WD2511A X.25
Packet Network Interface (LAPB)

FEATURES

Handles The Entire Link Level Communication Protocol

- International standard CCIT X.25 LAPB protocol for packet switching
- Programmable link level timer (T1) and retransmission counter (N2)
- Automatic framing, appending and testing of FCS, and zero bit insertion and deletion
- Automatic error detection and retransmission
- Automatic flow control when buffers are not available
- Certified by Telenet, Transpac, and other major packet switching networks

Custom Communications Architecture Provides Top Performance

- Full-duplex transmission speeds to 1.1 Mbps
- Dual-channel direct memory access (DMA) capability unburdens your CPU
- Bus-oriented control for easy computer interface
- Synchronous modem interface built in
- 5 Volt NMOS technology in 48-pin DIP and 68-pin CLCC
- TTL compatible on all inputs and outputs

X.25 Feature Extensions For Broader Applications

- Half-Duplex communication option
- Programmable address field for multipoint operation and any point-to-point application where throughput and data integrity are required
- Built-in loopback testing

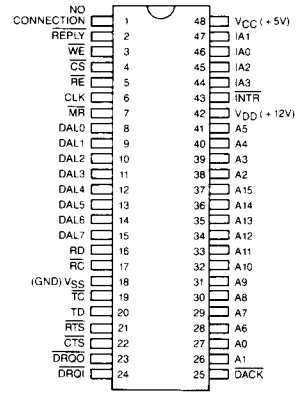


FIGURE 1. 48-PIN DESIGNATION

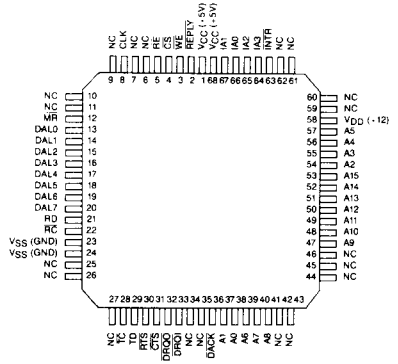


FIGURE 1A. 68-PIN DESIGNATION

DESCRIPTION

Western Digital is the first company to offer a complete implementation of the CCITT Recommendation X.25 LAPB protocol in a chip. The WD2511A provides an error free communication capability while offloading the Host from the responsibility of meeting the realtime requirements of the communication channel.

The WD2511A consists of a 48-pin DIP and 68-pin CLCC. These NMOS LSI devices provide a unique solution for point-to-point communication requirements. The WD2511A offloads the Host by implementing the full link layer protocol of the International Standard Recommendation X.25 using the two DMA channels.

This bit oriented serial link controller automatically handles the transmission and reception of packets of data, acknowledges the receipt of error free packets, and requests that lost or erroneous packets be retransmitted. The WD2511A interrupts the Host after receiving an error free packet, acknowledgement of a transmitted packet, and after detecting certain error or abnormal conditions on the link.

APPLICATIONS

The WD2511A can be used on dedicated or switched lines whenever reliable error free communication is needed. In fact, any point-to-point application concerned with high integrity and throughput will benefit from the WD2511A's integrated solution.

Examples of equipment using this chip include modems, PADs, switching nodes, statistical multiplexers, gateways from one network to another and equipment connecting computers or terminals to public or private networks.

TABLE OF CONTENTS

FIGURES

TABLES

1.0 ORGANIZATION

Figures 1 and 1A illustrate the Pin Designations and Table 1 describes the interface signals of the WD2511A. A detailed block diagram of the WD2511A is shown in Figure 2. A glossary of terms used throughout this document appears at the end of the Data Sheet.

Mode control and monitor of status by the user's CPU is performed through the Read/Write Control circuit which reads from or writes into I/O registers addressed by IAO-IA3.

Transmit and receive data are accessed through the DMA control. Serial data is generated and received by the bit-oriented controllers.

Internal Control of the WD2511A is by means of three internal microcontrollers: one for transmit, one for receive, and one for overall control.

Parallel transmit data are entered into the Transmitter Holding Register (THR) and then presented to the Transmitter Register (TR) which converts the data to a serial bit stream. The Cyclic Redundancy Check (CRC) is computed in the 16-bit CRC register and the result becomes the transmitted Frame Check Sequence (FCS).

Parallel receive data enters the Receiver Holding Register (RHR) from the 24-bit serial Receiver Register (RR). The 24-bit length of RR permits stripping of the FCS prior to transfer into the RHR. The receiver CRC register is used to test the validity of the received FCS. A 3-Byte FIFO is included in the receiver to decouple the speed of the user from the receiver.

Table 1. 48-PIN AND 68-PIN INTERFACE SIGNALS DESCRIPTION (All signals are TTL compatible.)

48-PIN NUMBER	68-PIN NUMBER	SYMBOL	PIN NAME	FUNCTION
1	-	NO CONNECTION	No Connection	Leave pin open.
2	2	$\overline{\text{REPLY}}$	Reply	An active low output indicates the WD2511A has either a $\overline{\text{CS}}\bullet\overline{\text{RE}}$ or a $\overline{\text{CS}}\bullet\overline{\text{WE}}$ condition.
3	3	$\overline{\text{WE}}$	Write Enable	The data on the DAL are written into the selected register when $\overline{\text{CS}}$ and $\overline{\text{WE}}$ are low.
4	4	$\overline{\text{CS}}$	Chip Select	Active low chip select for user control of I/O registers.
5	5	$\overline{\text{RE}}$	Read Enable	The contents of the selected register is placed on DAL when $\overline{\text{CS}}$ and $\overline{\text{RE}}$ are low.
6	8	CLK	Clock	Clock input used for internal timing. Must be square wave and should be 2MHZ $\pm$ 5%. For operation at less than 2MHZ, contact the factory.
7	12	$\overline{\text{MR}}$	Master Reset	Active low initializes the chip. All registers reset to zero, except control bits MDISC and $\overline{\text{LINK}}$ which are set to 1. $\overline{\text{DACK}}$ must be stable high before $\overline{\text{MR}}$ goes high.
8-15	13-20	DAL0-DAL7	Data Access Lines	An 8-bit bi-directional three-state data bus for user and DMA controller transfers.
16	21	RD	Receive Date	Receive serial data input.
17	22	$\overline{\text{RC}}$	Receive Clock	This is a 1x clock input. RD is sampled on the rising edge of $\overline{\text{RC}}$.
18	23,24	V_{SS}	Ground	Ground.
19	28	$\overline{\text{TC}}$	Transmit Clock	A 1x clock input. TD changes on the falling edge of $\overline{\text{TC}}$.
20	29	TD	Transmit Data	Transmit serial data output.
21	30	$\overline{\text{RTS}}$	Request-To-Send	An open collector (drain) output which goes low when the WD2511A is ready to transmit either flags or data.
22	31	CTS	Clear-To-Send	An active low output signals the WD2511A that transmission may begin. If high, the TD output is forced high. May be hard-wired to ground.

Table 1. 48-PIN AND 68-PIN INTERFACE SIGNALS DESCRIPTION (continued) (All signals are TTL compatible.)

48-PIN NUMBER	68-PIN NUMBER	SYMBOL	PIN NAME	FUNCTION
23	32	$\overline{DRQO}$	DMA Request Out	An active low output signal which initiates user bus request so the WD2511A can output data onto the bus.
24	33	$\overline{DRQI}$	DMA Request In	An active low output signal which initiates user bus request so that data may be input to the WD2511A.
25	36	$\overline{DACK}$	DMA Acknowledge	An active low input from the user in response to $\overline{DRQI}$ or $\overline{DRQO}$. $\overline{DACK}$ must not be low if $\overline{CS}$ and $\overline{RE}$ are low or if $\overline{CS}$ and $\overline{WE}$ are low.
27,26 38-41, 28-37	38,37, 54-57, 39-41, 47-53	A0-A15	Address Lines Out	Sixteen address outputs from the WD2511A for DMA operation. If the control bit ADRV is 1, the outputs are TTL drives at all times. If ADRV is 0, the outputs are three-state, and are HI-Z whenever $\overline{DACK}$ is high. (ADRV is in Control Register #1.)
42	58	V_{DD}	Power Supply	+12VDC power supply input.
43	63	$\overline{INTR}$	Interrupt Request	An active low interrupt service request output. Returns to high when Status Register #1 is read.
46,47, 45,44	64,65, 66,67	IA0-IA3	Address Lines In	Four address inputs for user controlled read/write operation of the I/O registers in the WD2511A. If ADRV = 0, these may be tied to A0-A3. (ADRV is in Control Register = 1.)
48	1,68	V_{CC}	Power Supply	+5VDC power supply input.
-	6,7, 9-11, 25-27, 34,35, 42-46, 59-62	NC	No Connection	Leave pins open.

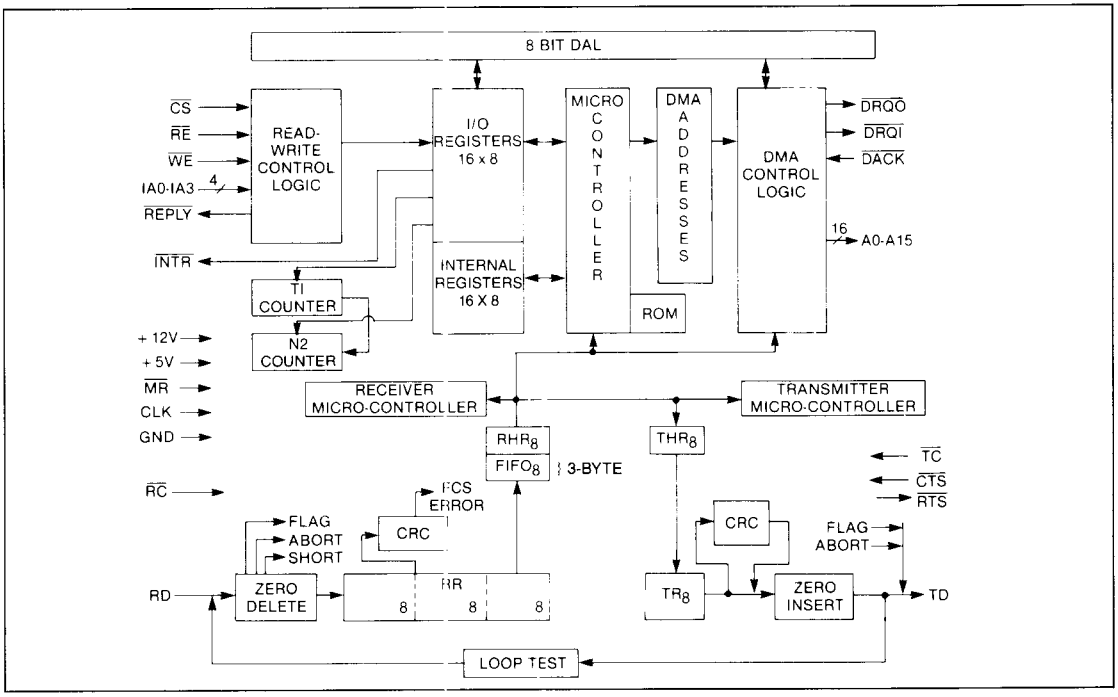


Figure 2. WD2511A BLOCK DIAGRAM — DETAILED

2.0 THE STRUCTURE OF X.25

CCITT Recommendation X.25 is an international standard for connecting computer equipment to a Public Data Network (PDN). Its uses, however, go far beyond PDN'S. X.25 is used for connecting dissimilar computer equipment together, as a gateway from one Local Area Network (LAN) to another, and in many private Wide Area Networks (WAN).

THE WD2511A AND THE ISO MODEL

The CCITT X.25 recommendation comprises three levels of protocols (Level 1 to 3).

LEVEL 3	PACKET LAYER	SOFTWARE
LEVEL 2	DATA LINK LAYER	LAPB (WD2511A)
LEVEL 1	PHYSICAL LAYER	RS232, X.21

Level 1 is the physical level, which concerns the actual means of bit transmission across a physical medium.

Level 2 is the link level which includes frame formatting, error control and link control.

Level 3 is the packet (network) level which controls the traffic of the different virtual calls and multiplexes these for passage over the physical line.

These three levels are completely independent of each other, which allows changes to be made to one level without disrupting the operation of any other level. An adjacent level is affected only if the changes affect the interface to that level.

Each level performs one well defined set of functions, using only a well defined set of services provided by the level below. These functions implement a set of service that can be accessed only from the level above. Each level is strictly controlled by the systems engineer according to formal functional and interface specifications.

The WD2511A implements level 2. Without additional logic, it generates the frame, performs error checking, performs link management (set up/disconnect) and ensures reliable data transmission by evaluating the sequence number associated with each I-frame. The device automatically acknowledges received I-frames and fully supports up to 7 outstanding (unacknowledged) frames, including retransmission if required.

2.1 FRAME FORMAT

The WD2511A performs LAPB, which is a "bit oriented" data communications protocol. According to the general format for bit-oriented procedures (HDLC, SDLC, ADCCP) each serial block of data is called a frame. Figure 3 shows the frame format.

Each frame starts and ends with a FLAG (01111110). A single FLAG may be used both as the closing flag of one frame and the opening flag of the next frame. The WD2511A can detect this shared flag if it receives one. However, the WD2511A generates at least three flags in-between each frame it transmits with the exception of ABORT frames. In this case the WD2511A could generate a single flag after an aborted frame.

In between flags, data transparency to easily distinguish flags from data is provided by the insertion of a 0 bit after

all sequences of 5 contiguous 1 bits. The receiver will strip 0 these inserted 0 bits to reconstruct the original data.

Each frame begins with an address field followed by a control field (the A and C fields). The last 16-bits before the closing flag make up the Frame Check Sequence (FCS.)

The Address field consists of one byte.

The Control field consists of one byte. The content of this field is described below.

The Information field of a frame is unrestricted with respect to code or grouping of bits. The maximum length of an I-field is a system parameter.

a. FRAME FORMATS

Bit order of transmission

12345678	12345678	12345678	16 to 1	12345678
Flag	Address	Control	FCS	Flag
F 01111110	A 8-bits	C 8-bits	FCS 16-bits	F 01111110

Bit order of transmission

12345678	12345678	12345678		16 to 1	12345678
Flag	Address	Control	Information	FCS	Flag
F 01111110	A 8-bits	C 8-bits	I N-bits	FCS 16-bits	F 01111110

FCS Frame Checking Sequence

b. WD2511A IMPLEMENTATION

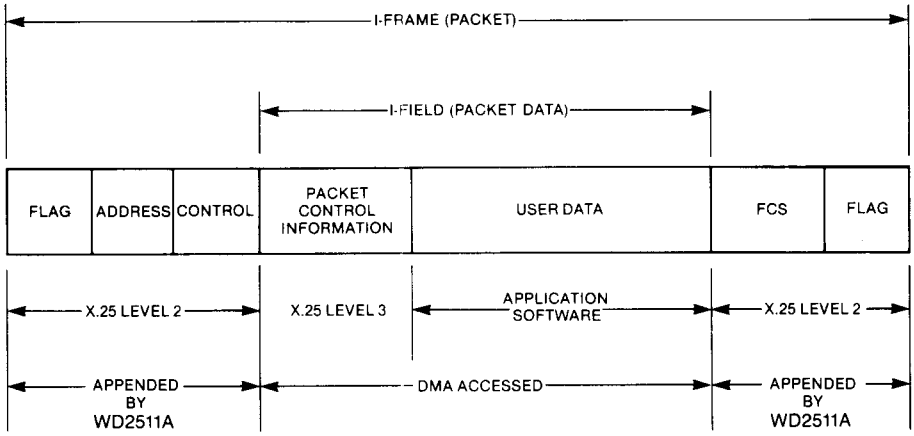


Figure 3. LAPB FRAME FORMAT

The FCS calculation includes all data between the opening flag and the first bit of the FCS, except for 0's inserted for transparency. The 16-bit FCS has the following characteristics:

Polynomial = $X^{16} + X^{12} + X^5 + 1$
Transmitted Polarity – Inverted
Transmitted Order – High Order Bit First
Preset Value – All 1's

After the frame is received, if there were no errors, the remainder in the CRC register (internal in the WD2511A) will be:

1111000010111000 (FOB8).

The WD2511A generates and tests the opening and closing Flag, FCS, A-Field, and C-Field and performs zero bit insertion and deletion.

According to the X.25 protocol, there are three types of frames: supervisory (S-frame), un-numbered (U-frame), and information (I-frame). The WD2511A performs datalink layer access control. All S- and U-frames are automatically generated and tested by the WD2511A. The user need only be concerned with the I-frames, (i.e. packets).

The WD2511A will transmit contiguous flags for interframe time fill in full duplex mode.

Addresses, commands, responses and sequence numbers are transmitted with the low order bit first. The order of transmitting bits within the information field is not specified in X.25. The WD2511A transmits the low order bit first. The FCS is transmitted to the line commencing with the coefficient of the highest term. (*Note* – The low order bit is defined as bit 1, as depicted in Figure 3.)

A frame not properly bounded by two flags, or having fewer than 32 bits between flags, is an invalid frame.

Aborting a frame is performed by transmitting at least seven contiguous 1s (with no inserted 0s).

Interframe time fill is accomplished by transmitting contiguous flags between frames.

2.2 LINK STATES

The Link is in an ACTIVE condition when a station is actively transmitting a frame, an abort sequence, or interframe time fill.

The Link is defined to be in an IDLE condition when a contiguous 1s state is detected that persists for at least 15 bit times.

2.3 CONTROL FIELD FORMATS

The control field contains a command or a response, and sequence numbers where applicable.

Three types of control field formats (see Table 2.) are used to perform the LAPB protocol: numbered information transfer (I frames), numbered supervisory functions (S frames) and unnumbered control functions (U frames).

The I format is used to perform an information transfer. The functions of N(S), N(R) and P/F are independent; i.e., each I frame has an N(S), an N(R) which may or may not acknowledge additional I frames received by the DTE or DCE, and a P/F bit.

The S format is used to perform link supervisory control functions such as to acknowledge I frames, to request retransmission of I frames, and to request a temporary suspension of transmission of I frames.

The U format is used to provide additional link control functions. This format contains no sequence numbers.

Modulus

Each I frame is sequentially numbered and may have the value 0 through modulus minus 1 (where "modulus" is the modulus of the sequence numbers). The modulus equals 8 and the sequence numbers cycle through the range of 0 to 7.

Frame variables and sequence numbers

The send state variable denotes the sequence number of the next in-sequence I frame to be transmitted. The send state variable can take on the value 0 through 7. The value of the send state variable is incremented by 1 with each successive I frame transmission, but cannot exceed N(R) of the last received I or S frame by more than the maximum number of outstanding I frames (7). Only I frames contain N(S), the send sequence number of transmitted frames. Prior to transmission of an in-sequence I frame, the value of N(S) is set equal to the value of the send state variable.

The receive state variable denotes the sequence number of the next in-sequence I frame to be received. The receive state variable can take on the value 0 through 7. The value of the receive state variable is incremented by one with the receipt of an error free, in-sequence I frame whose send sequence number N(S) equals the receive state variable. All I frames and S frames contain N(R), the expected sequence number of the next received I frame. Prior to transmission of a frame of the above types, the value of N(R) is set equal to the current value of the receive state variable. N(R) indicates that the DTE or DCE transmitting the N(R) has received correctly all I frames numbered up to and including N(R) – 1.

2.4 SUPERVISORY COMMANDS AND RESPONSES

The Receive Ready (RR) supervisory frame is used to indicate it is ready to receive an I frame or to acknowledge previously received I frames numbered up to and including N(R) – 1. RR may be used to clear a busy condition that was initiated by the transmision of an RNR.

The Reject (REJ) supervisory frame is used to request retransmission of I frames starting with the frame numbered N(R). I frames numbered N(R) – 1 and below are acknowledged. Additional I frames pending initial transmission may be transmitted following the retransmitted I frame(s).

Only one REJ exception condition for a given direction of information transfer may be established at any time. The REJ exception condition is cleared (reset) upon the receipt of an I frame with an N(S) equal to the N(R) of the REJ.

The Receive Not Ready (RNR) supervisory frame is used to indicate a busy condition: i.e., temporary inability to accept additional incoming I frames. I frames numbered up to and including N(R) – 1 are acknowledged. I frame N(R) and subsequent I frames received, if any, are not acknowledged; the acceptance status of these I frames will be indicated in subsequent exchanges. An indication that the busy condition has cleared is communicated by the transmission of a UA, RR, REJ or SABM. The RNR, REJ or RR command with the P bit set to 1 may be used to ask for the status of the remote device.

2.5 UNNUMBERED COMMANDS AND RESPONSES

The SABM unnumbered command is used to place the addressed station in the asynchronous balanced mode (ABM) information transfer phase. No information field is permitted with the SABM command. A station confirms acceptance of SABM by the transmission at the first opportunity of a UA response. Upon acceptance of this command the send state variable and receive state variable are set to 0. Previously transmitted I frames that are unacknowledged when this command is actioned remain unacknowledged and must be resent after the Link has been brought back up.

The DISC unnumbered command is used to terminate the mode previously set. It is used to inform the station

receiving the DISC that the sender is suspending operation. No information field is permitted with the DISC command. The receiving station confirms the acceptance of DISC by the transmission of a UA response. Previously transmitted I frames that are unacknowledged when this command is actioned remain unacknowledged.

The UA unnumbered response is used to acknowledge the receipt and acceptance of the U format commands. No information field is permitted with the UA response.

The DM unnumbered response is used to report a status where a station is logically disconnected from the link, and is in the disconnected phase. The DM response may be sent in this phase to request a set mode command, or, if sent in response to the reception of a set mode command, to say that the station is still in the disconnected phase and cannot action the set mode command. No information field is permitted with the DM response.

The FRMR response is used to report an error condition not recoverable by retransmission of the identical frame; i.e., one of the following conditions, which results from the receipt of a frame without FCS error:

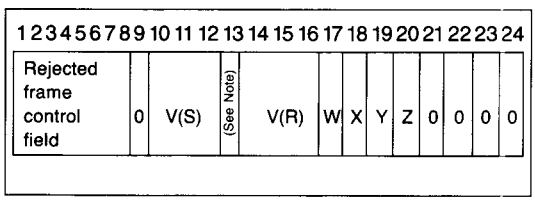
1) the receipt of a command or response that is invalid or not implemented;
2) the receipt of an I-frame with an information field which exceeds the maximum established length;
3) the receipt of an invalid N(R);
4) the receipt of a frame with an information field which is not permitted or the receipt of an S or U frame with incorrect length.

An invalid N(R) is defined as one which points to an I-frame which has previously been transmitted and acknowledged or to an I-frame which has not been transmitted and is not the next sequential I-frame pending transmission.

An information field which immediately follows the control field, and consists of 3 octets, is returned with this response and provides the reason for the FRMR response. This format is given in Figure 4 below.

Figure 4: FRMR information field format

Information field bits



- Rejected frame control field is the control field of the received frame which caused the command (frame) reject.
- V(S) is the current send state variable value at the station reporting the rejection condition (bit 10 = low order bit).
- V(R) is the current receive state variable value at the station reporting the rejection condition (bit 14 = low order bit).
- W set to 1 indicates that the control field received and returned in bits 1 through 8 was invalid or not implemented.
- X set to 1 indicates that the control field received and returned in bits 1 through 8 was considered invalid because the frame contained an information field which is not permitted or is an S or U frame with incorrect length. Bit W must be set to 1 in conjunction with this bit.
- Y set to 1 indicates that the information field received exceeded the maximum established capacity of the station reporting the rejection condition.
- Z set to 1 indicates that the control field received and returned in bits 1 through 8 contained an invalid N(R).

