ADAPTING A WAN CONTROLLER TO A LAN ENVIRONMENT

Robert W. O'Dell and Lloyd A. Hasley
Applications Engineer and Design Manager
Motorola
6501 William Cannon Drive West
Austin, TX 78735-8598 (NC-025)

ABSTRACT

This article describes how a popular multi-protocol communications controller called the MC68302, is adapted to meet the requirements of the popular AppleTalk™ LAN. The major interest in this effort is that the MC68302 was designed principally with wide area network (WAN) applications in mind -- not LAN applications. For instance, even though many WAN and LAN protocols are based on high level data-link control (HDLC), significant differences in HDLC-based WAN and LAN design still exist. The solution to the problem has uncovered some principles about the design, and some principal areas to look out for, when adapting WAN-type controllers to LAN environments.

INTRODUCTION

The MC68302 is a multiprotocol communications controller that includes a 68000 processor core, a system integration block, and a communications processor (see figure 1). The 68000 core is that of a standard 68HC000, except that it also has the capability to operate in an 8-bit mode (68008) for lower cost applications.

The systems integration block contains generally useful peripherals such as 3 timers, an interrupt controller, one independent DMA controller, a DRAM refresh controller, 4 chip selects, 1152 bytes of dual-port RAM, bus arbiter, and parallel I/O lines.

The communications processor (CP) contains three serial channels (SCCs), each of which can run HDLC, BISYNC, UART, DDCMP, V.110 or totally transparent protocols. With each SCC are provided 2 serial DMA (SDMA) channels for transfer of data to/from external memory. The CP contains 2 SMC channels useful in ISDN applications, and an SCP for communication with simple peripherals like EEPROMs, Real-time clocks, etc. Finally the CP contains a physical interface multiplexor for connecting the three serial channels to an ISDN basic rate, or PCM highway application. At the heart of the communications processor is a RISC engine that assists with the execution of the various protocols that the chip offers.

The MC68302 has applications in many communications applications such as modems, switches, and multiplexors, as well as more general purpose applications such as laser printers, medical equipment, and industrial control. In terms of its use in AppleTalk applications, the MC68302 is currently being used in laser printers and gateways with the adaption strategies outlined here.

AppleTalk is a set of protocols developed by Apple Computer to provide a local area network service between Macintosh™ computers and printers. LocalTalk™ is used to refer to the two lowest layers of AppleTalk, that uses an HDLC-based protocol at a rate of 230.4 Kbps.

The adaption ofthe MC68302 controller to the AppleTalk canbe categorized into two areas of concern: hardware and software.

The hardware portion of the solution for this WAN to LAN adaptation takes place inside a MC68302 sister chip, called the MC68195 LocalTalk Adaptor (LA). This device implements the features that the MC68302 lacks in interfacing to LocalTalk. These requirements include:

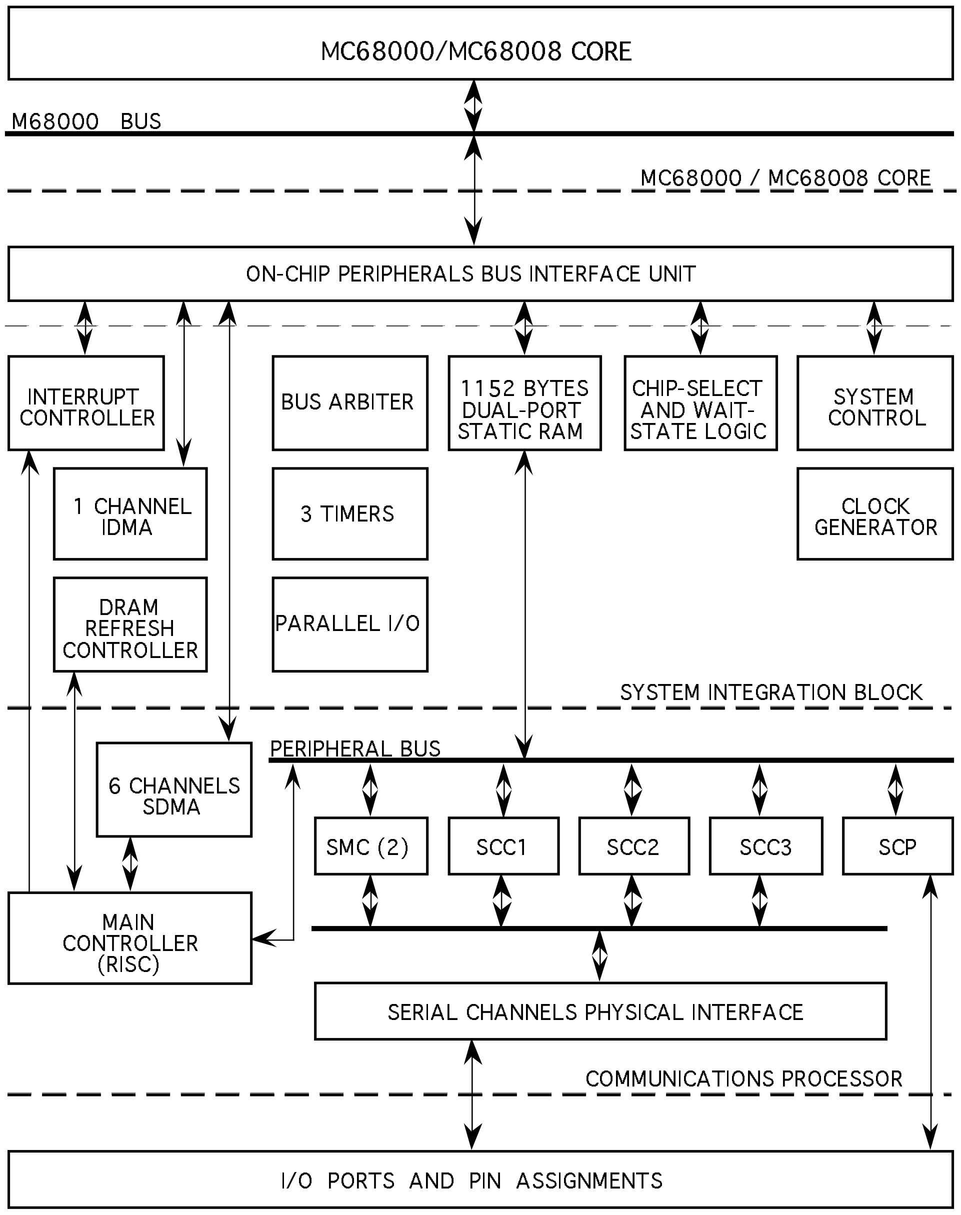

Figure 1: MC68302 Block Diagram

(1) FM0 data clock encoding/decoding,
(2) The preamble synchronization sequence,
(3) The postamble frame abort sequencing, and
(4) Providing the physical transceiver interface.

