pool Device Driver Data Base Definitions
```

```
NON-POOL DATA BASE DEFINITIONS  MACRO V04.00  25-AUG-82 17:30:56 PAGE 8
|              CONTENT DEFINITIONS

                                        .sbttl |              Content Definitions
                                        ;
                                        ;  Interprocessor Protocol Message Content Definitions
                                        ;
                 000000                 p.cts   = 0;    /* Clear-to-Send "n" bytes. */
                 000001                 p.rts   = 1;    /* Request-to-Send "n" bytes. */
                 000002                 p.df    = 2;    /* Data Follows */
                 000003                 p.dfe   = 3;    /* Data Follows End-of-Stream */
                 000004                 p.abort = 4;    /* Abort */
                 000005                 p.aback = 5;    /* Abort Ack */
                 000010                 p.debug = 10;   /* Debug Channel Qualifier */
                 000400                 p.wchn = 0400;  /* Write Channel Qualifier */

                                        ;
                                        ;  Protocol Tag Word Bit Masks
                                        ;
                 177767                 m$dbug = ^Cp.debug;     /* isolate debug channel flag */
                 177417                 m$usbf = ^C0360;        /* isolate user subfunction bits */
                 177770                 m$mtyp = ^C07;          /* isolate message type code */
                 177717                 m$mext = ^C060;         /* extended address bits */
```

```
NON-POOL DATA BASE DEFINITIONS  MACRO V04.00  25-AUG-82 17:30:56 PAGE 9
|               FORMAT DEFINITIONS

      1                         .sbttl |                Format Definitions
      2        000041           unpcnt = 33.;      /* count of full-duplex data paths */
      3        001430           cqsize = 6*2*2*unpcnt; /* byte count of message circle queue */
      4
      5                         ;
      6                         ; Channel Block Entry Definitions.
      7                         ;
      8                         ;       Each Channel Block relates RSX-11M QIOs to XQ/CP data paths.
      9                         ;       Two half-duplex XQ/CP channels are paired to produce each full-duplex
     10                         ;       RSX-11M channel.
     11                         ;       The Task TCB address is used for channel allocation.  No task may
     12                         ;       diddle a channel to which it has not connected.  (A horrible exception
     13                         ;       is the "Debug Channel" which is first-come/first-served).
     14                         ;       The TAG word contains a prototype header for protocol messages for
     15                         ;       the channel.  It contains the Read LCN, Debug Channel Flag, Message
     16                         ;       type code, and the user subfunction bits from the last RTS received.
     17                         ;       The Read byte count is copied out of the I/O packet to allow
     18                         ;       the received byte count to be summed there.
     19                         ;       The RTS byte count is used for byte count arbitration.
     20                         ;       The Read-Status-Word and Write-Status-Word contain the address
     21                         ;       of the state sub-table for the channel.
     22                         ;       Read and Write QIO Queue Headers implement a queue for multiple
     23                         ;       QIO requests.  While the requests are handled sequentially and in
     24                         ;       the order in which they are received, the capability exists to start
     25                         ;       a new I/O without going all the way back up to the controlling task.
     26                         ;
     27                         .macro cbe
     28                                 asect 0;        /* Each Channel Block Entry has this format */
     29                                 word c..tcb;    /* TCB Address */
     30                                 word c..tag;    /* Prototype Protocol Header */
     31                                 word c..rts;    /* RTS Byte Count */
     32                                 word c..rsw;    /* Read Status Word */
     33                                 word c..qrd, 2; /* Read QIO Queue Header */
     34                                 word c..wsw;    /* Write Status Word */
     35                                 word c..qwr, 2; /* Write QIO Queue Header */
     36                         .endm
```

```
388   NON-POOL DATA BASE DEFINITIONS  MACRO V04.00  25-AUG-82 17:30:56 PAGE 10
389   |                FORMAT DEFINITIONS
390
391        1                                     ;
392        2                                     ;  Channel Block Table Definition
393        3                                     ;
394        4                                     ;       The offsets defined here must be reflected by routines
395        5                                     ;       SETRCB, SETXCB, etc.
396        6                                     ;       Unlabelled words are plugged at driver load time at UNPWF.
397        7                                     ;
398        8 000012  000000                      rstate: .word   0       ; address of receiver state sub-table
399        9 000014  000000                      rdbcxs: .word   0       ; excess read byte count (see rcv)
400       10 000016  000000                      xstate: .word   0       ; address of transmitter state sub-table
401       11 000020  000000                      wrbcxs: .word   0       ; excess write byte count (see xmt)
402       12 000022  000000                              .word   0       ; Address of Receive $FORK1 Context Block
403       13 000024  000000                              .word   0       ; Address of Transmit $FORK1 Context Block
404       14 000026  000000                              .word   0       ; Address of Receive Command Protocol Block
405       15 000030  000000                              .word   0       ; Address of Transmit Command Protocol Block
406       16 000032                          un0tbl:
407       17         000041                        .rept unpcnt
408       18                                             cbe
409       19                                       .endr
410       20
411       21                                     ;  Special Debug Channel Block
412       22 001154                              dbugcb: cbe
413       23
414       24                                     .sbttl |
415       25                                     .title XQ/CP Register Definitions
416
```

```
416  XQ/CP REGISTER DEFINITIONS       MACRO V04.00  25-AUG-82 17:30:56 PAGE 11
417  |       ACC XQ/CP REGISTER DEFINITIONS
418
419          1                              .sbttl |        ACC XQ/CP Register Definitions
420          2
421          3                              ;  Receive XQ/CP Registers
422          4          176200              rcsr = 0176200;           /* Receive Command/Status Register */
423          5          100000                RERR   = 0100000;       /* Composite Read Error Flag */
424          6          040000                RNXM   = 0040000;       /* Missed! */
425          7          036000                RSTS   = 0036000;       /* Status Bits. Unused */
426          8          001000                RHALT  = 0001000;       /* Z80 is Halted */
427          9          000400                RDBF   = 0000400;       /* Data Available */
428         10          000200                RRDY   = 0000200;       /* Ready to Go */
429         11          000100                RIEN   = 0000100;       /* Interrupt Enable */
430         12          000060                RADX   = 0000060;       /* Extended Address Bits for Read */
431         13                                ;        0000010;       /* UNUSED */
432         14          000004                RSIG   = 0000004;       /* Lights to Cause Receive Interrupt */
433         15          000002                RRESET = 0000002;       /* Set to Init Receive Side */
434         16          000001                RGO    = 0000001;       /* Command Strobe */
435         17          176202              rdb = rcsr + 2;
436         18          176204              rba = rdb + 2;
437         19          176206              rwc = rba + 2;
438         20
439         21                              ;  Transmit XQ/CP Registers
440         22          176210              xcsr = 0176210;           /* Transmit Command/Status Register */
441         23          100000                XERR   = 0100000;       /* Composite Transmit Error Flag */
442         24          040000                XNXM   = 0040000;       /* Missed Again! */
443         25          036000                XSTS   = 0036000;       /* Status Bits. Unused */
444         26                                ;        0001000;       /* UNUSED */
445         27          000400                XDBF   = 0000400;       /* Ready to Accept Data */
446         28          000200                XRDY   = 0000200;       /* Ready to Go */
447         29          000100                XIEN   = 0000100;       /* Interrupt Enable */
448         30          000060                XADX   = 0000060;       /* Extended Address Bits for Transmit */
449         31          000010                X280   = 0000010;       /* Set to Reset Z80 */
450         32          000004                XSIG   = 0000004;       /* Set to Interrupt Z80 */
451         33          000002                XRESET = 0000002;       /* Set to Init Transmit Side */
452         34          000001                XGO    = 0000001;       /* Command Strobe */
453         35          176212              xdb = xcsr + 2;
454         36          176214              xba = xdb + 2;
455         37          176216              xwc = xba + 2;
456         38
457         39
458         40                              ;  Convience Definitions
459         41          000004              ba = rba - rcsr;
460         42          000006              wc = rwc - rcsr;
461         43
462         44
463         45                              ;  Definitions for RSX-11M Data Base
464         46          176210              un0csr = xcsr;
465         47          000200              un0pri = PR4;
466         48          000140              un0vec = 0140;
467         49
468         50                              .if df DVLP
469         51                                RRESET = 0;
470         52                                XRESET = 0;
471         53                                X280 = 0;
472         54                              .endc
473
```

```
473  XQ/CP REGISTER DEFINITIONS      MACRO V04.00  25-AUG-82 17:30:56 PAGE 12
474  |         ACC XQ/CP REGISTER DEFINITIONS
475
476          1                                 ;
477          2                                 ;  Finite State Automaton Event Code Numbers
478          3                                 ;
479          4          000000                 READ    = 0;    /* Read Request */
480          5          000001                 CTS     = 1;    /* Clear-to-Send */
481          6
482          7          000000                 WRITE   = 0;    /* Write Request */
483          8          000001                 RTS     = 1;    /* Request-to-Send */
484          9
485         10          000002                 LABORT  = 2;    /* Local Abort */
486         11          000003                 RABORT  = 3;    /* Remote Abort */
487         12          000004                 ABOACK  = 4;    /* Abort ACK */
488         13          000005                 MORE    = 5;    /* More Data to come */
489         14          000006                 IOC     = 6;    /* I/O Completion and/or End-of-Stream */
490         15
491         16          000000                 XMTCMD = 0;     /* Send RTS, CTS, etc. */
492         17          000001                 XMTDFE = 1;     /* Send some flavor of a DATA FOLLOWS message */
493         18          000002                 XMTIOC = 2;     /* I/O Completion Event
494         19
495         20          000000                 RCVCMD = 0;     /* Received RTS, CTS, etc. */
496         21          000001                 RCVDFE = 1;     /* Received some flavor of DATA FOLOWS */
497         22
498         23                                 .sbttl |
499         24                                 .title Device Driver Major Routines
500         25                                 .sbttl | RSX-11M Device Driver Major Routines
501         26
502
```

```
DEVICE DRIVER MAJOR ROUTINES    MACRO V04.00  25-AUG-82 17:30:56 PAGE 13
|       POWER-FAIL / DRIVER LOAD

                                     .sbttl |        Power-Fail / Driver Load
                                     ;
                                     ;       This routine is called upon device driver load.
                                     ;       Thus, it is a handy place to put a breakpoint for debugging.
                                     ;       Also, general device initialization.
                                     ;
                                     ;       Entry:
                                     ;               R3 = controller index
                                     ;               R4 ->SCB
                                     ;               R5 ->UCB
 001176                              unpwf:
                                     ;  Allow other driver breakpoints to be set at load time
                                     .iif df DEBUG BPT

                                     ;  Buy blocks from executive pool, but just once.
 001176  005767  176626                      tst     un0tbl-2
 001202  001002                              bne     30$             ; (already allocated)
 001204  004767  004134                      jsr     pc,setpool      ; setup pool blocks
 001210                              30$:

                                     ;  clear channel blocks.
 001210  012703  005260'                     mov     #initcb,r3      ; R3 ->Channel Block Initialization Routine
 001214  004767  003714                      jsr     pc,do4all       ; init all channel blocks

                                     ;  flag debug channel block
 001220  052767  000010  177730              bis     #p.debug,dbugcb+c..tag

                                     ;  init RSTATE & XSTATE
 001226  012767  003456' 176556              mov     #rccmd,rstate   ; rstate is address of state subtable
 001234  012767  004110' 176554              mov     #xmidle,xstate  ; xstate is address of state subtable

                                     ;  Reset Harware Pipes and Processor
 001242  052767  000002  176200              bis     #RRESET,RCSR
 001250  052767  000012  176210              bis     #XRESET!XZ80,XCSR

                                     ;  Enable Interrupts
 001256  052767  000100  176200              bis     #RIEN,RCSR
 001264  052767  000100  176210              bis     #XIEN,XCSR

                                     ;  Issue first read
 001272  012700  000004                      mov     #4,r0           ; R0 = Byte Count
 001276  004767  004256                      jsr     pc,setrcb       ; R1 ->Receive Block
 001302  005002                              clr     r2              ; R2 = extenede address bits
 001304  004767  003332                      jsr     pc,rcv
 001310  000207                              rts     pc

                                      .sbttl |
                                      .sbttl |        I/O Packet Processor
```

```
DEVICE DRIVER MAJOR ROUTINES     MACRO V04.00  25-AUG-82 17:30:56 PAGE 14
|                ENTRY AND VALIDATION

        1                                       .sbttl |                Entry and Validation
        2                                       ;
        3                                       ;  Here upon QIO
        4                                       ;
        5                                       ;       Accept a QIO request and dispatch processing for it.
        6                                       ;       NOTE: This is very different from the usual device driver handling.
        7                                       ;
        8                                       ;       Entry:
        9                                       ;               R1 ->I/O Packet
       10                                       ;               R4 ->SCB
       11                                       ;               R5 ->UCB
       12 001312                                unini:
       13 001312  016100  000012                        mov     i.fcn(r1),r0    ; get the QIO Function Code from the I/O Packet
       14 001316  140000                                bicb    r0,r0           ; (drop subfunction bits for dispatch compare)
       15 001320                                        dispatch IO.CON,10$
       16 001326                                        dispatch IO.DSC,20$
       17 001334                                        dispatch IO.DBR,30$
       18 001342                                        dispatch IO.RLB,31$
U          001342  022700  000000                        cmp     #IO.RLB,r0
       19 001350                                        dispatch IO.DBW,40$
       20 001356                                        dispatch IO.WLB,41$
U          001356  022700  000000                        cmp     #IO.WLB,r0
       21                                       ;
       22                                       ;  Invalid Request (includes RSX ATTACH and DETACH)
       23                                       ;
U       24 001364  012700  000000                        mov     #IE.IFC&377,r0  ; INVALID FUNCTION CODE
       25 001370  000512                        9$:     br      90$             ; call $iofin and return
       26
       27
       28                                       .sbttl |
       29                                       .sbttl |                Non-Transfer Requests
```