Note – For FRMR, bits 9 and 21 to 24 shall be set to 0. Bit 13 shall be set to 1 if the frame rejected was a response, and set to 0 if the frame rejected was a command.

2.6 EXCEPTION CONDITION REPORTING AND RECOVERY

The error recovery procedures which are available to effect recovery following the detection/occurrence of an exception condition at the link level are described below. Exception conditions described are those situations which may occur as the result of transmission errors, device malfunction or operational situations.

A busy condition results when a station is temporarily unable to continue to receive I-frames due to internal constraints, e.g., receive buffering limitations. In this case an RNR frame is transmitted from the busy station. I-frames pending transmission may be transmitted from a busy station.

An N(S) sequence exception condition occurs in the receiver when an I-frame received error-free (no FCS error) contains an N(S) which is not equal to the receive state variable at the receiver. The receiver does not acknowledge (increment its receive state variable) the I-frame causing the sequence error, or any I-frames which may follow, until an I-frame with the correct N(S) is received. The information field of all I-frames whose N(S) does not equal the receive state variable will be discarded.

A station receiving one or more I-frames having sequence errors but otherwise error free accepts the control information contained in the N(R) field and the P bit to perform link control functions; e.g., to receive acknowledgement of previously transmitted I-frames. Therefore, the retransmitted I-frame may contain an N(R) field and P bit that are updated from, and therefore different from, the ones contained in the originally transmitted I-frame.

The REJ is used to initiate an exception recovery (retransmission) following the detection of a sequence error. Only one "sent REJ" exception condition from a station is established at a time. A sent REJ exception condition is cleared when the requested I-frame is received. A station receiving REJ initiates sequential (re-) transmission of I-frames starting with the I-frame indicated by the N(R) obtained in the REJ frame.

If a station, due to a transmission error, does not receive (or receives and discards) a single I-frame or the last I-frame in a sequence of I-frames, it will not detect an out-of-sequence exception condition and therefore will not transmit REJ. The station that transmitted the unacknowledged I-frame(s), following the completion of a system specified time-out period, takes recovery action as described in section 2.8 to determine at which I-frame retransmission must begin.

2.7 LINK SET UP AND DISCONNECTION

Should a station wish to set up the link, it sends an SABM command to which it expects a UA response.

Should a station wish to disconnect for any reason, it sends a DISC command, to which it expects a UA or DM response.

2.8 USE OF POLL BIT

One use of the Poll bit (P) is in conjunction with Time-Out Recovery. Timer T1 is started at the beginning of a transmitted command provided it has not been previously started. If T1 runs out, the command will be retransmitted with P = 1. If T1 runs out again, the command will again be retransmitted, with P = 1 up to N2 times. At N2 + 1, an error interrupt will occur. If the command was an S-frame, the WD2511A will reset the link by transmitting a SABM. If the command was a SABM, the WD2511A will send a DISC. If a DISC, the WD2511A will continue to send a DISC indefinitely until the remote station responds.

2.9 TRANSMISSION OF ABORT

An Abort (seven contiguous 1's) is transmitted to terminate a frame in such a manner that the receiving station will ignore the frame. There are two conditions which will cause the WD2511A to transmit an ABORT.

1. Transmitter Under-Run
2. While transmitting a packet, a REJ is received.

2.10 LAPB PROCEDURE

The Link Access Procedure Balanced (LAPB) is described in CCITT Recommendation X.25 as the Level 2 protocol for the Asynchronous Balanced Mode (ABM).

The DTE is the Data Terminal Equipment and the DCE is the Data Circuit Termination Equipment (the network side of the DTE-DCE connection).

Unlike the earlier Master/Slave Protocols, the DTE and DCE are each "combined" stations in that each can transmit and receive commands and responses. Whether a particular frame is to be taken as a Command or a Response is determined by the contents of the address field. Commands from the DCE and the associated responses from the DTE use address A (hex 03). Commands from the DTE and the associated responses from the DCE use address B (hex 01).

Refer to CCITT Recommendation X.25, 1980 (Yellow Book) for more details on the X.25 LAPB protocol itself. Appendix D contains additional information pertaining to the WD2511A's specific implementation. The individual commands and responses are summarized in Table 2.

2.11 THE DIFFERENCE BETWEEN LAP AND LAPB

In March 1976, the CCITT adopted Recommendation X.25 as an interface standard for public packet-switching networks. The link level procedure adopted was called Link Access Procedure (LAP) and used the HDLC Asynchronous Response Mode (ARM). However, ARM was not designed for peer-to-peer communications so LAP had some subtle problems. Therefore, in 1977, when Provisional Recommendation X.25 was adopted, a procedure called LAPB was added. LAPB is Link Access Procedure-Balanced and operates under the HDLC Asynchronous Balanced Mode (ABM). Unfortunately, the 1977 LAPB lacked good symmetry between the DTE and DCE, and was unworkable.

In the April 1979 CCITT meeting, the LAPB was greatly enhanced, especially in the DTE/DCE symmetry. This enhanced version was approved in the February 1980 Plenary meeting of the CCITT. We now have a good, workable LAPB standard. LAPB is a superior procedure and the usage of LAP is being replaced with LAPB.

FRAME TYPE	COMMAND	RESPONSE	BIT #							
			7	6	5	4	3	2	1	0
INFORMATION (I)	I-FRAME (PACKET)		N(R)			P	N(S)			0
UNNUMBERED (U)	SABM		0	0	1	P	1	1	1	1
	DISC		0	1	0	P	0	0	1	1
		UA	0	1	1	F	0	0	1	1
		FRMR	1	0	0	F	0	1	1	1
		DM	0	0	0	F	1	1	1	1
SUPERVISORY (S)	RR	RR	N(R)			P/F	0	0	0	1
	RNR	RNR	N(R)			P/F	0	1	0	1
	REJ	REJ	N(R)			P/F	1	0	0	1

LAPB Commands and Responses (Bit 0 is transmitted first). Only the FRMR and I-frame contain I-fields.

Table 2. FRAME TYPES

N(S) Transmitter send sequence number (bit 2 = low order bit)
N(R) Transmitter receive sequence number (bit 6 = low order bit)
S Supervisory function bit
M Modifier function bit
P/F Poll bit when issued as a command, final bit when issued as a response (1 = Poll/Final)

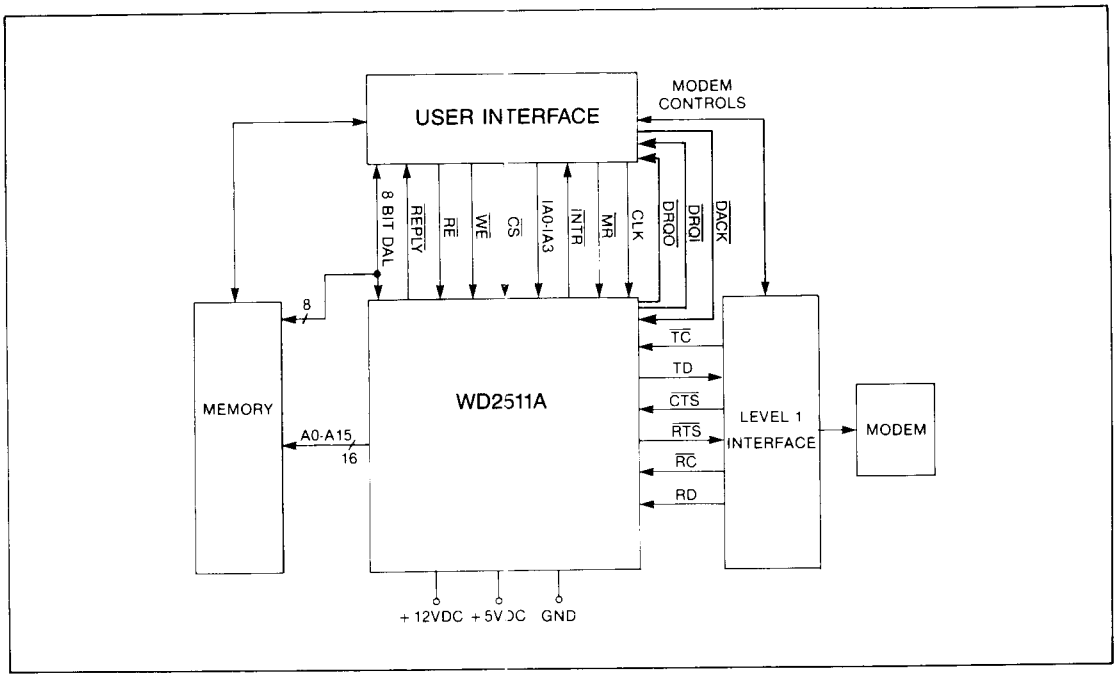


Figure 5. SYSTEM CONNECTION INTERFACE

3.0 REGISTER DESCRIPTION

Figure 5 shows how the WD2511A interfaces to other system components.

The WD2511A is controlled and monitored by sixteen I/O registers. Control, status, and error bits will be referred to as CR, SR, or ER, respectively, along with two digits. For example, SR16 refers to status register #1 and bit 6, which is "XBA."

Table 3. REGISTER DEFINITION

REG #	IA3	IA2	IA1	IA0	REGISTER	REGISTER GROUPING
0	0	0	0	0	CR0	OVERALL CONTROL AND MONITOR
1	0	0	0	1	CR1	
2	0	0	1	0	*SR0	
3	0	0	1	1	*SR1	
4	0	1	0	0	*SR2	
5	0	1	0	1	*ER0	
6	0	1	1	0	*CHAIN MONITOR	RECEIVER MONITOR
7	0	1	1	1	*RECEIVED C-FIELD	
8	1	0	0	0	T1	TIMER
9	1	0	0	1	N2/T1	
A	1	0	1	0	TLOOK HI	DMA SET-UP
B	1	0	1	1	TLOOK LO	
C	1	1	0	0	CHAIN/BUFFER SIZE	
D	1	1	0	1	NOT USED	
E	1	1	1	0	XMT COMMAND "E"	"A" FIELD
F	1	1	1	1	XMT RESPONSE "F"	

*USER READ ONLY. (Write Not Possible)

NOTE:

Registers 8 through F should be set-up while MDISC = 1.

Table 4. CONTROL, STATUS, ERROR REGISTERS

REGISTER	BIT #							
	7	6	5	4	3	2	1	0
CR0	$\overline{\text{ADISC}}$	0	H/$\overline{\text{F}}$	ACTIVE / $\overline{\text{PASSIVE}}$	LOOP TEST	RAMT	RECR	MDISC
CR1	TXMT	TRCV	$\overline{\text{XI}}$	ADRV	0	0	0	SEND
SR0	NA2	NA1	NA0	RNRR	NB2	NB1	NB0	RNRX
SR1	[1]PKR	[1]XBA	[1]ERROR	0	NE2	NE1	NE0	0
SR2	T1OUT	$\overline{\text{IRTS}}$	REC IDLE	0	0	0	0	$\overline{\text{LINK}}$
ER0	ER07	ER06	ER05	ER04	ER03	ER02	ER01	ER00

[1]Causes interrupt ($\overline{\text{INTR}}$ Goes Low).

3.1 CONTROL REGISTERS (CR0/CR1)

Table 5. CONTROL REGISTER 0

REGISTER	CR07	CR06	CR05	CR04	CR03	CR02	CR01	CR00
CR0	$\overline{\text{ADISC}}$	0	H/$\overline{\text{F}}$	ACTIVE / $\overline{\text{PASSIVE}}$	LOOP TEST	RAMT	RECR	MDISC

BIT	DESCRIPTION
CR00	MDISC (Mandatory Disconnect command). After Master Reset MDISC will be set. Clearing MDISC causes the W2511A to move to its initialization state where it will be ready to activate the Link Layer protocol. Setting MDISC causes the chip to send one or more DISC frames to disconnect the link. No DMA accessed data will be transferred as long at MDISC is set.
CR01	RECR (Receiver Ready). Indicates the CPU's receiver buffer is Ready (CR01 = 1). If RECR = 1, the WD2511A may begin receiving I-frames. (See SR00)
CR02	RAMT (RAM Test). This bit activates an internal register test. Refer to SELF TESTS section for details.
CR03	Loop Test. The LOOP TEST bit will connect the transmit data output to the receive data input. The receiver input pins RD an $\overline{\text{RC}}$ are then logically disconnected from the internal circuitry. The "E" and "F" data register of the A-field must be equal and CR04-7 should equal 1001.
CR04	ACTIVE/$\overline{\text{PASSIVE}}$. Once MDISC has been cleared this bit takes effect. When set, the WD2511A will initiate the link up procedures; when cleared, it will wait for a link up command or DM response from the remote station. (See Note)
CR05	H/$\overline{\text{F}}$. Selects full duplex if CR05 = 0 and half duplex if CR05 = 1 (See Appendix B for half-duplex operation)
CR06	Reserved, should be Zero
CR07	$\overline{\text{ADISC}}$. Used when CR04 = 1 (ACTIVE). When the WD2511A actively initiates link set-up, a DISC will be transmitted and acknowledged prior to transmission of the SABM if CR07 = 0. If CR07 = 1, The WD2511A will send only the SABM. (see Note)

NOTE:

Changes to CR07 and CR04 only take affect when the chip is disconnected (MDISC = 1).

Table 6. CONTROL REGISTER 1

BIT	DESCRIPTION
CR10	SEND. The SEND bit is used to command the WD2511A to send one or more packets of user data. If SEND is set, the WD2511A will read from TLOOK the BRDY bit of the next segment to be transmitted. If BRDY is clear, the Send bit will be cleared, and no transmission of user data takes place. If BRDY is set, however, the WD2511A will read TSADR and TCNT from that TLOOK segment and transmit the data accordingly. After transmitting the data, the WD2511A clears BRDY of the segment just sent and reads BRDY for the next TLOOK segment. If BRDY is set, the user data is sent as above. If BRDY is not set, the SEND bit is cleared and transmission of user data stops until SEND is set again. As a matter of good practice, SEND should be set each time a BRDY bit is set.
CR11-13	Reserved – should be zeros.
CR14	ADRV. The Address Valid bit is the control for the 16-bit output addresses (A0-A15). If ADRV is clear, the outputs are tri-state and are in Hi-z (except when DACK is low). If ADRV is set, the outputs are always low impedance (TTL) and are forced hi-level (logical 1) when $\overline{DRQ0}$, $\overline{DRQ1}$, and $\overline{DACK}$ are all high.
CR15	$\overline{XI}$. This bit is used in conjunction with CR17 (TXMT) for transparent mode transmission. If $\overline{XI}$ is clear, the frame to be sent has an I-field, i.e. is 3 or more bytes long; if XI is set, the frame is 2 bytes long. (excluding FCS and Flag). (See Note)
CR16	TRCV. The Transparent Receive bit, when set, causes the WD2511A to pass all received unknown frames to the user. (See Note)
CR17	TXMT. The Transparent Transmit bit, when set, allows the user to build a frame for the WD2511A. (See Note).

NOTE:

Refer to Appendix A for a complete description of Transparent MODE.

3.2 STATUS REGISTERS (SR0/SR1/SR2)

Table 7. STATUS REGISTER 0

REGISTER	SR07	SR06	SR05	SR04	SR03	SR02	SR01	SR00
SR0	NA2	NA1	NA0	RNRR	NB2	NB1	NB0	RNRX

BIT	DESCRIPTION
SR00	RNRX. An RNR has been transmitted or will be at the next opportunity. The CPU should set RECR when receive buffers are available and leave it set.
SR03-SR01	NB2-NB0. Next block to be transmitted. (See Note)
SR04	RNRR. This bit is set when an RNR frame is received. Once set, it is cleared when an RR, REJ, SABM, or UA is received.
SR07-SR05	NA2-NA0.Next block of transmitted data to be acknowledged.

NOTE:

Values zero through 7 correspond to NS in the control field of transmitted information frames. When reading the values in SR0, read the value until 2 consecutive reads produce identical values.

Table 8. STATUS REGISTER 1

REGISTER	SR17	SR16	SR15	SR14	SR13	SR12	SR11	SR10
SR1	↑PKR	↑XBA	↑ERROR	0	NE2	NE1	NE0	0

BIT	DESCRIPTION
SR10	0 (not used)
SR13-SR11	NE2-NE0. Next expected packet number and next RLOOK segment number. (See Note 2)
SR14	0 (not used)
SR15	ERROR. The ERROR bit indicates: An error has occurred which is not recoverable by the WD2511A or a significant event has occurred. (see Note 3) For the specific reason for the ERROR bit being set, see error register (ER0) on next page.
SR16	XBA. The XBA (transmitted block acknowledgement) bit set indicates that one or more previously transmitted Blocks have been acknowledged by the remote Station. Upon acknowledgement, the ACK'ED bit is set for each segment in TLOOK which was acknowledged.
SR17	PKR. The PKR (Packet Received) bit set indicates one or more packets have been received error-free and in correct sequence according to the received NS field. The I-field data is in the user's RAM memory. NE is advanced.

Table 9. STATUS REGISTER 2

REGISTER	SR27	SR26	SR25	SR24	SR23	SR22	SR21	SR20
SR2	T1OUT	$\overline{\text{IRTS}}$	REC IDLE	0	0	0	0	$\overline{\text{LINK}}$

BIT	DESCRIPTION
SR20	$\overline{\text{LINK}}$. If the link is established, $\overline{\text{LINK}}$ = 0. If the link is logically disconnected, $\overline{\text{LINK}}$ = 1.
SR24-SR21	Unused Bits-0.
SR25	REC IDLE. Receiver Idle indicates that the WD2511A has received at least 15 contiguous 1's.
SR26	$\overline{\text{IRTS}}$. Internal Request-To-Send indicates that the transmitter is attempting (successful or not) to send either data or flags.
SR27	T1OUT. Timer T1 has timed-out when set. This bit returns to 0 when T1 is re-started. When T1OUT = 1, T1 is not running. NOTE: This bit could be a 1 for a few microseconds in between intervals when T1 stops and is restarted.

NOTE 1:

The three interrupt-causing bits are SR17, SR16, and SR15. Any of the three will cause an interrupt request ($\overline{\text{INTR}}$ goes low). After SR1 is read, all three bits are rest to 0, and $\overline{\text{INTR}}$ returns to high. Before exiting the interrupt service routine, re-read SR1 to insure there are no new interrupts to be processed.

NOTE 2:

Values 0 through 7 correspond to NS in the Control field of received information frames. When reading this field, read the value until the results of 2 consecutive reads are identical.

NOTE 3:

After MR, there exists a possibility that the chip will falsely report one transmitter underrun when MDISC is cleared. This error after MR is not significant; clear it by reading register SR1 then continue with initialization.

3.3 ERROR REGISTER (ER0)

Table 10. ERROR REGISTER 0

HEX VALUE	ERROR/EVENT
02	Receiver overrun (ROR). The Receiver Register (RR) had a character to load into the FIFO but the FIFO was full. (Note 2.)
04	Transmitter underrun (TUR). The transmitter (TR) needed a character from the Transmitter Holding Register (THR) but the THR was not ready. The frame being transmitted is aborted. (Note 2.)
10	RLOOK not ready. REC RDY bit of next segment is 0 but RECR = 1. This interrupt will not occur if RECR = 0.
21	Link is up. Was down.
22	DISC sent. REC IDLE for time T1 x N2.
24	DISC sent. SABM sent N2 times without receiving UA.
30	Received DISC or DM while link was up.
41	Going to next receive chain segment.
42	Next chain segment of the Receiver was not ready.
80	Link reset (SABM) received.
88	S-command sent N2 times without acknowledgement.
C0	Frame Reject (FRMR) received. (Note 1.)
C1	Frame Reject (FRMR) transmitted. (Note 3.) The received C-Field (returned in the first I-field byte of the FRMR (frame) was invalid. (W)
C3	Frame Reject (FRMR) transmitted. (Note 3.) The received and rejected frame contained an I-field which is not permitted with this frame type. (W,X)
C4	Frame Reject (FRMR) transmitted. (Note 3.) Received I-field exceeded the total amount of I-field data bytes established in Register C. (Y)
C8	Frame Reject (FRMR) transmitted. (Note 3.) The received frame contained an invalid N(R). (Z)

NOTES:

1. Whenever a Frame Reject (FRMR) is received, the I-field will have been placed in the next available RLOOK memory location by the DMA. A link reset (SABM) will be transmitted but no additional interrupt is generated. The NB is not advanced.
2. Receiver overrun and Transmitter underrun are an indication that the $\overline{TC}/\overline{RC}$ clocks are either too fast for the WD2511A, or the $\overline{DACK}$ response is too slow, or both. The chip recovers from these errors the same as for line noise errors – the protocol time out recovery. The host need only monitor TUR/ROR for excessive occurance. An initial TUR interrupt (at start up) is not valid and should be ignored unless it occurs a second time.
3. As a result of FRMR transmitted, a SABM is received, causing link reset. In this case, only the Frame Reject interrupt is indicated.

FRMR FORMAT

A frame reject (FRMR) contains a three byte I-field. The first byte is the rejected frame control field. The second byte contains the current N(S) and N(R) counts of the station reporting the reject condition. The third byte contains W-X-Y-Z-0-0-0-0 where W is the LSB.

W indicates that the control field received and returned in the first I-field byte was invalid.

X indicates the rejected frame contained an I-field which is not permitted with this command. W is also set in this case.

Y indicates the received I-field exceeded the maximum I-field data byte count established (CHAIN/BUFFER SIZE). Y is mutually exclusive with W.

Z indicates the received control field contained an invalid N(R). Z is mutually exclusive with W.

In the FRMR I-field, bit 4 of the second byte is set if the rejected frame was a response.

Upon receiving a FRMR, the WD2511A will place the 3 byte I-field in memory by DMA, just as if the FRMR were a packet.

When the WD2511A transmits a FRMR, the frame reject condition is entered. Only a received SABM or DISC will clear this condition. If any other command is received, the WD2511A will re-transmit the FRMR. Also, the WD2511A will not transmit packets while in the frame reject condition. If no SABM is received after N2 X T1 time, the WD2511A will send a SABM.

3.4 RECEIVED C-FIELD REGISTER

Register 7 contains the C-field of the last received frame, provided the A-field of the frame was equal to either register E or F, the FCS was good, the frame contained 32 or more bits, and the WD2511A is not waiting for a SABM or DISC in response to a transmitted FRMR.

3.5 TIMER/COUNTER REGISTER

Registers 8 and 9 define a 10-bit timer (T1) and a 6-bit Maximum Number of Transmission/Retransmissions counter (N2).

	BIT #							
REGISTER	7	6	5	4	3	2	1	0
8	T1							LSB
9	N2							
	MSB					LSB	MSB	

MSB = Most Significant Bit
LSB = Least Significant Bit

T1 provides the value of a delay in waiting for a response and/or acknowledgement. The delay is the binary count multiplied by time CT where:

$$CT = \frac{16384}{CLK} \text{sec}$$

Thus, if CLK = 2 MHz, then T1 may be set in increments 8,192 milliseconds, to a maximum delay of 8.39 seconds. All ones in T1 is the maximum delay.

Once the CPU establishes T1 and N2, there is no need to write into T1 and N2 again unless a master reset (MR) has occurred, there is a power loss, or the user needs to change T1 or N2. If a time-out occurs, the WD2511A will still retain T1 and N2.

The conditions for starting, stopping, or restarting T1 are shown in Figure 6. ("Re-start" means starting T1 before it ran-out).