These required additions are discussed and contrasted with typical WAN features.

The other portion of the solution, is the software and MC68302 configuration techniques that are required to fully implement LocalTalk. These include:

(1) Meeting the required inter-frame gap timing by synchronizing the M68000 core to the on-chip RISC processor,
(2) Dealing with the half-duplex nature of the protocol on a full duplex device, and
(3) Buffer management including the generation of "minute-man" frames to speed transmission.

It will be shown how to solve these problems using the MC68302.

Finally, the results of the solutions are generalized to using WAN-type controllers in LAN environments. The problem areas are generalized, and the board designer is cautioned about key questions to ask the manufacturer.

HARDWARE ISSUES

The solutions to the hardware requirements in adapting the MC68302 to an AppleTalk environment are solved in a MC68302 sister chip, called the MC68195 LocalTalk Adaptor (LA). This device implements the features that the MC68302 lacks in interfacing to AppleTalk.

For a LocalTalk channel, the M68000 core on the MC68302 runs the link layer and higher layer software (see Figure 2). Data is received and transmitted through the MC68302 serial channels, which are configured in the high-level data link control (HDLC) mode. TheLA device handles the FM0 encoding, decoding, and clockrecovery as well as the generation of the special LocalTalk sync and abort sequences. Finally, RS-422 line drivers/receivers are used to interface theLA device to the network media. The LA device hastwo channels that allow it to interface to two LocalTalk networks simultaneously.

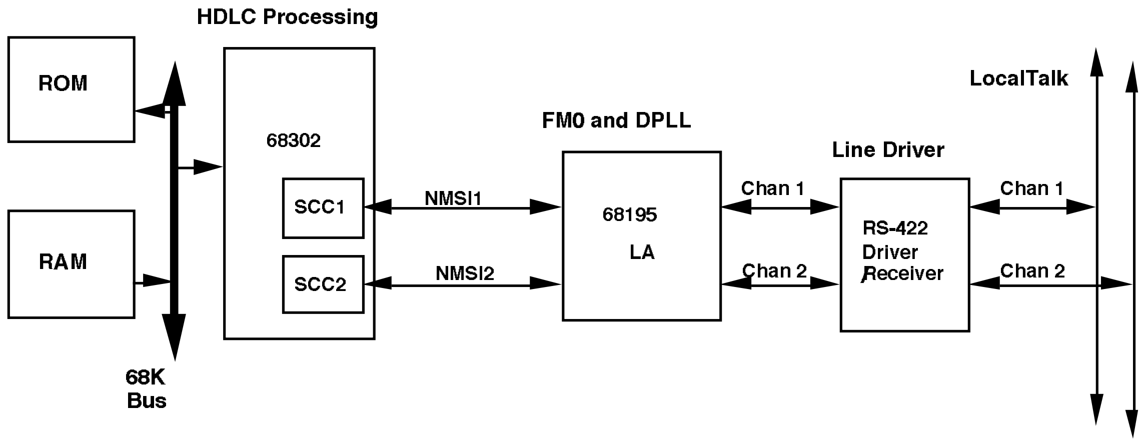

Figure 2: MC68195 LA Adapts the MC68302 to AppleTalk

Figure 3 shows a daughter card for the LA devicethat can plug into the ADS302 development board. The ADS302 board is widely used for development of MC68302 software. The daughter card shown in the figure, plugs into the ADS302 board, and provides the hardware solution for adapting the MC68302 to LocalTalk. With this board, all the major features of the LA device can be evaluated.

(See section D.9 “An Appletalk® Node with the MC68302 and MC68195” in the MC68302 User’s Manual for this schematic.)

Figure 3: Schematic Diagram: LA Daughter Board

LOCALTALK OVERVIEW

To understand the functions of the LA device and the hardware problems it solves, a basic understanding of LocalTalk is required.

A LocalTalk frame is a modified HDLC frame (see Figure 4).

SYNC SEQ.	HDLC FLAG	CONTROL ADDRESS	DEST. ADDRESS	SOURCE BYTE	DATA BYTES	CRC-16	ENDING FLAG	ABORT SEQUENCE
>3 bits	2 or more bytes	1 byte	1 byte	1 byte	0-600 bytes	2 bytes	1 byte	bits

Figure 4: LocalTalk Frame Format

First, a synchronization sequence of greater than three bits is sent. This sequence consists of at least one logical one bit, FM0 encoded, followed by greater than two bit times of line idle. No particular maximum time is specified for this line idle time.

The idle time allows some LocalTalk equipment to sense carrier by detecting a "missing clock" on the line. This protocol "feature" is a method of reducing the latency imposed by the Z8530 serial communication controller, which, otherwise, would not be able to detect the start of the frame until a time point one byte time after the receipt of the opening flags.

This is the first major departure from typical WAN protocols that use HDLC. Most WAN-based HDLC protocols have no such non-flag "preamble", and it was immediately clear that the MC68302 did not support the transmission of such a preamble automatically. In most WAN-based systems, the preamble is nothing more than a line idle (all ones) condition. Thus the LocalTalk preamble must be implemented in the LA device.

The remainder of the frame is a typical half-duplex HDLC frame. Two or more flags are sent allowing bit, byte, and frame delineation/detection. Two bytes of address, destination and source, are transmitted next. This is followed by a byte of control, and zero to 600 data bytes. Next, two bytes of CRC are sent. The CRC is the common CRC16 polynomial referenced in the HDLC standard. These features are well within the domain of typical WAN-controllers including the MC68302.

The LocalTalk frame is then terminated by a flag and a restricted HDLC abort sequence (a sequence of twelve to eighteen logical ones). The transmitter's driver is then disabled.

In a WAN-based HDLC protocol, the termination of an HDLC frame, is usually a sequence of ones, which repeats infinitely until another frame is sent, or continuous, repeating flags. LocalTalk, then, differs from the WAN-based HDLC protocols, in that the number of ones sent after the frame cannot be infinite, at least from the line transmitter point-of-view; the line transmitter must turn off after 12-18 ones.