```
DEVICE DRIVER MAJOR ROUTINES    MACRO V04.00  25-AUG-82 17:30:56 PAGE 15
|                         CONNECT REQUEST

      1                                    .sbttl |                       CONNECT Request
      2                                    ;
      3                                    ;       Provide single-access functionality lost in making the unit selection
      4                                    ;       a parameter of the QIO request.
      5                                    ;
      6                                    ;       Entry:
      7                                    ;               R1 ->I/O Packet
      8                                    ;               R4 ->SCB
      9                                    ;               R5 ->UCB
     10 001372                             10$:
     11 001372  004767  004240                     jsr     pc,setqrc       ; R2 ->Channel Block
     12 001376  103507                             bcs     90$
U    13 001400  012700  000000                     mov     #IE.DAA&377,r0  ; presume DEVICE ALREADY ATTACHED
     14 001404  005762  000000                     tst     c..tcb(r2)      ; if already allocated . . .
     15 001410  001102                             bne     90$             ; user wins error
     16 001412  016162  000004  000000             mov     i.tcb(r1),c..tcb(r2); else plug tcb address
     17 001420  000474                             br      80$             ; and return with success code
     18
     19
     20
     21
     22                                    .sbttl |                       DISCONNECT Request
     23                                    ;
     24                                    ;       Entry:
     25                                    ;               R1 ->I/O Packet
     26                                    ;               R4 ->SCB
     27                                    ;               R5 ->UCB
     28 001422                             20$:
     29                                    ;  verify channel ownership
     30 001422  004767  004210                     jsr     pc,setqrc       ; R2 ->Channel Block
     31 001426  103473                             bcs     90$
U    32 001430  012700  000000                     mov     #IE.DNA&377,r0  ; presume DEVICE NOT ATTACHED
     33 001434  026162  000004  000000             cmp     i.tcb(r1),c..tcb(r2); is this guy the owner?
     34 001442  001065                             bne     90$             ; (jerk!)
     35
     36                                    ;  disown channel & abort all i/o
     37 001444  005062  000000                     clr     c..tcb(r2)
     38 001450  004767  003516                     jsr     pc,haltio
     39 001454  000456                             br      80$
     40
     41                                    .sbttl |
DEVICE DRIVER MAJOR ROUTINES    MACRO V04.00  25-AUG-82 17:30:56 PAGE 16
|                    DATA TRANSFER REQUESTS

      1                                    .sbttl |                 Data Transfer Requests
      2                                    .sbttl |                           Debug Read Request
      3                                    ;
      4                                    ;        Entry:
      5                                    ;                  R1 ->I/O Packet
      6                                    ;                  R4 ->SCB
      7                                    ;                  R5 ->UCB
      8 001456                             30$:
      9 001456  004767  004112                      jsr     pc,setdbc       ; R2 ->Debug Channel Block
     10 001462  103455                              bcs     90$             ; use common error exit
     11 001464  000403                              br      35$             ; merge into common Read code
     12
     13
     14
     15
     16                                    .sbttl |                           Read Request
     17                                    ;
     18                                    ;        Entry:
     19                                    ;                  R1 ->I/O Packet
     20                                    ;                  R4 ->SCB
     21                                    ;                  R5 ->UCB
     22 001466                             31$:
     23 001466  004767  004160                      jsr     pc,setxrc       ; R2 ->Channel Block
     24 001472  103451                              bcs     90$
     25                                    ;        br      35$             ; (fall into 35$)
     26
     27
     28
     29
     30                                    ;
     31                                    ;  Common Read Request Code
     32                                    ;
     33                                    ;        R1 ->I/O Packet
     34                                    ;        R2 ->Channel Block
     35                                    ;        R4 ->SCB
     36                                    ;        R5 ->UCB
     37 001474                             35$:
     38                                    ;  add I/O Packet to read queue for this channel
     39 001474  010200                              mov     r2,r0
     40 001476  062700  000010                      add     #c..qrd,r0
     41 001502  010246                              mov     r2,-(sp)
     42 001504                                      CALL    $QINSF
0       001504  004767  000000                      JSR     PC,$QINSF
     43 001510  012602                              mov     (sp)+,r2
     44
     45                                    ;  Declare Read Request Event for this channel
     46 001512  016200  000006                      mov     c..rsw(r2),r0
     47 001516  012701  000000                      mov     #READ,r1
     48 001522  004767  000120                      jsr     pc,fsa
     49 001526  010062  000006                      mov     r0,c..rsw(r2)
     50
     51 001532  000435                              br      100$            ; iofin called at completion time, not now.
```

```
DEVICE DRIVER MAJOR ROUTINES    MACRO V04.00  25-AUG-82 17:30:56 PAGE 17
                               DEBUG WRITE REQUEST

      1                                      .sbttl |                        Debug Write Request
      2                                      ;
      3                                      ;       Entry:
      4                                      ;               R1 ->I/O Packet
      5                                      ;               R4 ->SCB
      6                                      ;               R5 ->UCB
      7 001534                               40$:
      8 001534  004767  004034                       jsr     pc,setdbc       ; R2 ->Debug Channel Block
      9 001540  103426                               bcs     90$             ; use common error exit
     10 001542  000403                               br      45$             ; merge with common Write code
     11
     12
     13
     14
     15                                      .sbttl |                        Write Request
     16                                      ;
     17                                      ;       Entry:
     18                                      ;               R1 ->I/O Packet
     19                                      ;               R4 ->SCB
     20                                      ;               R5 ->UCB
     21 001544                               41$:
     22 001544  004767  004102                       jsr     pc,setxrc       ; R2 ->Channel Block
     23 001550  103422                               bcs     90$             ; Error code in R0
     24                                      ;       br      45$             ; (fall into 45$)
     25
     26
     27
     28
     29                                      ;
     30                                      ;  Common Write Request Code
     31                                      ;
     32                                      ;       R1 ->I/O Packet
     33                                      ;       R2 ->Channel Block
     34                                      ;       R4 ->SCB
     35                                      ;       R5 ->UCB
     36 001552                               45$:
     37                                      ;  add I/O Packet to write queue for this channel
     38 001552  010200                               mov     r2,r0
     39 001554  062700  000016                       add     #c..qwr,r0
     40 001560  010246                               mov     r2,-(sp)
     41 001562                                       CALL    $QINSF
1)         001562  004767  000000                       JSR     PC,$QINSF
     42 001566  012602                               mov     (sp)+,r2
     43
     44                                      ;  Declare Write Request Event for this channel
     45 001570  016200  000014                       mov     c..wsw(r2),r0
     46 001574  012701  000000                       mov     #WRITE,r1
     47 001600  004767  000042                       jsr     pc,fsa
     48 001604  010062  000014                       mov     r0,c..wsw(r2)
     49
     50 001610  000406                               br      100$            ; iofin called at completion time, not now.
     51
     52                                      .sbttl |
742   DEVICE DRIVER MAJOR ROUTINES    MACRO V04.00  25-AUG-82 17:30:56 PAGE 18
743   |       COMMON QIO REQUEST EXIT CODE
744
745         1                                    .sbttl |        Common QIO Request Exit Code
746         2                                    ;
747         3                                    ;  Return Success Code
748         4                                    ;
749         5 001612                             80$:
750         6 001612  012700  000001                     mov     #1,r0
751         7                                    ;       br      90$             ; (fall into 90$)
752         8
753         9
754        10
755        11
756        12                                    ;
757        13                                    ;  Call $IOFIN
758        14                                    ;
759        15 001616                             90$:
760        16 001616  010103                             mov     r1,r3           ; R3->I/O Packet addr
761        17 001620  005001                             clr     r1              ; byte count (=0 for error)
762        18 001622                                     CALL    $IOFIN
763   U       001622  004767  000000                     JSR     PC,$IOFIN
764        19                                    ;       br      100$            ; (fall into 100$)
765        20
766        21                                    ;
767        22                                    ;  QIO Request Routine Exit
768        23                                    ;
769        24 001626  000207                     100$:   rts     pc
770        25
771        26                                    .sbttl |
772
```

```
DEVICE DRIVER MAJOR ROUTINES    MACRO V04.00  25-AUG-82 17:30:56 PAGE 19
|       CANCEL I/O

                                     .sbttl |        Cancel I/O
                                     ;
                                     ; Abort all waiting I/O for this task, perform an implicit DISCONNECT
                                     ; for every data path owned by this task, and schedule an abort sweep
                                     ; for this controller.
                                     ;
                                     ;       Entry:
                                     ;               Device interrupts locked out.
                                     ;               R1 ->TCB
                                     ;               R3 = Controller index
                                     ;               R4 ->SCB
                                     ;               R5 ->UCB
   001630                            UNCAN:
   001630  010104                            mov     r1,r4           ; R4 = TCB address
   001632  012703  005234'                  mov     #lokill,r3      ; R3 ->routine
   001636  004767  003272                   jsr     pc,do4all       ; scan all channels
   001642  000207                           rts     pc




                                     .sbttl |
                                     .sbttl |        Device Timeout

   001644                            untmo:
   001644  000207                           rts     pc      ; **** UNUSED ****

                                     .sbttl |
                                     .sbttl |
                                     .title XQ/CP FSA
                                     .sbttl | RSX-11M XQ/CP Device Driver Protocol Finite State Automaton

```

```
XQ/CP FSA       MACRO V04.00  25-AUG-82 17:30:56 PAGE 20
|       FINITE STATE AUTOMATON STATE TRANSITION DRIVER

                                .sbttl |        Finite State Automaton State Transition Driver
                                .sbttl |                fsa - execute state routine indexed by event
                                ;
                                ;       Entry:
                                ;               R0 ->State Transition Subtable
                                ;               R1 = Event Code Number
                                ;
                                ;       Exit:
                                ;               R0 ->New State Transition Subtable
   001646                       fsa:
                                ;  Preserve registers
   001646  010546                       mov     r5,-(sp)
   001650  010446                       mov     r4,-(sp)
   001652  010346                       mov     r3,-(sp)
   001654  010246                       mov     r2,-(sp)

                                ;  Convert event code to routine address pointer
   001656  060101                       add     r1,r1
   001660  060101                       add     r1,r1
   001662  060100                       add     r1,r0

                                ;  service the event
   001664  012046                       mov     (r0)+,-(sp)     ; Save new state value
   001666  004730                       jsr     pc,@(R0)+       ; call routine
   001670  012600                       mov     (sp)+,r0        ; return new state value

                                ;  Restore caller's registers
   001672  012602                       mov     (sp)+,r2
   001674  012603                       mov     (sp)+,r3
   001676  012604                       mov     (sp)+,r4
   001700  012605                       mov     (sp)+,r5
   001702  000207               ignore: rts     pc
```

```
XQ/CP FSA       MACRO V04.00  25-AUG-82 17:30:56 PAGE 21
|       READ STATE TABLES

      1                                 .sbttl |        Read State Tables
      2                                 ;  (Wait for Read or RTS)
      3 001704  001774' 002154'         rdidle: .word   RTSwait, idlerd         ;*READ
      4 001710  001740' 002170'                 .word   rdwait,  idlrts         ;*RTS
      5 001714  002064' 002552'                 .word   rdapn,   rdsabo         ; LOCAL ABORT
      6 001720  001704' 002570'                 .word   rdidle,  rdsaba         ; REMOTE ABORT
      7 001724  002064' 002552'                 .word   rdapn,   rdsabo         ; ABORT ACK
      8 001730  000001  005132'                 .word      1,    die            ; MORE
      9 001734  000002  005132'                 .word      2,    die            ; IOC
     10
     11                                 ;  (Read Wait)
     12 001740  002030' 002224'         rdwait: .word   rdiop,   gotrd          ;*READ
     13 001744  002064' 002552'                 .word   rdapn,   rdsabo         ; RTS
     14 001750  002120' 002552'                 .word   rdaps,   rdsabo         ; LOCAL ABORT
     15 001754  001704' 002570'                 .word   rdidle,  rdsaba         ; REMOTE ABORT
     16 001760  002064' 002552'                 .word   rdapn,   rdsabo         ; ABORT ACK
     17 001764  000003  005132'                 .word      3,    die            ; MORE
     18 001770  000004  005132'                 .word      4,    die            ; IOC
     19
     20                                 ;  (RTS Wait)
     21 001774  001774' 001702'         RTSwait:.word   RTSwait, ignore         ; READ
     22 002000  002030' 002270'                 .word   rdiop,   gotRTS         ;*RTS
     23 002004  002120' 002552'                 .word   rdaps,   rdsabo         ; LOCAL ABORT
     24 002010  001704' 002522'                 .word   rdidle,  rdabort        ; REMOTE ABORT
     25 002014  002120' 002552'                 .word   rdaps,   rdsabo         ; ABORT ACK
     26 002020  000005  005132'                 .word      5,    die            ; MORE
     27 002024  000006  005132'                 .word      6,    die            ; IOC
     28
     29                                 ;  (Read I/O in Progress)
     30 002030  002030' 001702'         rdiop:  .word   rdiop,   ignore         ; READ
     31 002034  002120' 002552'                 .word   rdaps,   rdsabo         ; RTS
     32 002040  002120' 002552'                 .word   rdaps,   rdsabo         ; LOCAL ABORT
     33 002044  001704' 002522'                 .word   rdidle,  rdabort        ; REMOTE ABORT
     34 002050  002120' 002552'                 .word   rdaps,   rdsabo         ; ABORT ACK
     35 002054  001774' 002352'                 .word   RTSwait, rmore          ;*MORE
     36 002060  001704' 002404'                 .word   rdidle,  rioc           ;*IOC
     37
     38                                 ;  (Read Abort Pending, No Signal)
     39 002064  002120' 001702'         rdapn:  .word   rdaps,   ignore         ; READ
     40 002070  002064' 001702'                 .word   rdapn,   ignore         ; RTS
     41 002074  002120' 001702'                 .word   rdaps,   ignore         ; LOCAL ABORT
     42 002100  002064' 002570'                 .word   rdapn,   rdsaba         ; REMOTE ABORT
     43 002104  001704' 001702'                 .word   rdidle,  ignore         ;*ABORT ACK
     44 002110  002064' 001702'                 .word   rdapn,   ignore         ; MORE
     45 002114  002064' 001702'                 .word   rdapn,   ignore         ; IOC
     46
     47                                 ;  (Read Abort Pending and Signal)
     48 002120  002120' 001702'         rdaps:  .word   rdaps,   ignore         ; READ
     49 002124  002120' 001702'                 .word   rdaps,   ignore         ; RTS
     50 002130  002120' 001702'                 .word   rdaps,   ignore         ; LOCAL ABORT
     51 002134  002120' 002570'                 .word   rdaps,   rdsaba         ; REMOTE ABORT
     52 002140  001704' 002540'                 .word   rdidle,  rraps          ;*ABORT ACK
     53 002144  002120' 001702'                 .word   rdaps,   ignore         ; MORE
     54 002150  002120' 001702'                 .word   rdaps,   ignore         ; IOC
     55
```