3.6 ADDRESS REGISTERS

Registers E and F provide a programmable A-field. This allows the WD2511A to be a super-set of the X.25 document. That is, the WD2511A can handle a wider range of applications than the DTE-DCE links defined in X.25. These wider ranges include: DTE-to-DTE connection, multipoint, and loop-back testing. If the WD2511A is strictly in an X.25 DTE-DCE link, use the values shown below:

DTE Register E = 01
Register F = 03

DCE Register E = 03
Register F = 01

If performing a loop-back test, either internal (CR03 = 1) or external (CR03 = 0), registers E and F should be the same.

4. MEMORY ACCESS METHOD

The WD2511A memory access is accomplished by the use of DMA and two look-up tables. These tables are set-up to allow up to 7 I-frames to be outstanding in each direction of the communication link. The look-up tables are divided into a transmit and a receive area. (TLOOK and RLOOK) and are located in memory external to the WD2511A.

TLOOK
RLOOK

These tables contain address and control information for individual Transmit/Receive packets.

To provide the WD2511A access to TLOOK and RLOOK load only the starting address of TLOOK into the WD2511A registers A and B.

REG A	A15	A14	A13	A12	A11	A10	A9	A8
REG B	A7	A6	A5	A4	A3	A2	A1	A0

A0-A15 16 bit TLOOK starting address

The TLOOK and RLOOK tables are each divided into 8 segments and each segment contains 8 bytes. Figure 7 illustrates the segmentation of TLOOK and RLOOK. Figures 9 and 11 illustrate the contents of a single TLOOK and RLOOK segment.

START T1	RE-START T1	STOP T1
1. *I-frame sent and T1 not already in progress due to previous I-frame.	*Acknowledgement received to some, but not all, I-frames.	Acknowledgement received for all I-frames.
2. —	*RNR received while link up.	
3. *SABM or DISC sent. (N2 restarted at first occurrence)	—	UA or DM Received
4. Receiver Idle (REC IDLE = 1)	*Frame sent, while REC IDLE = 1	Detected REC IDLE = 0
5. S — command sent		—

*N2 is restarted.

Figure 6. TIME OUT CONDITIONS

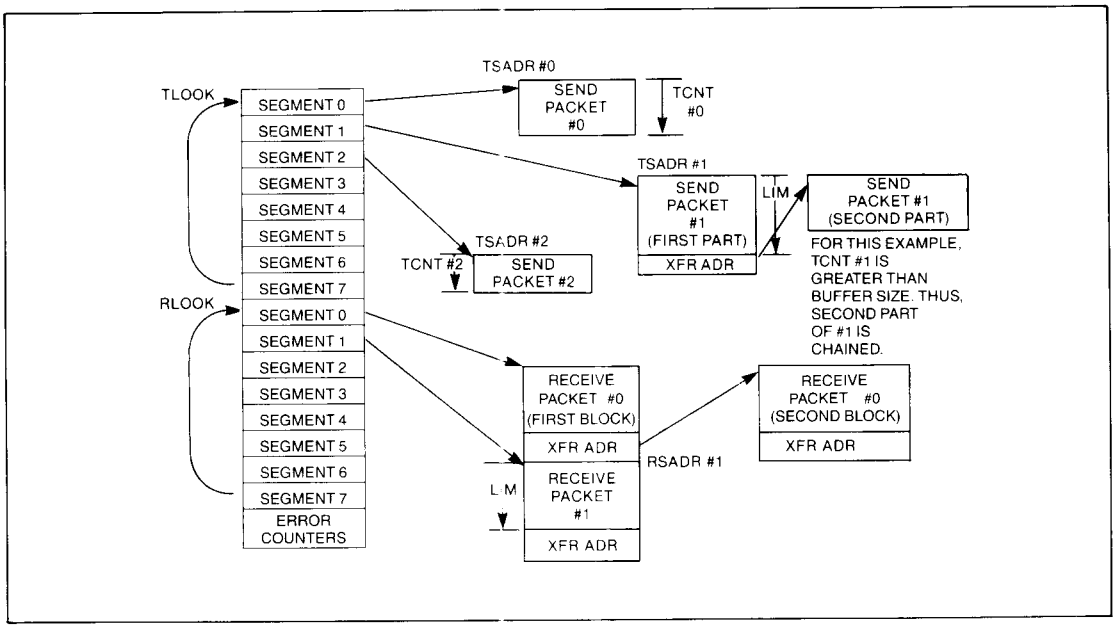


Figure 7. MEMORY ACCESS SCHEME

4.1 TRANSMIT (TLOOK)

Figure 9 details the individual segments of the TLOOK table which the user must program to cause data to be sent. The WD2511A will transmit data from host memory buffers when the next sequential TLOOK segment (starting at 0, modulo 8) has a buffer available, indicated by BRDY = 1 in TLOOK, and SEND = 1 in CR1.

The address field (TSADR) contains a 16 bit binary field pointing to the user's memory buffer. HI represents the upper 8 bits; LO the lower 8. The length field (TCNT) is a 12 bit binary field representing the length of the Information field to be sent including any Chain Pointers (refer to section 4.5 on Chaining even if you do not intend to use Chaining.)

To transmit, the WD2511A reads from TLOOK the starting address and length of the packet to be transmitted. It automatically transmits the Flag, Address, and Control fields. Next, the Information field data is transmitted from the memory buffer using DMA. At the end of the Information field, the WD2511A automatically appends the FCS and the closing Flag.

After the frame has been sent, the WD2511A will clear BRDY in the current TLOOK segment and move to the next. If BRDY is set, the process will repeat. If the WD2511A encounters a TLOOK segment where BRDY = 0, it will stop transmitting and clear SEND. To cause new buffers to be sent, the user need only set up the next sequential TLOOK segment(s), set BRDY, and then set SEND again.

After transmitting a packet, LAPB requires that an acknowledgement (ACK) be received from the remote device. ACKs are contained in the N(R) field of a received S-frame or I-frame. Upon receipt of an ACK, the WD2511A will set the ACK'ED bit in the appropriate TLOOK segment(s) and generate a Transmitted Block Acknowledged interrupt (XBA = 1). Note that more than one packet at a time can be ACK'd by the remote. This means that more than one TLOOK's ACK'ED bit could be set on one XBA interrupt.

Before assigning a new buffer to a TLOOK segment, the user must make sure that the previous block which used the segment has been ACK'd, i.e. ACK'ED = 1.

The WD2511A sets a timer (T1) to monitor the receipt of frame acknowledgements. If the timer expires, the WD2511A will poll the remote device for its status. The WD2511A also responds to requests to resend packets that were missed or received in error. If retransmission of one or more (up to a maximum of 7 out of the 8) packets becomes necessary, the WD2511A automatically resends the required packets. The user's software does not become involved in this retransmission. An error counter is incremented when this occurs (see Error Counters, section 4.8). The status of SEND has no effect on the WD2511A's monitoring of frame acknowledgement.

Under normal operation, the user brings up the link by programming CR0. Once communication with the remote device has been established and the link has been initialized (LINK = 0), user data can be sent using the above procedures. Note, however, that if the user sets up TLOOK and sets SEND, the WD2511A will begin to send the user's data, even if LINK = 1 (Link is "down"). In this state, the WD2511A will not monitor for ACKs received and will reset itself once the Link Initialization does occur. Therefore, the user should never set SEND until after the Link is up (LINK = 0) unless operating in the Transparent mode (see Appendix A).

4.2 RECEIVE (RLOOK)

Figure 11 details the individual segments for the RLOOK table which the user must program to cause data to be received from the remote device. The WD2511A will DMA data to host memory buffers when the next sequential RLOOK segment (starting at 0, modulo 8) has a buffer available, indicated by REC RDY = 1 in RLOOK, and RECR = 1 in CR0.

The address field (RSADR) contains a 16 bit binary field. HI represents the upper 8 bits; LO the lower 8. The length field (RCNT) is a 12 bit binary field representing the length of the received Information field including any Chain Pointers (refer to section 4.5 on Chaining even if you do not intend to use Chaining).

To receive user data into memory, the WD2511A reads from RLOOK the starting address and length of an empty buffer. As an incoming frame is being received, it is checked for correct Address and FCS fields and for Control field content. If it is an Information frame, the Information field is placed in the empty user buffer. If it arrives without an FCS error and with the proper N(S) sequence number, the WD2511A will, in order:

- Clear REC RDY, set FRCML (frame complete), and store any received residual count for non-8 bit byte data) in byte 0 of the current RLOOK segment,
- Store the received length in bytes of the I-field in RCNT HI and RCNT LO (includes chain pointers)
- Advance the NE count (modulo 8) in SR1,
- Generate a Packet Received (PKR) interrupt (even if the N(R) field is not valid).

Then, if the N(R) field was valid, the WD2511A prepares for the next packet by moving to the next RLOOK segment (modulo 8) and acknowledges the received I-frame at the first opportunity.

Note that if the host is processing the RLOOK entries, sequentially examining the FRCML bits, there is a possibility that the host could read the byte count of a completed frame before the chip updates it. For this reason, the RCNT HI/LO should be initialized to 0 each time a new empty buffer is stored into an RLOOK segment. Then, if the host ever reads an RCNT of 0, it should read RCNT once more.

If the WD2511A encounters an RLOOK segment where REC RDY = 0, it will stop receiving Information frames and transmit an RNR (Receiver Not Ready) to the remote device. It will also generate an Error interrupt 10 (RLOOK not ready) if RECR = 1. To cause new frames to be received, the user must set up the next sequential RLOOK segment(s) again and set REC RDY. The WD2511A will begin to receive data again after the remote device sends its next command. Refer to Appendix D, State Table IV, and to appendix E for more details.

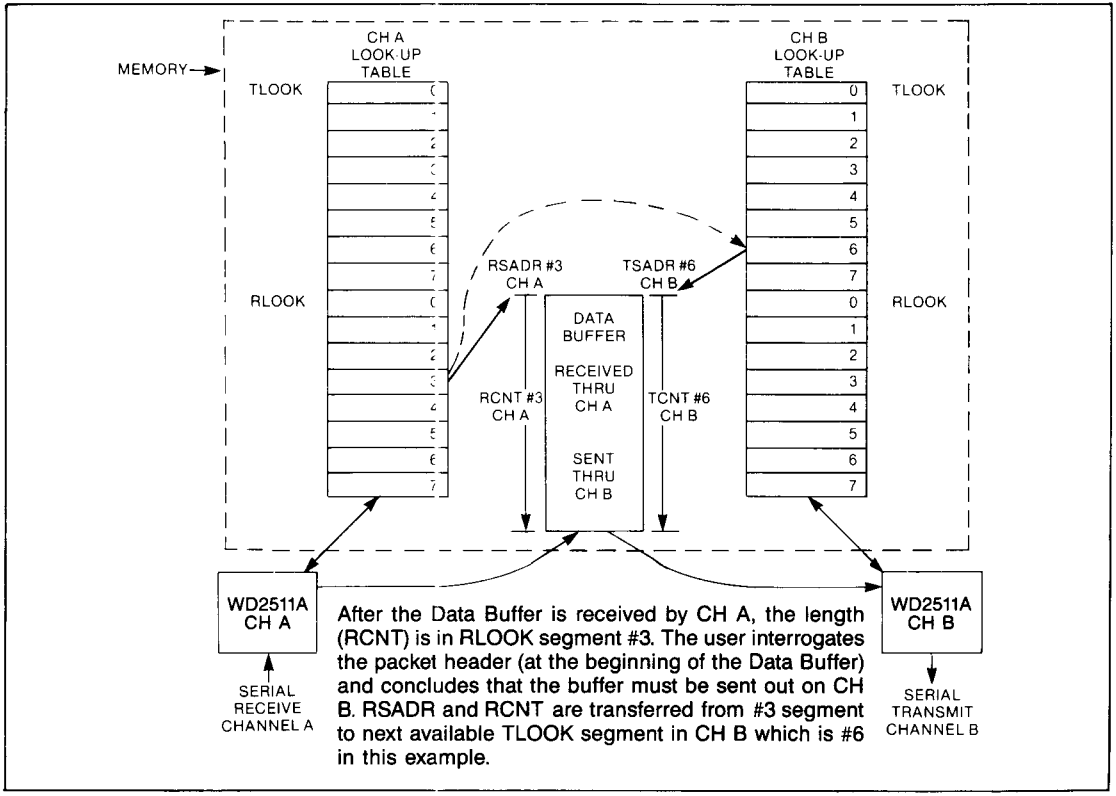


Figure 8. STORE-AND-FORWARD EXAMPLE

Note that, unlike SEND described above, RECR is not cleared by the WD2511A and may be set up before the link is up. If RECR is set and if there is a buffer available in RLOOK segment 0, DMA into that buffer can occur even when the link has not been brought up. If the WD2511A has been programmed to come up as a passive station (CR04 = 1), then an arriving Information frame with a good FCS and an N(S) = 0, will cause the WD2511A to generate a PKR interrupt as described above. The WD2511A will not, however, send out the ACK. Therefore, the user will have to initialize RLOOK again. The remote device will resend the frame after the link is up. The PKR is not generated if the WD2511A has sent a SABM or FRMR.

Whether the WD2511A accesses RLOOK/TLOOK information or a user buffer, a DMA cycle is required for each access.

Figure 8 shows a Store and Forward example that is useful in a network node.

BYTE # IN SEGMENT	BIT #							
	7	6	5	4	3	2	1	0
1	ACK'ED	0	0	0	0	0	BRDY	BRDY*
2	TSADR HI							
3	TSADR LO							
4	0				TCNT HI			
5	TCNT LO							
6	SPARE FOR USER DEFINITION							
7	SPARE							
8	SPARE							

* BRDY is cleared after the last flag is transmitted and NB is incremented.

Figure 9. TLOOK SEGMENT

The control bits in TLOOK (BRDY and ACK'ED) and in RLOOK (FRCML and REC RDY) define various states for each segment. These states are shown below:

TLOOK STATES

ACK'ED	BRDY	STATE
0	1	Ready To Transmit (user set BRDY, cleared ACK'ED)
0	0	*Transmitted and Awaiting Acknowledge (WD2511A cleared BRDY)
1	0	Received Acknowledge (WD2511A set ACK'ED)
1	1	This state not allowed

*State 0-0 could also occur whenever there is no data ready to send.

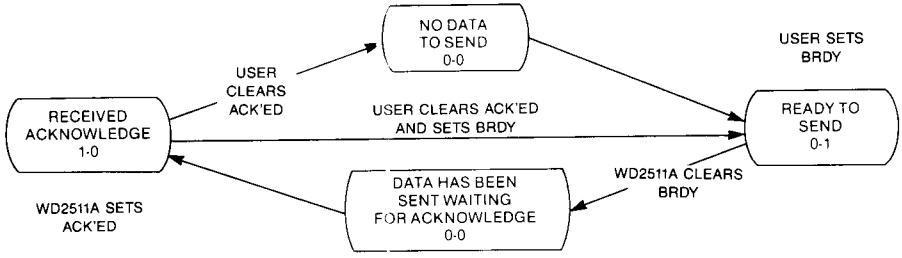


Figure 10. TLOOK SEGMENT STATE FLOW

Notice that in a TLOOK segment, the 0-0 state could have two meanings. Due to control internal to the WD2511A this will not pose an ambiquity to the WD2511A. However, if it is a difficulty to the user, the user could at start-up, set all ACK'ED bits. Since this would only be a start-up procedure, this would not violate the "deadly embrace" rule.

In the "WAITING FOR ACKNOWLEDGE" state, one or more re-transmissions could occur.

RLOOK STATES

BYTE # IN SEGMENT	BIT #							
	7	6	5	4	3	2	1	0
1	*FRCML	0	0	0	RES2	RES1	RES0	REC RDY
2	RSADR HI							
3	RSADR LO							
4	NOT USED				RCNT HI			
5	RCNT LO							
6	SPARE FOR USER DEFINITION							
7	SPARE							
8	SPARE							

*FRCML = Frame Complete

Figure 11. RLOOK SEGMENT

FRCML	REC RDY	STATE
0	1	Ready to Receive (user set REC RDY, cleared FRCML)
1	0	Received Packet (WD2511A set FRCML, cleared REC RDY)
0	0	Not Ready (user cleared FRCML)
1	1	This state not allowed

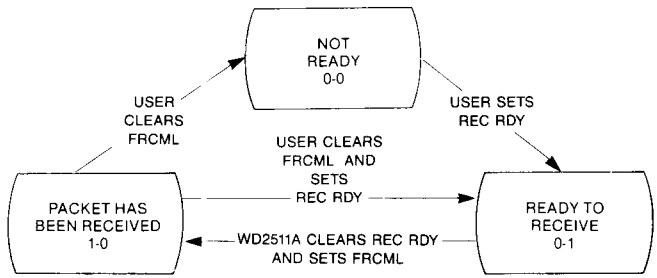


Figure 12. RLOOK SEGMENT STATE FLOW

4.3 "DEADLY EMBRACE" PREVENTION

A "deadly embrace" can occur when two processes reach a state where each is waiting for the other. In this case, the two processes are the user and the micro-controller inside the WD2511A. Therefore, to prevent "deadly embrace," the following rule is obeyed by the WD2511A and should also be obeyed by the user.

RULE: If a bit is set by the user, it will not be set by the WD2511A, and vice versa. If a bit is cleared by the WD2511A, it will not be cleared by the user, and vice versa.

This rule applies to TLOOK, RLOOK and to the I/O registers. This rule does not apply to the error counters. As an example, the BRDY bit in a TLOOK segment and the SEND bit in CRI are only set by the user and only cleared by the WD2511A.

4.4 SEND BIT CONTENTION

The WD2511A could be clearing the SEND bit when the Host is setting it. To insure that the bit is set the Host should read the status of the SEND bit after it is set. If the SEND bit is cleared, the Host should set it again.

REGISTER	CHAIN				BUFFER SIZE			
C	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0

4.5 BUFFER SIZE/CHAINING REGISTER

The lower 4 bits of Register C define the user's buffer size in multiples of 64 bytes, including a two byte transfer address (XFR ADR). If BUFFER SIZE is 0000, all buffers are 64 bytes long. For 0001, the size is 128, for 0010, the size is 192, and so on up to a maximum buffer size of 1024.

The maximum amount of I-field data bytes that can be contained in any buffer is the buffer size minus 2 bytes (XRF ADR) for all Transmit and Receive buffers, except for the last Receive buffer. For this buffer, the maximum amount of I-field bytes is the buffer size minus 3.

If you need to send and receive more bytes than your defined buffer size minus 3, you must select chaining. If you need to send and receive more than 1021 bytes of data (i.e. the maximum buffer size minus 3), you must select chaining.

The CHAIN value is not checked on transmit. The WD2511A uses the TCNT and buffer size as the user specifies it. It is up to the user not to set up a TCNT that exceeds the BUFFER/CHAIN Register definition.

The WD2511A includes a chained-buffer feature which allows the user more efficient use of memory, particularly in situations where the maximum packet size is much larger than the average packet size.

Chaining is restricted to transmit speeds less than or equal to 100 KBPS and to memory locations above 00FF hex.

Register C is used to program the chaining feature. The upper 4 bits, CHAIN, define the number of chain segments allowed in addition to the first segment. (If this feature is not used, make CHAIN all 0's).

For example, suppose that the buffer size defines a segment size of 128 and that CHAIN defines 8 additional segments in addition to the first. (Register C would be hex 81 in this example.) When 126 bytes of I-field data have been received, the WD2511A will read the next two buffer bytes as a transfer address (XFR ADR) pointing to another segment. At the end of that segment is another XFR ADR, and so on, up to a maximum of 9 total segments, (in this example.)

For the receiver, a XFR ADR of 00XX will mean that the next segment is not ready. If the WD2511A reaches a XFR ADR on the receiver with 00XX, there will be an Error Interrupt Code 42; otherwise there will be an Interrupt Code 41 which is a status indication that the WD2511A is going to the next segment. The upper 4 bits of I/O Register 6 gives a status of which chain segment is currently being used.

The transmitter chaining works like the receiver with the following exceptions:

1. XFR ADR = 00XX will not indicate next segment not ready.
2. There is no interrupt when going from one segment to another.
3. There is no status of the current segment being used.
4. Last chaining block is allowed to contain one more I-field data byte.

Total amount of I-field data bytes in receiver = (64 x (1 + BUFFER SIZE) −2) x (1 + CHAIN) − 1.

The total amount of I-field data bytes in transmitter = (64 x (1 + BUFFER SIZE) −2) x (1 + CHAIN).

Also, note that the transmitter and receiver counts are modified by 2 for each time a chain boundary is crossed. For example, if BUFFER SIZE = 0001 (segment size = 128 bytes including XFR ADR) and if an I-field of 270 bytes is to be transmitted, then there will be two times that a chain boundary is crossed. The TCNT must equal 274 to send the 270 bytes. The same is true for RCNT. Note that the largest block of data that can be received without chaining is 1021 bytes.

4.6 TLOOK AND RLOOK POINTERS

There are three 3-bit counters for the status of the segments in TLOOK and RLOOK. Status Register #0 (SRO) contains counters NA and NB which are used in conjunction with TLOOK. NB is the TLOOK segment number of the next block to be transmitted and is advanced at the end of each block transmission. NA is the value of the TLOOK segment of the next block to be acknowledged. If all transmitted blocks have been acknowledged, then NA = NB.

In SR1 is a 3-bit counter, NE, used in conjunction with RLOOK. NE is the value of the RLOOK segment number where the next received packet will be placed.

NA = Next to be Acknowledged

NB = Next Block to be Transmitted

NE = Next Expected to be Received

4.7 VARIABLE BIT LENGTH AND RESIDUAL BITS

The WD2511A will send only 8 bits per character and all transmitted frames will have an integral number of bytes.

The WD2511A can receive a packet with, or without, an integral number of bytes. The "RES" bits in the RLOOK tables indicate the number of received residual bits. The residual bits occupy the lower portion of the last received character.

RES 2	RES 1	RES 0	RECEIVED RESIDUAL BITS
0	0	0	0
0	0	1	7
0	1	0	6
0	1	1	5
1	0	0	4
1	0	1	3
1	1	0	2
1	1	1	1

4.8 ERROR COUNTERS

Following contiguously after RLOOK are six 8-bit error counters. The WD2511A will increment each counter at the occurrence of the defined event. However, the WD2511A will not increment past 255 (all 1's). The user has the responsibility of clearing each counter. The first counter past RLOOK is #1, etc.

ERROR COUNTER	TYPE OF ERROR
1	*Received Frames with FCS Error (includes frames ABORTed in the I-field).
2	Received Short Frames (less than 32 bits)
3	**Number of times T1 expired
4	Not used
5	*REJ Frames Received
6	REJ Frames Transmitted

*These counters are incremented only if the received A-field is equal to either Register E or F.

**Incremented only when attempting to transmit a command.

The Error Counters are accessed by the WD2511A transmitter DMA channel. Therefore, if multiple errors are received while the WD2511A is transmitting a long frame, only the last error will be counted. The only Counters which could miss counts because of this are Counters #1, #2, and #5. The error Counters are incremented only when the link is up (LINK = 0).

5.0 SELF-TESTS

There are two self-test features: 1) Internal RAM Register Test and 2) Loop-Back Test. Both tests are suitable for manufacturing testing, user incoming inspection testing, or system diagnostics and trouble-shooting.

5.1 INTERNAL RAM REGISTER TEST

There are eleven 8-bit registers internal to the WD2511A which are not directly accessible by the user's CPU. Seven of these registers can be tested by the Loop-Back Test. The internal RAM test provides a means to check the other four registers.