The line transmitter is controlled in the RS422 driver chip, thus, the hardware must provide a transmit enable signal to the RS422 chip, to control the transmission (both on and off conditions). Thus, once again, some hardware support is needed to implement LocalTalk, as compared to a standard HDLC frame. This includes the generation of the 12-18 bit postamble, and the ability to control the RS-422 transceiver.

The control byte within the LocalTalk frame indicates the type of frame. Control byte values from 0x01 to 0x7f are data frames and control byte values from 0x80 to 0xff are control frames. Four different control frames are currently defined: ENQ (Enquiry), ACK (ENQ acknowledgement), RTS (request to send a data frame), and CTS (clear to send a data frame).

The control byte has many different configurations in WAN-based HDLC protocols, and is often 2 bytes, not just one. Most HDLC controllers leave the control byte to the domain of the software, thus, the differences in the LocalTalk control byte has no effect on the hardware implementation. This is the case with the MC68302.

Frames are sent in groups known as dialogs. For instance, to transfer a data frame, three frames are actually sent over the network: a RTS frame (not to be confused with the RS-232 pin RTS) is sent requesting the network, a CTS frame is sent by the destination node, and finally the data frame is sent by the requesting node. These three frames comprise one possible type of dialog. Once a dialog has begun, other nodes cannot begin transmission until the dialog is complete. Dialogs are typically handled in software.

Frames within a dialog are transmitted with a maximum interframe gap (IFG) of 200 μS. Although the LocalTalk specification does not state it, there is also a minimum "recommended" IFG of 50 μS. Dialogs must be separated by a minimum interdialog gap (IDG) of 400 μS. In Apple's present LocalTalk implementations, these gaps are implemented via software.

Although the special preamble synchronization sequence only needs to be sent preceding RTS frames, the sequence can be sent prior to any frame and still comply with the LocalTalk specification. Due to the protocol definition, collisions should only be encountered during RTS and ENQ frames. Once a frame's transmission is started, it is fully transmitted regardless of whether it collides with another frame. ENQ frames are infrequent, being sent only when a node is powered up and "enters" the network. A higher level protocol controls the uniqueness and transmission of ENQ frames.

In addition to the frame fields, LocalTalk requires that the frame be FM0 (differential manchester space) encoded. Refer to Figure 5. FM0 requires one level transition on every bit boundary. If the value to be encoded is a logic zero, FM0 also requires a second transition in the middle of the bit time. The purpose of the FM0 encoding is to eliminate the need to transmit clocking information on a separate wire. With FM0 the clocking information is present whenever valid data is present.

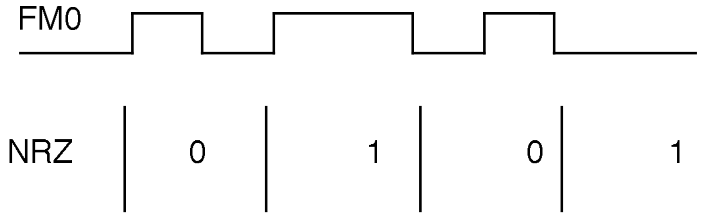

Figure 5: FM0 Encoding Method

FM0 is not often used in WAN-based protocols. The MC68302 contains no on-chip FM0 capability, thus the encoding and decoding of FM0 must be integrated into the hardware.

We can then summarize the tasks ofthe hardware that should be integrated into the LA device. These tasks ardisted in order of decreasing complexity:

(1) FM0 data clock encoding/decoding,
(2) The preamble synchronization sequence,
(3) The postamble frame abort sequencing, and
(4) Providing the physical transceiver interface.

Now we discuss the solution tothese problems as embodied in the LA.

THE LA DEVICE DESCRIBED

Figure 6 shows the block diagram of the MC68195 LocalTalk Adaptor (LA) device. The LA provides all of the digital logic necessary to interface the MC68302 integrated multi-protocol processor (IMP) to LocalTalk. The LA device consists of four blocks: acommon clock, a reset block, and two serial adaptor blocks. Each serial adaptor is identical.