```
  899        56                    .sbttl |
  900        57                    .sbttl | Finite State Automaton Read Routines
  901
```

```
901   XQ/CP FSA        MACRO V04.00  25-AUG-82 17:30:56 PAGE 22
902   |         IDLERD - READ REQUEST WITH NO PENDING RTS
903
904          1                                      .sbttl |        idlerd - Read request with no pending RTS
905          2                                      ;
906          3                                      ;       Copy count of desired bytes and clear received byte count
907          4                                      ;
908          5                                      ;       Entry:
909          6                                      ;               R2 ->Channel Block
910          7                                      ;
911          8                                      ;       Exit:
912          9                                      ;               R2 Preserved
913         10 002154                               idlerd:
914         11 002154  016201  000010                       mov     c..qrd(r2),r1   ; R1 ->I/O packet
915         12 002160  016161  000030  000034               mov     i..cnt(r1),i..xfr(r1); note desired byte count
916         13 002166  000207                               rts     pc
917         14
918         15
919         16
920         17
921         18                                      .sbttl |        idlrts - RTS Received with no outstanding read request
922         19                                      ;
923         20                                      ;       Note RTS byte-count and User Subfunction bits and return
924         21                                      ;
925         22                                      ;       Entry:
926         23                                      ;               R2 ->Channel Block
927         24                                      ;
928         25                                      ;       Exit:
929         26                                      ;               R2 Preserved
930         27 002170                               idlrts:
931         28                                      ;  copy user subfunction bits into channel block tag word
932         29 002170  004767  003364                       jsr     pc,setrcb       ; R1 ->Receive Command Block
933         30 002174  011100                               mov     @r1,r0
934         31 002176  042700  177417                       bic     #m$usbf,r0      ; isolate user subfunction bits
935         32 002202  042762  000360  000002               bic     #^Cm$usbf,c..tag(r2)
936         33 002210  050062  000002                       bis     r0,c..tag(r2)   ; store them in prototype tag word
937         34
938         35                                      ;  note RTS byte count
939         36 002214  016162  000002  000004               mov     2(r1),c..rts(r2); note byte count
940         37 002222  000207                               rts     pc
941         38
942         39
943         40
944         41
945         42                                      .sbttl |        gotrd - Read Request after previous RTS
946         43                                      ;
947         44                                      ;       Arbitrate byte count from previous RTS and send CTS
948         45                                      ;
949         46                                      ;       Entry:
950         47                                      ;               R2 ->Channel Block
951         48                                      ;
952         49                                      ;       Exit:
953         50                                      ;               CTS message queued to transmitter
954         51                                      ;
955         52 002224                               gotrd:
956         53                                      ;  note desired byte count and clear received byte count
957         54 002224  016201  000010                       mov     c..qrd(r2),r1   ; R1 ->I/O Packet
958         55 002230  016161  000030  000034               mov     i..cnt(r1),i..xfr(r1)
```

```
  959        56
  960        57                          ; arbitrate byte count
  961
```

```
XQ/CP FSA          MACRO V04.00  25-AUG-82 17:30:56 PAGE 22-1
|          QUTRD - READ REQUEST AFTER PREVIOUS RTS

     58 002236  016200  000004                  mov     c..rts(r2),r0    ; get RTS byte count in R0
     59 002242  016101  000034                  mov     i..xfr(r1),r1    ; R1 = read request byte count
     60 002246  004767  002562                  jsr     pc,calcbc        ; R1 = min(R0, R1)
     61 002252  010162  000004                  mov     r1,c..rts(r2)    ; remember actual byte count of tranfer
     62
     63                                 ;  Send CTS to other side
     64 002256  012703  000000                  mov     #p.cts,r3        ; R3 = Message Type
     65 002262  004767  002052                  jsr     pc,sndrd         ; (byte count already in R1)
     66 002266  000207                          rts     pc
     67
     68
     69
     70
     71                                 .sbttl |        gotrts - RTS received with read pending
     72                                 ;
     73                                 ;       Arbitrate byte count and send CTS
     74                                 ;
     75                                 ;       Entry:
     76                                 ;               R2 ->Channel Block
     77                                 ;
     78                                 ;       Exit:
     79                                 ;               CTS message queued to transmitter
     80 002270                           gotrts:
     81                                 ;  copy user subfunction bits into channel block tag word
     82 002270  004767  003264                  jsr     pc,setrcb        ; R1 ->Receive Command Block
     83 002274  011100                          mov     @r1,r0
     84 002276  042700  177417                  bic     #m$usbf,r0       ; isolate user subfunction bits
     85 002302  042762  000360  000002          bic     #^Cm$usbf,c..tag(r2)
     86 002310  050062  000002                  bis     r0,c..tag(r2)    ; store them in prototype tag word
     87
     88                                 ;  arbitrate byte count
     89 002314  016100  000002                  mov     2(r1),r0         ; R0 = RTS byte count
     90 002320  016201  000010                  mov     c..qrd(r2),r1
     91 002324  016101  000034                  mov     i..xfr(r1),r1    ; R1 = read request byte count
     92 002330  004767  002500                  jsr     pc,calcbc        ; select minimum
     93 002334  010162  000004                  mov     r1,c..rts(r2)    ; remember actual transfer byte count
     94
     95                                 ;  Send CTS to other side
     96 002340  012703  000000                  mov     #p.cts,r3        ; R3 = Message Type
     97 002344  004767  001770                  jsr     pc,sndrd         ; (R1 already holds byte count)
     98 002350  000207                          rts     pc
     99
    100
    101
    102
    103                                 .sbttl |        rmore - more data to come
    104                                 ;
    105                                 ;       Add count of bytes transfered and wait for next RTS
    106                                 ;
    107                                 ;       Entry:
    108                                 ;               R2 ->Channel Block
    109 002352                           rmore:
    110 002352  016201  000010                  mov     c..qrd(r2),r1    ; R1 ->I/O packet
    111 002356  016200  000004                  mov     c..rts(r2),r0    ; R0 = transfer byte count
    112 002362  160061  000034                  sub     r0,i..xfr(r1)    ; count down bytes to receive
```

```
1019         113 002366  060061  000026                   add     r0,i..buf+2(r1) ; update buffer address
1020         114 002372  103003                           bcc     10$
1021
```

```
    XQ/CP FSA       MACRO V04.00  25-AUG-82 17:30:56 PAGE 22-2
    !       XMORE - MORE DATA TO COME

        115 002374  062761  000020  000024          add     #020,i..buf(r1) ; (18-bit format)
        116 002402  000207                  10$:    rts     pc
        117
        118
        119
        120
        121                                 .sbttl !        rioc - Read Completion
        122                                 ;
        123                                 ;       Entry:
        124                                 ;               R2 ->Channel Block
        125 002404                          rioc:
        126                                 ;  get first I/O Packet
        127 002404  010200                          mov     r2,r0
        128 002406  062700  000010                  add     #c..qrd,r0
        129 002412  010246                          mov     r2,-(sp)        ; (save channel block pointer)
        130 002414                                  CALL    $QRMVF          ; (returns pointer in R1)
    U       002414  004767  000000                  JSR     PC,$QRMVF
        131 002420  103001                          bcc     10$
        132 002422                                  CRASH
        133 002424  010103                  10$:    mov     r1,r3           ; ($IOFIN wants I/O Packet addr in R3)
        134 002426  012602                          mov     (sp)+,r2        ; (get stacked R2)
        135 002430  166261  000004  000034          sub     c..rts(r2),i..xfr(r1)
        136
        137                                 ;  return user subfunction bits
        138 002436  016200  000002                  mov     c..tag(r2),r0
        139 002442  010001                          mov     r0,r1           ; (copy tag word for later test)
        140 002444  042700  177417                  bic     #m$usbf,r0
        141 002450  000300                          swab    r0              ; (CS is upper half)
        142
        143                                 ;  return completion code
        144 002452  004767  003102                  jsr     pc,setrcb
        145 002456  011101                          mov     @r1,r1          ; get DF or DFE code
        146 002460  042701  177770                  bic     #m$mtyp,r1
        147 002464  022701  000003                  cmp     #p.dfe,r1       ; END?
        148 002470  001403                          beq     20$             ; EQ - yes
    U   149 002472  052700  000000                  bis     #IS.TNC,r0      ; else indicate MORE
        150 002476  000402                          br      30$
    U   151 002500  052700  000000          20$:    bis     #IS.SUC,r0      ; indicate END
        152 002504                          30$:
        153
        154                                 ;  return byte count
        155 002504  016301  000030                  mov     i..cnt(r3),r1   ; R1 = byte count
        156 002510  166301  000034                  sub     i..xfr(r3),r1   ; minus residual byte count
        157 002514                                  CALL    $IOFIN          ; R3 ->I/O Packet
    U       002514  004767  000000                  JSR     PC,$IOFIN
        158 002520  000207                          rts     pc
        159
        160
        161
        162
        163                                 .sbttl !        rdabort - received remote read abort
        164                                 ;
        165                                 ;       Entry:
        166                                 ;               R2 ->Channel Block
        167                                 ;
```

```
 1079       168 002522                    rdabort:
 1080       169                           ;  send abort ACK and abort all I/O Packets
 1081
```

```
XQ/CP FSA        MACRO V04.00  25-AUG-82 17:30:56 PAGE 22-3
|       RDABORT - RECEIVED REMOTE READ ABORT

    170 002522  004767  000042              jsr     pc,rdsaba
    171 002526  012700  000010              mov     #c..qrd,r0
    172 002532  004767  002232              jsr     pc,drain
    173 002536  000207                      rts     pc
    174
    175
    176
    177
    178                             .sbttl |        rraps - finish up read abort
    179                             ;
    180                             ;       Entry:
    181                             ;               R2 ->Channel Block
    182 002540                      rraps:
    183 002540  012700  000010              mov     #c..qrd,r0
    184 002544  004767  002220              jsr     pc,drain
    185 002550  000207                      rts     pc
    186
    187
    188
    189
    190                             .sbttl |        rdsabo - Send Read Abort
    191                             ;
    192                             ;       Entry:
    193                             ;               R2 ->Channel Block
    194                             ;
    195                             ;       Exit:
    196                             ;               (FSA preserves)
    197 002552                      rdsabo:
    198                             ;   Send Abort to other side
    199 002552  005000                      clr     r0
    200 002554  005001                      clr     r1
    201 002556  012703  000004              mov     #p.abort,r3     ; R3 = message code
    202 002562  004767  001552              jsr     pc,sndrd
    203 002566  000207                      rts     pc
    204
    205
    206
    207
    208                             .sbttl |        rdsaba - Send Read Abort ACK
    209                             ;
    210                             ;       Entry:
    211                             ;               R2 ->Channel Block
    212                             ;
    213                             ;       Exit:
    214                             ;               (FSA preserves)
    215 002570                      rdsaba:
    216 002570  005000                      clr     r0
    217 002572  005001                      clr     r1
    218 002574  012703  000005              mov     #p.aback,r3
    219 002600  004767  001534              jsr     pc,sndrd
    220 002604  000207                      rts     pc
```