The contents of Register A are placed in two even internal registers and the contents of Register B in two odd internal registers. The four registers are then added together without carry and the result is placed in Registers 2, 5, 6 and 7. This test is initiated when RAMT (CR02) = 1. Use the following procedure:

1. Set-up Registers A and B.
2. Set RAMT.
3. Wait at least 50 times the CLK Period.
4. Read Registers 2, 5, 6 and 7.

To repeat the test for new values in Registers A and B:

5. Clear RAMT.
6. Wait at least 100 times the CLK period.
7. Go back to step 1.

Refer to Appendix E for more details.

5.2 LOOP-BACK TEST

The loop-back may be internal (CR03 = 1) or external (CR03 = 0). If external, RD and TD must be tied together either directly or remotely.

If CR03 = 1, TD is internally tied to RD and the RD signal (pin 16) is internally disconnected. Also, $\overline{TC}$ is internally tied to $\overline{RC}$ and the pin at $\overline{RC}$ (pin 17) is internally disconnected.

Early versions of the WD2511 required CTS to be connected externally to GND or to RTS. This is not true for the WD2511 "A".

CR07 and CR04 (active, no disc) should be set when running Loopback and Registers E and F must be identical.

Refer to Appendix E for more detailed information.

6.0 WD2511A ELECTRICAL SPECIFICATIONS:

Tables 11 and 12 list the DC Operating Characteristics and AC Timing Characteristics. Refer to Figure 13 for the Timing Diagrams.

ABSOLUTE MAXIMUM RATINGS:

Voltages referenced to V_{SS}

High Supply Voltage (V_{DD}) . . . . . −0.3V to +15V
Voltage at any Pin (except V_{DD}) . . . . . −0.3V to +7.0V
Operating Temperature Range . . . . . 0°C to 70°C*
Storage Temperature Range . . . . . −55°C to +125°C

Maximum limits indicate where permanent device damage occurs. Continuous operation at these limits is not intended and should be limited to those conditions specified in the DC Electrical characteristics.

*Contact WD for information regarding extended temperature screening.

Table 11. DC OPERATING CHARACTERISTICS

DC Operating Characteristics: V_{SS} = 0V, V_{CC} = 5.0V ± 0.25V, V_{DD} = 12.0V ± 0.6V, T_A = 0°C to 70°C

SYMBOL	PARAMETER	MIN	TYP	MAX	UNIT	CONDITIONS
I_{DD}	V_{DD} Supply Current		20	50	mA	
I_{CC}	V_{CC} Supply Current		200	280	mA	
V_{DD}	High Voltage Supply	11.4	12	12.6	V	
V_{CC}	Low Voltage Supply	4.75	5	5.25	V	
V_{IH}	Input High Voltage	2.0			V	
V_{IL}	Input Low Voltage			0.8	V	
V_{OH}	Output High Voltage	2.4			V	I_O = −0.1mA
V_{OL}	Output Low Voltage			0.4	V	I_O = 1.6mA
I_{LH}	Input Source Current			+10	μA	Vin = V_{CC}
I_{LL}	Input Sink Current			−10	μA	Vin = +0.4V
I_{OZH}	Output Leakage High (High Impedance)	−10		+10	μA	Vout = V_{CC}
I_{OZL}	Output Leakage Low (High Impedance)	−10		+10	μA	Vout = +0.4V

Table 12. AC TIMING CHARACTERISTICS

AC Timing Characteristics: V_{SS} = 0V, V_{CC} = 5V ± .25V, V_{DD} = + 12.0V ± .6V, T_A = 0° to + 70°C

SYMBOL	PARAMETER	MIN	TYP	MAX	UNIT	CONDITIONS
CLK	Clock Frequency	1.9	2.0	2.1	MHz	Note 1
RC	Receive Clock Frequency	0		1.1	MHz	Note 4
TC	Transmit Clock Frequency	0		1.1	MHz	Note 4
T_{RC}	Receive Clock Period			900	ns	
T_{TC}	Transmit Clock Period			900	ns	
T_{MR}	Master Reset Pulse Width ($\overline{MR}$)	1			ms	
T_{AR}	Input Address Set-up Time to $\overline{RE}$ Enable	0			ns	
T_{HA}	Input Address Hold Time after $\overline{RE}$ Disable	0			ns	
T_{RR}	$\overline{RE}$ PULSE Width	375			ns	Note 2, 3, 7
T_{RDD}	Data Valid Delay from $\overline{RE}$ Enable	2		375	ns	Note 2, 3, 5, 7
T_{HD}	Data Valid after $\overline{RE}$ Disable	20		100	ns	Note 2, 3, 5, 7
T_{AW}	Input Address Set-up Time to $\overline{WE}$ Enable	50			ns	
T_{AHW}	Input Address Hold Time after $\overline{WE}$ Disable	50			ns	
T_{WW}	$\overline{WE}$ Pulse Width	150			ns	Note 2, 3, 7
T_{DW}	Data Set-up Time to $\overline{WE}$ Disable or $\overline{DACK}$ Disable for DMA In	50			ns	
T_{DHW}	Data Hold Time after $\overline{WE}$ Disable or after $\overline{DACK}$ Disable for DMA In	100			ns	
T_{CSR}	Chip Select Set-up Time to $\overline{RE}$ Enable	0			ns	
T_{CHR}	Chip Select Hold Time after $\overline{RE}$ Disable	0			ns	
T_{CSW}	Chip Select Set-up Time to $\overline{WE}$ Enable	0			ns	
T_{CHW}	Chip Select Hold Time after $\overline{WE}$ Disable	0			ns	
T_{RRE}	$\overline{REPLY}$ Enable from $\overline{CS}$-$\overline{RE}$ Enable			325	ns	
T_{RRD}	$\overline{REPLY}$ Disable from $\overline{CS}$-$\overline{RE}$ Disable			325	ns	
T_{RWE}	$\overline{REPLY}$ Enable from $\overline{CS}$-$\overline{WE}$ Enable			200	ns	
T_{RWD}	$\overline{REPLY}$ Disable from $\overline{CS}$-$\overline{WE}$ Disable			225	ns	
T_{DKO}	$\overline{DACK}$ Pulse Width – DMA Out	375			ns	Note 2, 3, 6
T_{DKI}	$\overline{DACK}$ Pulse Width – DMA In	375			ns	Note 2, 3, 6, 9

Table 12. AC TIMING CHARACTERISTICS (continued)

AC Timing Characteristics: V_{SS} = 0V, V_{CC} = 5V ± .25V, V_{DD} = + 12.0V ± .6V, T_A = 0° to + 70°C

SYMBOL	PARAMETER	MIN	TYP	MAX	UNIT	CONDITIONS
T_{DA1}	Output Address Valid from $\overline{DRQO}$ (or $\overline{DRQI}$) Enable if ADRV = 1			80	ns	Note 5
T_{DA0}	Output Address Valid from $\overline{DACK}$ Enable if ADRV = 0			375	ns	Note 5, 6, 9
T_{DAH}	Output Address Valid after $\overline{DACK}$ Disable	20		125	ns	Note 5, 6
T_{CSH}	$\overline{CS}$-$\overline{WE}$ Disable or $\overline{CS}$-$\overline{RE}$ Disable Time (High)	500			ns	Note 2, 3, 7
T_{DCS}	$\overline{DACK}$ Enable after $\overline{CS}$-$\overline{WE}$ Disable or $\overline{CS}$-$\overline{RE}$ Disable	500			ns	Note 3
T_{DKH}	$\overline{DACK}$ Disable Time (High)	500			ns	Note 2, 3, 6
T_{CSD}	$\overline{CS}$-$\overline{WE}$ Enable or $\overline{CS}$-$\overline{RE}$ Enable after $\overline{DACK}$ Disable	500			ns	Note 2, 3, 6
T_{DKD}	Data Valid Delay from $\overline{DACK}$ Enable	2		375	ns	Note 5, 6
T_{DMW}	Data Valid after $\overline{DACK}$ Disable for DMA Out	20		125	ns	Note 5, 6
T_{DV}	Transmit Data Valid Delay			120	ns	
T_{SRD}	Receive Data Setup Time	0			ns	
T_{HRD}	Receive Data Hold Time	350			ns	
T_{DD}	$\overline{DRQO}$ or $\overline{DRQI}$ Disable after $\overline{DACK}$ Enable			375	ns	Note 5, 6, 9
T_{DQO}	$\overline{DACK}$ Enable after $\overline{DRQO}$ Enable	0		7	T_{RC}	Note 5, 6, 8
T_{DQI}	$\overline{DACK}$ Enable after $\overline{DRQI}$ Enable	0		8	T_{TC}	Note 5, 6, 8

NOTES

1. Clock must have 50% duty cycle ± 10%.
2. No user READ ($\overline{CS}$-$\overline{RE}$) or WRITE ($\overline{CS}$-$\overline{WE}$) is allowed within 500 nanoseconds after the trailing (rising) edge of $\overline{DACK}$.
3. The leading (falling) edge of $\overline{DACK}$ is not allowed within 500 nanoseconds after the completion of a user WRITE ($\overline{CS}$-$\overline{WE}$) or READ ($\overline{CS}$-$\overline{RE}$).
4. See "Ordering Information" for maximum serial data rates.
5. C(load) = 100pf.
6. There must be at least 500 ns between the trailing (rising) edge of $\overline{DACK}$ to the next leading (falling) edge of $\overline{DACK}$.
7. There must be at least a 500 ns time interval between any two or more consecutive user READs or WRITEs.
8. Maximum time for $\overline{DACK}$ to become active (i.e. leading (falling) edge of $\overline{DACK}$) after the WD2511A issued a DMA request ($\overline{DRQO}$ or $\overline{DRQI}$). Exceeding of this time interval will cause a Receiver Overrun ($\overline{DRQO}$) or a Transmitter Underrun ($\overline{DRQI}$) Error to occur.
9. These parameters can be screened for 250ns. Contact factory.

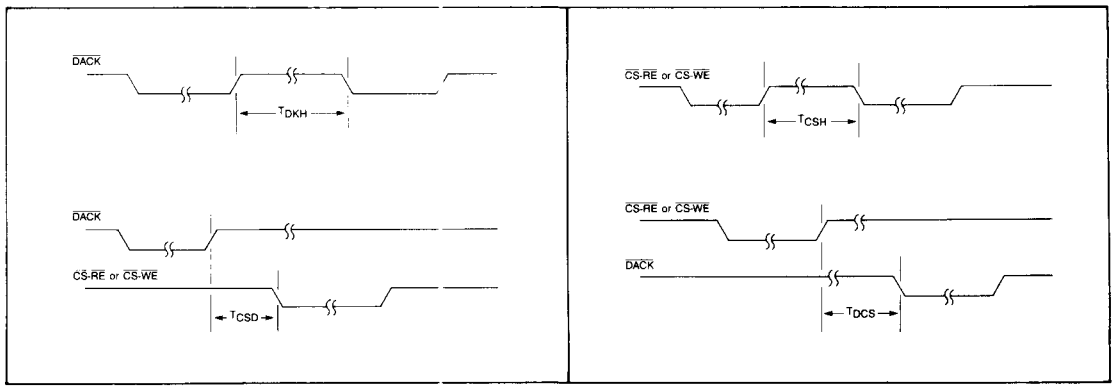
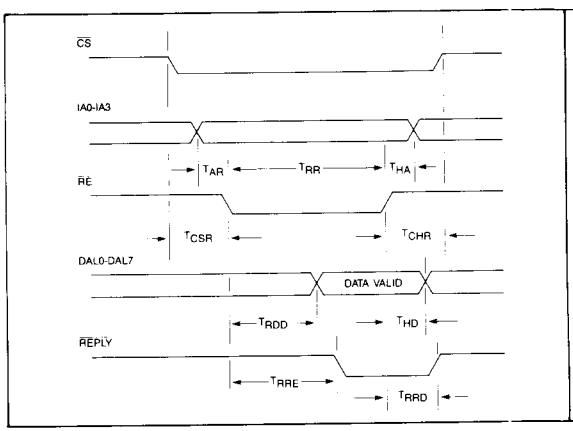
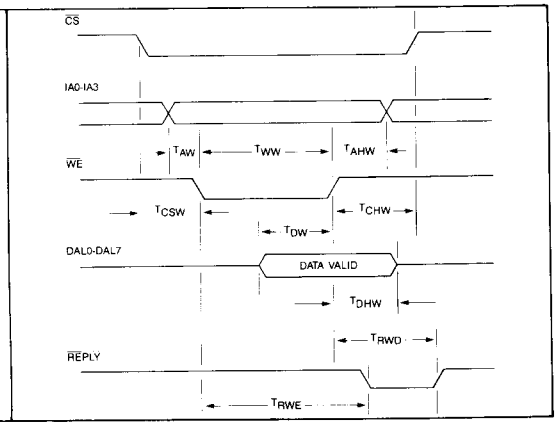


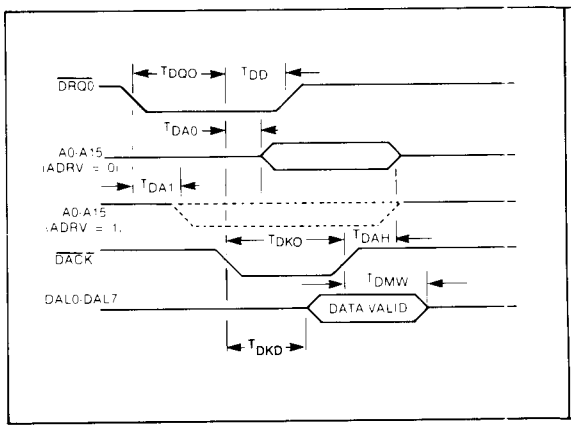
Figure 13. TIMING DIAGRAMS
a. INTER ACCESS TIMING



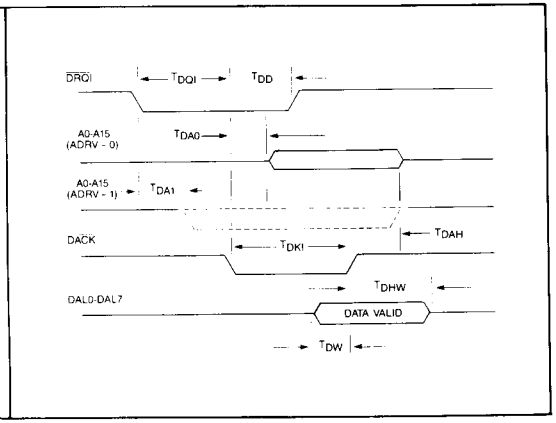
b. USER (CPU) READ



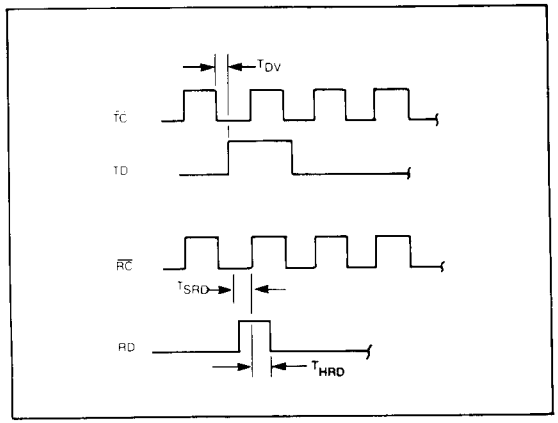
c. USER (CPU) WRITE



d. DMA OUT



e. DMA IN



f. TD-RD TIMING

7.0 ORDERING INFORMATION

ORDER NUMBER	MAXIMUM DATA RATE
WD2511A AN-01	100 Kbps
WD2511A AN-05	500 Kbps
WD2511A AN-11	1.1 Mbps

Contact the factory for information on other speeds or package types.

Figure 13. TIMING DIAGRAMS (continued)

APPENDIX A
TRANSPARENT MODES

The WD2511A is intended to be a link level controller meeting the requirements of X.25 LAPB. However, there has been an increasing demand from potential WD2511A users for additional frame types not included in the LAPB repertoire.

For example, the Bell System standard, BX.25, calls for the use of XID (exchange identification) in LAPB connections of DTE-to-DTE and in Dial access. (DTE-to-DTE and Dial access are not X.25 in the strictest sense.) Also, Western Digital has received several requests for the use of a SIM (Set Initialization Mode). Also, there has been one request to allow "unknown" frames to pass through the chip for the purpose of teleloading.

For this reason we have added two selectable modes to the WD2511A: transparent transmit and transparent receive. Basically, these two modes allow the user the option to pass certain non-LAPB frames thru the chip without the chip controlling these frames according to the LAPB protocol.

FEATURES OF THE TRANSPARENT MODES

- May transmit any A and C field under transparent control.
- May receive any U-frame not part of the LAPB repertoire when Tranparent-Receive enabled if address equals register E or F.
- Transparent modes are link state independent.

1.0 HOW TRANSPARENT MODES WORK

Two control bits activate this mode. TXMT (CR17) enables the Transparent Transmit and TRCV (CR16) enables the Transparent Receive.

1.1 TRANSPARENT TRANSMIT

When TXMT = 1, the WD2511A will transmit the frame in the next TLOOK segment provided SEND (CR10) = 1 and BRDY of that TLOOK segment is 1. The link may be either UP or DOWN. The WD2511A will not add the A and C fields to the Transparent Transmitted frame. The user must add these fields as the first two bytes in the transmit buffer. Thus the significance of the transmit count (TCNT) is different from normal packet transmission. In packet transmission, TCNT is the count of the I-field. In transparent transmission, TCNT is the I-field plus the A and C fields (I-field plus two bytes).

The timer, T1, will be disabled in transparent transmission. Therefore, if using this feature while the link is UP, it is advised that TXMT be set only when there are no outstanding (unacknowledged) packets (indicated whenever NA = NB.)

At the end of the transparent transmission, there will be an interrupt with XBA = 1. The SEND bit will be cleared but the BRDY will not be cleared. The NB pointer will not be incremented. To send another transparent frame, store its address in the same TLOOK segment and set SEND. To resume packet transmission, clear TXMT and set SEND. (Of course the TLOOK segment must be set-up prior to setting SEND.)

If SEND is set while the link is down, a transmission will occur even if TXMT = 0. Under this condition, a packet will be transmitted from current TLOOK SEGMENT, NB and V(S) will be incremented, and the chip will go on to the next TLOOK segment (up to a maximum of seven frames), just as if link were UP. However, the WD2511A will expect no acknowledgement to the packet(s). If the link is brought UP later, NB and V(S) are cleared to 0 at the time the link comes UP.

The bit $\overline{XI}$ (CR15) is used only when TXMT = 1. $\overline{XI}$ stands for Transmit I-field. If the frame contains three, or more bytes, not counting FCS, clear $\overline{XI}$. If the frame contains two bytes not counting FCS, set $\overline{XI}$. When $\overline{XI}$ = 1, only two frame bytes will be transmitted regardless of TCNT. DO NOT attempt to transmit a frame with TXMT = 1 and $\overline{XI}$ = 0 if TCNT is 2, 1, or 0.

1.2 TRANSPARENT RECEIVE

For the purposes of this discussion, it is necessary to define an "unknown frame" (that is, a frame which is "unknown" to the WD2511A).

Unknown Frame: A U-frame (unnumbered) frame which is not part of the LAPB repertoire. The U-frame repertoire in LAPB is SABM, DISC, DM, UA, and FRMR. For the purpose of this discussion, "UF" will refer to an unknown frame without an I-field, and "UFI" will refer to an unknown frame with an I-field.

A received SREJ (Selective REJect), which is an S-frame, is not considered an unknown frame by the WD2511A. If the link is DOWN and a SREJ command is received, a DM response will be sent or the frame will be ignored. If the link is down and a SREJ response is received, the SREJ is disregarded. If the link is UP and a SREJ command or response is received, a FRMR will be sent. The WD2511A will treat a received SREJ the same whether TRCV is 0 or 1.

A received packet (I-frame) response is not considered an unknown frame by the WD2511A. If the link is DOWN, the frame is disregarded. If the link is UP, a FRMR will be sent with W = 1 and X = 1. The received packet response is treated the same whether TRCV is 0 or 1.

Whether TRCV is 0 or 1, the WD2511A will check all received frames to insure that the A-field equals either Register E or F, that the FCS is correct, and that the frame contains 32 bits or more. If TRCV = 0, and if a UF or UFI is received, and if the link is UP, the WD2511A will send a FRMR with W = 1 (W and X are 1 in case of UFI). Refer to Appendix D.

When TRCV = 1, the WD2511A will be enabled to receive all frames. If the frame is "known" by the WD2511A, it will be treated according to the protocol just as if TRCV = 0. However, if the frame is a UF or UFI, it will be passed on to the user.

When an unknown frame is received while TRCV = 1, there will be an interrupt with ERROR = 1 and the Error

Register (ERO) will contain one of the following hexidecimal values:

ERO	FRAME RECEIVED
60	UFI Response
61	UFI Command
62	UF Response
63	UF Command

The C-field of the received frame is contained in Register #7. If the frame had an I-field, the frame will be placed in the next RLOOK segment and the value of RCNT will represent the count of bytes in the I-field (not including the A and C fields). The RLOOK pointer, NE, will be incremented. Therefore, the relationship between NE and V(R) will not be guaranteed if transparent receive is used while the link is UP. However, this will not cause a sequence problem in the protocol since the actual V(R) is maintained in an internal register in the WD2511A. Note that NE is cleared when the link is brought UP. Thus, if transparent receive is used only when the link is DOWN, then NE will be equal to V(R).

A word of caution. If the next RLOOK segment is not ready when a UFI is received, the Error Register (60 or 61) will be overwritten almost immediately with an error code 10 (RLNR) and the user will not know if the received UFI was a command or a response.

If RECR is set while the link is DOWN, the WD2511A will prepare to receive I-fields, whether TRCV is 0 or 1. If a packet command is received, there will be a PKR interrupt and the NE and V(R) will be incremented. Of course, no acknowledgement is sent and NE and V(R) are cleared once the link is brought up.

The following tables show what action the WD2511A will take when various frames are received.

TABLE A.1 PACKET RECEIVED (command, not response)

LINK	RLOOK READY	TRCV	ACTION BY WD2511A
DOWN	NO	0 or 1	DISREGARD
DOWN	YES	0 or 1	If N(S) = V(R), PKR interrupt, V(R) and NE incremented.
UP	NO	0 or 1	No ack transmitted. If N(S) not = V(R), DISREGARD.
UP	YES	0 or 1	If N(S) = V(R), RNR sent, else, REJ condition entered.
			If N(S) = V(R), PKR interrupt, V(R) and NE incremented. Acknowledgement sent at next opportunity. If N(S) not = V(R), enter REJ condition.

TABLE A.2 UFI RECEIVED

LINK	RLOOK READY	TRCV	ACTION BY WD2511A
DOWN	NO	0 or 1	DISREGARD
DOWN	YES	0	DISREGARD
DOWN	YES	1	Error interrupt 60 or 61. NE incremented.
UP	NO	0	FRMR sent. W = 1, X = 1
UP	NO	1	DISREGARD
UP	YES	0	FRMR sent. W = 1, X = 1
UP	YES	1	Error interrupt 60 or 61. NE incremented.

If TRVC = 1 and UF (no I-field) is received, there will be an Error interrupt 62 or 63, independent of the link state or the readiness of RLOOK.

Of course, the received C-field of any frame will be in Register 7 provided the A-field matched either Register E or F, the FCS was good, and the frame contained 32, or more, bits.

APPENDIX B

HALF DUPLEX OPTION (LAPX)

The WD2511A is basically a full duplex device. The receiver is maintained in an "always ready" condition even if the receive buffer is not ready. Thus, whether the received frame came from a full or half duplex system is of no consequence to the WD2511A.

Therefore, the half duplex option affects only the WD2511A transmitter. Half duplex is enabled when H/F (CRO5) = 1.