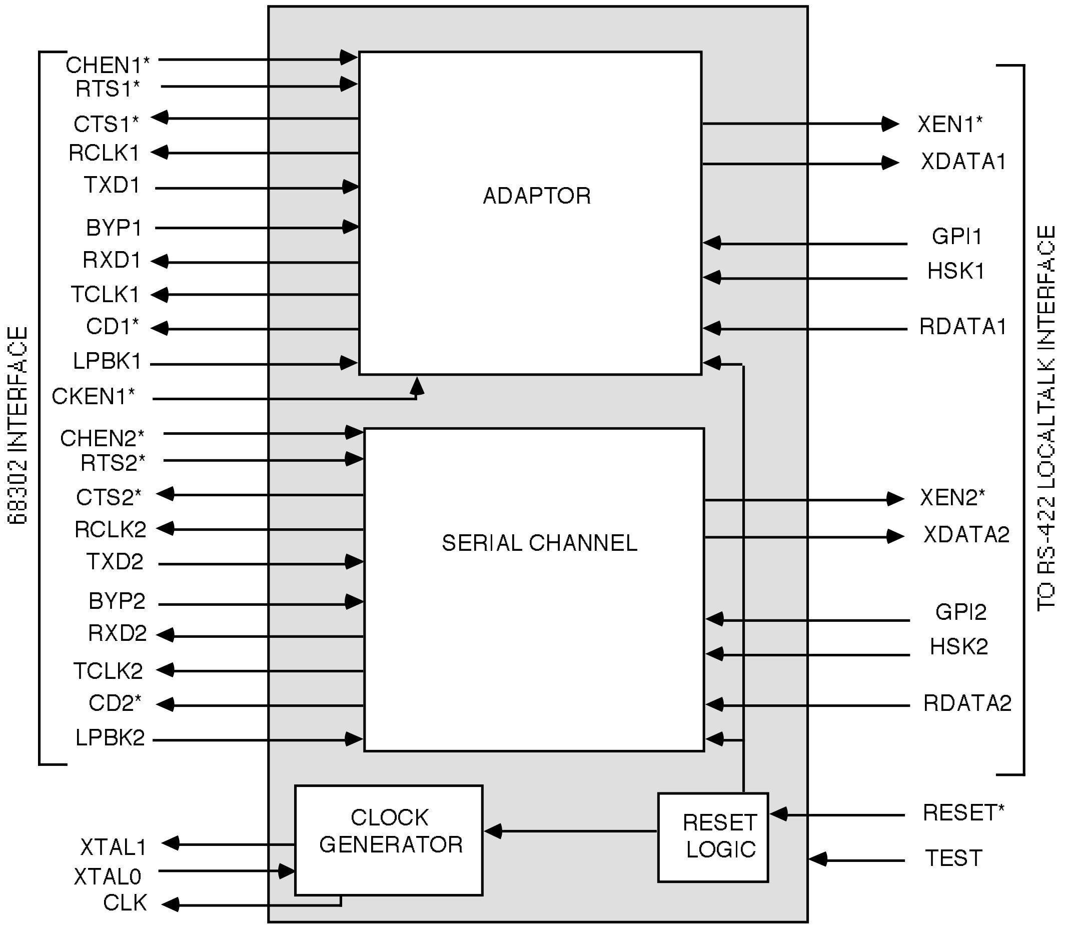

Figure 6: LA Block Diagram.

The common clock block consists of a crystal oscillator operating at 2.304 MHz, giving a 230.4 Kbps data rate for LocalTalk. Higher clock rates are also possible, giving data rates up to 2.5 Mbps. A CMOS level clock may bedriven into the crystal input, if the oscillator function is not desired. The oscillator's output is fed to the CLK pin. This permits LAs to be cascaded in applications where multiple IMPs and multiple LAs are used. This primary clock is internally divided by ten to produce a 230.4 KHz clock. This divided clock is output on the LA TCLK pins.

A LocalTalk bypass capability is provided via the pins BYP1 and BYP2. When BYP is asserted (high), the specified channel disables its LocalTalk encoder/decoder. Data and controlis then passed transparently through the LA. The primary purpose of bypass mode is to allow software selection between non-LAN (e.g. UART, or USART) and LocalTalk operation, similar to what is currently implemented in the Macintosh computer family.

In addition to the above bypass capability, each channel can also be disabled via CHEN1 and CHEN2. When enabled, the serial channels behave normally. When disabled, all of the LA outputs that goto the IMP (TCLK, CTS*, CD*, RCLK, and RXD) are three-stated. This disabled modeallows other network interface adaptors to connected to the IMP simultaneously with the LA device.

A loopback capability is provided via the pins LPBK1 and LPBK2. Loopback causes the LA transmit data (XDATA) to be internally connected to theLA receive data (RDATA) pin.

FM0 Data Clock Decoding

On the transmit side, the encoding of FM0 data is straight forward, only requiring a 2x transmit clock, which is available within the LA. We will thus concentrate on the receive side only.

On the receive side, a number of functions are required: noise filter, squelch, all digital phase-locked loop (ADPLL), and finally data decode. See Figure 7 for the LA serial adaptor block diagram. The noise filter removes undesired transitions. The squelch circuitry detects incoming line activity. The ADPLL logic provides clock recovery. The data decode circuitry converts the recoveredFM0 data back to normal NRZ for output to the MC68302.

The noise filter uses a three-of-five majority technique to remove any received pulses of less than three clock samples. Thus, noise pulses of duration less than ≈ 868 ns (for LocalTalk) are filtered and removed from the incoming data stream. Thisimposes a slight delay of three 10x high speed clocks (or 0.3 bit time) to the incoming data.

The squelch block decides when the incoming data from the noise filter indicates activity. Any transition triggers a 1.5 bit time retriggerable pulse stretcher. The output of thispulse stretcher is carrier-detect, CD*. CD* is output tothe MC68302 to indicate a busy media (refer to transmit description above). The pulse stretcher onthe CD* pin, allows the MC68302 to see a constant level on the CD* pin during the synchronization sequence. This prevents additional interrupts on changes in the CD* pin state that could affect software performance. CD* is also used internally by the receive block to activate the ADPLL.