```
XQ/CP FSA        MACRO V04.00  25-AUG-82 17:30:56 PAGE 23
|

      1                             .sbttl |
      2                             .sbttl |        Write State Tables
      3                             ;  (Wait for Write Request)
      4 002606  002642' 003022'     wrwait: .word  CTSwait, gotwr          ;*WRITE
      5 002612  002732' 003302'             .word  wrapn,   wrsabo         ; CTS
      6 002616  002766' 003302'             .word  wraps,   wrsabo         ; LOCAL ABORT
      7 002622  002606' 003320'             .word  wrwait,  wrsaba         ; REMOTE ABORT
      8 002626  002732' 003302'             .word  wrapn,   wrsabo         ; ABORT ACK
      9 002632  000007  005132'             .word    7,     die            ; MORE
     10 002636  000010  005132'             .word   10,     die            ; IOC
     11
     12                             ;  (Wait for Clear-to-Send)
     13 002642  002642' 001702'     CTSwait:.word  CTSwait, ignore         ; WRITE
     14 002646  002676' 003056'             .word  wriop,   gotCTS         ;*CTS
     15 002652  002766' 003302'             .word  wraps,   wrsabo         ; LOCAL ABORT
     16 002656  002606' 003252'             .word  wrwait,  wrabo          ; REMOTE ABORT
     17 002662  002766' 003302'             .word  wraps,   wrsabo         ; ABORT ACK
     18 002666  000011  005132'             .word   11,     die            ; MORE
     19 002672  000012  005132'             .word   12,     die            ; IOC
     20
     21                             ;  (Write I/O in Progress)
     22 002676  002676' 001702'     wriop:  .word  wriop,   ignore         ; WRITE
     23 002702  002766' 003302'             .word  wraps,   wrsabo         ; CTS
     24 002706  002766' 003302'             .word  wraps,   wrsabo         ; LOCAL ABORT
     25 002712  002606' 003252'             .word  wrwait,  wrabo          ; REMOTE ABORT
     26 002716  002766' 003302'             .word  wraps,   wrsabo         ; ABORT ACK
     27 002722  002642' 003136'             .word  CTSwait, wmore          ;*MORE
     28 002726  002606' 003210'             .word  wrwait,  wioc           ;*IOC
     29
     30                             ; (Write Abort Pending, No Signal)
     31 002732  002766' 001702'     wrapn:  .word  wraps,   ignore         ; WRITE
     32 002736  002732' 001702'             .word  wrapn,   ignore         ; CTS
     33 002742  002766' 001702'             .word  wraps,   ignore         ; LOCAL ABORT
     34 002746  002732' 003320'             .word  wrapn,   wrsaba         ; REMOTE ABORT
     35 002752  002606' 001702'             .word  wrwait,  ignore         ;*ABORT ACK
     36 002756  002732' 001702'             .word  wrapn,   ignore         ; MORE
     37 002762  002732' 001702'             .word  wrapn,   ignore         ; IOC
     38
     39                             ; (Write Abort Pending and Signal)
     40 002766  002766' 001702'     wraps:  .word  wraps,   ignore         ; WRITE
     41 002772  002766' 001702'             .word  wraps,   ignore         ; CTS
     42 002776  002766' 001702'             .word  wraps,   ignore         ; LOCAL ABORT
     43 003002  002766' 003320'             .word  wraps,   wrsaba         ; REMOTE ABORT
     44 003006  002606' 003270'             .word  wrwait,  wrabort        ;*ABORT ACK
     45 003012  002766' 001702'             .word  wraps,   ignore         ; MORE
     46 003016  002766' 001702'             .word  wraps,   ignore         ; IOC
     47
     48                             .title FSA Write Routines
     49                             .sbttl |
     50                             .sbttl | Finite State Automaton Write Routines
```

```
FSA WRITE ROUTINES      MACRO V04.00  25-AUG-82 17:30:56 PAGE 24
|       GOTWR - WRITE REQUEST

      1                                          .sbttl |       gotwr - Write Request
      2                                          ;
      3                                          ;       Send RTS for QIO byte count
      4                                          ;
      5                                          ;       Entry:
      6                                          ;               R2 ->Channel Block
      7 003022                                   gotwr:
      8                                          ; Get QIO byte count and send RTS to other side
      9 003022  016201  000016                          mov     c..qwr(r2),r1   ; (R1 ->I/O Packet)
     10 003026  016100  000012                          mov     i.fcn(r1),r0    ; R0 = QIO code plus user subfunction bits
     11 003032  016161  000030  000034                  mov     i..cnt(r1),i..xfr(r1)
     12 003040  016101  000034                          mov     i..xfr(r1),r1   ; R1 = Byte Count
     13 003044  012703  000001                          mov     #p.cts,r3       ; R3 = Message Type
     14 003050  004767  001304                          jsr     pc,sndwr
     15 003054  000207                                  rts     pc
     16
     17
     18
     19
     20                                          .sbttl |       gotCTS - Received Clear-to-Send
     21                                          ;
     22                                          ;       Entry:
     23                                          ;               R2 ->Channel Block
     24 003056                                   gotcts:
     25                                          ; arbitrate byte count
     26 003056  004767  002476                          jsr     pc,setrcb
     27 003062  016101  000002                          mov     2(r1),r1        ; R1 = Byte Count
     28
     29                                          ; Send "DATA FOLLOWS" or "DATA FOLLOWS and END" to other side
     30 003066  016200  000016                          mov     c..qwr(r2),r0   ; R0 ->I/O Packet
     31 003072  012703  000002                          mov     #p.df,r3        ; select "Data Follows" message type
     32 003076  020160  000034                          cmp     r1,i..xfr(r0)   ; size we wanted?
     33 003102  001006                                  bne     10$             ; no, don't end
     34 003104  032760  000004  000012                  bit     #sf.end,i.fcn(r0); does user want end?
     35 003112  001402                                  beq     10$             ; EQ - no
     36 003114  012703  000003                          mov     #p.dfe,r3       ; else select "Data Follows and End" message type
     37 003120  160160  000034                   10$:   sub     r1,i..xfr(r0)
     38 003124  016000  000012                          mov     i.fcn(r0),r0
     39 003130  004767  001314                          jsr     pc,snddata      ; R1 = byte count, R3 = Message Type
     40 003134  000207                                  rts     pc
     41
     42
     43
     44
     45                                          .sbttl |       wmore - Write Completion but not for user task
     46                                          ;
     47                                          ;       Entry:
     48                                          ;               R2 ->Channel Block
     49 003136                                   wmore:
     50                                          ; update I/O Packet
     51 003136  004767  002424                          jsr     pc,setxcb       ; R1 ->Transmit Block
     52 003142  016100  000002                          mov     2(r1),r0        ; R0 = transfer byte count
     53 003146  016201  000016                          mov     c..qwr(r2),r1   ; R1 ->I/O packet
     54 003152  060061  000026                          add     r0,i..buf+2(r1) ; update buffer address
     55 003156  103001                                  bcc     10$
```

```
 1246        56 003160  062761  000020  000024          add     #020,1..buf(r1) ; (18-bit format)
 1247        57 003166                          10$:
 1248
```

```
FSA WRITE ROUTINES      MACRO V04.00  25-AUG-82 17:30:56 PAGE 24-1
|        WMORE - WRITE COMPLETION BUT NOT FOR USER TASK

         58
         59                              ;  send RTS for residual bytes
         60 003166  016100  000012                mov     i.fcn(r1),r0    ; R0 = user subfunction bits
         61 003172  016101  000034                mov     i..xfr(r1),r1   ; R1 = byte count
         62 003176  012703  000001                mov     #p.rts,r3       ; R3 = message type
         63 003202  004767  001152                jsr     pc,sndwr
         64 003206  000207                        rts     pc
         65
         66
         67
         68
         69                              .sbttl |        wioc - Write Completion
         70                              ;
         71                              ;       Entry:
         72                              ;               R2 ->Channel Block
         73                              ;
         74                              ;       Exit:
         75                              ;               (FSA Preserves)
         76 003210                       wioc:
         77                              ;  use first I/O Packet
         78 003210  010200                        mov     r2,r0
         79 003212  062700  000016                add     #c..qwr,r0
         80 003216                                CALL    $QRMVF
U           003216  004767  000000                JSR     PC,$QRMVF
         81 003222  103001                        bcc     10$
         82 003224                                CRASH
         83 003226                       10$:
         84
         85                              ;  give I/O Completion to user
         86 003226  010103                        mov     r1,r3           ; R3 ->I/O Packet
         87 003230  016301  000030                mov     i..cnt(r3),r1   ; R1 = byte count (just GOTTA be zero!)
         88 003234  166301  000034                sub     i..xfr(r3),r1
         89 003240  012700  000001                mov     #1,r0           ; R0 = Completion Code
         90 003244                                CALL    $IOFIN
U           003244  004767  000000                JSR     PC,$IOFIN
         91
         92 003250  000207                        rts     pc
         93
         94
         95
         96
         97                              .sbttl |        wrabo - received remote write abort
         98                              ;
         99                              ;       Entry:
        100                              ;               R2 ->Channel Block
        101                              ;
        102                              ;       Exit:
        103                              ;               (FSA preserves)
        104 003252                       wrabo:
        105 003252  004767  000042                jsr     pc,wrsaba
        106 003256  012700  000016                mov     #c..qwr,r0
        107 003262  004767  001502                jsr     pc,drain
        108 003266  000207                        rts     pc
        109
        110
```

```
 1306        111
 1307        112
 1308
```

```
FSA WRITE ROUTINES      MACRO V04.00  25-AUG-82 17:30:56 PAGE 24-2
!       WRABORT - FINISH UP WRITE ABORT

    113                                 .sbttl !        wrabort - finish up write abort
    114                                 ;
    115                                 ;       Entry:
    116                                 ;               R2 ->Channel Block
    117                                 ;
    118                                 ;       Exit:
    119                                 ;               (FSA preserves)
    120 003270                          wrabort:
    121 003270  012700  000016                  mov     #c..qwr,r0
    122 003274  004767  001470                  jsr     pc,drain
    123 003300  000207                          rts     pc
    124
    125
    126
    127
    128                                 .sbttl !        wrsabo - Send Write Abort
    129                                 ;
    130                                 ;       Entry:
    131                                 ;               R2 ->Channel Block
    132                                 ;
    133                                 ;       Exit:
    134                                 ;               (FSA preserves)
    135 003302                          wrsabo:
    136                                 ;   Send Abort to other side
    137 003302  005000                          clr     r0
    138 003304  005001                          clr     r1
    139 003306  012703  000004                  mov     #p.abort,r3     ; R3 = message code
    140 003312  004767  001042                  jsr     pc,sndwr
    141 003316  000207                          rts     pc
    142
    143
    144
    145
    146                                 .sbttl !        wrsaba - Send Write Abort ACK
    147                                 ;
    148                                 ;       Entry:
    149                                 ;               R2 ->Channel Block
    150                                 ;
    151                                 ;       Exit:
    152                                 ;               (FSA preserves)
    153 003320                          wrsaba:
    154 003320  005000                          clr     r0
    155 003322  005001                          clr     r1
    156 003324  012703  000005                  mov     #p.aback,r3
    157 003330  004767  001024                  jsr     pc,sndwr
    158 003334  000207                          rts     pc
    159
    160                                 .sbttl !
```

```
FSA WRITE ROUTINES      MACRO V04.00  25-AUG-82 17:30:56 PAGE 25
|        RECEIVE INTERRUPT SERVICE ROUTINE

      1                                   .sbttl |        Receive Interrupt Service Routine
      2                                   ;
      3                                   ;  Receive Interrupt Service Routine
      4                                   ;
      5 003336                            $uninp::
      6 003336                                    INTSV$ un,PR4,1         ;;; generate interrupt save code
      7 003342  004767  002340                    jsr     pc,frkinp       ;;; R4 ->$FORK1 Context block in pool
      8 003346                                    CALL    $FORK1          ;;; create fork process
U         003346  004767  000000                  JSR     PC,$FORK1
      9
     10                                   ;  Handle 64K boundary
     11 003352  016700  174436                    mov     rdbcxs,r0       ; more to do?
     12 003356  001412                            beq     10$             ; EQ - no
     13 003360  005001                            clr     r1              ; new low buffer address bits are all clear
     14 003362  013702  176200                    mov     @#rcsr,r2       ; new high "     "     "   are
     15 003366  042702  177717                    bic     #^C060,r2       ;  inc'd from csr
     16 003372  062702  000020                    add     #020,r2
     17 003376  004767  001240                    jsr     pc,rcv          ; now receive the second half
     18 003402  000424                            br      100$            ;  and wait for the next interrupt
     19
     20 003404                            10$:
     21
     22                                   ;  Determine the event code from receive block contents
     23 003404  004767  002150                    jsr     pc,setrcb       ; R1 ->receive command block in pool
     24 003410  004767  002076                    jsr     pc,setint       ; R0 = CMD type, R2 ->Channel Block
     25 003414  012701  000001                    mov     #RCVDFE,r1
     26 003420  022700  000002                    cmp     #p.df,r0        ; Data Follows?
     27 003424  001405                            beq     20$
     28 003426  022700  000003                    cmp     #p.dfe,r0       ;  of either flavor?
     29 003432  001402                            beq     20$
     30 003434  012701  000000                    mov     #RCVCMD,r1      ; no, just some ordinary command
     31 003440                            20$:
     32
     33                                   ;  Declare the event
     34 003440  016700  174346                    mov     rstate,r0
     35 003444  004767  176176                    jsr     pc,fsa
     36 003450  010067  174316                    mov     r0,rstate
     37
     38                                   ; Dismiss Receive Interrupt
     39 003454  000207                    100$:   rts     pc
```

```
FSA WRITE ROUTINES      MACRO V04.00  25-AUG-82 17:30:56 PAGE 26
|

      1                                   .sbttl |
      2                                   .sbttl |       Receiver State Tables
      3                                   ; (Ready to Receive a Command)
      4 003456  003456' 003476'           rccmd:  .word   rccmd,  rcvcom          ;*RCVCMD
      5 003462  003466' 003576'                   .word   rcdata, rcvhdr          ;*RCVDFE
      6
      7                                   ; (Receive User Data)
      8 003466  000013  005132'           rcdata: .word   13,     die             ; RCVCMD
      9 003472  003456' 003634'                   .word   rccmd,  rcvdat          ;*RCVDFE
     10
     11                                   .sbttl |
     12                                   .title FSA Receiver  Routines
     13                                   .sbttl |    Finite State Automaton Receiver Routines
```