The WD2511A will transmit one frame at a time according to the following procedure:

A. Enable $\overline{\text{RTS}}$ ($\overline{\text{RTS}}$ goes low).

B. Wait for $\overline{\text{CTS}}$ ($\overline{\text{CTS}}$ input goes low).

C. Transmit frame (when $\overline{\text{CTS}}$ is active).

D. Remove $\overline{\text{RTS}}$ ($\overline{\text{RTS}}$ goes high 2 ½ bit-time after the last 0 of the trailing flag).

NOTES:

The leading flag will be transmitted somewhere between 5 and 13 bits after $\overline{\text{CTS}}$ goes low.

Interframe fill will be all 1's (IDLE): If no frames are received after T1 × N2 time, a link reset will occur due to RECIDL (SR25).

If T1 is internally activated, it is started when $\overline{\text{RTS}}$ goes low.

After $\overline{\text{RTS}}$ goes low, the frame will not begin transmission until $\overline{\text{CTS}}$ goes low. After the frame has started, the transmission of that frame is completed even if $\overline{\text{CTS}}$ returns high during the frame.

See section titled USE OF FLAGS BY THE WD2511A in Appendix D.

APPENDIX C

OPERATING THE WD2511A IN A MULTIPOINT CONFIGURATION

A typical multipoint consists of a primary station controller connected to one, or more, secondary controllers by means of a four wire connection as shown in Figure C.1. One wire pair carries serial data from the primary for broadcast to all secondaries. One wire pair carries serial data from all secondaries to be received by the primary. Thus, the primary communicates with secondaries, and secondaries never communicate directly with one another. Also, the primary can only "talk" to one secondary at a time.

ADCCP (Advanced Data Communication Control Procedures, ANSI X3.66) specifies three modes: Normal Response Mode (NRM), Asynchronous Response Mode (ARM), and Asynchronous Balanced Mode (ABM). NRM is strictly an association between a primary and one, or more, secondaries. Therefore NRM has been specifically designed for use with multipoint, and is the best choice for multipoint. ABM and ARM are not as well suited for multipoint as NRM, but can be used for multipoint operation.

The WD2511A is LAPB controller, which is an implementation of ABM. For multipoint, the primary will establish a link with one secondary and communicate I-frames with that secondary. After this "session," the primary will initiate link disconnect and go on to another secondary.

The control of the WD2511A is by means of Registers E and F which control the command/response definitions for the A-field. This may be handled in one of two ways: First, make Register E the same for all secondaries; this value will be used in Register F of the primary. Register F will be unique for each secondary, and the primary will modify Register E for each session with an individual secondary. Second, Register E and F could be equal in each secondary, but unique for that secondary. The primary would modify Registers E and F for each session with a given secondary. Neither of the above methods appears to have an advantage over the other.

For example, suppose the second method is chosen, and there are five secondaries. Register E and F for the five will be 1, 3, 5, 7, and 9. Each secondary will be an idle state with CR0 bits 5 and 1 set (hex 22: half-duplex). The $\overline{RTS}$ pin will be off for the secondaries. As soon as a secondary receives DISC with an A field which matches Register F, $\overline{RTS}$ will be asserted. If the A field does not match Register F, the frame is discarded, and $\overline{RTS}$ remains off.

When the primary needs to communicate to #1, Register E and F in the primary are set to 1. The primary will set CR0 bits 4 and 1 (hex 12). The primary will initiate a link set-up procedure with #1. After the link is brought up with #1, there will be a link-up interrupt with ER0 = hex 21 in both the primary and secondary. If, however, #1 is off-line, there will be no link-up time-out from the WD2511A for failure to bring-up a link. The user must provide this time-out.

After the link is up, if a WD2511A is being used, the user may wish to clear CR0 bit 5 to become full duplex when the bit is cleared.

To discontinue a session, the primary will set bit 0 in CR0 (hex 01). This will cause a DISC to be generated to the secondary. The user may change Registers E and F for the next secondary after waiting a few milliseconds (2 or 3) after the $\overline{LINK}$ bit becomes set.

When the secondary receives the DISC, the secondary will generate a disconnect interrupt (ER0 = hex 30) and transmit a UA.

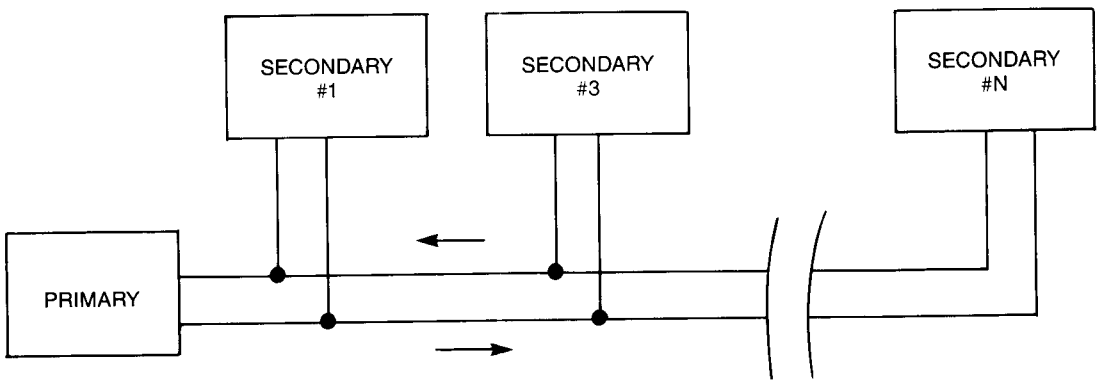


Figure C.1 TYPICAL MULTIPOINT CONFIGURATION

APPENDIX D
STATE DESCRIPTIONS
1.0 OVERVIEW OF WD2511A OPERATION

The following flow diagrams show a general overview of how the chip begins operation after MDIC is cleared and how it reacts to link resetting and disconnect conditions. These charts do not show the details of S-frame and I-frame management. These actions are shown in State Tables III-V. In full duplex implementations of LAPB, the device must account for independent TRANSMIT and RECEIVE State machines with signals between them. Therefore, boxes marked "SEND" indicate where the chip decides it must send a certain frame and sets a flag for the transmitter, not where the frame is actually transmitted.

State Tables I and II describe the action of the chip for certain frames received in the link up and link down states. They are referenced in the Flow Charts as I or II, where applicable. These diagrams are meant as a general overview of chip operation and not intended to reflect all of the internal processes.

The flow diagrams begin at START, after Master Reset. Initially the chip can only run the RAM test. To activate the protocol, the user programs CR0. The chip can be a passive device (CR04 = 0) where it will wait for a DM, DISC, or SABM from the remote device. If the chip is active, (CR04 = 1), it can be made to send a SABM (CR07 = 1) or a DISC (CR07 = 0) to the remote device. This is where STATE TABLE I begins. Note that once programmed for ACTIVE/PASSIVE, this cannot be changed without setting MDISC first or Master Reset.

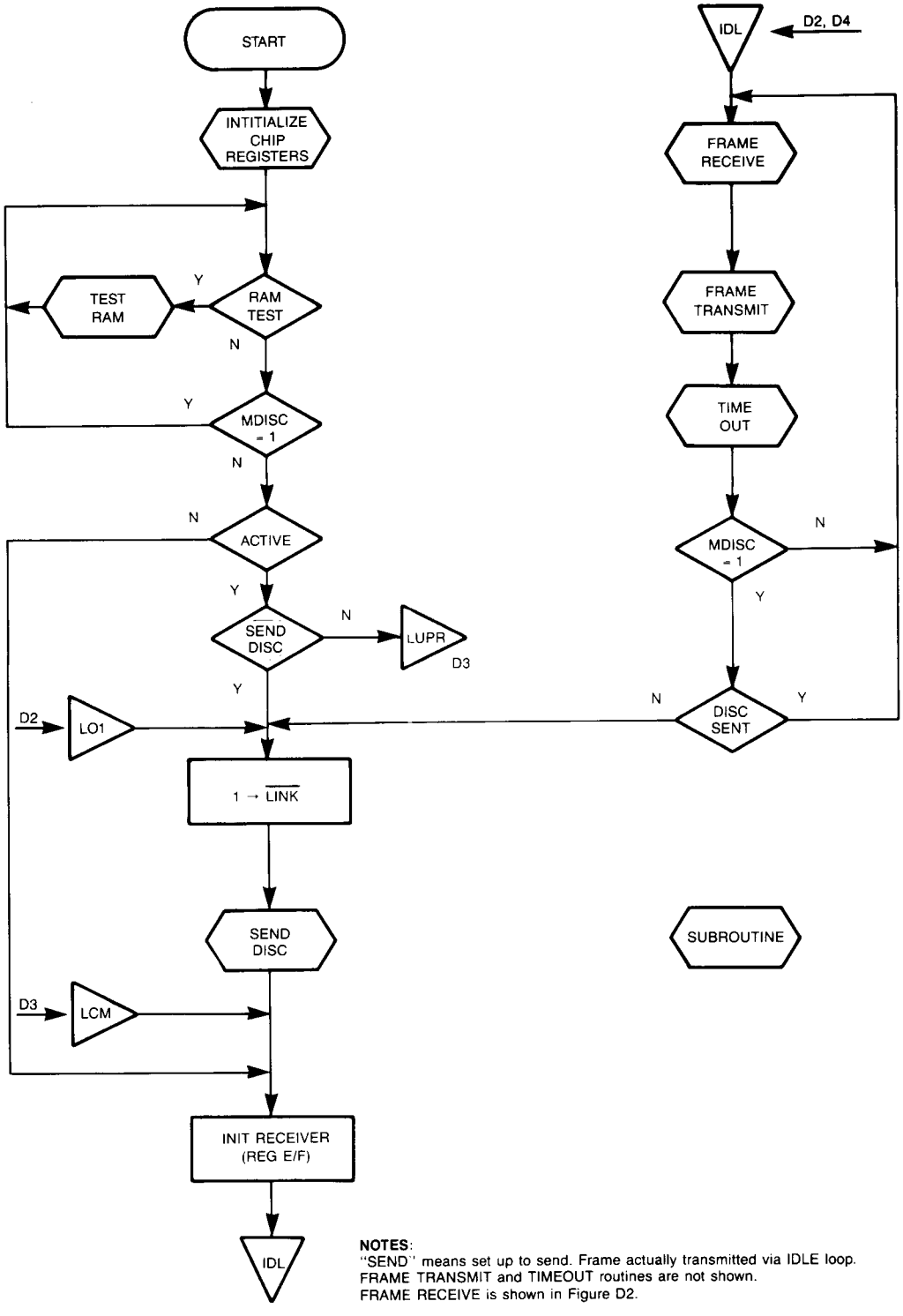
To bring down the link, the user sets MDISC (CR00 = 1). The chip will then send a DISC. The chip could also send a DISC if it has not received an answer to its SABM's (based on N2, registers 8/9). Either of these reasons moves the chip to STATE TABLE II.

After the link has been disconnected successfully, i.e. either:

a) Chip sends DISC, receives DM/UA, or

b) Chip gets DISC, transmits DM/UA,

if MDISC is clear (CR00 = 0) and the chip is in the ACTIVE mode (CR04 = 1), it will try to re-establish the link.



NOTES:
"SEND" means set up to send. Frame actually transmitted via IDLE loop.
FRAME TRANSMIT and TIMEOUT routines are not shown.
FRAME RECEIVE is shown in Figure D2.

Figure D.1

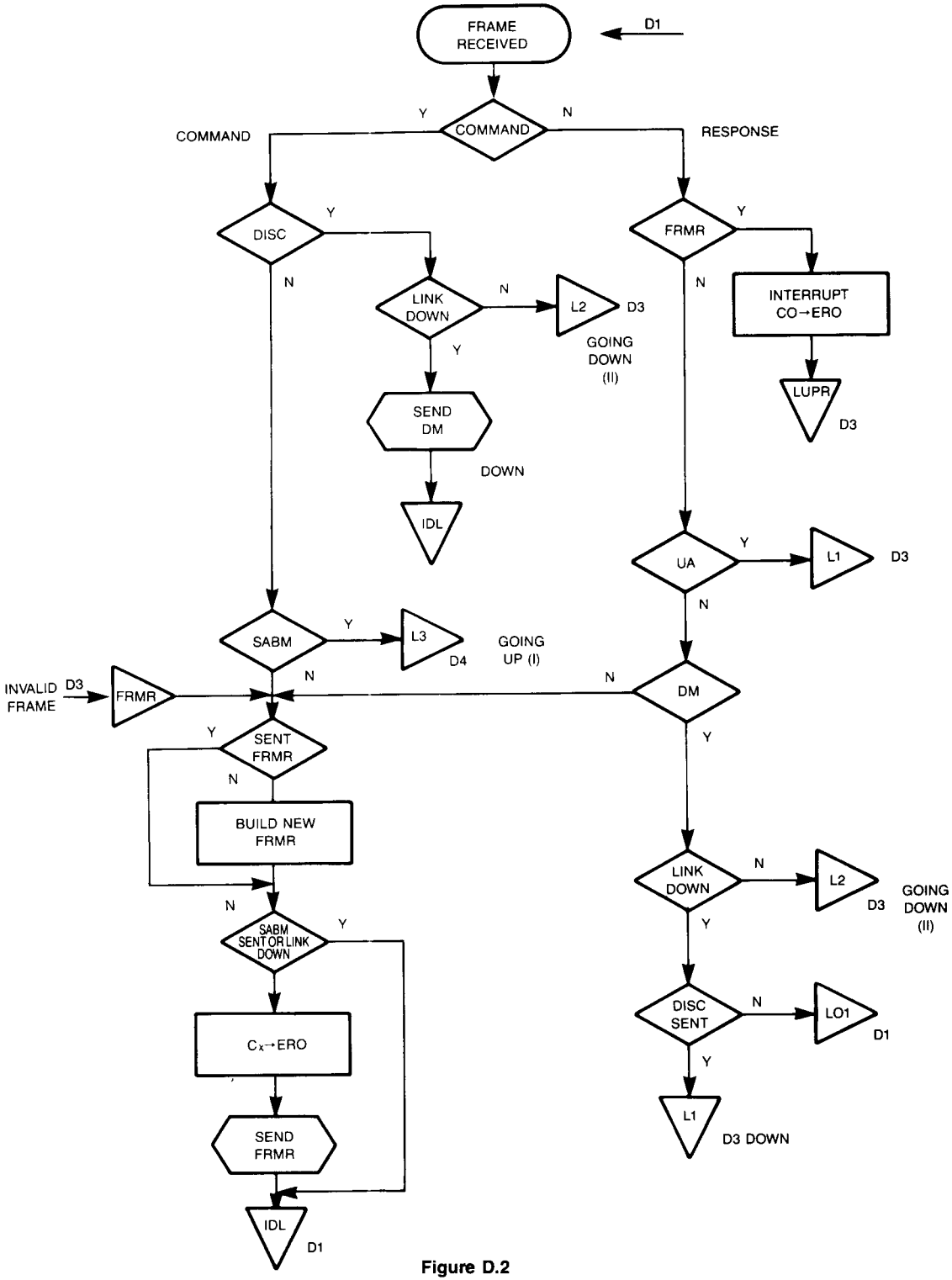

FRAME RECEIVED
D1
COMMAND
Y
N
RESPONSE
DISC
FRMR
LINK DOWN
L2
D3
GOING DOWN (II)
SEND DM
DOWN
IDL
INTERRUPT CO→ERO
LUPR
UA
L1
SABM
L3
D4
GOING UP (I)
INVALID FRAME
FRMR
DM
SENT FRMR
BUILD NEW FRMR
SABM SENT OR LINK DOWN
Cx→ERO
SEND FRMR
DISC SENT
LO1
D3 DOWN

Figure D.2

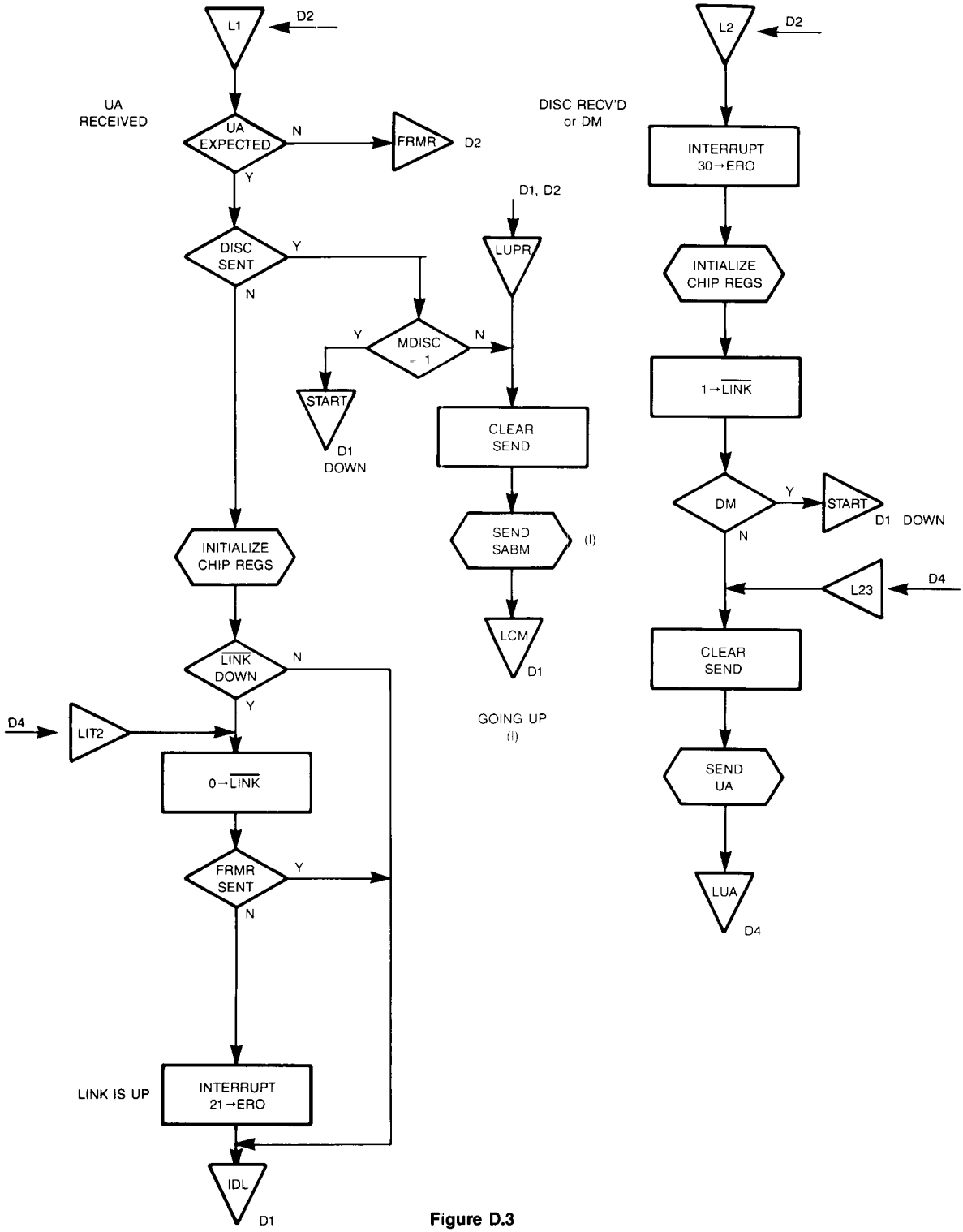

L1
D2
UA RECEIVED
UA EXPECTED
N
FRMR
D2
Y
DISC SENT
Y
N
D1, D2
LUPR
MDISC = 1
Y
N
START
D1 DOWN
CLEAR SEND
SEND SABM
(I)
LCM
D1
GOING UP
(I)
INITIALIZE CHIP REGS
LINK DOWN
N
Y
D4
LIT2
0→LINK
FRMR SENT
Y
N
LINK IS UP
INTERRUPT 21→ERO
IDL
D1
L2
D2
DISC RECV'D or DM
INTERRUPT 30→ERO
INTIALIZE CHIP REGS
1→LINK
DM
Y
START
D1 DOWN
N
L23
D4
CLEAR SEND
SEND UA
LUA
D4

Figure D.3

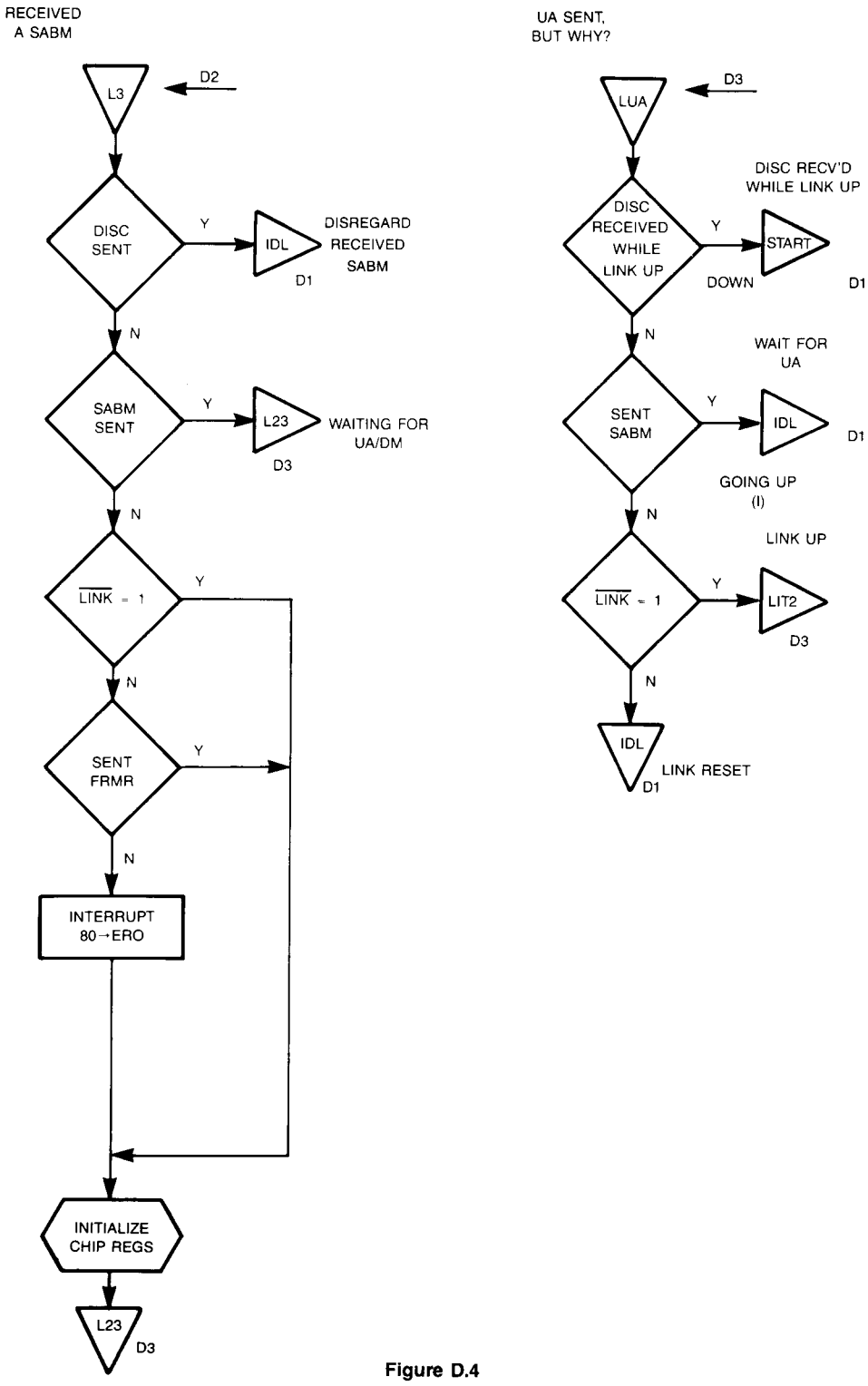

RECEIVED
A SABM
L3
D2
DISC
SENT
Y
IDL
D1
DISREGARD
RECEIVED
SABM
N
SABM
SENT
Y
L23
D3
WAITING FOR
UA/DM
N
LINK = 1
Y
N
SENT
FRMR
Y
N
INTERRUPT
80→ERO
INITIALIZE
CHIP REGS
L23
D3
UA SENT,
BUT WHY?
LUA
D3
DISC
RECEIVED
WHILE
LINK UP
Y
START
DOWN
D1
DISC RECV'D
WHILE LINK UP
N
SENT
SABM
Y
IDL
D1
WAIT FOR
UA
GOING UP
(I)
N
LINK = 1
Y
LIT2
D3
LINK UP
N
IDL
D1
LINK RESET

Figure D.4

2.0 STATE TABLE I LINK DOWN, BUT GOING UP

(Column 2 also applies to link reset)

ACTION BY WD2511A

STIMULUS:	COLUMN 1: DISC sent. Waiting for UA or DM.	COLUMN 2: SABM sent. Waiting for UA.
T1 runs out	Re-send DISC. P=1.	Re-send SABM. P=1.
T1 and N2 run out	Re-send DISC. P=1.	Send DISC, interrupt ER0 = 24. Go to column 1.
Received UA	Send SABM Go to column 2.	Clear NA, NB, NE, V(R), V(S). Go to IDL loop.
Received DISC	Send DM.	Send DM (NOTE 2)
Received SABM	Disregard.	Send UA. Clear NA, NB, NE, V(R), V(S). Keep waiting for UA.
Received DM	Send SABM. Go to column 2.	Send DISC. Go to column 1 (NOTE 3)
Received something (NOTE 1) other than UA, DM, DISC, or SABM	Disregard.	Disregard.