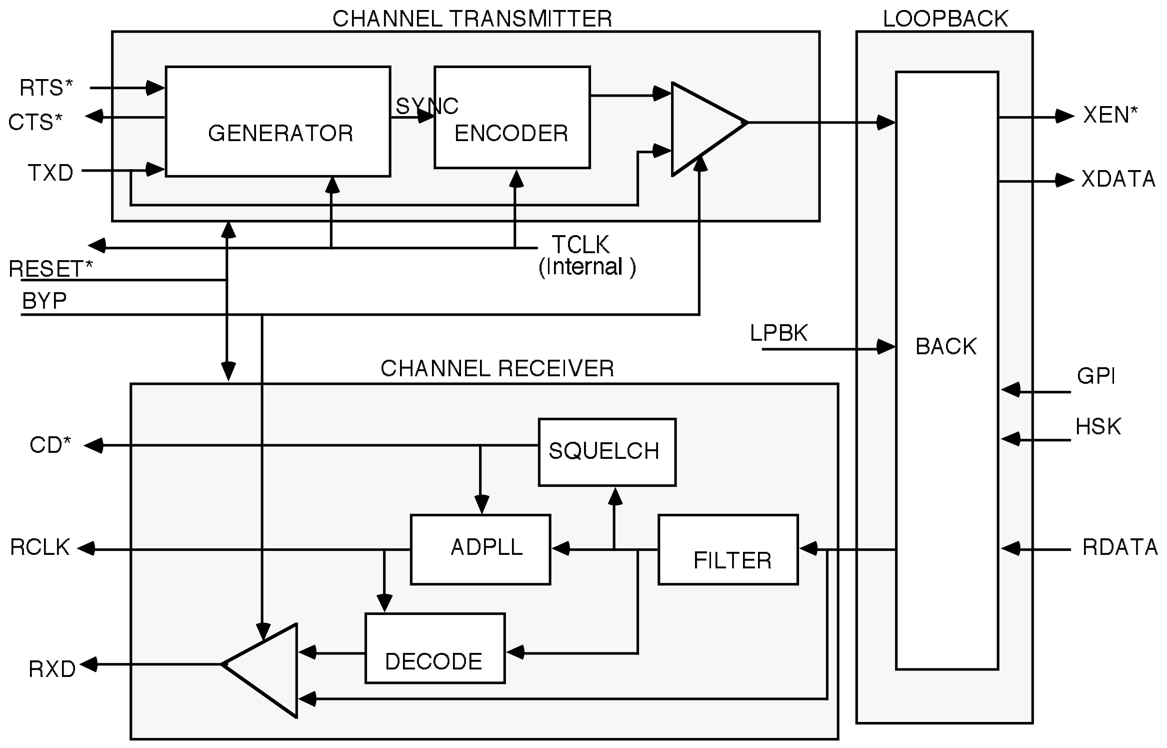

Figure 7: Serial Channel Block Diagram

The ADPLL uses an all-digital-PLL with arithmetic decode and 10X over sampling to recover an appropriate clock from the incoming data stream. This gives ≈ 860 ns of peak-to-peak edge jitter tolerance. This allows network lengths in excess of 1500 meters if signal amplitude levels and round-trip-delay considerations could be met (i.e. the LA will not be the limiting factor in network length). LocalTalk has a ±1% clock tolerance specification. Thus, the data rate can vary from 228.3 Kbps to 232.9 Kbps. Clock tolerance can be treated as a jitter component, giving an effective jitter tolerance of ≈ 850 ns peak-to-peak in LocalTalk.

The ADPLL nominally produces a recovered clock that is a nominal 230.4 KHz square wave, output to the MC68302 via the RCLK pin. Any received jitter is passed on to the MC68302. Thus, while the received clock consists of 2.17 μS pulse widths nominally (230.4 KHz of fifty percent duty cycle for LocalTalk applications), the recovered clock pulse width can vary from 1.30 μS up to 4.34 μS. (Actually the pulse width may be as low as 434 ns (i.e. one high speed clock in LocalTalk) at the start of the frame, as initial lock is acquired). The ADPLL will always acquire lock within two bit times.

The recovered clock is used to synchronize and to decode the received data stream. The decoded data bit is output to the MC68302's RXD pin. One bit time of delay is encountered in the data decode block.

The Preamble Synchronization Sequence

The LA is responsible for transmitting the synchronization sequence prior to each frame (see Figure 8). In the LA, this sequence consists of enabling the RS422 driver with FM0-encoded logical ones for two bit times (internal state SYNC1), followed by the disabling of the driver for three bit times (internal state SYNC2), followed by the first bit of the frame (D1). Data in this context means "that which is supplied from the MC68302". So the first "data" bit that the MC68302 actually provides is the first bit of the opening flag.

The implementation of the synchronization sequence takes advantage of an existing characteristic of the MC68302. When the MC68302 is transmitting HDLC data, the first bit of the HDLC frame is guaranteed to be sent 4 clocks after CTS* the rising TCLK edge that samples CTS* as low. The LA's synchronization pattern is 5 bits long. Thus, all the LA need do is delay the assertion of CTS* one TCLK after it recognizes RTS* from the MC68302. In this way the synchronization sequence fits perfectly within the startup delay of the MC68302.

Although the synchronization sequence needs to be sent only preceding RTS control frames, the LA will send the synchronization sequence before each frame. This eliminates the IMP having to instruct the LA asto the frame type. Sending the synchronizations sequence prior to each frame does notviolate the LocalTalk rules. The only disadvantage is an extra bit time of overhead per frame, as compared to what the MC68302 would provide on its own.

The Postamble Frame Abort Sequencing

After the closing flag at the end of the frame, the IMP deasserts RTS*, and begins sending a continuous stream of ones, until the next frame is transmitted. However, LocalTalk specifies that 12 to 18 ones must be sent, and then the RS422 driver must be turned off.