```
1418   FSA RECEIVER  ROUTINES  MACRO V04.00  25-AUG-82 17:30:56 PAGE 27
1419   |      RCVCOM - PROCESS PROTOCOL COMMAND MESSAGE
1420
1421        1                              .sbttl |        rcvcom - Process protocol command message
1422        2                              ;
1423        3                              ;       Last Command Received was not "Data Follows" or "Data Follows and End".
1424        4                              ;
1425        5                              ;       Entry:
1426        6                              ;               Command in receive block
1427        7                              ;
1428        8                              ;       Exit:
1429        9                              ;               (FSA preserves)
1430       10 003476                       rcvcom:
1431       11                              ;  Decode message type into event code
1432       12 003476  004767  002056               jsr     pc,setrcb       ; R1 ->Receive block
1433       13 003502  004767  002004               jsr     pc,setint       ; R2 ->Channel Block
1434       14 003506  011100                       mov     @r1,r0          ; (copy command code/LCN)
1435       15 003510  010003                       mov     r0,r3           ; (and save it for later read/write test)
1436       16 003512  004767  001326               jsr     pc,decode       ; R1 = Event Code Number
1437       17 003516  032703  000400               bit     #p.wchn,r3      ; read or write?
1438       18 003522  001007                       bne     5$
1439       19
1440       20                              ;  read-class command
1441       21 003524  016200  000006               mov     c..rsw(r2),r0   ; R0 ->Channel State Subtable
1442       22 003530  004767  176112               jsr     pc,fsa
1443       23 003534  010062  000006               mov     r0,c..rsw(r2)
1444       24 003540  000406                       br      9$
1445       25                              .
1446       26                              ;  write-class command
1447       27 003542  016200  000014       5$:     mov     c..wsw(r2),r0   ; R0 ->Channel State Subtable
1448       28 003546  004767  176074               jsr     pc,fsa
1449       29 003552  010062  000014               mov     r0,c..wsw(r2)
1450       30
1451       31 003556                       9$:
1452       32                              ;  Load regs and receive next command into pool block
1453       33 003556  012700  000004               mov     #4,r0           ; R0 = byte count
1454       34 003562  004767  001772               jsr     pc,setrcb       ; R1 ->Receive Command Block
1455       35 003566  005002                       clr     r2              ; R2 = Extended Address Bits
1456       36 003570  004767  001046               jsr     pc,rcv          ; start the hardware
1457       37 003574  000207                       rts     pc
1458
```

```
FSA RECEIVER  ROUTINES  MACRO V04.00  25-AUG-82 17:30:56 PAGE 28
|       RCVHDR - PREPARE FOR USER DATA

      1                                 .sbttl |        rcvhdr - prepare for user data
      2                                 ;
      3                                 ;       Command was "Data Follows" or "Data Follows and End".
      4                                 ;
      5                                 ;       Entry:
      6                                 ;               Command in receive block
      7                                 ;
      8                                 ;       Exit:
      9                                 ;               (FSA preserves)
     10 003576                          rcvhdr:
     11                                 ;  receive data into users buffer.
     12 003576  004767  001756                  jsr     pc,setrcb       ; R1 ->Receive Block
     13 003602  004767  001704                  jsr     pc,setint       ; R2 ->Channel Block
     14 003606  016100  000002                  mov     2(r1),r0        ; R0 = Already Negotiated Byte Count
     15 003612  016203  000010                  mov     c..qrd(r2),r3   ; (R3 ->I/O Packet)
     16 003616  016301  000026                  mov     i..buf+2(r3),r1 ; R1 ->User Buffer
     17 003622  016302  000024                  mov     i..buf(r3),r2   ; R2 = Extended Address Bits
     18 003626  004767  001010                  jsr     pc,rcv
     19 003632  000207                          rts     pc
```

```
FSA RECEIVER  ROUTINES  MACRO V04.00  25-AUG-82 17:30:56 PAGE 29
|       RCVDAT - PROCESS USER DATA MESSAGE

      1                                 .sbttl |        rcvdat - process user data message
      2                                 ;
      3                                 ;  Data Message Received.
      4                                 ;
      5                                 ;       Entry:
      6                                 ;               Command still in receive block
      7                                 ;               user data in user buffer
      8                                 ;
      9                                 ;       Exit:
     10                                 ;               (FSA preserves)
     11 003634                          rcvdat:
     12                                 ; set status code from byte count and/or command type
     13 003634  004767  001720                  jsr     pc,setrcb       ; R1 ->Receive Block
     14 003640  004767  001646                  jsr     pc,setint       ; R2 ->Channel Block
     15 003644  010246                          mov     r2,-(sp)
     16 003646  016103  000002                  mov     2(r1),r3        ; R3 = byte count for later end test
     17 003652  012701  000006                  mov     #IOC,R1         ; assume END
     18 003656  022700  000003                  cmp     #p.dfe,r0       ; sender said end?
     19 003662  001407                          beq     25$             ; that settle it!
     20 003664  016202  000010                  mov     c..qrd(r2),r2   ; (R2 ->I/O Packet)
     21 003670  020362  000034                  cmp     r3,l..xfr(r2)   ; or byte count fulfilled?
     22 003674  001402                          beq     25$             ; likewise means end
     23 003676  012701  000005                  mov     #MORE,r1
     24 003702  012602                  25$:    mov     (sp)+,r2
     25
     26                                 ;  declare the event
     27 003704  016200  000006                  mov     c..rsw(r2),r0
     28 003710  004767  175732                  jsr     pc,fsa
     29 003714  010062  000006                  mov     r0,c..rsw(r2)
     30
     31                                 ;  Load regs and receive next command into pool block
     32 003720  012700  000004                  mov     #4,r0           ; R0 = byte count
     33 003724  004767  001630                  jsr     pc,setrcb       ; R1 ->Receive Command Block
     34 003730  005002                          clr     r2              ; R2 = Extended Address Bits
     35 003732  004767  000704                  jsr     pc,rcv          ; start the hardware
     36 003736  000207                          rts     pc
     37
     38                                 .sbttl |
FSA RECEIVER  ROUTINES  MACRO V04.00  25-AUG-82 17:30:56 PAGE 10
|       TRANSMIT INTERRUPT SERVICE ROUTINE

      1                                       .sbttl |        Transmit Interrupt Service Routine
      2                                       ;
      3                                       ;  Transmit Interrupt Service Routine
      4                                       ;
      5 003740                                $unout::
      6 003740                                        INTSV$ un,PR4,1         ;;; generate interrupt save code
      7 003744  004767  001744                        jsr     pc,frkout       ;;; R4 ->$FORK1 Context block in pool
      8 003750                                        CALL    $FORK1          ;;; create fork process
U       003750  004767  000000                        JSR     PC,$FORK1
      9
     10                                       ;  handle 64K boundary
     11 003754  016700  174040                        mov     wrbcxs,r0
     12 003760  001412                                beq     10$
     13 003762  005001                                clr     r1
     14 003764  013702  176210                        mov     @#xcsr,r2
     15 003770  042702  177717                        bic     #^C060,r2
     16 003774  062702  000020                        add     #020,r2
     17 004000  004767  000672                        jsr     pc,xmt
     18 004004  000440                                br      100$
     19
     20 004006                                10$:
     21
     22                                       ;  Declare I/O Completion Event
     23 004006  016700  174004                        mov     xstate,r0
     24 004012  012701  000002                        mov     #XMTIOC,r1
     25 004016  004767  175624                        jsr     pc,fsa
     26 004022  010067  173770                        mov     r0,xstate
     27
     28                                       ;  check for more to do
     29 004026  026767  001670  001670                cmp     putter,taker    ; queue empty
     30 004034  001424                                beq     100$            ; if EQ, yes
     31
     32                                       ;  determine event code from message type
     33 004036  012701  000001                        mov     #XMTDFE,r1      ; assume DATA FOLLOWS message
     34 004042  017700  001656                        mov     @taker,r0
     35 004046  042700  177770                        bic     #m$mtyp,r0
     36 004052  022700  000002                        cmp     #p.df,r0
     37 004056  001405                                beq     20$
     38 004060  022700  000003                        cmp     #p.dfe, r0
     39 004064  001402                                beq     20$
     40 004066  012701  000000                        mov     #XMTCMD,r1      ; correct assumption to COMMAND message
     41 004072                                20$:
     42
     43                                       ;  declare the message event
     44 004072  016700  173720                        mov     xstate,r0
     45 004076  004767  175544                        jsr     pc,fsa
     46 004102  010067  173710                        mov     r0,xstate
     47 004106  000207                        100$:   rts     pc
```

```
FSA RECEIVER  ROUTINES  MACRO V04.00  25-AUG-82 17:30:56 PAGE 31
|

      1                               .sbttl |
      2                               .sbttl |        Transmitter State Tables
      3                               ;  (Idle, Ready for work)
      4 004110  004124' 004170'       xmidle: .word   xmiop0, cmddfe          ;*XMTCMD
      5 004114  004140' 004170'               .word   xmiop1, cmddfe          ;*XMTDFE
      6 004120  000014  005132'               .word      14, die              ; XMTIOC
      7
      8                               ;  (Command I/O Pending)
      9 004124  004124' 001702'       xmiop0: .word   xmiop0, ignore          ; XMTCMJ
     10 004130  004124' 001702'               .word   xmiop0, ignore          ; XMTDFE
     11 004134  004110' 001702'               .word   xmidle, ignore          ;*XMTIOC
     12
     13                               ;  (Data Follows Command I/O Pending)
     14 004140  004140' 001702'       xmiop1: .word   xmiop1, ignore          ; XMTCMD
     15 004144  004140' 001702'               .word   xmiop1, ignore          ; XMTDFE
     16 004150  004154' 004226'               .word   xmiop2, xmtdat          ;*XMTIOC
     17
     18                               ;  (User Data I/O Pending)
     19 004154  004154' 001702'       xmiop2: .word   xmiop2, ignore          ; XMTCMD
     20 004160  004154' 001702'               .word   xmiop2, ignore          ; XMTDFE
     21 004164  004110' 004264'               .word   xmidle, datioc          ;*XMTIOC
     22
     23                               .title FSA Transmitter Routines
     24                               .sbttl | Finite State Automaton Transmitter Routines
```

```
FSA TRANSMITTER ROUTINES        MACRO V04.00  25-AUG-82 17:30:56 PAGE 32
|       CMDDFE - SEND PROTOCOL COMMAND MESSAGE

      1                                   .sbttl |        cmddfe - send protocol command message
      2                                   ;
      3                                   ;       Entry:
      4                                   ;               Command must already be in circle-queue
      5                                   ;
      6                                   ;       Exit:
      7                                   ;               (FSA preserves)
      8 004170                            CMDDFE:
      9                                   ;  load next command from queue into transmit block
     10 004170  004767  001372                    jsr     pc,setxcb       ; R1 ->Transmit Control Block
     11 004174  004767  000376                    jsr     pc,getcq
     12 004200  010011                            mov     r0,@r1
     13 004202  004767  000370                    jsr     pc,getcq
     14 004206  010061  000002                    mov     r0,2(r1)
     15
     16                                   ;  send the command
     17 004212  012700  000004                    mov     #4,r0           ; R0 = byte count
     18 004216  005002                            clr     r2              ; R2 = Extended Address Bits
     19 004220  004767  000452                    jsr     pc,xmt          ; R1 ->Buffer
     20 004224  000207                            rts     pc
```

```
1622   FSA TRANSMITTER ROUTINES        MACRO V04.00  25-AUG-82 17:30:56 PAGE 33
1623   |       XMTDAT - TRANSMIT USER DATA
1624
1625         1                                 .sbttl |        xmtdat - transmit user data
1626         2                                 ;
1627         3                                 ;       Entry:
1628         4                                 ;               none
1629         5                                 ;
1630         6                                 ;       Exit:
1631         7                                 ;               (FSA preserves)
1632         8 004226                          XMTDAT:
1633         9 004226  004767  001334                  jsr     pc,setxcb       ; R1 ->Transmit Control Block
1634        10 004232  004767  001254                  jsr     pc,setint       ; R0 = cmd code, R2 ->Channel Block
1635        11 004236  016100  000002                  mov     2(r1),r0        ; R0 = Already Negotiated Byte Count
1636        12 004242  016203  000016                  mov     c..qwc(r2),r3   ; (R3 ->I/O Packet)
1637        13 004246  016301  000026                  mov     i..buf+2(r3),r1 ; R1 = User Buffer Address
1638        14 004252  016302  000024                  mov     i..buf(r3),r2   ; R2 = Extended Address Bits
1639        15 004256  004767  000414                  jsr     pc,xmt          ; "light fuse, stand well back."
1640        16 004262  000207                          rts     pc
1641
```

```
FSA TRANSMITTER ROUTINES        MACRO V04.00  25-AUG-82 17:30:56 PAGE 34
|       DATIOC - PROCESS WRITE COMPLETION

                                .sbttl |        datioc - process write completion
                                ;
                                ;       Entry:
                                ;               none
                                ;
                                ;       Exit:
                                ;               (FSA preserves)
  004264                        DATIOC:
                                ;  Determine completion status
  004264  004767  001276                jsr     pc,setxcb
  004270  004767  001216                jsr     pc,setint       ; R0 = command code, R2 ->Channel Block
  004274  016100  000002                mov     2(r1),r0        ; (R0 = byte count)
  004300  016203  000016                mov     c..qwr(r2),r3
  004304  012701  000006                mov     #IOC,r1         ; (presume END)
                                ;  *** Begin - HGM 20-Jul-82 ***
  004310  005763  000034                tst     i..xfr(r3)
  004314  001402                        beq     10$
                                ;       cmp     r0,i..xfr(r3)
                                ;       bhis    10$
                                ;  *** End - HGM 20-Jul-82 ***
  004316  012701  000005                mov     #MORE,r1
  004322                        10$:

                                ;  declare event
  004322  016200  000014                mov     c..wsw(r2),r0
  004326  004767  175314                jsr     pc,fsa
  004332  010062  000014                mov     r0,c..wsw(r2)

  004336  000207                        rts     pc

                                .title FSA Support Routines
                                .sbttl |
                                .sbttl | Finite State Automaton Support Routines
```

```
FSA SUPPORT ROUTINES    MACRO V04.00  25-AUG-82 17:30:56 PAGE 35
|       SNDRD - SEND READ-CLASS MESSAGE

      1                                     .sbttl |        sndrd - send read-class message
      2                                     ;
      3                                     ;       Entry:
      4                                     ;               R1 = Byte Count or 0
      5                                     ;               R2 ->Channel Block
      6                                     ;               R3 = Message Type
      7                                     ;
      8                                     ;       Exit:
      9                                     ;               (FSA Preserves)
     10 004340                              sndrd:
     11 004340  016200  000002                      mov     c..tag(r2),r0
     12 004344  142700  177767                      bicb    #m$dbug,r0      ; (preserve debug channel flag)
     13 004350  050300                              bis     r3,r0           ; insert message type
     14 004352  004767  000036                      jsr     pc,sndcmd
     15 004356  000207                              rts     pc
     16
     17
     18
     19
     20                                     .sbttl |        sndwr - send write-class message
     21                                     ;
     22                                     ;       Entry:
     23                                     ;               R0 = User Subfunction Bits
     24                                     ;               R1 = Byte Count or 0
     25                                     ;               R2 ->Channel Block
     26                                     ;               R3 = Message Type
     27                                     ;
     28                                     ;       Exit:
     29                                     ;               (FSA Preserves)
     30 004360                              sndwr:
     31 004360  042700  177417                      bic     #m$usbf,r0
     32 004364  010046                              mov     r0,-(sp)
     33 004366  016200  000002                      mov     c..tag(r2),r0
     34 004372  142700  177767                      bicb    #m$dbug,r0      ; (preserve debug channel flag)
     35 004376  050300                              bis     r3,r0           ; insert message type
     36 004400  052700  000400                      bis     #p.wchn,r0      ; set WRITE channel
     37 004404  052600                              bis     (sp)+,r0
     38 004406  004767  000002                      jsr     pc,sndcmd
     39 004412  000207                              rts     pc
     40
     41
     42
     43
     44                                     .sbttl |        sndcmd - Send Protocol Command Message
     45                                     ;
     46                                     ;       Entry:
     47                                     ;               R0 = LCN, User Subfunction Bits, Debug Channel Flag, & Message Type
     48                                     ;               R1 = Byte Count or 0
     49                                     ;
     50                                     ;       Exit:
     51                                     ;               (FSA Preserves)
     52 004414                              sndcmd:
     53                                     ;  Add message to queue
     54 004414  004767  000112                      jsr     pc,putq
     55 004420  010100                              mov     r1,r0
```

```
 1735        56 004422  004767  000104            jsr     pc,putcq
 1736        57
 1737
```

```
FSA SUPPORT ROUTINES    MACRO V04.00  25-AUG-82 17:30:56 PAGE 35-1
|       SNDCMD - SEND PROTOCOL COMMAND MESSAGE

                                ;  signal transmitter to begin work
004426  016700  173364                  mov     xstate,r0       ; R0 ->Transmitter State Subtable
004432  012701  000000                  mov     #XMTCMD,r1      ; R1 = "Send Protocol Message"
004436  004767  175204                  jsr     pc,fsa
004442  010067  173350                  mov     r0,xstate

004446  000207                          rts     pc




                                .sbttl |        snddata - send DATA FOLLOWS or DATA FOLLOWS END
                                ;
                                ;       Entry:
                                ;               R0 = User Subfunction Bits
                                ;               R1 = Byte Count
                                ;               R2 ->Channel Block
                                ;               R3 = p.df or p.dfe
                                ;
                                ;       Exit:
                                ;               (FSA preserves)
004450                          snddata:
                                ;  build header word in R0
004450  042700  177417                  `       #m$usbf,r0
004454  010046                                  r0,-(sp)
004456  016200  000002                          c..tag(r2),r0
004462  142700  177767                  bicb    #m$dbug,r0
004466  050300                          bis     r3,r0
004470  052700  000400                  bis     #p.wchn,r0
004474  052600                          bis     (sp)+,r0

                                ;  place request in message queue
004476  004767  000030                  jsr     pc,putcq
004502  010100                          mov     r1,r0
004504  004767  000022                  jsr     pc,putcq

                                ;  notify transmitter
004510  016700  173302                  mov     xstate,r0
004514  012701  000001                  mov     #XMTDFE,r1
004520  004767  175122                  jsr     pc,fsa
004524  010067  173266                  mov     r0,xstate

004530  000207                          rts     pc




                                .sbttl |        putcq - add word to the circle queue
                                ;
                                ;       Entry:
                                ;               R0 = Data Word
                                ;
                                ;       Exit:
                                ;               R0 Preserved
004532                          putcq:
 1795       113 004532  010077  001164               mov     r0,@putter
 1796       114 004536  062767  000002  001156       add     #2,putter
 1797
```

```
FSA SUPPORT ROUTINES    MACRO V04.00  25-AUG-82 17:30:56 PAGE 35-2
|       PUTCQ - ADD WORD TO THE CIRCLE QUEUE

    115 004544  026727  001152  007356'          cmp     putter,#cq+cqsize
    116 004552  001003                           bne     50$
    117 004554  012767  005726' 001140           mov     #cq,putter
    118 004562  026767  001134  001134  50$:     cmp     putter,taker
    119 004570  001001                           bne     100$
    120 004572                                   crash
    121 004574  000207                  100$:    rts     pc
    122
    123
    124
    125
    126                                 .sbttl |         getcq - get a word from the circle queue
    127                                 ;
    128                                 ;        Entry:
    129                                 ;                None
    130                                 ;
    131                                 ;        Exit:
    132                                 ;                R0 = Data Word
    133                                 ;                Preserves all (others)
    134 004576                           getcq:
    135 004576  026767  001120  001120           cmp     putter,taker
    136 004604  001001                           bne     10$
    137 004606                                   crash                   ; shouldn't be called if q is empty
    138 004610  017700  001110          10$:     mov     @taker,r0
    139 004614  062767  000002  001102           add     #2,taker
    140 004622  026727  001076  007356'          cmp     taker,#cq+cqsize
    141 004630  001003                           bne     100$
    142 004632  012767  005726' 001064           mov     #cq,taker
    143 004640  000207                  100$:    rts     pc
    144
    145
    146
    147
    148                                 .sbttl |         rcv - transfer next message from xq/cp to pdp-11
    149                                 ;
    150                                 ;        Entry:
    151                                 ;                R0 = Byte Count
    152                                 ;                R1 ->Buffer
    153                                 ;                R2 = Extended Address Bits
    154                                 ;
    155                                 ;        Exit:
    156                                 ;                none
    157 004642                           rcv:
    158                                 ;  handle 64K boundary
    159 004642  005067  173146                   clr     rdbcxs          ; clear the excess byte count
    160 004646  010103                           mov     r1,r3           ; R3 = buffer address
    161 004650  060003                           add     r0,r3           ;  plus byte count
    162 004652  020103                           cmp     r1,r3           ; check for 64K overflow
    163 004654  103403                           blo     10$
    164 004656  010367  173132                   mov     r3,rdbcxs       ; store the excess for second receive
    165 004662  160300                           sub     r3,r0           ; get size of first receive
    166 004664                          10$:
    167
    168 004664  012703  176200                   mov     #rcsr,r3        ; R3 ->Receive CSR
    169 004670  004767  000016                   jsr     pc,xqrp
```

```
 1855       170 004674  000207                rts    pc
 1856       171
 1857
```

```
1857  FSA SUPPORT ROUTINES    MACRO V04.00  25-AUG-82 17:30:56 PAGE 35-3
1858  |       RCV - TRANSFER NEXT MESSAGE FROM XQ/CP TO PDP-11
1859
1860       172
1861       173
1862       174
1863       175                                  .sbttl |       xmt - tranfer next message from pdp-11 to xq/cp
1864       176                                  ;
1865       177                                  ;       Entry:
1866       178                                  ;               R0 = Byte Count
1867       179                                  ;               R1 ->Buffer
1868       180                                  ;               R2 = Extended Address Bits
1869       181 004676                           xmt:
1870       182                                  ;  handle 64K boundary
1871       183 004676  005067  173116                   clr     wrbcxs          ; clear the excess byte count
1872       184 004702  010103                           mov     r1,r3           ; R3 = buffer address
1873       185 004704  060003                           add     r0,r3           ;  plus byte count
1874       186 004706  020103                           cmp     r1,r3           ; check for 64K overflow
1875       187 004710  103403                           blo     10$
1876       188 004712  010367  173102                   mov     r3,wrbcxs       ; store the excess for second transmit
1877       189 004716  160300                           sub     r3,r0           ; get size of first transmit
1878       190 004720                           10$:
1879       191
1880       192 004720  012703  176210                   mov     #xcsr,r3
1881       193 004724  004767  000002                   jsr     pc,xqcp
1882       194 004730  000207                           rts     pc
1883       195
1884       196
1885       197
1886       198
1887       199                                  .sbttl |       xqcp - set XQ/CP registers
1888       200                                  ;
1889       201                                  ;       Entry:
1890       202                                  ;               R0 = Byte Count
1891       203                                  ;               R1 ->Data Buffer
1892       204                                  ;               R2 = Extended Address Bits 4 & 5
1893       205                                  ;               R3 ->Receive/Transmit CSR
1894       206                                  ;
1895       207                                  ;       Exit:
1896       208                                  ;               none
1897       209 004732                           xqcp:
1898       210 004732  005200                           inc     r0              ; round up to word count
1899       211 004734  006200                           asr     r0
1900       212 004736  005400                           neg     r0              ; hardware wants 2's-complement byte count
1901       213 004740  010063  000006                   mov     r0,wc(r3)       ; load word count register
1902       214 004744  010163  000004                   mov     r1,ba(r3)       ; load buffer address
1903       215 004750  042702  177717                   bic     #^C060,r2       ; isolate memory extension bits in R2
1904       216 004754  042713  000060                   bic     #060,@r3        ; (reset previous memory extension state)
1905       217 004760  050213                           bis     R2,@r3
1906       218 004762  052713  000001                   bis     #1,@r3          ; light the GO bit
1907       219 004766  000207                           rts     pc
1908       220
1909       221
1910       222
1911       223
1912       224                                  .sbttl |       drain - abort pending I/O
1913       225                                  ;
1914       226                                  ;       Entry:
 1915        227                            ;               R0 = Queue Header Offset (c..qrd, etc.)
 1916        228                            ;               R2 ->Channel Block
 1917
```

```
FSA SUPPORT ROUTINES    MACRO V04.00  25-AUG-82 17:30:56 PAGE 35-4
|       DRAIN - ABORT PENDING I/O

      229 004770                          drain:
      230 004770  010246                          mov     r2,-(sp)        ; preserve channel block pointer
      231 004772  060200                          add     r2,r0           ; R0 ->wait queue header
      232 004774                          10$:    CALL    $QRMVF          ; point R1 at a waiting packet, if any
U         004774  004767  000000                  JSR     PC,$QRMVF
      233 005000  103413                          bcs     20$             ; none left
      234 005002  010046                          mov     r0,-(sp)        ; save header pointer
U     235 005004  012700  000000                  mov     #IE.ABO&377,r0  ; R0 = ABORT Condition code
      236 005010  010103                          mov     r1,r3           ; R3 ->I/O Packet
      237 005012  005001                          clr     r1              ; R1 = byte count (0 for abort)
      238 005014  010446                          mov     r4,-(sp)        ; protect R4 from $IOFIN . . .
      239 005016                                  CALL    $IOFIN          ; abort this packet to user
U         005016  004767  000000                  JSR     PC,$IOFIN
      240 005022  012604                          mov     (sp)+,r4        ; restore *SCB
      241 005024  012600                          mov     (sp)+,r0        ; restore queue header pointer
      242 005026  000762                          br      10$             ; loop for next packet, if any
      243
      244 005030  012602                  20$:    mov     (sp)+,r2        ; restore channel block pointer
      245 005032  000207                          rts     pc
      246
      247
      248
      249
      250                                 .sbttl |        calcbc - return minimum of two byte-counts
      251                                 ;
      252                                 ;       Entry:
      253                                 ;               R0 = first byte count
      254                                 ;               R1 = second byte count
      255                                 ;
      256                                 ;       Exit:
      257                                 ;               R1 = min(R0, R1);
      258 005034                          calcbc:
      259 005034  020100                          cmp     r1,r0
      260 005036  101401                          blos    10$
      261 005040  010001                          mov     r0,r1
      262 005042  000207                  10$:    rts     pc
      263
      264
      265
      266
      267                                 .sbttl |        decode - convert message type to event code
      268                                 ;
      269                                 ;       Entry:
      270                                 ;               R0 = type
      271                                 ;
      272                                 ;       Exit:
      273                                 ;               R1 = Event Code
      274 005044                          decode:
      275 005044  042700  177770                  bic     #m$mtyp,r0
      276 005050                                  dispatch p.rts,10$
      277 005056                                  dispatch p.cts,20$
      278 005064                                  dispatch p.abort,30$
      279 005072                                  dispatch p.aback,40$
      280 005100                                  crash                   ; something is VERY wrong
      281 005102  012701  000001          10$:    mov     #RTS,r1
```

```
 1975      282 005106  000410                        br      100$
 1976      283 005110  012701  000001       20$:     mov     #CTS,r1
 1977
```