NOTES ON STATE TABLE 1

1. This applies to ADM, even if no DISC was sent, such as initially coming up in Passive Mode.
2. A UA will be sent if the chip sent the SABM in response to a FRMR.
3. If the chip sent SABM in response to a FRMR, a DM will cause the chip to begin as if from Master Reset.

3.0 STATE TABLE II LINK UP, BUT GOING DOWN

ACTION BY WD2511A

STIMULUS	DISC sent. Waiting for UA.
T1 runs out	Re-send DISC. P = 1.
Received UA or DM	Go to start
Received SABM	Disregard
Received DISC	Send DM. Wait for DM.
Received something other than DISC. SABM, UA, or DM	Disregard

4.0 USE OF FLAGS BY THE WD2511A

Once MDISC has been reset, the WD2511A will send interframe flags (hex 7E) if full duplex is selected (CRO5 = 0). If half duplex is selected, (CRO5 = 1), interframe fill will be all 1's (IDLE).

The WD2511A does not require the interframe time fill flags. Either idle or flags will be accepted. However, if the receiver detects idle for time T1 × N2, the WD2511A will send a DISC. If a UA is received, the link will be brought back up with a SABM from the chip.

When sending continuous flags, the WD2511A will send:

01111110011111100111111 0011.....

The WD2511A will accept either the above sequence as continuous flags, or the "shared zero" pattern:

01111110111111101111110111111.....

With one exception, the chip will not send a shared flag to terminate one frame and begin another but is capable of receiving this. If a frame is aborted, there may be only one flag after the aborted frame, if there is an I-frame pending transmission.

5.0 DEFINITIONS OF COMMAND AND RESPONSE

A transmitted or received command or response is a frame with the A-field defined below:

FRAME	A-FIELD
Transmitted Command	Register E
Received Command	Register F
Transmitted Response	Register F
Received Response	Register E

For non-transparent transmitted frames, only commands or responses are transmitted. A transparent transmitted frame (TXMT) may have any A-field the user chooses.

All received frames must be either commands or responses or the frame is disregarded ("thrown away"), even if transparent receive is enabled (TRCV = 1).

6.0 STATE TABLE III SENDING I-FRAMES (PACKETS) AND S-COMMANDS

NOTES:
In all subsequent pages, the link is considered UP ($\overline{LINK}$ = 0) unless otherwise stated. X = don't care. TXMT = 0 for Table III.

SEND	BRDY	NA AND NB	RNRR	T1 EXPIRES	RCVD REJ	ACTION BY WD2511A
1	0	X	0	No	No	Clear SEND (CR10)
1	1	X	0	No	No	Send next packet with N(S) = NB. After transmission complete, increment NB. Exception: if NB + 1 = NA, do not send next packet. There are 7 outstanding.
X	X	X	1	Yes	No	Send S-command, P = 1.
X	X	not =	X	Yes	No	Send S-command, P = 1.
X	X	not =	X	No	Yes	Make NA = received N(R). Start sequential retransmission of packets beginning with N(S) = NA. (Note 3)

NOTES ON STATE TABLE III

1. Received S-frames in the Table are assumed to have valid N(R)'s.
2. When an acknowledgement of one or more previously transmitted packets is received, NA is set equal to the received N(R). All TLOOK segments from the old value of NA up to N(R) -1 are acknowledged and the appropriate ACKED bits in the TLOOK segments will be set. After setting the ACKED bits, an XBA interrupt is generated.
3. Assuming appropriate TLOOK segments are ready, packets are transmitted sequentially without waiting for an acknowledgement, with three exceptions:
   a. There are already seven outstanding (unacknowledged) packets (NB + 1 = NA).
   b. The remote station has indicated a busy condition by sending an RNR frame (RNRR). T1 is started and an S-command will be transmitted with P = 1 when T1 expires.
   c. T1 expired and there are one or more outstanding packets. An S-command will be transmitted with P =1.
4. If an S-frame command with P = 1 is received, the WD2511A will transmit an S-frame response with F = 1 at the next opportunity.
5. If SEND is set and TXMT is set, a frame will be transmitted from the next TLOOK segment if BRDY is set. After transmission, SEND is cleared by the WD2511A.

7.0 RECEIVING AND TRANSMITTING A NULL PACKET

If an error-free (FCS good) packet is received with a correct N(S) but has no I-field, that packet will be treated the same as a packet with an I-field. The fact that there was no I-field is shown by RCNT equal to 0.

The WD2511A will not transmit a null packet. TCNT must not be allowed to be all 0's.

8.0 SENDING A REJ (RESPONSE)

1. The REJ condition is entered any time an error-free packet is received with an out-of-sequence N(S).
2. When the REJ condition is entered, the REJ frame with N(R) = V(R) is transmitted immediately if a packet is not being transmitted, or, at the completion of the current packet. There are two exceptions, as noted in 3 and 4 below.
3. If a link resetting SABM needs to be transmitted, the SABM is sent. When the UA is received for the SABM, the REJ condition is cleared.
4. If the receiver is not ready (RNRX = 1), the REJ is not sent.
5. Once the REJ condition is entered, only one REJ will be transmitted. Another REJ is not transmitted unless the REJ condition is cleared and re-entered. The REJ condition is cleared if a packet is received with correct N(S), if a SABM is received, or if a SABM is transmitted and a UA received.
6. When the REJ is transmitted, error counter #6 is incremented.

9.0 RECEIVING A REJ (RESPONSE OR COMMAND)

If a REJ is received error-free, then:

1. If the N(R) is not valid, an interrupt is generated with ERO = C8 and a FRMR is transmitted.
2. If the N(R) is valid and greater than NA, at least one transmitted packet is acknowledged. The appropriate ACKED bits in TLOOK are set and an XBA interrupt is generated.
3. If the N(R) is valid and less than NB, the WD2511A will begin sequential retransmission starting with V(S) = received N(R). If a packet is being transmitted when the REJ is received, that packet is aborted. If the N(R) is valid and equal to NB and a packet is being transmitted, that packet (which will be #NB) is aborted and retransmission will begin.
4. If the N(R) is valid and equal to NB and there is no packet being transmitted, there is no retransmission

initiated. In this case the REJ has the same effect as an RR.

5. If in 2, 3, or 4 above, the received REJ with P = 1 is a command, the WD2511A will transmit a RR or RNR response with F = 1 at the next respond opportunity.

10.0 RECEIVING A SREJ (RESPONSE OR COMMAND)

The SREJ (Selective Reject) is an unimplemented frame in X.25, and, as such, results in FRMR being transmitted if received by the WD2511A. If the N(R) field in the SREJ is valid, a FRMR + W is sent with Vr incremented to reflect the Nr in the SREJ. If the Nr in the SREJ is invalid, a FRMR + Z is sent to the remote. In both cases an Error interrupt is generated for the user and the remote will send a link resetting SABM.

11.0 STATE TABLE IV LOCAL STATION BUSY (SENT RNR: RNRX = 1)

$\overline{\text{LINK}}$	RECR	REC RDY	ACTION
1	X	X	No S-frame transmitted when link down.
1 → 0	1	1	RLOOK ready. No RNR frame sent.
1 → 0	1	0	RNR response sent immediately after link Up. RNRS set. RLNR interrupt.
1 → 0	0	X	RNR response sent immediately after link UP. RNRX set. No RLNR interrupt.
0	1	1	Receiver ready to accept packets.
0	1 → 0	1	Receiver ready to accept packets.
0	1	0	RNR response sent. RNRX set. RLNR interrupt.
0	1 → 0	1	If RNRX was set, then RNRX will be cleared after the next received packet or S-COMMAND. After that, an RR or REJ response is sent. RR response is not automatically sent upon setting of RECR.
0	0	0	RNR response sent. RNRX set. There is no RLNR interrupt.

NOTES ON STATE TABLE IV

1. The arrow → indicate a change in state from the value on the left to the value on the right. The user should not clear RECR, once set.
2. The RNRX status bit is set at the time the receiver-not-ready condition was established. The RNR frame will be sent immediately if no packet is being sent or after the end of the current packet.
3. When a received packet is brought into memory with RNRX = 0, the packet will be accepted provided the FCS and N(S) are correct and the I-field is not too long. The N(R) may or may not be correct but is checked separately. If N(R) is not valid, a FRMR is transmitted.
4. Whenever RNRX = 1, the I-field of a received frame is not brought into memory. For received packets, the N(S) and N(R) are checked as usual. If the N(S) is out-of-sequence, the REJ will not be transmitted.
5. If a link resetting SABM is transmitted when RNRX = 1, RNRX will be cleared when the UA is received. If the condition which caused receiver-not-ready still exists, an RNR is sent and RNRX is set. However, if the receiver instead is ready, I-field data may be brought into memory. The same also applies when a link resetting SABM is received.

12.0 STATE TABLE V REMOTE STATION BUSY (RECEIVED RNR: RNRR = 1)

SEND	NA AND NB	RECVD ACK	RECVD RNR	RECVD RR, REJ OR UA	T1 EXPIRES	ACTION
X	not equal	Yes	Yes	No	No	Set RNRR. Restart T1 and N2. Update NA.
0	equal	No	Yes	No	No	Set RNRR. Start T1.
X	not equal	No	No	No	Yes	Send S-Command (P = 1). If RNR subsequently received, restart T1 and N2.
X	not equal	Yes	No	Yes, but not UA	No	Clear RNRR. Restart T1 and N2. Update NA.
X	X	X	No	Yes	No	Clear RNRR.
0 → 1	equal	No	No	No	No	Send next packet. Increment NB after transmission. (Then, NB does not = NA). Start T1 and N2.

NOTES ON TABLE V

1. If SEND = 1, it is assumed for this table that BRDY of the next TLOOK segment is set.
2. IF RNRR = 1, an RR or RNR command is transmitted at T1 intervals.

13.0 DEFINITION OF VALID RECEIVED N(R)

Reference

CCITT Recommendation X.25 paragraphs 2.4.10 and 2.3.4.10.

Definition

A valid received N(R) is greater than or equal to NA, and less than or equal to NB.

1. The "greater than" and "less than" relationships must be understood in a circular sense. 0 could be greater than 7 depending on the values of NA and NB.
2. If NA = NB, there is only one possible valid received N(R): N(R) = NA.
3. If NB + 1 = NA, there are seven outstanding packets and any received N(R) will be valid.
   N(R) =NB ACK'S all of the out-standing frames,
   N(R) =NA ACK'S none of them, and an
   N(R) in between ACK's some of the packets.
4. Basically, a received N(R) which is not valid is one which acknowledges a packet, or packets, not yet transmitted.

14.0 SENDING S-FRAME COMMANDS

When an S-frame command is to be transmitted, a RR command is transmitted if RNRX = 0 or a RNR command is transmitted if RNRX = 1. If RNRX = 0 and a REJ is waiting to be transmitted, a REJ command is transmitted.

For all transmitted S-commands, the P bit is set to 1.

An S-command will be transmitted at T1 intervals if an RNR is received (RNRR = 1) or if T1 has expired due to waiting for an acknowledgement to previously transmitted packets.

15.0 CONDITIONS FOR SENDING SABM (LINK RESET)

1. FRMR received.
2. Have sent an S-command N2–1 times with P = 1 (at T1 intervals) without receiving an S-response with F =1.

UNSOLICITED UA OR UNSOLICITED F BIT

If an unsolicited UA or an unsolicited F bit is received with the link up, a FRMR will be transmitted with W = 1.

SENDING AN FRMR

AN FRMR may be transmitted for any of the reasons indicated in X.25 (W,X,Y,Z). An FRMR is transmitted only if the link is up.

Upon sending a FRMR, the WD2511A will not send a packet until the FRMR condition is cleared. The WD2511A will also discard any received I-fields. The FRMR condition is cleared when either a SABM or DISC is received.

While in the FRMR condition, the WD2511A will act as shown below:

FRAME RECEIVED	ACTION BY WD2511A
SABM	Send UA. Clear FRMR condition. Enter information transfer phase.
DISC	Send UA. Clear FRMR Condition. Enter logical disconnect state.
Packet with good N(R)	Retransmit FRMR
S-FRAME with good N(R) (Command or response)	Retransmit FRMR
Packet or S-frame with bad N(R)	Retransmit FRMR
Any frame with violation W,X,Y,Z	Retransmit FRMR
1st FRMR	Send SABM
Subsequent FRMR	ignore, let T1 expire

RECEIVING AN FRMR

After a FRMR has been received:

1. The FRMR I-field will be in the memory referenced by the current NE segment, provided the receiver was ready. RECRDY is not cleared.
2. The SEND bit is cleared.
3. No more I-field data is allowed to come into memory until the user makes the receiver memory ready.
4. A link resetting SABM is transmitted and an error interrupt, ERO = C0 is generated.
5. After the UA is received for the SABM, the NA, NB, NE, V(R), and V(S) are cleared to 0.

16.0 PROTOCOL SIGNIFICANCE OF TLOOK/RLOOK POINTERS

The NE, NA, and NB pointers have a relationship with the sequence counters used in the LAPB protocol.

The RLOOK pointer NE is equal to V(R) at all times if TRCV = 0. However TRCV = 1 and the link is UP, there is no guaranteed relationship between NE and V(R).

TLOOK pointer NB is the Next Blook to be transmitted. If the chip is not in packet retransmission, NB is equal to the V(S) of the next packet to be transmitted.

TLOOK pointer NA is the Next packet to be Acknowledged. It represents the V(S) number of the oldest packet in the retransmission buffer.

APPENDIX E
SOFTWARE APPLICATION NOTES
1.0 PROGRAMMING

Initialization of the WD2511A and I-field data processing (level 3) is accomplished by user software. The real time requirements of a driver for the WD2511A are less than for other drivers because the WD2511A responds to link exceptions and overhead functions on its own.

Configuring the WD2511A for certain test functions, modes, timer values, location of initial memory pointers, chain buffer lengths and link level addresses is performed via the sixteen I/O registers.

All buffer management support, buffer chaining and free/busy flags occur in user memory. Two look-up tables (TLOOK/RLOOK), located in the user memory, contain pointers/counters for up to eight outstanding transmit/receive packets. The WD2511A contains only one address pointer: the starting address of Segment #0 in the TLOOK table. Segment #0 in the RLOOK table always begins 40(Hex) bytes after TLOOK, Segment #0, byte #0. See section "Memory Access Method" in the WD2511A specifications, section 4.

Link monitoring is done by use of the I/O registers and the memory buffers. The WD2511A indicates to the system CPU that a certain event has occurred by setting a bit in status register 1 and setting the interrupt flag. This indicates whether a packet has been received a transmitted packet has been acknowledged, a non-recoverable error condition or some other condition needs the attention of the CPU.

The sample driver PDL (program design language) included in Figures E.1 through E.6 is an example of how a general purpose WD2511A driver might be constructed. It is published in PDL, rather than a real programming language, in order to address the largest possible audience of programmers.

The driver is constructed with 3 major routines:

1. Initialization – to initialize the WD2511A, set up registers, and RLOOK segments, and to bring up a link.
2. Transmit a frame – to send a frame to WD2511A.
3. Interrupt handler – to handle interrupts generated from WD2511A.

This is just a simplified example for the WD2511A driver; all error detection and recovery logic is omitted. However, notes are added appropriate for readers who need to develop a real WD2511A driver. A more detailed PDL for a WD2511A driver to run on IBM PC is included in Section 4 for reference.

```
Initialization
    Reset WD2511A
    Set Register 8 (T1)
    Set Register 9 (N2)
    Set Register A (TLOOK HI)
    Set Register B (TLOOK LO)
    IF to operate as DTE
        Set Register E (COMMAND) = 1
        Set Register F (RESPONSE) = 3
    ELSE to Operate as DCE
        Set Register E (COMMAND) = 3
        Set Register F (RESPONSE) = 1
    ENDIF
    Set Register C (CHAIN/BUFFER SIZE)
    Initialize Buffer Pools
    DO UNTIL Window Full
        Get Buffers from Pool (Receive Buffer)
        IF (CHAIN)
          Set XFR ADR in each Buffer
        ENDIF
        Set RLOOK parameters for this
          segment (RSADR, RCNT, RECRDY)
    ENDDO
    Set Register 0 (CRO)
    ACTIVE and RECR
    (to bring up link)
```

Figure E.1 INITIALIZATION

```
Transmit a Frame
    IF outstanding LE. Window
        Get Buffers from Pool (transmit buffer)
        IF (CHAIN)
          Set XFR ADR in each Buffer
        ENDIF
        Copy User Data to Transmit Buffer
        Set Next Available TLOOK Segment
        Parameters (TSADR, TCNT, BRDY)
        Increment Outstanding
        Set SEND in Register 1 (CRI)
    ENDIF
```

Figure E.2 TRANSMIT A FRAME

NOTE:
When presenting packets to the chip for transmission, the host should implement a Level 3 timer. The value of the timer is system dependent and varies with packet size and line speed but should be in the order of seconds. If a packet has not been acknowledged by the time the timer expires, the host should check the SEND bit. If it is reset, set it to 1 again and restart the timer. If it was still set, the link must be reset. Do this by setting the MDISC bit (CR00 = 1), waiting for the link to go down ($\overline{LINK}$ = 1), then resetting MDISC (CR00 = 0), and waiting for the link to come back up (ERO = 21 or, if RLOOK0 was not ready, 10 and $\overline{LINK}$ = 0). The reason for this timer exists in the X.25 protocol itself. There are certain conditions where the link layer is operating according to protocol but transmission of user packets is blocked. For example, if the remote station is busy and sends an RNR, the WD2511A will not be able to successfully transmit user packets to it, as long as this condition persists. The WD2511A will poll for the remote status at T1 intervals to insure that leaving the busy state is detected. However, as long as the remote responds RNR to the poll, no more user data will be sent.

```
Interrupt Handler
    Read Register 3 (SRI)
    IF PKR
        Process Received Data
    ENDIF
    IF XBA
        Process Acknowledgement
    ENDIF
    IF ERROR
        Process Error
    ENDIF
```

Figure E.3 INTERRUPT HANDLER

```
Process Received Data
    DO WHILE RLOOK Segment FRMCL
        Remove Buffer from Segment to
          Receive Queue
        Indicate Data to User (to remove data
          from receive queue)
        Get Buffers from Pool (receive buffer)
        IF (CHAIN)
          Set XFR ADR in each Buffer
        ENDIF
          Set Next Unused RLOOK Segment
          (RSADR, RCNT, BRDY)
    ENDDO
```

Figure E.4 PROCESS RECEIVED DATA

NOTE:

USE OF THE RECR BIT

The RECR (CR01) bit should be understood as an instruction to the WD2511A to enable the receiver function. The WD2511A will test RECR as soon as MDISC is cleared and will retest RECR after each link set-up and each link reset. Once the receiver is ready, the WD2511A will not test RECR again unless there is a link set-up, a link reset, or a receiver not ready condition. Once the RECR bit has been set by the user, it should not be cleared by the user.

The receiver-not-ready condition is indicated by RNRX = 1. This condition is cleared after the user sets RECR and RECRDY (in the next RLOOK) and after either a packet or an S-frame is received from the remote station.

If RECRDY of the next RLOOK is 0 and RECR = 0, there will not be an RLNR interrupt, but RNRX will be set. If RECR = 1 but the RECRDY bit of the next RLOOK segment is 0, there will be an RLNR interrupt (error code 10) and RNRX will be set.

```
Process Acknowledgement
    DO WHILE (outstanding GT.0) and (TLOOK
        Segment ACKED)
        Confirm to User
          (data successfully transmitted)
        Release Buffers in this TLOOK Segment
          to Pool
        Decrement Outstanding
        TLOOK Segment = Next TLOOK Segment
    ENDDO
```

Figure E.5 PROCESS ACKNOWLEDGEMENT

```
Process Error
    Read Register 5 (ER0)
    IF (ER0) .EQ. 0 X 21
        Indicate LINKUP
    ELSE IF [(ER0) .EQ. 0 X 22]
        OR [(ER0) .EQ. 0 X 24]
        OR [(ER0) .EQ 0 X 30]
        Indicate LINKDOWN
    ELSE
        Indicate LINK RESET
    ENDIF
```

Figure E.6 PROCESS ERROR

NOTE:

The host should keep its own set of variables to determine the index of the Next Packet to be Received and the Next Packet to be Acknowledged because if a Link Reset occurs, the chip resets its NA, NB, and NE counters. After a Link Reset the host should look for unprocessed received packets (FRCML = 1) in the RLOOK table beginning at its Next Packet to be Received segment and proceeding in order until it finds FRMCML = 0. Furthermore, if RLOOK0 has RECRDY = 1 and RECR is set to 1, a packet can be stored into RLOOK0 immediately after a Link Reset. Therefore, the host should also look for received packets beginning at RLOOK0 after a Link Reset.

The chip resets the SEND bit after a Link Reset so no new TLOOK buffers will be sent until the host sets SEND again. After a Link Reset the host should look for any unprocessed acknowledged packets (ACKED = 1) in the TLOOK table beginning at its Next Packet to be Acknowledged segment and proceeding in order until it finds a segment with ACKED = 0. Then the host must set up the TLOOK segments again so that the oldest unacknowledged packet is in TLOOK0, the next in TLOOK1, and so on, setting the BRDY = 1 in each occupied segment. (New packets may be added to the TLOOK at the next available segment.) When the host has finished setting up the TLOOK segments, it should set the SEND bit to 1. At this point packet transmission will resume if the remote station is up and is not in a receiver not ready (RNRR = 1) condition.