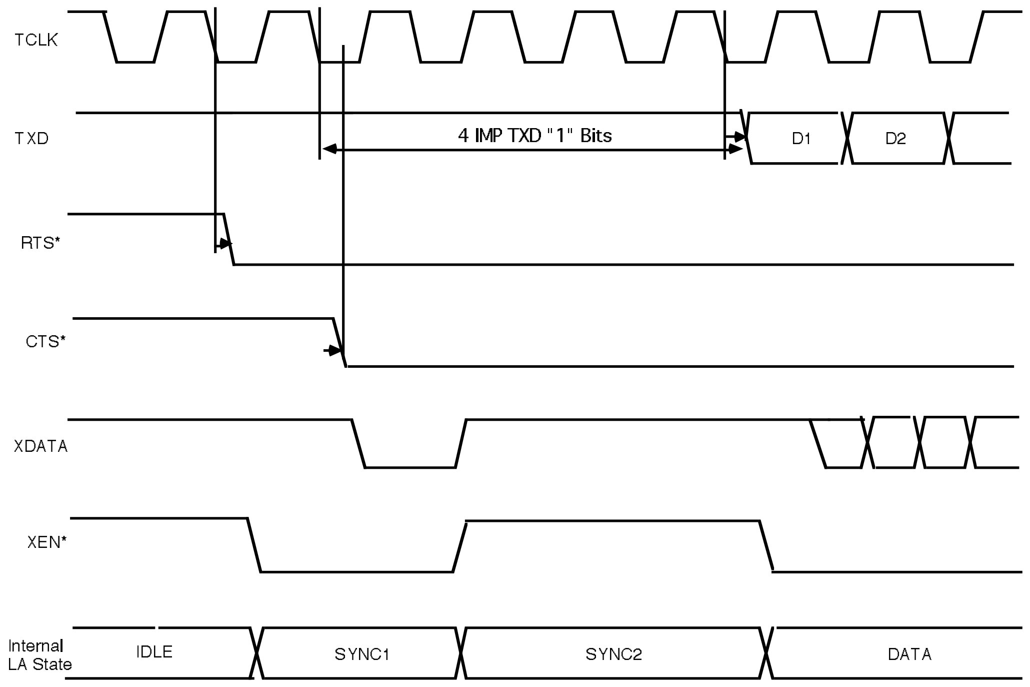

Figure 8: Transmitter Start-of-Frame Timing

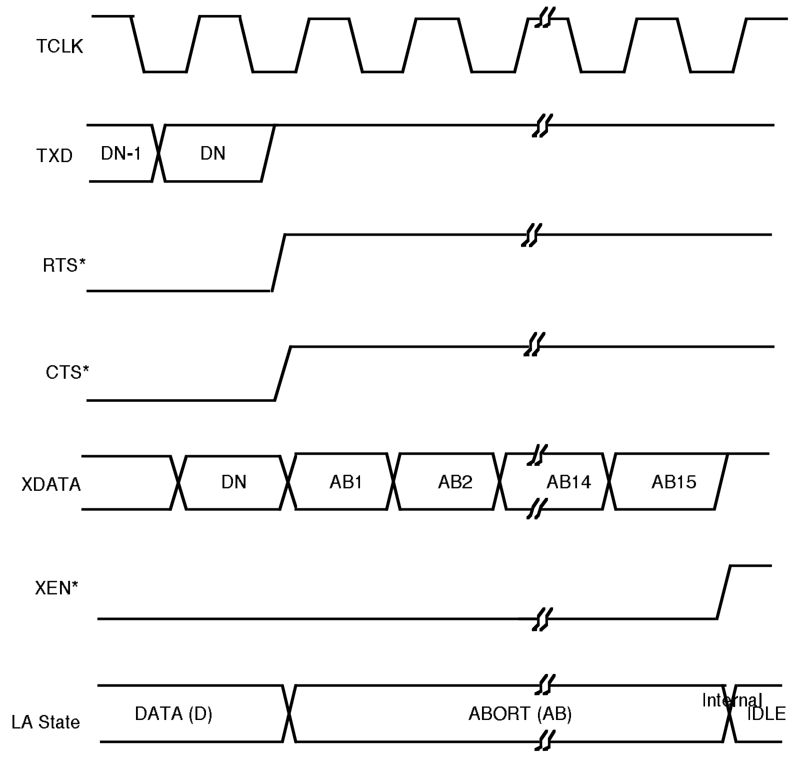

Figure 9: Transmitter End-of-Frame Timing

The LA meets this condition bywaiting for the RTS* line from the MC68302 to be deasserted, and then sends fifteen ones, which are FM0 encoded (see Figure 9). During this fifteen bit time period, the incoming MC68302 RTS* line and TXD lines are ignored. (The MC68302 must be configured in software, not to send a new frame too quickly. Doing so would violate the LocalTalk protocol inter-frame gap (IFG) rules).

Providing the Physical Transceiver Interface

The XEN* line in figures 8 and 9 show the timing of the transmitter enable. The RTS* pin from the MC68302 may be directly used as the transmit driver enable, except in two instances. TheLA must negate XEN* during the preamble sequence (figure 8), and must continue to assert XEN* for 15 bit times after the RTS* pin from the MC68302 goes low.

SOFTWARE ISSUES

To complete the job of adapting the MC68302 to an AppleTalk LAN, the software drivers had to be written to meet the AppleTalk LAN requirements. The developed software is called LAbug. LAbug consists of two distinct layers: application and LocalTalk Link-Layer Driver. Refer to figure 10. The LocalTalk driver consists of routines to initialize itself, to transmit packets, and receive packets. The application layer consists of a command interpreter and various command level routines to exercise and validate the link layer.

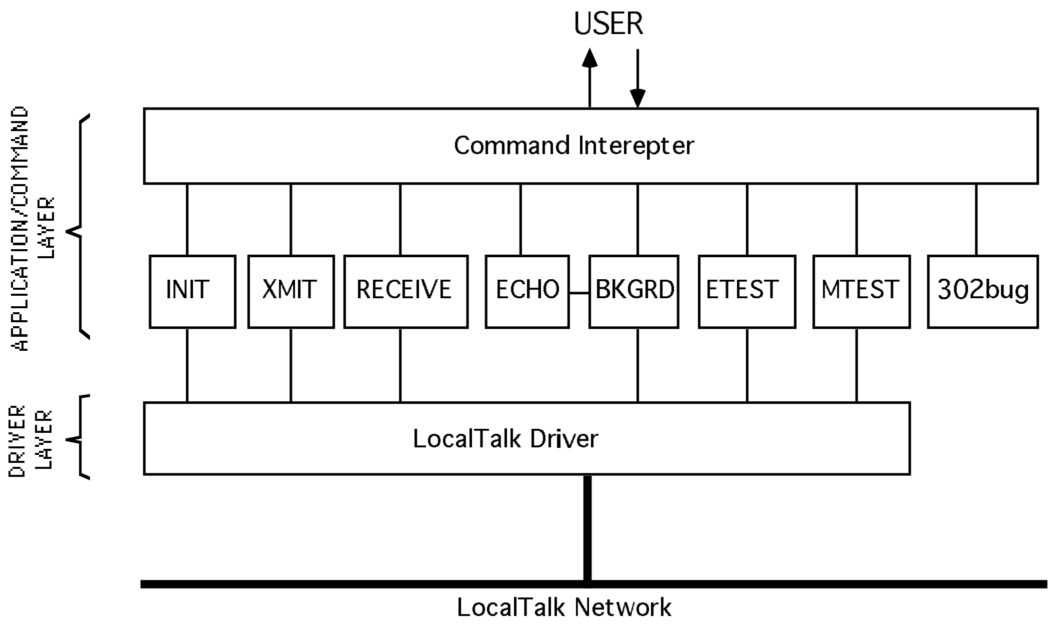

Figure 10: Software Structure.

The LocalTalk driver is patterned after the psuedocode driver given in the back of *Inside AppleTalk, Version 2.0* [R-5]. The link access protocol is identical and the presented higher-layer interfaces have been preserved. An additional receive interface is optionally available to eliminate the need for higher layers to copy the received data.

The application layer consists of a command interpreter that runs on top of 302bug that already exists on the ADS302 board. This command interpreter polls the user for commands and determines if the enteredcommands are LocalTalk application commands or not. If they are, the command routines are executed. Otherwise, the commands are given to 302bug for execution. Thus, while running LAbug, all of the 302bug commands are available.

The application layer provides a portable mechanism to validate the LocalTalk Driver. This driver module is called LAdriver.

Driver

LAdriver implements all of the LocalTalk link layer. For the most part, LAdriver is a port of the link-layer driver given in Appendix B of *Inside AppleTalk* [R-5]. The driver has been translated from Pascal into "C" and ported to operate on the MC68302 processor. Figure 11 shows the various routines that comprise LAdriver and their inter-relationship. LAdriver is completely documented in [R-2]. Where possible, the names have been preserved.

The driver is almost identical to its Pascal ancestor. The receive routines have been modified to be interrupt driven, instead of polled. Also, a second receive interface, via the routines: RPacketAvail, RPacket, GetRBuf, and FreePacket, is provided that allows packet buffers to be passed directly to upper layers without requiring that the received data be copied. Also, the ReceiveFrame routines have been made into an interrupt handler instead of the polled one given in [R-5].