```
FSA SUPPORT ROUTINES    MACRO V04.00  25-AUG-82 17:30:56 PAGE 35-5
|        DECODE - CONVERT MESSAGE TYPE TO EVENT CODE

     284 005114  000405                           br      100$
     285 005116  012701  000003           30$:    mov     #RABORT,r1
     286 005122  000402                           br      100$
     287 005124  012701  000004           40$:    mov     #ABOACK,r1
     288                                  ;       br      100$            ; (fall into 100$)
     289 005130  000207                   100$:   rts     pc
     290
     291
     292
     293
     294                                  .sbttl |        die - illegal state/event trap
     295                                  ;
     296                                  ;  die - illegal state/event trap
     297                                  ;
     298 005132                           die:    CRASH
     299
     300
     301                                  .sbttl |
     302                                  .title Common Channel Block Routines
     303                                  .sbttl | Common Channel Block Routines
```

```
COMMON CHANNEL BLOCK ROUTINES   MACRO V04.00  25-AUG-82 17:30:56 PAGE 36
|       DO4ALL - CALL ROUTINE FOR ALL CHANNEL BLOCKS

       1                                .sbttl |        do4all - call routine for all channel blocks
       2                                ;
       3                                ;       Entry:
       4                                ;               R3 ->Routine to be called with R2 ->Channel Block
       5 005134                         do4all:
       6 005134  012700  000040                 mov     #unpcnt-1,r0    ; check all channels, top to bottom
       7 005140  010046                 10$:    mov     r0,-(sp)
       8 005142  010002                         mov     r0,r2           ; R2 = RSX-11M Logical Unit Number
       9 005144  004767  000300                 jsr     pc,setcbp       ; R2 ->Channel Block
      10 005150  004713                         jsr     pc,@r3          ; call service routine
      11 005152  012600                         mov     (sp)+,r0
      12 005154  005300                         dec     r0
      13 005156  002370                         bge     10$             ; include channel 0
      14
      15 005160  005000                         clr     r0              ; (Debug channel is zero)
      16 005162  012702  001154'                mov     #dbugcb,r2      ; include Debug Channel
      17 005166  004713                         jsr     pc,@r3
      18 005170  000207                         rts     pc
      19
      20
      21
      22
      23                                .sbttl |        haltio - cause local abort event for read and write channels
      24                                ;
      25                                ;       Entry:
      26                                ;               R2 ->channel block
      27                                ;
      28                                ;       NOTE: R2 is preserved.
      29 005172                         haltio:
      30 005172  012701  000002                 mov     #LABORT,r1
      31 005176  016200  000006                 mov     c..rsw(r2),r0
      32 005202  004767  174440                 jsr     pc,fsa
      33 005206  010062  000006                 mov     r0,c..rsw(r2)
      34
      35 005212  012701  000002                 mov     #LABORT,r1
      36 005216  016200  000014                 mov     c..wsw(r2),r0
      37 005222  004767  174420                 jsr     pc,fsa
      38 005226  010062  000014                 mov     r0,c..wsw(r2)
      39
      40 005232  000207                         rts     pc
      41
      42
      43
      44
      45                                .sbttl |        iokill - abort ALL I/O for ALL channels owned by task
      46                                ;
      47                                ;       Entry:  (Called via do4all mechanism) .
      48                                ;               R0 = Channel Index
      49                                ;               R2 ->Channel Block
      50                                ;               R4 ->TCB
      51 005234                         iokill:
      52 005234  020462  000000                 cmp     r4,c..tcb(r2)   ; this the guy?
      53 005240  001006                         bne     20$             ; no, spare him
      54 005242  005062  000000                 clr     c..tcb(r2)      ; yes, knock down the CONNECTION
      55 005246  010446                         mov     r4,-(sp)
```

```
 2058        56 005250  004767  177716            jsr     pc,haltio       ; abort all I/O
 2059        57 005254  012604                    mov     (sp)+,r4
 2060
```

```
COMMON CHANNEL BLOCK ROUTINES   MACRO V04.00  25-AUG-82 17:30:56 PAGE 36-1
|       IOKILL - ABORT ALL I/O FOR ALL CHANNELS OWNED BY TASK

     58 005256  000207                  20$:    rts     pc
     59
     60
     61
     62
     63                                 .sbttl |        initcb - initialize a channel block
     64                                 ;
     65                                 ;       Entry:  (Called via do4all)
     66                                 ;               R0 = channel index
     67                                 ;               R2 ->channel Block
     68 005260                          initcb:
     69                                 ;  clear owned flag
     70 005260  005062  000000                  clr     c..tcb(r2)
     71
     72                                 ;  init prototype LCN
     73 005264  060000                          add     r0,r0
     74 005266  110062  000003                  movb    r0,c..tag+1(r2)
     75
     76                                 ;  init read/write states
     77 005272  012762  001704' 000006          mov     #rdidle,c..rsw(r2)
     78 005300  012762  002606' 000014          mov     #wrwait,c..wsw(r2)
     79
     80                                 ; init Queue Headers
     81 005306  010200                          mov     r2,r0
     82 005310  062700  000010                  add     #c..qrd,r0
     83 005314  005010                          clr     @r0
     84 005316  010060  000002                  mov     r0,2(r0)
     85
     86 005322  010200                          mov     r2,r0
     87 005324  062700  000016                  add     #c..qwr,r0
     88 005330  005010                          clr     @r0
     89 005332  010060  000002                  mov     r0,2(r0)
     90
     91                                 ;  init RTS byte count word
     92 005336  005062  000004                  clr     c..rts(r2)
     93
     94 005342  000207                          rts     pc
     95
     96
     97
     98
     99                                 .sbttl |
    100                                 .title Executive Pool Pointer Routines
    101                                 .sbttl | Executive Pool Pointer Routines
```

```
EXECUTIVE POOL POINTER ROUTINES MACRO V04.00  25-AUG-82 17:30:56 PAGE 37
|       SETPOOL - SETUP EXECUTIVE POOL BLOCKS

          1                                     .sbttl |        setpool - setup executive pool blocks
          2                                     ;
          3                                     ;       Entry:
          4                                     ;               R4 ->SCB
          5                                     ;
          6                                     ;       Exit:
          7                                     ;               Pool blocks setup
          8 005344                              setpool:
          9                                     ; Buy pool space from the executive
         10 005344  012701  000010                      mov     #10,r1          ; want 2 command blocks @ 4 bytes each
         11 005350  062701  000024                      add     #24,r1          ; plus 2 $FORK1 context blocks @ 10. bytes each
         12 005354                                      CALL    $ALOCB          ; allocate space from RSX pool
U             005354  004767  000000                      JSR     PC,$ALOCB
         13 005360  103432                              bcs     100$            ; Couldn't!
         14
         15                                     ; stuff pool block addresses at negative offsets from channel block table
         16 005362  012701  000032'                     mov     #un0tbl,r1      ; stuff addresses at negative offsets
         17 005366  010041                              mov     r0,-(r1)        ; R0 ->transmit command block in pool
         18 005370  062700  000004                      add     #4,r0           ; (step past the block itslef)
         19 005374  010041                              mov     r0,-(r1)        ; R0 ->receive command block in pool
         20 005376  062700  000004                      add     #4,r0           ; (step past the block itslef)
         21 005402  062700  000006                      add     #6,r0           ; ($FORK1 wants address of third word + 2)
         22 005406  010041                              mov     r0,-(r1)        ; R0 ->transmit $FORK1 context block
         23 005410  062700  000012                      add     #12,r0          ; (point past next block)
         24 005414  010041                              mov     r0,-(r1)        ; R0 ->receive $FORK1 context block
         25
         26                                     ; load driver context information into alternate $FORK1 context blocks
         27 005416  010446                              mov     r4,-(sp)        ; (save SCB pointer)
U        28 005420  016400  000010                      mov     s.frk+10(r4),r0 ; R0 = driver context word from SCB
         29 005424  004767  000256                      jsr     pc,frkinp       ; R4 ->$FORK1 input context block
         30 005430  010064  000002                      mov     r0,2(r4)
         31 005434  004767  000254                      jsr     pc,frkout       ; R4 ->$FORK1 output context block
         32 005440  010064  000002                      mov     r0,2(r4)
         33 005444  012604                              mov     (sp)+,r4        ; (restore SCB pointer)
         34
         35 005446  000207                      100$:   rts     pc
```

```
2146   EXECUTIVE POOL POINTER ROUTINES MACRO V04.00  25-AUG-82 17:30:56 PAGE 38
2147   |       SETCBP - SET CHANNEL BLOCK POINTER
2148
2149        1                               .sbttl |        setcbp - Set Channel Block Pointer
2150        2                               ;
2151        3                               ;       Entry:
2152        4                               ;               R2 = RSX-11M Full-Duplex Channel Number
2153        5                               ;               R5 ->UCB
2154        6                               ;
2155        7                               ;       Exit:
2156        8                               ;               CARRY CLEAR ON SUCCESS and
2157        9                               ;               R2 ->Channel Block
2158       10                               ;
2159       11                               ;       Error:
2160       12                               ;               CARRY SET
2161       13                               ;               R0 = RSX-11M Error Code
2162       14                               ;
2163       15                               ;       NOTE:
2164       16                               ;               Preserves R1, & R3
2165       17 005450                        setcbp:
2166       18 005450  022702  000041                cmp     #unpcnt,r2
2167       19 005454  103412                        blo     90$             ; bad unit number (RSX can't check)
2168       20 005456  060202                        add     r2,r2           ; *2
2169       21 005460  010246                        mov     r2,-(sp)        ;       *2
2170       22 005462  060202                        add     r2,r2           ; *4
2171       23 005464  060202                        add     r2,r2           ; *10
2172       24 005466  060202                        add     r2,r2           ; *20
2173       25 005470  062602                        add     (sp)+,r2        ; *22
2174       26 005472  062702  000032'               add     #un0tbl,r2      ; R2 ->Channel Block
2175       27 005476  000241                        clc                     ; flag no error
2176       28 005500  000403                        br      100$
2177       29
2178  U    30 005502  012700  000000        90$:    mov     #IE.IDU&0377,r0
2179       31 005506  000261                        sec
2180       32 005510  000207                100$:   rts     pc
2181       33
2182       34
2183       35
2184       36
2185       37                               .sbttl |        setint - Setup Interrupt Registers
2186       38                               ;
2187       39                               ;       Entry:
2188       40                               ;               R1 ->Command Block in Executive Pool
2189       41                               ;
2190       42                               ;       Exit:
2191       43                               ;               R0 = Command Type
2192       44                               ;               R2 ->Channel Block
2193       45                               ;
2194       46                               ;       Error:
2195       47                               ;               CARRY SET
2196       48                               ;               R0 = RSX-11M Error Code
2197       49                               ;
2198       50                               ;       Note:
2199       51                               ;               R1 is Preserved
2200       52 005512                        setint:
2201       53 005512  005000                        clr     r0
2202       54 005514  151100                        bisb    @r1,r0          ; R0 = command code
2203       55 005516  042700  177760                bic     [illegible],r0; Isolate message type and delete channel flag
```

```
2204        56 005522  032700  000010             bit     #p.debug,r0
2205        57 005526  001007                     bne     10$
2206
```

```
EXECUTIVE POOL POINTER ROUTINES MACRO V04.00  25-AUG-82 17:30:56 PAGE 38-1
|        SETINT - SETUP INTERRUPT REGISTERS

     58 005530  005002                         clr     r2
     59 005532  156102  000001                 bisb    1(r1),r2
     60 005536  006202                         asr     r2
     61 005540  004767  177704                 jsr     pc,setcbp       ; R2 ->Channel Block
     62 005544  000404                         br      20$
     63
     64 005546  042700  000010         10$:    bic     #^CmSdbug,r0    ; clear debug flag for later compares
     65 005552  012702  001154'                mov     #dbugcb,r2
     66 005556  000207                 20$:    rts     pc
     67
     68
     69
     70
     71                                .sbttl |        setrcb - return address of receive control block
     72                                ;
     73                                ;       Entry:
     74                                ;               none
     75                                ;
     76                                ;       Exit:
     77                                ;               R1 ->Receive Control Block in Executive Pool
     78 005560                         setrcb:
     79 005560  016701  172242                 mov     un0tbl-4,r1
     80 005564  000207                         rts     pc
     81
     82
     83
     84
     85                                .sbttl |        setxcb - return address of transmit control block
     86                                ;
     87                                ;       Entry:
     88                                ;               none
     89                                ;
     90                                ;       Exit:
     91                                ;               R1 ->Transmit Control Block in Executive Pool
     92 005566                         setxcb:
     93 005566  016701  172236                 mov     un0tbl-2,r1
     94 005572  000207                         rts     pc
     95
     96
     97
     98
     99                                .sbttl |        setdbc - return address of Debug Channel Block
    100                                ;
    101                                ;       Entry:
    102                                ;               R1 ->I/O Packet
    103                                ;
    104                                ;       Exit:
    105                                ;               R2 ->Debug Channel Block
    106                                ;
    107                                ;       Error:
    108                                ;               CARRY SET
    109                                ;               R0 = RSX-11M Error Code
    110                                ;
    111                                ;       Notes:
    112                                ;               Preserves R1
```

```
 2264        113 005574                         setdbc:
 2265        114 005574  012702  001154'                  mov     #dbugcb,r2
 2266
```