2.0 INTERNAL LOOP-BACK TEST

The loop-back test feature is an internal programmable loop-back of data, enabling the user to make an almost complete test of the WD2511A. It allows diagnostic testing of the WD2511A and the interfacing circuitry. In this mode, transmitted data to the TD pin is internally routed to the received data input circuitry, thus allowing this WD2511A to set-up a link, send a number of packets to itself and then reset the link.

The $\overline{RC}$ clock is internally connected to $\overline{TC}$ clock and the $\overline{CTS}$ input is internally connected to the $\overline{RTS}$ output.

The loop-back test allows the verifying of proper operation of practically all the various functions of the WD2511A. The features tested here, the addresses and values of the variables chosen are only used as examples and are as follows:

TLOOK segments starting address = 0800H
Transmit Data buffer #0 starting address = 1000H
Received Data buffer #0 starting address = 1800H
Number of packets transferred = 1
Number of I-field bytes per packet = 1024
Number of residual bits = 0
T1 = 101H
N2 = 20H

Chaining is used in this example. The 1024 bytes are divided into 256 byte chain segments. Five segments are needed for this operation with 254 bytes of I-field data and two XFR ADR bytes per segment in the first four chain-segments. The rest of the I-field data (8 bytes) are located in the fifth chain-segment.

Programming:

CHAIN = 4 = number of CHAIN segments – 1
LIMIT = 3 = (number of bytes per segment divided by 64) – 1

For buffer management programming, see memory access scheme, Figure E.7.

XMIT Command Address and XMIT Response Address (REG E and F) must be the same value.

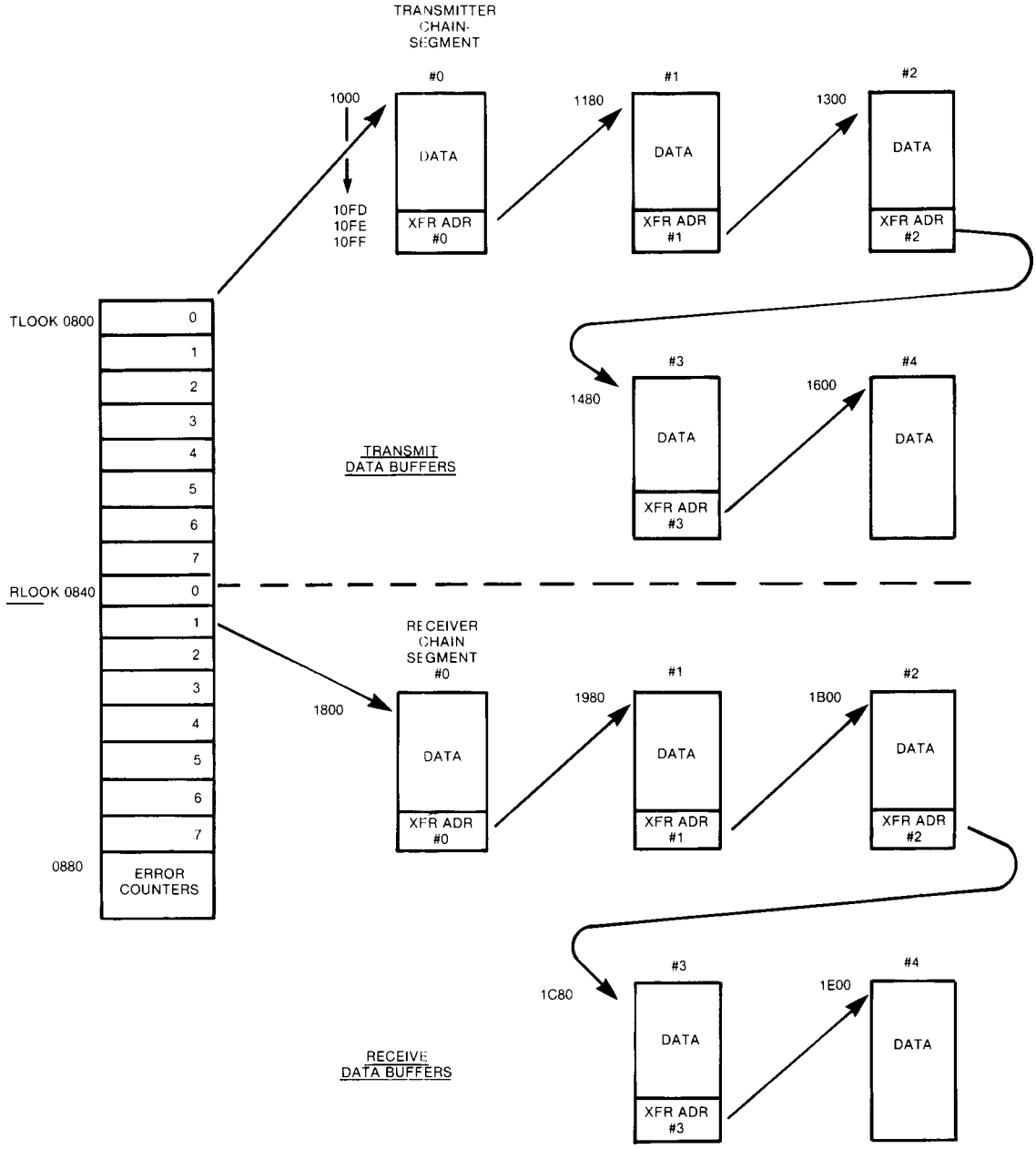


Figure E.7 MEMORY ACCESS METHOD FOR LOOP-BACK TEST

```
Loop-Back Test
    Reset WD2511A
    Set Registers T1, N2, TLOOK HI, TLOOK
      LO, CHAIN/BUFFER SIZE
    Set Register E (COMMAND) = 1
    Set Register F (RESPONSE) = 1
    Allocate Transmit Buffers
    Set up all 8 TLOOK Segments Parameters
      (TSADR, TCNT, BRDY)
    Allocate Receive Buffers
    Set up all 8 RLOOK Segments Parameters
      (RSADR, RCNT, RECRDY)
    Set Register 0 (CR0) to bring up link
      (LOOP TEST, RECR, ACTIVE, NO.ADISC)
    Set Register 1 (CR1) to Start Transmission
      (SEND, ADRV)
    Delay
    DO FOR i = 1 to 8
        Compare Data in TLOOK Segment [ i ]
        and
          Data in RLOOK Segment [ i ]
        IF .NE.
          Test Failed
          UNDO
        ENDIF
    ENDDO
    Reset WD2511A
```

Figure E.8 LOOP BACK-TEST

3.0 INTERNAL RAM REGISTER TEST

There are eleven 8-bit registers internal to the WD2511A which are not directly accessible by the user's CPU. Seven of these registers can be tested by the Loop-Back Test. This test provides the means to check the other four registers.

The contents of Register A are placed in two even internal registers, and the contents of Register B in two odd internal registers. The four registers are then added together without carry and the result is placed in Registers 2, 5, 6 and 7. This test is initiated when RAMT (CR02) = 1. Use the following procedure:

1. Set-up Registers A and B.
2. Set RAMT.
3. Wait at least 50 times the CLK Period.
4. Read Registers 2, 5, 6 and 7.

To repeat the test for new values in Registers A and B:

5. Clear RAMT.
6. Wait at least 100 times the CLK period.
7. Go back to step 1.

The following is the PDL to outline the software design for internal RAM Register Test.

```
Internal RAM Register Test
    Reset WD2511A
    Test-failed = FALSE
    DO FOR i = 1 to 2
        UNDO IF Test-failed
        Select Register A and Register B Values
        Calculate Expected Result
          ([Reg A]) + [Reg B]) * 2
        Set MDISC and RAMT in Register 0 (CR0)
        Delay 50 Clock Ticks
        Read Registers 2,5,6 and 7
        IF [Reg 2] .NE. Expected Result
          Test-failed = TRUE
        ELSEIF [Reg 5] .NE. Expected Result
          Test-failed = TRUE
        ELSEIF [Reg 6] .NE. Expected Result
          Test-failed = TRUE
        ELSEIF [Reg 7] .NE. Expected Result
          Test-failed = TRUE
        ENDIF
        Reset RAMT in Register 0 (CR0)
        Delay 100 Clock Ticks
    ENDDO
```

Figure E.9 INTERNAL RAM TEST

4.0 EXAMPLE PDL DRIVER

This is an example driver which was originally implemented on an IBM PC. In this example, data is copied between system buffers and WD2511A buffers in order to simplify OSI layer interactions. However, two stages of buffering are not a requirement of the WD2511A. The driver implements a variable link layer window.

Not provided in PDL are the Kernel and Timer routines used by this driver. These are somewhat generic and vary with each user's system. This information is believed to be accurate and reliable. However, no responsibility is assumed by Western Digital for its use.

X.25 LEVEL 2 PDL

CONTENTS

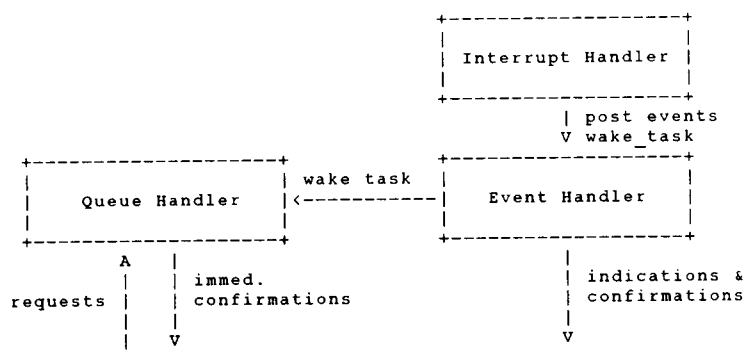
Overview of Main Driver Organization

The central part of the 2511 driver is organized around two semi-asynchronous tasks. The two tasks work together to drive the 2511. One task is responsible for managing a queue of service requests. The second task processes asynchronous events.

The queue handler task (QUEUE_HNDLR) accepts service requests in the form of request blocks (RBs). The request blocks are processed on a FIFO basis. At certain times, for example when the transmit window is closed, the queue may be blocked from proceeding. When asynchronous events occur to unblock the queue handler, it will be rescheduled by the event handler task.

The event handler task (EVENT_HNDLR) processes events, most of which are posted by the interrupt handler. Sometimes the processing of an event may cause the event handler to reschedule the queue handler.

The relationship of the two tasks and the interrupt handler is illustrated below:



The two main driver tasks depend heavily upon the services of the underlying tasking kernel and timer (see external procedure definitions). The tasking kernel provides for the ability to schedule tasks at one of several entry points, depending on the disposition of the task, and the contents of queued request blocks.

Overview of Main Driver Organization (continued)

For specifics relating to kernel and timer services refer to their respective interface specifications.

The queue handler may be scheduled at one of three entry points, depending upon the index field of RBs which are enqueued on it. The queue handler task may be properly described as a purely "message-driven" task (see kernel documentation. The currently defined entry points for the queue handler are:

- process_connect (taskid, rb, rbcnt)
- process_disconnect (taskid, rb, rbcnt)
- process_transmit_request (taskid, rb, rbcnt)

The event handler task is a purely "state_driven" task. Currently, it operates in only one state; event_processing. When events require service, the interrupt handler schedules the event handler task. The event handler continues processing events until there are no more to process. Once the events are all serviced, the event handler puts itself to sleep. The main entry point for the event handler is:

- process_event (taskid, rb, rbcnt)

External procedures refererenced by LAPBMAIN

```
.******************************
. Kernel and Timer Services:  *
.******************************

sleep()                            put current task to sleep
wake_task(taskid)                  reschedule specified task
DESCHEDULE                         deschedule current task and delete RB
..                                   note that this may not be implemented
..                                   as a function call
free_chain (pointer_to_hdr)        free chain of sys. bufs
cancel_timer (taskid, timerid)     cancel specified timer entry
create_timer (taskid, timerid,     create a timer entry
..          num_ticks,
..          service_routine,
..          extra_parm,errortable)
restart_timer (taskid, timerid,    restart previously existing timer
..          num_ticks,
..          service_routine,
..          extra_parm, errortable)

.*******************************************
.. L2 Service Primitives invoked by LAPBMAIN:*
.*******************************************

l_conn_conf()                      connect confirmation service
l_conn_ind()                       connect indication service *
l_disc_conf()                      disconnect confirmation service
l_disc_ind()                       disconnect indication service
l_data_conf()                      data transfer confirmation
l_data_ind()                       received data indication
l_sta_ind()                        status indication
l_rst_ind()                        reset indication

.. Note, l_conn_ind() is defined in ISO service specifications, but cannot ever happen using the 2511, since
    any connection must always be anticipated. This is due to the operation of the ACTIVE bit in the CR0
    command register.
```

X.25 LEVEL 2 SOFTWARE

External procedures refererenced by LAPBMAIN (continued)

```
.****************************************
. Utility Subroutines for 2511 board: *
.****************************************

copy_to_2511(frame_ptr,tlook_seg_adr)
..      copies a chain of buffers containing a frame into the 2511 board's
..      local memory.  Also fills in the TLOOK table.

copy_from_2511(rlook_seg_adr)
..      copies a frame from 2511 local board memory into system memory.

free_local_chain(adr_of_chain_hdr)
..      frees a buffer chain which was local to the 2511 board.

get_2511_buf(adr_of_chain_hdr)
..      get a buffer in 2511 board's local memory and attach it to
..      the specified chain

reset_tables()
..      reinitializes TLOOK and RLOOK tables, and buffer pool located on
..      the 2511 board
```

Global Definitions (Available to other modules)

```
D                                                                                                                    D
D                                                                                                                    D
D  1    driver_state represents the current link state of the driver.  Possible  values for driver_state are:       D
D  2                                                                                                                 D
D  3            up                  driver is currently in connected state                                           D
D  4            up_in_progress      driver is currently in process of connection                                     D
D  5            down                driver is currently disconnected                                                 D
D  6            down_in_progress    driver is in process of disconnection                                            D
D  7                                                                                                                 D
D  8    save_ER0 saves most recently read 2511 error register                                                        D
D  9    save_SR1 saves most recently read 2511 SR1                                                                   D
D 10    save_CR0 saves the command register image                                                                    D
D 11    save_CR1 saves the command register image                                                                    D
D 12                                                                                                                 D
D 13    frames_tx is a counter for number of frames transmitted                                                      D
D 14    frames_rx is a counter for number of frames received                                                         D
D 15                                                                                                                 D
D 16    tag[8] contains tag fields (or pointers) for each transmitted frame The tag fields are actually an extension of D
D               the TLOOK  table; an entry is filled in for each frame which is transmitted.  When the frame is      D
D               acknowledged, the driver passes the value of the tag field in the data confirmation so that the higher  D
D               layer may match requests with confirms.                                                              D
D 17    RB_2511 is a type definition for a request block containing the following  fields:                            D
D 18                                                                                                                 D
D 19        rb_next_rb  pointer to next rb in chain, NULL if this RB is last                                         D
D 20        rb_index    entry point index, used for scheduling                                                       D
D 21        rb_frame    pointer to buffer chain containing the frame                                                 D
D 22        rb_tag      an identifier associated with the frame                                                      D
D 23                                                                                                                 D
D 24    When one of the RB_2511 fields is referenced, it will be represented as:                                    D
D 25                                                                                                                 D
D 26         <pointer to rb> -> <field name>                                                                         D
D 27                                                                                                                 D
D 28    For example,  rb->rb_tag is the tag field of the request block  pointed to by rb.                            D
D 29                                                                                                                 D
D                                                                                                                    D
```

Local Data Definitions for LAPBMAIN

```
int_status records status accumulated by the interrupt handler Events are OR'd into this variable waiting for
        service by the event handler.  Possible values for int_status are:

        LNK_UP          link up event
        LNK_DWN         link down event
        TX_UR           transmit underrun event
        RX_OR           receive overrun event
        PKT_RCVD        packet arrived in receive queue
        TX_ACK          transmitted frame has been acknowledged
        RST             a link reset (or FRMR sequence) has occurred
        CHN_LEN         tried to receive frame longer than chain

event is a snapshot of int_status

..The following local data items are associated with TLOOK and RLOOK tables

t_next_in       index of TLOOK seg. where next frame will be enqueued
t_next_out      index of TLOOK seg. where next ACK is expected
r_next_in       index of RLOOK seg. where next empty chain should be placed
..              for receive
r_next_out      index of RLOOK seg. where next frame is expected to arrive
t_outstanding   number of outstanding transmit frames
```

```
interrupt handler
REF
PAGE
*******************************************************************************************
*                                                                                         *
*  1    ..This routine reads the 2511 interrupt status register. The status bits are partially decoded and OR'd into an *
*              interrupt status variable. The background task (event handler) is rescheduled to service the     *
*              appropriate events. Notice that all of the events associated with a link reset, for example, an FRMR *
*              condition, are lumped together into one event: RST. This is because the higher layer will probably make *
*              no distinction                                                                *
*  2    ..ENTRY: This routine is expected to be called by an assembly language interface routine. The assembly language *
*              routine may also invoke the kernel scheduler if the scheduler is not already running and an event *
*              requires service.                                                             *
*  3    ..EXIT: The return value will indicate whether or not the event handler background task requires rescheduling. *
*              ..                                                                            *
*  4    .......................................................................              *
*  5    ..                                                                                   *
*                                                                                         *
```