In the figure solid lines represent traditional procedure calls and dashed lines indicate the interrupt and non-interrupt interface. Arrow heads indicate the principal direction of data flow.

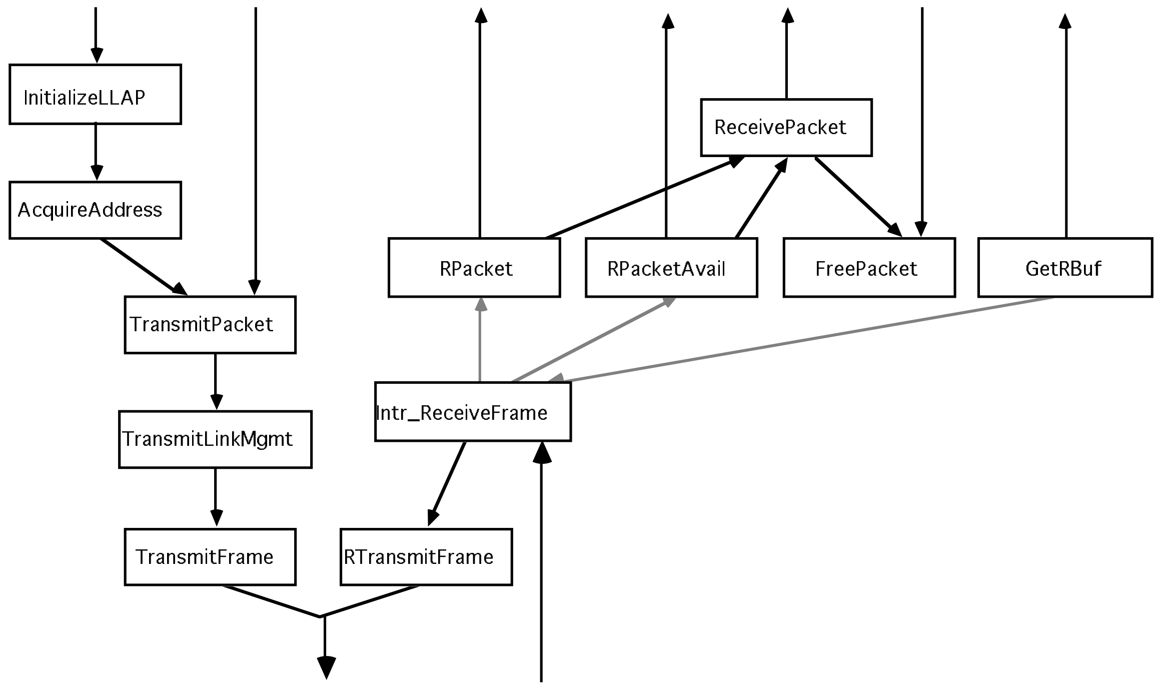

Figure 11: LocalTalk Driver Structure.

Software and Configuration Issues

LAdriver takes advantage of a number of configuration possibilities of the MC68302. They can be grouped into three major categories:

(1) Meeting the required inter-frame gap (IFG) timing by synchronizing the M68000 core to the on-chip RISC processor,
(2) Dealing with the half-duplex nature of the protocol on a full duplex device, and
(3) Buffer management including the generation of "minute-man" frames to speed transmission.

The first and third deal with making the software fast enough to meet the LocalTalk specification and the second deals the inherent difficulty of using a full-duplex device (the MC68302) to implement a completely half-duplex protocol (LocalTalk). Each of these will be discussed in greater detail below.

Software Performance. The MC68302 actually contains two processors: a 68000 - which runs user software, and a RISC processor - called the CP that actually controls and moves data to/from each of the serial channels and other internal hardware. Refer to [R-3]. In order to meet the required IFG these two processors need to be synchronized. The CP operates in a polled environment and only periodically checks that a packet needs to be sent for any given SCC. By default, each SCC and its corresponding descriptors are checked once every sixteen transmit clocks (TCLK). For LocalTalk this polling interval is 69.4 μS. This default value is unacceptably high for the LocalTalk environment.

Two features of the MC68302 were used by LAdriver to bring this default polling interval down to an acceptable level. Refer to Figure 12. By disabling and then immediately re-enabling the transmitter of the SCC near the beginning of the receive interrupt handler synchronizing the CP with the MC68000 interrupt handler. Disabling the transmitter causes it to reset. Re-enabling causes the transmitter to send the IDLE sequence (sixteen ones in HDLC mode) before beginning normal operation. Thus by strategically placing this disable/enable sequence, the CP's SCC can be synchronized to the interrupt handler such that the poll will occur shortly following the frame being made ready for transmission (by its ready bit in the descriptor being set). A second feature can be used to decrease the time between subsequent polls to 34.7 μS. By instructing the CP to transmit flags during this critical time, the CP polls the SCC every eight bit times, as these flags are generated in the CP. Obviously, this second feature is not needed if the base interrupt handler takes less the initial sixteen TCLK polling time; though it is still useful for normal transmissions (just not those from the interrupt handler).

Half-Duplex Protocol. Another issue handled by the LAdriver is the inherent complications associated with operating a half-duplex protocol with a full-duplex device. The MC68302 is primarily a general purpose WAN controller. Most WAN protocols, i.e. X25-LAPB, are full-duplex in nature. Transmission and reception occurring simultaneously and the protocol stack keeps up with the data the best that it can. LocalTalk is a half-duplex protocol. Data frames are sent over the media only when both entities are ready. Readiness is determined through RTS and CTS frames which are also sent only under very specific guidelines. Due to the rigidness and the half-duplex nature of the protocol, much error checking is performed by the link-layer software. If frames are received at incorrect times, recovery procedures are initiated usually resulting in the driver turning itself off until it is reset by higher level software.