```
2266   EXECUTIVE POOL POINTER ROUTINES MACRO V04.00  25-AUG-82 17:30:56 PAGE 38-2
2267   |       SETDBC - RETURN ADDRESS OF DEBUG CHANNEL BLOCK
2268
2269        115 005600  005762  000000                     tst     c..tcb(r2)
2270        116 005604  001410                             beq     50$
2271        117 005606  026162  000004  000000             cmp     l.tcb(rl),c..tcb(r2)
2272        118 005614  001407                             beq     100$
2273   U    119 005616  012700  000000                     mov     #IE.DAA&0377,R0
2274        120 005622  000261                             sec
2275        121 005624  000403                             br      100$
2276        122 005626  016162  000004  000000  50$:       mov     l.tcb(rl),c..tcb(r2)
2277        123                                 ;          br      100$            ; (fall into 100$)
2278        124 005634  000207                  100$:      rts     pc
2279        125
2280        126
2281        127
2282        128
2283        129                                 .sbttl |           setqrc - return address of channel block for non-transfer requests
2284        130                                 ;
2285        131                                 ;          Entry:
2286        132                                 ;                  Rl ->I/O Packet
2287        133                                 ;
2288        134                                 ;          Exit:
2289        135                                 ;                  R2 ->Channel Block
2290        136                                 ;
2291        137                                 ;          Error:
2292        138                                 ;                  CARRY SET
2293        139                                 ;                  R0 = RSX-11M Error Code
2294        140                                 ;
2295        141                                 ;          Note:
2296        142                                 ;                  Preserves Rl
2297        143 005636                          setqrc:
2298        144 005636  005002                             clr     r2
2299        145 005640  156102  000030                     bisb    l..chn-2(rl),r2
2300        146 005644  004767  177600                     jsr     pc,setcbp
2301        147 005650  000207                             rts     pc
2302        148
2303        149
2304        150
2305        151
2306        152                                 .sbttl |           setxrc - return address of channel block for transfer requests
2307        153                                 ;
2308        154                                 ;          Entry:
2309        155                                 ;                  Rl ->I/O Packet
2310        156                                 ;
2311        157                                 ;          Exit:
2312        158                                 ;                  R2 ->Channel Block
2313        159                                 ;
2314        160                                 ;          Error:
2315        161                                 ;                  CARRY SET
2316        162                                 ;                  R0 = RSX-11M Error Code
2317        163                                 ;
2318        164                                 ;          Note:
2319        165                                 ;                  Preserves Rl
2320        166 005652                          setxrc:
2321        167 005652  005002                             clr     r2
2322        168 005654  156102  000012                     bisb    l..bn(rl),r2
2323        169 005660  004767  177[illegible]                     jsr     pc,setcbp
```

```
 2324        170 005664  103407                           bcs     100$
 2325        171 005666  026162  000004  000000           cmp     i.tcb(rl),c..tcb(r2)
 2326
```

```
EXECUTIVE POOL POINTER ROUTINES MACRO V04.00  25-AUG-82 17:30:56 PAGE 38-3
|       SETXRC - RETURN ADDRESS OF CHANNEL BLOCK FOR TRANSFER REQUES

     172 005674  001403                              beq     100$
U    173 005676  012700  000000                      mov     #IE.DNA&0377,r0
     174 005702  000261                              sec
     175 005704  000207                     100$:    rts     pc
     176
     177
     178
     179
     180                                    .sbttl |        frkinp - return pointer to input context block for $FORK1
     181                                    ;
     182                                    ;        Entry:
     183                                    ;                None
     184                                    ;
     185                                    ;        Exit:
     186                                    ;                R4 ->$FORK1 input context block
     187 005706                             frkinp:
     188 005706  016704  172110                      mov     un0tbl-10,r4
     189 005712  000207                              rts     pc
     190
     191
     192
     193
     194                                    .sbttl |        frkout - return pointer to output context block for $FORK1
     195                                    ;
     196                                    ;        Entry:
     197                                    ;                None
     198                                    ;
     199                                    ;        Exit:
     200                                    ;                R4 ->$FORK1 output context block
     201 005714                             frkout:
     202 005714  016704  172104                      mov     un0tbl-6,r4
     203 005720  000207                              rts     pc
     204
     205                                    .sbttl |
     206                                    .sbttl |
     207                                    .title Non-Pool Data-Base
```

```
NON-POOL DATA-BASE      MACRO V04.00  25-AUG-82 17:30:56 PAGE 39
| NON-POOL DATA-BASE

      1                                .sbttl | Non-Pool Data-Base
      2                                ;
      3                                ;  message queue
      4                                ;
      5 005722  005726'                putter: .word   cq
      6 005724  005726'                taker:  .word   cq
      7 005726                         cq:     .blkw   cqsize/2
      8
      9                                .sbttl |
     10                                .sbttl |
     11                                .title RSX-11M Pool Data-Base
     12                                .sbttl | RSX-11M Pool Data-Base
```

```
RSX-11M POOL DATA-BASE  MACRO V04.00  25-AUG-82 17:30:56 PAGE 40
|         CONTENT DEFINITIONS

      1                               .sbttl |        Content Definitions
      2                               .list me
      3
      4                               ; Function Mask Mask Bits
      5               000001          .iokil = 1;      /* cancel I/O */
      6               000002          .iowlb = 2;      /* write block */
      7               000004          .iorlb = 4;      /* read block */
      8               000100          .iocon = 0100;  /* connect to data-path; ACC ATTACH replacement */
      9               000200          .iodsc = 0200;  /* disconnect data-path; ACC DETACH replacement */
     10               000400          .iodbr = 0400;  /* Debug Read */
     11               001000          .iodbw = 1000;  /* Debug Write */
     12
     13               001707          .iomsk = .iokil ! .iorlb ! .iowlb ! .iocon ! .iodsc ! .iodbr ! .iodbw;
     14               000300          .ioctl = .iocon ! .iodsc;
     15
     16                               ;
     17                               .sbttl |        Format Definitions
     18                               .sbttl |                Device Control Block (DCB)
     19                               ;
     20 007356                        $UNDAT::
     21 007356  000000                UNDCB:   .WORD   0                      ;POINTER TO NEXT DCB
     22 007360  007416'                        .WORD   UN0UCB                 ;POINTER TO FIRST UCB
     23 007362     125        116              .ASCII  /UN/                   ;DEVICE NAME
     24 007364     000        000              .BYTE   0,0                    ;UNIT #LIMITS
     25 007366  000034                         .WORD   UCBLEN                 ;LENGTH OF EACH UCB
     26 007370  000000                         .WORD   0                      ;DRIVER DISPATCH TABLE
     27 007372  001707                         .WORD   .iomsk                 ;FUNCTION MASK 0-15
     28 007374  000300                         .WORD   .ioctl                 ;CONTROL MASK 0-15
     29 007376  000000                         .WORD   0                      ;NOP MASK 0-15
     30 007400  000000                         .WORD   0                      ;ACP MASK 0-15
     31 007402  000000                         .WORD   0                      ;FUNCTION MASK 16-31
     32 007404  000000                         .WORD   0                      ;CONTROL MASK 16-31
     33 007406  000000                         .WORD   0                      ;NOP MASK 16-31
     34 007410  000000                         .WORD   0                      ;ACP MASK 16-31
     35                                   .IFDF      L$$DRV
     36 007412  000000                         .WORD   0                      ;PARTITION CONTROL BLOCK POINTER
     37                                   .ENDC
     38
     39                               ;
     40                               .sbttl |                Unit Control Block (UCB)
     41                               ;
     42                               .macro ucb unit
     43                                  .IIF EQ unit .list
     44                                  .IIF DF M$$MUP .WORD 0               ;OWNING TERMINAL UCB ADDR
     45                               .list
     46                               UN'unit'UCB::
     47                               .nlist
     48                                        .WORD   UNDCB                  ;POINTER BACK TO DCB
     49                                        .WORD   .-2                    ;REDIRECT POINTER
     50                                        .BYTE   UC.PWF!UC.NPR!UC.QUE!UC.KIL;CONTROL FLAGS
     51                                        .BYTE   0                      ;STATUS FLAGS (comm regs busy flag)
     52                                        .BYTE   unit,US.PUB            ;UNIT #, UNIT STATUS (disallow ALLOCATE)
     53                                        .WORD   DV.SQD                 ;DEVICE CHARACTERISTICS
     54                                        .WORD   0                      ;U.CW2
     55                                        .WORD   0                      ;U.CW3
```

```
 2438        56                          .WORD   512.                  ;DEFAULT BUFFER SIZE
 2439        57                          .WORD   UN'unit'SCB                   ;POINTER TO SCB
 2440
```

```
2440   RSX-11M POOL DATA-BASE  MACRO V04.00  25-AUG-82 17:30:56 PAGE 40-1
2441   |               UNIT CONTROL BLOCK (UCB)
2442
2443       58                                        .WORD   0                      ;TCB POINTER
2444       59                                        .BLKW   3                      ;U.BUF, U.CNT
2445       60
2446       61                                   .if df UMR
2447       62                                  .iif eq unit u..umo = . - unucbs;       /* define UMR allocation block offset */
2448       63                                        .word   0                      ; link
2449       64                                        .word   0                      ; first UMR
2450       65                                        .word   0                      ; # of UMR's * 4
2451       66                                        .word   0
2452       67                                        .word   0
2453       68                                        .word   0
2454       69                                   .endc ; df UMR
2455       70
2456       71                                   .IF EQ unit
2457       72                                  ucblen = . - unucbs;    /* measure first UCB for size */
2458       73                                  .nlist
2459       74                                     .ENDC
2460       75                                  .endm
2461       76
2462       77                                  ;
2463       78                                  .sbttl |                 Status Control Block (SCB)
2464       79                                  ;
2465       80                                  .macro scb unit
2466       81                                  .list
2467       82                                  UN'unit'SCB:
2468       83                                  .nlist
2469       84                                     .IIF EQ unit .list
2470       85                                         .WORD   0,.-2                  ;I/O QUEUE LISTHEAD
2471       86                                         .BYTE   un'unit'pri,un'unit'vec/4;PRIORITY, DEVICE INTERRUPT VECTOR
2472       87                                         .BYTE   0,0                    ;CURRENT TIMEOUT, INITIAL TIMEOUT
2473       88                                         .BYTE   unit'*2,0                      ;CONTROLLER INDEX, CONTROLLER STATU
2474       89                                         .WORD   un'unit'csr                    ;DEVICE CSR ADDRESS
2475       90                                         .BLKW   1                      ;I/O PACKET ADDRESS
2476       91                                         .BLKW   4                      ;FORK BLOCK STORAGE
2477       92                                     .IFDF L$$DRV & M$$MGE
2478       93                                         .BLKW   1                      ;DRIVER RELOCATION BIAS
2479       94                                     .ENDC
2480       95                                     .IIF EQ unit .nlist
2481       96                                  .endm
2482       97
2483       98                                  ;
2484       99                                  ; build UCB
2485      100                                  ;
2486      101 007414                           unucbs: ucb 0
2487          007414  000000                      .IIF DF M$$MUP .WORD 0             ;OWNING TERMINAL UCB ADDR
2488          007416                           UN0UCB::
2489          007416  007356'                        .WORD   UN0DCB                 ;POINTER BACK TO DCB
2490          007420  007416'                        .WORD   .-2                    ;REDIRECT POINTER
2491          007422     164                         .BYTE   UC.PWF!UC.NPR!UC.QUE!UC.KIL;CONTROL FLAGS
2492          007423     000                         .BYTE   0                      ;STATUS FLAGS (comm regs busy flag)
2493          007424     000      004                .BYTE   0,US.PUB               ;0 #, 0 STATUS (disallow ALLOCATE)
2494          007426  000040                         .WORD   DV.SQD                 ;DEVICE CHARACTERISTICS
2495          007430  000000                         .WORD   0                      ;U.CW2
2496          007432  000000                         .WORD   0                      ;U.CW3
2497          007434  001000                         .WORD   512.                   ;DEFAULT BUFFER SIZE
```

```
 2498          007436  007450'               .WORD   UNOSCB              ;POINTER TO SCB
 2499          007440  000000                .WORD   0                   ;TCB POINTER
 2500
```

```
RSX-11M POOL DATA-BASE  MACRO V04.00  25-AUG-82 17:30:56 PAGE 40-2
|                   STATUS CONTROL BLOCK (SCB)

         007442                               .BLKW   3                     ;U.BUF, U.CNT

                                           .if df UMR
                                        .iif eq 0 u..umo = . - unucbs;  /* define UMR allocation block offset */
                                              .word   0                     ; link
                                              .word   0                     ; first UMR
                                              .word   0                     ; # of UMR's * 4
                                              .word   0
                                              .word   0
                                              .word   0
                                           .endc ; df UMR

                                           .IF EQ 0
                 000034                 ucblen = . - unucbs;    /* measure first UCB for size */
                                           .ENDC
      102
      103                               ;
      104                               ; build SCB
      105                               ;
      106 007450                        unucbs: scb 0
          007450                        UN0SCB:
          007450  000000  007450'             .WORD   0,.-2                 ;I/O QUEUE LISTHEAD
          007454     200     030              .BYTE   un0pri,un0vec/4;PRIORITY, DEVICE INTERRUPT VECTOR
          007456     000     000              .BYTE   0,0                   ;CURRENT TIMEOUT, INITIAL TIMEOUT
          007460     000     000              .BYTE   0*2,0                 ;CONTROLLER INDEX, CONTROLLER STATUS
          007462  176210                      .WORD   un0csr                ;DEVICE CSR ADDRESS
          007464                              .BLKW   1                     ;I/O PACKET ADDRESS
          007466                              .BLKW   4                     ;FORK BLOCK STORAGE
                                           .IFDF L$$DRV & M$$MGE
          007476                              .BLKW   1                     ;DRIVER RELOCATION BIAS
                                           .ENDC
      107
      108 007500                        $UNEND::
      109         000001                      .end
```