interrupt handler (continued)

```
*                                                                                                                                *
*  6     save_SR1 = input 2511 interrupt status register                                                                       *
*  7     IF (transmit acknowledgement)                                                                                         *
*  8         set TX_ACK in int_status                                                                                          *
*  9     ENDIF                                                                                                                 *
* 10     IF (packet arrived )                                                                                                  *
* 11         set PKT_RCVD in int_status                                                                                        *
* 12     ENDIF                                                                                                                 *
* 13     IF (error or event occurred)                                                                                          *
* 14         save_ER0 = input 2511 error status register                                                                       *
* 15         IF (linkup event)                                                                                                 *
* 16             set LNK_UP in int_status                                                                                      *
* 17         ENDIF                                                                                                             *
* 18         IF (we received a disconnect) OR (we transmitted a disconnect)                                                    *
* 19             set LNK_DWN in int_status                                                                                     *
* 20         ENDIF                                                                                                             *
* 21         IF (receive overrun occurred)                                                                                     *
* 22             set RX_OR in int_status                                                                                       *
* 23         ENDIF                                                                                                             *
* 24         IF (transmit underrun occurred)                                                                                   *
* 25             set TX_UR in int_status                                                                                       *
* 26         ENDIF                                                                                                             *
* 27         IF (chain overflow error occurred)                                                                                *
* 28             set CHN_LEN in int_status                                                                                     *
* 29         ENDIF                                                                                                             *
* 30         IF (any of the reset situations occurred)                                                                         *
* 31             set RST in int_status                                                                                         *
* 32         ENDIF                                                                                                             *
* 33         IF (an RNR interrupt occurred) .. check for missed reset                                                          *
* 34             temp = r_next_out                                                                                             *
* 35             DO WHILE (rlook_table[temp] has FRMCL set)                                                                    *
* 36                 temp = (temp + 1) mod 8                                                                                   *
* 37                 UNDO IF (temp == r_next_in)                                                                               *
* 38             ENDDO                                                                                                         *
* 39             IF (rlook_table[temp] is ready)                                                                               *
* 40                 set RST in int_status                                                                                     *
* 41             ENDIF                                                                                                         *
* 42         ENDIF                                                                                                             *
* 43     ENDIF                                                                                                                 *
* 44     IF (int_status has any event turned on)                                                                               *
4* 45         wake_task (event_handler)                                                                                         *
* 46     ENDIF                                                                                                                 *
* 47     IF (int_status is not 0)                                                                                              *
* 48         RETURN (invoke scheduler)                                                                                         *
* 49     ELSE                                                                                                                  *
* 50         RETURN (no scheduling required)                                                                                   *
* 51     ENDIF                                                                                                                 *
*                                                                                                                                *
**********************************************************************************************************************************
```

```
  process_connection (taskid, rb, rbcnt) .. QUEUE_HNDLR task
REF
PAGE
   ********************************************************************************************************************************
   *                                                                                                                              *
   *   1     .. This routine examines the driver state and, if the state is 'down', then it attempts to bring up the link.      *
   *               If there is an error in the request, immediate confirmation is issued. Otherwise, the confirmation will *
   *               be issued by the event handler as the result of an event.                                                    *
   *   2     ..ENTRY: rb is a pointer variable which points to an RB_2511 data                                                  *
   *   3     .. structure.                                                                                                      *
   *   4     ..                                                                                                                 *
   *   5     .. rb->rb_tag == 0 if this station should wait for remote to connect                                               *
   *   6     .. == 1 if this station should assume the active role                                                              *
   *   7     ..                                                                                                                 *
   *   8     ...................................................................                                                *
   *   9     IF (driver_state == down)                                                                                          *
   *  10         IF (rb->rb_tag <> 0)                                                                                           *
   *  11             set ACTIVE bit in save_CR0                                                                                 *
   *  12         ELSE                                                                                                           *
   *  13             reset ACTIVE bit in save_CR0                                                                               *
   *  14         ENDIF                                                                                                          *
   *  15         reset MDISC in save_CR0;                                                                                       *
   *  16         output (save_CR0 + RECR) to the 2511 CR0 register                                                              *
   *  17         driver_state = up_in_progress                                                                                  *
  12 * 18         check_receive() to initialize receive timer                                                                    *
   4 * 19         sleep()                                                                                                        *
   *  20         DESCHEDULE                                                                                                     *
   *  21     ELSE ..the only other case allowed on the queue handler is link already up                                         *
   4 * 22         l_conn_conf(link already up)                                                                                   *
   *  23         DESCHEDULE                                                                                                     *
   *  24     ENDIF                                                                                                              *
   *                                                                                                                              *
   ********************************************************************************************************************************
```

```
  process_disconnect (taskid, rb, rbcnt)..QUEUE_HNDLR task
REF
PAGE
  ********************************************************************************************************************************
  *                                                                                                                              *
  *   1    ..This routine examines the current state of the link and initiates link-down processing. An attempt is made to *
  *               shut the link down gracefully by allowing outstanding I-frames to be acknowledged first. Before the     *
  *               confirmation is generated, received frames are cleared out of the RLOOK table.                           *
  *   2    ..                                                                                                              *
  *   3    ..ENTRY: rb is a pointer variable which points to a RB_2511 data structure.                                     *
  *   4    .. no fields of the rb are currently used                                                                       *
  *   5    ..                                                                                                              *
  *   6    ................................................................                                                *
  *   7    ..                                                                                                              *
  *   8    IF (driver_state == down)                                                                                       *
 4*   9        l_disc_conf (link already down)                                                                             *
  *  10        DESCHEDULE                                                                                                  *
  *  11    ELSE only other case allowed for queue handler is link up                                                       *
  *  12        driver_state = down_in_progress                                                                             *
  *  13        IF (t_outstanding)                                                                                          *
  *  14            DO WHILE (there are more transmit frames outstanding,                                                   *
  *  15                i.e. t_outstanding > 0)                                                                             *
 11*  16                init_send () to make sure the transmit isn't stuck                                                  *
 4 *  17                sleep()                                                                                             *
  *  18                UNDO IF (driver_state has changed)                                                                  *
  *  19            ENDDO                                                                                                   *
  *  20        ELSE                                                                                                        *
 16*  21            process_link_down()                                                                                     *
 4 *  22            l_disc_conf(GOOD)                                                                                       *
  *  23        ENDIF                                                                                                       *
  *  24        DESCHEDULE                                                                                                  *
  *  25    ENDIF                                                                                                           *
  *                                                                                                                              *
  ********************************************************************************************************************************
```

```
  process_transmit_request (taskid,rb,rbcnt).. QUEUE_HNDLR task
REF
PAGE
   ********************************************************************************************************************************
   *                                                                                                                              *
   *   1     ..This routine attempts to enqueue the next frame for the 2511. If the window is closed, then the Queue Handler   *
   *               task sleeps until the window opens or a DISC or RESET occurs.                                                  *
   *   2     ..                                                                                                                     *
   *   3     ..ENTRY: rb is a pointer variable pointing to a RB_2511 data structure                                                 *
   *   4     .. rb->rb_frame is a pointer to a chain of data buffers                                                                *
   *   5     .. rb->rb_tag is an identifier associated with the frame for                                                           *
   *   6     .. acknowledgment purposes                                                                                             *
   *   7     ..........................................................................                                             *
   *   8     ..                                                                                                                     *
   *   9     DO WHILE (window is closed, i.e. t_outstanding == window)                                                              *
   * 10          IF (driver_state <> up)                                                                                           *
 4 * 11              l_data_conf (link already down, rb->rb_tag)                                                                   *
   * 12              DESCHEDULE                                                                                                    *
   * 13          ENDIF                                                                                                             *
 4 * 14          sleep()                                                                                                           *
   * 15      ENDDO                                                                                                                 *
   * 16      status =                                                                                                              *
 4 * 17      copy_to_2511(rb->rb_frame, pointer to tlook_table[t_next_in])                                                         *
   * 18      IF (status is not good) .. no buffers available, etc.                                                                 *
   * 19          DESCHEDULE (leave current RB queued and try later)                                                                *
   * 20      ENDIF                                                                                                                 *
   * 21      set BRDY in tlook_table[t_next_in]                                                                                    *
   * 22      t_outstanding = t_outstanding + 1                                                                                     *
11 * 23      init_send() to set SEND bit and start deadman timer for transmit                                                      *
   * 24      tag[t_next_in] = rb->rb_tag                                                                                           *
   * 25      t_next_in = (t_next_in + 1) MOD 8                                                                                     *
   * 26      DESCHEDULE                                                                                                            *
   *                                                                                                                              *
   ********************************************************************************************************************************
```

```
   init_send() -- initiate/reinitiate frame transmission
REF
PAGE
   ********************************************************************************************************************************
   *                                                                                                                              *
   *  1     .. This routine restarts a timer which is used to set the SEND bit again in the event that a transmission fails *
   *              to start.                                                                                                       *
   *  2     ..                                                                                                                    *
   *  3     ..............................................................                                                        *
   *  4     status =                                                                                                              *
 4 *  5     restart_timer (QUEUE_HNDLR, tx_timer_id, tx_timeout, init_send, NULL,0)                                               *
   *  6     IF (status is not good)                                                                                               *
 4 *  7        create_timer (QUEUE_HNDLR,tx_timer_id, tx_timeout, init_send, 0, NULL)                                             *
   *  8     ENDIF                                                                                                                 *
   *  9     set the SEND bit in CR1                                                                                               *
   * 10     RETURN                                                                                                                *
   *                                                                                                                              *
   ********************************************************************************************************************************
```

```
   check_receive() -- if missed interrupt, check queue
REF
PAGE
   ********************************************************************************************************************************
   *                                                                                                                              *
   *   1     ..This routine is invoked when a timeout occurs to scan the RLOOK table for newly arrived frames or missed link     *
   *               reset events. The timer is restarted before exiting.                                                           *
   *   2     ..                                                                                                                   *
   *   3     ..............................................................                                                       *
   *   4     IF (rlook_table[r_next_out] had FRMCL set)                                                                           *
   *   5         disable interrupts                                                                                               *
   *   6         set PKT_RCVD in int_status                                                                                       *
   *   7         enable interrupts                                                                                                *
 4 *   8         wake_task(EVENT_HNDLR)                                                                                           *
   *   9     ENDIF                                                                                                                *
   *  10     IF (RNRX is set in SR0) .. possibly overwritten RST                                                                  *
   *  11         temp = r_next_out                                                                                                *
   *  12         DO WHILE (rlook_table[temp] has FRMCL set)                                                                       *
   *  13             temp = (temp + 1) mod 8                                                                                      *
   *  14             UNDO IF (temp == r_next_in)                                                                                  *
   *  15         ENDDO                                                                                                            *
   *  16         IF (rlook_table[temp] is ready)                                                                                  *
   *  17             disable interrupts                                                                                           *
   *  18             set RST in int_status                                                                                        *
   *  19             enable interrupts                                                                                            *
   *  20         ENDIF                                                                                                            *
   *  21     ENDIF                                                                                                                *
   *  22     status =                                                                                                             *
 4 *  23     restart_timer (EVENT_HNDLR, receive_timer_id, receive_timeout,                                                       *
   *  24     .. check_receive, NULL, 0)                                                                                           *
   *  25     IF (status is not good)                                                                                              *
 4 *  26         create_timer (EVENT_HNDLR, receive_timer_id, receive_timeout,                                                    *
   *  27         .. check_receive, NULL, 0)                                                                                       *
   *  28     ENDIF                                                                                                                *
   *  29     RETURN                                                                                                               *
   *                                                                                                                              *
   ********************************************************************************************************************************
```

```
   process_event (taskid, rb, rbcnt) ..EVENT_HNDLR task
REF
PAGE
   ********************************************************************************************************************************
   *                                                                                                                              *
   *   1     ..This routine is the only entry point for the event handler task. The task is state-driven (see kernel              *
   *               documentation), so no RBs ever get created on it. When no events need to be processed, this task sleeps       *
   *               until the interrupt handler reschedules it.                                                                    *
   *   2     ..                                                                                                                   *
   *   3     ..ENTRY: The event handler is rescheduled (via wake_task()) from the interrupt handler routine.                      *
   *   4     ..                                                                                                                   *
   *   5     ....................................................................                                                 *
   *   6     ..                                                                                                                   *
   *   7     DO FOREVER                                                                                                           *
   *   8         old_state = driver_state                                                                                         *
   *   9         DO UNTIL (event is not equal to 0) -- do at least once                                                           *
   *  10             disable interrupts                                                                                           *
   *  11             event = int_status                                                                                           *
   *  12             int_status = 0                                                                                               *
   *  13             IF (event == 0) sleep()                                                                                      *
   *  14             ENDIF                                                                                                        *
   *  15         ENDDO                                                                                                            *
   *  16         enable interrupts                                                                                                *
   *  17         IF (event & TX_UR)                                                                                               *
   4* 18             l_sta_ind (transmit underrun occurred)                                                                       *
   *  19         ENDIF                                                                                                            *
   *  20         IF (event & RX_OR)                                                                                               *
   4* 21             l_sta_ind (receive overrun occurred)                                                                         *
   *  22         ENDIF                                                                                                            *
   *  23         IF (event & CHN_LEN)                                                                                             *
   4* 24             l_sta_ind (a buffer chain was exceeded)                                                                      *
   *  25         ENDIF                                                                                                            *
   *  26         IF (event & PKT_RCVD) AND (driver_state == up_in_progress                                                        *
   *  27             AND (link is up, i.e. SR2.LINK is reset) -- somehow missed linkup                                            *
   *  28             event = logical_or(event,LNK_UP)                                                                             *
   *  29         ENDIF                                                                                                            *
   *  30         IF (event & LNK_UP) AND (link is not already down)                                                               *
   *  31             turn off LNK_DWN and RST events, since they must have been previous                                          *
  14* 32             process_link_up()                                                                                            *
  17* 33             empty_rlook()                                                                                                *
   *  34         ENDIF                                                                                                            *
   *  35         IF (event & PKT_RCVD)                                                                                            *
   *                                                                                                                              *
```

process_event (taskid, rb, rbcnt) ..EVENT_HNDLR task (continued)

```
17              empty_rlook()
            ENDIF
            IF (event & TX_ACK) .. some frames have been ack'd
                .. note that we could be cleaning up in preparation for disconnect
15              process_tx_acks()
                IF (driver_state == down_in_progress)
                    AND (no transmit frames are outstanding, i.e. t_outstanding==0)
16                  process_link_down()
4                   l_disc_conf(graceful link down)
                ENDIF
            ELSEIF (event & LNK_DWN)
                DO CASE (old_state)
                up:
4                   l_disc_ind()
                up_in_progress:
4                   l_conn_conf(remote disconnected)
                down_in_progress:
4                   l_disc_conf (remote disconnected)
                    .. there may be unacknowledged frames outstanding
                down:
                    .. how did this happen? we're already down, so ignore it
                ENDDO
16              process_link_down()
            ELSEIF (event & RST)
17              empty_rlook() -- l_rst_ind will be generated from this routine
            ENDIF
            IF (old_state <> driver_state) OR (t_outstanding < window)
4               wake_task (QUEUE_HNDLR)
            ENDIF
        ENDDO
```

```
   process_link_up()
REF
PAGE
   ********************************************************************************************************************************
   *                                                                                                                              *
   *  1     .. This function performs the processing associated with a linkup event. Connect confirmations and indications  *
   *              are issued from this routine.                                                                                   *
   *  2     ..                                                                                                                    *
   *  3     ..............................................................                                                        *
   *  4     DO CASE (driver_state)                                                                                                *
   *  5     up: .                                                                                                                 *
 4 *  6         l_rst_ind()                                                                                                       *
   *  7         disable interrupts                                                                                                *
   *  8         set RST in int_status                                                                                             *
   *  9         enable interrupts                                                                                                 *
17 * 10         empty_rlook()                                                                                                     *
   * 11     down_in_progress:                                                                                                     *
 4 * 12         l_rst_ind()                                                                                                       *
   * 13         disable interrupts                                                                                                *
   * 14         set RST in int_status                                                                                             *
   * 15         enable interrupts                                                                                                 *
17 * 16         empty_rlook()                                                                                                     *
   * 17     up_in_progress:                                                                                                       *
   * 18         driver_state = up                                                                                                 *
 4 * 19         l_conn_conf (good, link is up)                                                                                    *
   * 20     down: .. This should never happen, since the chip is not enabled                                                      *
   * 21         .. to let the remote connect in the down state                                                                    *
   * 22         HALT (unknown chip state)                                                                                         *
   * 23     ENDDO                                                                                                                 *
   * 24     RETURN                                                                                                                *
   *                                                                                                                              *
   ********************************************************************************************************************************
```

```
  process_tx_acks()
REF
PAGE
   ********************************************************************************************************************************
   *                                                                                                                              *
   *  1     ..This routine performs the processing associated with frame acknowledgement events. The current 2511 transmit      *
   *              queue is searched for acknowledged frames. Data confirmations are issued for each acknowledged frame,          *
   *              and the frame buffers on the 2511 board RAM are freed for reuse.                                               *
   *  2     ..                                                                                                                   *
   *  3     ..........................................................................                                           *
   *  4     ..                                                                                                                   *
   *  5     DO WHILE (i.e. more acknowledged frames, i.e. t_outstanding >0)                                                      *
   *  6        UNDO IF (tlook_table[t_next_out] does not have ACKED bit set)                                                     *
 4 *  7        free_local_chain (ptr to tlook[t_next_out].tsadr_hi)                                                              *
 4 *  8        l_data_conf(frame acknowledged, tag[t_next_out])                                                                  *
   *  9        frames_tx = frames_tx + 1                                                                                         *
   * 10        tag[t_next_out] = NULL                                                                                            *
   * 11        t_next_out = (t_next_out + 1) MOD 8                                                                               *
   * 12        t_outstanding = t_outstanding - 1                                                                                 *
   * 13     ENDDO                                                                                                                *
   * 14     RETURN                                                                                                               *
   *                                                                                                                              *
   ********************************************************************************************************************************
```



```
  process_link_down()
REF
PAGE
   ********************************************************************************************************************************
   *                                                                                                                              *
   *  1     .. This function performs link_down event processing. Confirmations or indications are issued, depending upon       *
   *              the current state of the driver.                                                                               *
   *  2     ..                                                                                                                   *
   *  3     ..............................................................                                                       *
   *  4     reset ACTIVE in save_CR0                                                                                             *
   *  5     set MDISC in save_CR0                                                                                                *
   *  6     output save_CR0 to 2511 register CR0                                                                                 *
 17 *  7     empty_rlook()                                                                                                        *
   *  8     DO FOR each outstanding transmit frame                                                                               *
 4 *  9        l_data_conf(link already down, tag[t_next_out]                                                                    *
   * 10        t_next_out = (t_next_out = 1) MOD 8                                                                               *
   * 11     ENDDO                                                                                                                *
   * 12     t_outstanding = 0                                                                                                    *
 4 * 13     reset_tables() .. reinitialize 2511 tables and RAM                                                                   *
 4 * 14     cancel_timer (QUEUE_HNDLR, send_timer_id)                                                                            *
 4 * 15     cancel_timer (EVENT_HNDLR, receive_timer_id)                                                                         *
   * 16     driver_state = down                                                                                                  *
   * 17     RETURN                                                                                                               *
   *                                                                                                                              *
   ********************************************************************************************************************************
```

```
  empty_rlook() -- empty out frames in rlook
REF
PAGE
  ********************************************************************************************************************************
  *                                                                                                                              *
  *  1     .. This routine empties out the RLOOK table, taking care to adjust the pointers in the event of a link-reset      *
  *              condition. If a reset occurs it also calls a routine to restart the transmitter.                              *
  *  2     ..                                                                                                                    *
  *  3     ..............................................................                                                        *
  *  4     just_reset = event & RST                                                                                              *
  *  5     .. if we're currently processing a reset event, note it                                                               *
  *  6     DO UNTIL NOT (rlook[r_next_in] has FRMCL set)OR ( just_reset))                                                        *
  *  7          ..do at least once                                                                                               *
  *  8          IF NOT (just_reset)                                                                                              *
  *  9              disable interrupts ..pick up any RST since last reading                                                      *
  * 10              IF (int_status & RST )                                                                                       *
  * 11                  just_reset = TRUE                                                                                        *
  * 12              ENDIF                                                                                                        *
  * 13              reset RST in int_status                                                                                      *
  * 14              enable interrupts                                                                                            *
  * 15          ELSE                                                                                                             *
  * 16              IF (r_next_out == 0) OR (rlook[r_next_out has FRMCL reset)                                                   *
  * 17                  shift_segments                                                                                           *
  * 18                  r_next_out = 0                                                                                           *
  * 19                  r_next_in = window                                                                                       *
  * 20                  set RECR bit in CR0                                                                                      *
  * 21                  just_reset = FALSE                                                                                       *
18* 22                  restart_tx()                                                                                             *
 4* 23                  l_rst_ind()                                                                                              *
  * 24              ENDIF                                                                                                        *
  * 25          ENDIF                                                                                                            *
  * 26          IF (rlook[r_next_out] has FRMCL bit set)                                                                         *
  * 27              frame =                                                                                                      *
 4* 28              copy_from_2511 (ptr to rlook[r_next_out])                                                                    *
  * 29              .. possible deschedule here for buffer allocation                                                            *
 4* 30              l_data_ind(frame) .. pass frame to higher layer                                                              *
  * 31              frames_rx = frames_rx + 1                                                                                    *
  * 32              requeue empty buffer and bump rlook pointers                                                                 *
  * 33          ENDIF                                                                                                            *
  * 34     ENDDO                                                                                                                 *
  * 35     RETURN                                                                                                                *
  *                                                                                                                              *
  ********************************************************************************************************************************
```

```
   restart_tx()
REF
PAGE
   ********************************************************************************************************************************
   *                                                                                                                              *
   *  1     .. This function is used for recovering from a link reset condition. It cleans out the tlook, and shifts the      *
   *              currently waiting buffers into position at the tip of the tlook table. SEND is enabled at the end of            *
   *              this routine.                                                                                                   *
   *  2     ..                                                                                                                    *
   *  3     ..........................................................................                                            *
   *  4     ..                                                                                                                    *
15 *  5     process_tx_acks()                                                                                                     *
   *  6     ..                                                                                                                    *
   *  7     .. shift window to top of tlook                                                                                       *
   *  8     ..                                                                                                                    *
   *  9     copy current tlook entries into temp table                                                                            *
   * 10     copy temp table back to tlook starting at 0                                                                           *
   * 11     adjust tag entries                                                                                                    *
   * 12     ..                                                                                                                    *
11 * 13     init_send()                                                                                                           *
   * 14     t_next_out = 0                                                                                                        *
   * 15     t_next_in = t_outstanding                                                                                             *
   * 16     RETURN                                                                                                                *
   *                                                                                                                              *
   ********************************************************************************************************************************
```

APPENDIX F

HARDWARE APPLICATION NOTES

1.0 THE WD2511A GENERAL DESCRIPTION

The WD2511A is an LSI device that fully handles the link level (level 2) of the CCITT X.25 communications protocol.

In addition to the traditional parallel/serial converters and FCS logic, the WD2511A incorporates a highly efficient micro-programmed processor that fully handles the required link set-up and frame sequencing operations conventionally delegated to a "user defined" processor. The WD2511A also contains an intelligent two-channel DMA controller to further simplify its integration into a user's system plus the necessary T1 timer and N2 counter.

2.0 HARDWARE

The WD2511A must be connected to the Physical Level (Level 1), usually with simple line drivers/receivers.

A typical X.25 DTE/DCE station block diagram is shown in Figure F.1.

A modem is needed for long-distance communication lines.

2.1 READ/WRITE CONTROL OF I/O REGISTERS

The sixteen I/O registers are directly accessible from the CPU data bus (DAL0-DAL7) by a read and/or write operation by the CPU. The CPU must activate the WD2511A register address (IA0-IA3), Chip Select ($\overline{CS}$), Write Enable ($\overline{WE}$), or Read Enable ($\overline{RE}$) before each data bus transfer operation. The read/write operation is completed when $\overline{CS}$ or $\overline{RE}/\overline{WE}$ is brought high. During a write operation, the falling edge of $\overline{WE}$ will initiate a WD2511A write cycle. The addressed register will then be loaded with the content of the Data Bus. The rising edge of $\overline{WE}$ will latch that data into the addressed register.

During a read operation, the falling edge of $\overline{RE}$ will initiate a WD2511A read cycle. The addressed register will then place its contents onto the Data Bus.

The user must set-up all transmit data, TSADR HI and LO, and TCNT HI and LO, before setting BRDY in the applicable TLOOK segment.

The user must set aside receiver memory (at least one chain segment with transfer address), and set-up RSADR HI and LO before setting REC RDY in the applicable RLOOK segment.

This procedure is described in detail in Appendix E.

2.2 DMA IN/OUT OPERATION

The Direct Memory Access (DMA) operation is completely controlled by the WD2511A. During a DMA cycle, the CPU sets its address bus, data bus, and three-state control signals to their high impedance states.

During a DMA In cycle, the task of transferring one byte of I-field data from memory into the WD2511A is performed by the chip. The CPU time (in the example described in this paragraph to execute this task) is five T-states for a low speed CPU clock system and ten T-states for a high speed CPU clock system. The DMA In function starts when the WD2511A is ready to receive a byte from memory to be transmitted to the remote station.

The DMA out operation is very similar to the DMA In function. During this cycle, one byte of I-field data is transferred from the WD2511A to the memory. The CPU-time in this example described to perform this task is the same as for the DMA In cycle. The DMA Out function starts when the WD2511A is holding a received I-field byte and is ready to transfer this to the memory.

2.3 SERIAL INTERFACE

The receiver and transmitter sub-systems are completely independent of each other, the CPU Read/Write functions and the DMA In/Out functions.

The serial data is synchronized by the externally supplied $\overline{TC}$ clock and $\overline{RC}$ clock. The falling edge of $\overline{TC}$ generates new transmitted data and the rising edge of $\overline{RC}$ is used to sample the received data.

After initilization and before the first frame is sent, the TD output sends IDLEs (continuous 1s).

After the first frame is sent or the ACTIVE/$\overline{PASSIVE}$ bit is set, continuous flags are sent in between frames.

For detailed information on what type of frames are sent for certain conditions, see the WD2511A specifications.

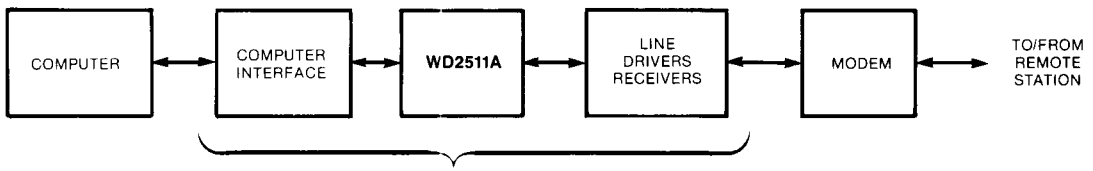


Figure 5 (Section 2, Data Sheet)

Figure F.1 DTE/DCE STATION BLOCK DIAGRAM

GLOSSARY OF DATA COMMUNICATIONS TERMS

The following is a list of industry-accepted data communications terms that are applicable to this specification.

ABM	Asynchronous Balanced Mode
ADCCP	Advanced Data Communications Control Procedure (ANSI BSR X3.66)
ANSI	American National Standards Institute
ARM	Asynchronous Response Mode
CCIT	International Consultative Committee for Telegraphy and Telephony
DCE	Data Circuit Termination Equipment (the network side of the DTE/DCE link)
DISC	Disconnect. A U-Frame Command
DTE	Data Terminal Equipment
DM	Disconnect Mode. A U-Frame Response
ECMA	European Computer Manufacturers Association
FCS	Frame Check Sequence
FDX	Full Duplex (also called: "two way simultaneous")
FRAME	Basic serial block of bit-oriented data. Includes leading and training flags, address field, control field. FCS field, and an optional information field.
FRMR	Frame Reject. A U-Frame
HDLC	High-Level Data Link Control (ISO 3309)
HDX	Half Duplex (also called "two way alternate")
HOST	Another name for a DTE
I-Frame	Information Frame. Control field bit 0 is 0. In X.25 an I-frame is a packet.
ISO	International Standards Organization
LINK	The logical and physical connection between two data terminals.
LAP	Link Access Procedure
LAPB	Link Access Procedure Balanced
N2	Maximum number of retransmissions of a frame. (Also called retransmission count variable.)
NODE	Another name for a DCE or DTE
N(R)	Sequence number of next frame expected to be received
N(S)	Sequence number of current frame being transmitted
OCTET	An 8-bit byte
PACKET	An I-Frame in X.25
PAD	Packet Assembly/Disassembly facility
REJ*	Reject. An S-Frame Command or Response
RNR*	Receiver Not Ready. An S-Frame Command or Response
RR*	Receiver Ready. An S-Frame Command or Response
S-Frame	Supervisory Frame. Control field bit 0 = 1 and bit 1 = 0
SABM	Set Asynchronous Balanced Mode. A U-Frame Command
SDLC	Synchronous Data Link Control (IBM document GA27-3093)
T1	A Primary Timer for a delay in waiting for a response to a frame
U-Frame	Unnumbered Frame. Control Field bit 0 = 1 and bit 1 = 1
UA	Unnumbered Acknowledge. A U-Frame Response
X.25	Recommendation by CCITT on Interfacing to Public Packet Switching Networks
X.3, X.28, X.29	Recommendations by CCITT involving PAD facilities

*There are also RR, RNR, and REJ packets which are not the same as the S-Frame RR, RNR and REJ discussed in this document.

Western Digital
2445 McCabe Way
Irvine, California 92714
(714) 474-2033 (714) 863-0102
FAX 714-660-4909 TLX 910-595-1139

For Information on WD Communications Products
Call: 1-800-NET LEADER (1-800-638-5323)

WESTERN DIGITAL

W1336C 8/87 5M