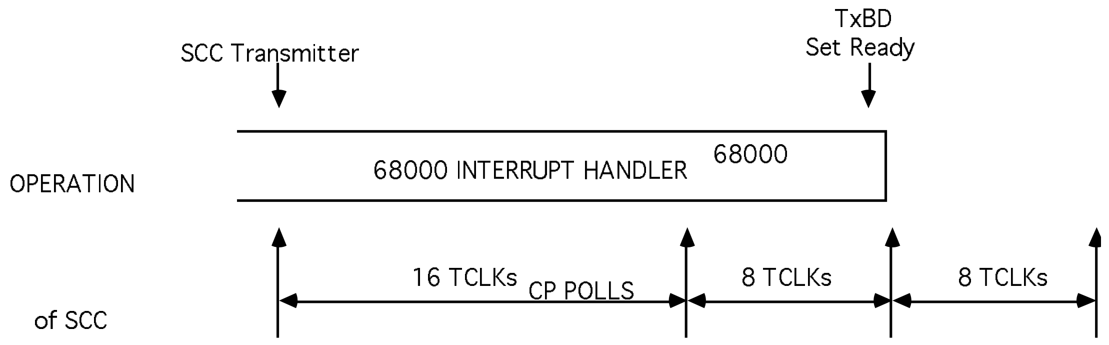

Figure 12: 68000 and CP Synchronization.

One area that this is significant for the MC68302 is that a node is allowed to receive its own transmissions. If a node receives its own frames, that node's protocol stack assumes that another node exists on the network using the same address and treats this as a fatal error. For the MC68302 this potential problem is simple to overcome. LAdriver's transmit frame procedures simply disable the receiver during transmission. The only complication is that, since the receiver may have been receiving another frame at the time of its being disabled (perhaps a collision is taking place), the receiver's internal state machine may be in the improper state. An "Enter-Hunt" mode command is given after re-initializing the receiver at the end of transmission, to insure proper operation.

Hot Frames. While the half-duplex nature of LocalTalk limits the network's effective throughput, it allows some creativity in the use of the associated buffer descriptors. The ring nature of the buffer descriptors is only useful in full-duplex protocols while a queue of both transmit and receive frames is possible and desired. In half-duplex protocols, no other frames may be transmitted or received over the network until the current one has been dealt with. Thus, only one descriptor can usually be used.

Due to the flexibility of the MC68302 these "un-usable" descriptors can actually be used to decrease transmit latency. Refer to Figure 13. Since LocalTalk only has five frame types to transmit, "minute-man" hot frames may be stored, ready for use in descriptors one through six. These ready frames already have the source address and control fields stored in them. To transmit one of these hot frames; the destination address is inserted and the CP is instructed to send that particular frame. This is a total of two one-byte transfers. To instruct the CP to send a frame from a different descriptor than the next one in the ring, the CP's internal buffer pointer is modified. This modification is possible through the dual-port RAM that the 68000 and CP share.

LAdriver never sets the ready bit for descriptor zero, and descriptor seven is unused. The control frames operate as described above. Data frames are similar but are a little different. The header portion (destination and source address, and frame type) are treated as a hot frame. Only the descriptor for the header is configured to "fall-through" to the next descriptor where the actual data portion of the frame is placed.

KEY QUESTIONS TO ASK

The above discussion of hardware and software lead to some generalizations in adapting WAN controllers to LAN environments. They can be summarized in terms of the questions to ask yourself or the manufacturer of the WAN-type equipment for which it is desired to interface to a LAN.

1. What is the data encoding method used on the LAN (if any)? If the WAN controller doesn't support it, some additional hardware will be required.

2. What is the LAN framing format? Often it is the LAN preambles and postambles that will cause trouble for WAN controllers. The solutions here may be solved by configuring the WAN controller in a different way, or may require some external hardware, or both. The CRC technique of the WAN controller should also be checked, to see if it matches what is required by the LAN.

3. What are the characteristics of frame transmission latency on this device? In the development of LAN firmware, it is usually advantageous for the controller to start the frame's transmission as soon as possible, once the decision to begin transmitting is

made. Find out what the transmission latencies are, and use them in the interframe gap calculations. In the case of the MC68302 adaption, the latency issue was solved with an interesting register configuration, and smart software.

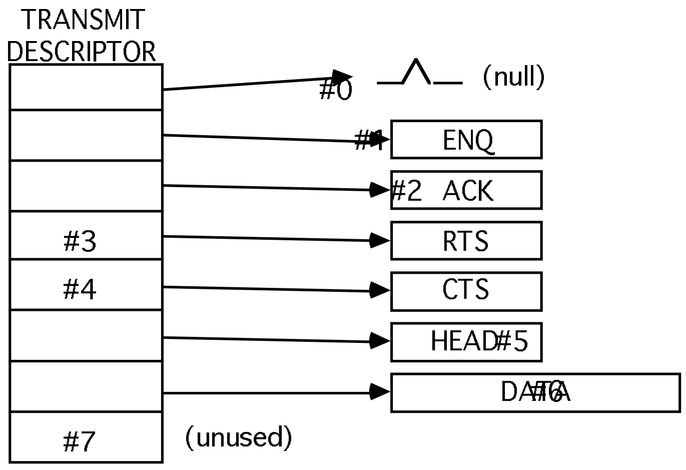

Figure 13: Transmit Descriptor Usage.

4. Can the WAN controller ignore its own transmission? Since most LANs are half duplex, it is important to be able to disregard one's own transmissions. This capability can be included in the external hardware if necessary by forcing the receive pin to an idle state during transmissions, but perhaps the WAN controller contains a method to disable its receiver in software. Another possible solution is to solve the problem in the software driver by identifying one's own frame in the receive queue and disregarding it. Depending on the protocol, this is not always feasible. In the case of the MC68302 adaption, the decision was made to usethe "disable the receiver" method.

5. What processor overhead is required during frame reception (and transmission)? In this category, less is always better. This is an area where the MC68302 excels, since it can wait until a correctly received frame arives in full, before it needs to notify the processor to take action.

REFERENCES

[R-1] Motorola, Inc. *Create an AppleTalk with an MC68302 and MC68195*. Austin, Texas: Motorola, Inc. 1990. (Included in Appendix D of [R-3] below).

[R-2] Motorola, Inc. *LocalTalk Adaptor Software Driver, Version 1.0* Austin, Texas: Motorola, Inc. 1991.

[R-3] Motorola, Inc. *MC68302 Integrated Multi-Protocol Processor User's Manual.* Rev. 2. MC68302UM/AD REV 2. Austin, Texas: Motorola, Inc. 1991

[R-4] Motorola, Inc. *MC68195 Datasheet.* MC68195/D. Austin, Texas: Motorola, Inc. 1991

[R-5] Sidhu, Gursharan S., Richard F. Andrews, and Alan B. Oppenheimer. *Inside AppleTalk*. 2nd ed. Reading, Massachusetts: Addison-Wesley Publishing Company, Inc. 1990