CALIFORNIA PATH PROGRAM
INSTITUTE OF TRANSPORTATION STUDIES
UNIVERSITY OF CALIFORNIA, BERKELEY

Design and Implementation of Digital Radio Communications Link for Platoon Control Experiments

Wei-Yi (William) Li
University of California, Berkeley

California PATH Research Report
UCB-ITS-PRR-95-2

This work was performed as part of the California PATH Program of the University of California, in cooperation with the State of California Business, Transportation, and Housing Agency, Department of Transportation; and the United States Department of Transportation, Federal Highway Administration.

January 1995

ISSN 10551425

Design and Implementation of Digital Radio Communications Link for Platoon Control Experiments

by Wei-Yi (William) Li

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II.**

Approval for the Report and Comprehensive Examination:

Committee:

Pravin Varaiya

Professor Pravin Varaiya
Research Advisor

25 June '93

(Date)

* * * * * * *

[illegible]

Professor Jean Walrand
Second Reader

6/25/93

(Date)

Table of Contents

Design and Implementation of Digital Radio Communication Link for Platoon Control Experiments

Abstract

This report discusses the design and implementation of inter-vehicle communication systems used in the longitudinal control platoon experiments conducted by PATH (Partners for Advanced Transit and Highways) in ITS (Institute of Transportation Studies) at the University of California at Berkeley. In the first section of this report, the role of vehicle communication in the platoon experiment is discussed. In later parts of this report, the inter-vehicle communication links implemented in both the two and the four vehicles platoon experiments are examined. Lastly, this report investigates some of possible future improvements of the communication link used for platoon control experimentation.

1. Introduction and Background Theories

1.1. Platoon Control and the Need of a Communication Link

The working hypothesis underlying PATH's experimental effort is that the capacity of a freeway can be increased by organizing freeway traffic into platoons. A platoon is a group of tightly spaced vehicles travelling using a vehicle follower control system[1][2][3][14][15]. The system implemented in the platoon experiments deals primarily with longitudinal control. The communication setup that is discussed here is employed to convey information within a platoon. However, inter-platoon communication required for coordinating maneuver may be developed from current work with little modification.

The longitudinal control algorithm for automating platoon control was developed by Professor Hedrick and his associates in the Department of Mechanical Engineering, University of California at Berkeley. It is based on a sliding control method appropriate for the non-linearities existing in vehicles[4]. Computer analysis of platoon control, using the above mentioned algorithm, indicated that control error amplification may occur if the platoon is controlled without a communication link. This problem is documented in the paper ***Longitudinal Vehicle Controller Design*** for *IVHS* (Intelligent Vehicles/Highway System) by Hedrick, et. al [4]. They conclude that control error amplification may be avoided by communicating the lead vehicle (first vehicle in the platoon) information, in addition to the preceding vehicle speed and acceleration which is already required by the control algorithm, to all of the vehicles in the platoon. Thus a communication link within the platoon became an essential part of platoon control.

1.2. Background on Computer Networking

The main responsibility of the communication link is to provide a reliable communication channel for vehicles involved in the platoon experiments. In an effort to develop the communication link in an orderly fashion, the different responsibilities of the link are organized into different layers of abstraction. These layers are modeled according to the International Standards Organization's (ISO) Open System's Interconnection (OSI) Reference Model [10].

The IS0 has specified seven layers in its OSI reference model. (see Figure 1) For the radio link, only the first three layers are implemented. These layers are listed as follows:

Layer 1: Physical Layer

The physical layer concerns itself with the communication devices needed for the transmission of bits over a communication channel. The design issues at this layer deal largely with the electrical and interfaces and the transmission medium underneath the physical layer. We avoid these considerations by incorporating commercially available communication and computing products in this project. For this project, the physical layer consists of the radio transceivers, the communication interface controllers, and the host computers.

Layer 2: Data Link Layer

The responsibility of the data link layer is to implement reliable packet transmission by using physical layer services. This layer is also responsible for regulating the access to the shared radio channel. Functions that are performed by this layer are implemented by the link layer protocol unit on the communication interface board and the communication protocol which is implemented in hardware.

Layer 3: The Network Layer

The network layer deals with routing of data packets from one location to another on the network. In addition, this layer provides congestion control. For the radio communication network implemented in this project, however, routing control is unnecessary as all data are communicated by broadcasting to every station on the network. In this project, the network layer filters received messages such that only the correctly designated messages are passed to the platoon controller. The functionality of this layer is achieved by the communication manager residing on the host computer.

Layer 4: The Transport Layer

This layer provides the function of formatting messages passed from the session layer to the network layer and fragments smaller packets out of the original message if necessary. Proper reconstruction of the original message from the received fragments is assured. However, since, in the platoon experiment, data messages are not fragmented for transmission, this layer is not implemented.

Layer 5: The Session Layer

The session layer establish dialogues and supervise between stations on the network. The session layer of different stations must agree on a common set of rules before a connection can be established between them. This layer is not implemented in this project.

Layer 6: The Presentation Layer

The presentation layer converts session layer messages to format that is understood by the application layer, and vice versa. In this project, since each network station has identical setup, compatibility is not an issue. This layer is not implemented.

Layer 7: The Application Layer

This layer defines the set of protocols for the user application to communicate with its peers. For this project, the application layer defines the format of the data messages. The generation and processing of these messages occur in the control mechanism.

2. Requirements and Designs

2.1. Platoon Control Experiments

The objective of the platoon control experiments is to demonstrate the viability of longitudinal platoon control using currently available technology. By longitudinal control we mean the automatic control of the spacing between vehicles. The lateral (directional) controls are made by the human driver in the experimental vehicles. This project investigates the viability of inter-vehicle and intra-platoon communications using existing communication technology.

The experiment is to be conducted in two phases. In the first phase, the platoon is composed of two vehicles: one of which is human-controlled, and the other is under automatic control. The human-driven vehicle is the "lead" vehicle. The computer-controlled vehicle is the "following" vehicle.

A typical two-vehicle platoon experiment is conducted as follows: the experiment starts with both vehicles being manually driven. The following vehicle switches into the automatic control mode while both vehicles are in motion. Using the information from the communication and sensors, control on the following vehicle attempts to maintain the desired distance from the lead vehicle throughout the duration of the experiment.

In the second phase, the two-vehicle platoon is replaced by a multiple automated vehicle platoon. Here, the control algorithm must account for the additional interactions between different numbers of computer-controlled vehicles in the platoon. Control experiments for this stage are conducted in a fashion similar to the two-vehicles platoon experiments, where automated control is initiated when all vehicles reach the desired speed and separation from the preceding vehicle. The increased platoon size, however, necessiates more complex control and communication mechanisms, as discussed later in this report.

2.2. Responsibilities of the Communication Link

The communication link is the information transport mechanism for the platooning vehicles. This link is expected to perform three tasks described below in the platoon control experiments:

1. Data Transport: The control mechanism requires the communication link to transport data from each vehicle participating in the platoon. This data includes the vehicle velocity, acceleration, and a timestamp identifying when the velocity and acceleration data are sampled. The lead vehicle is distributed to all vehicles in the platoon. Data from all other vehicles is sent to the vehicle immediately behind originating vehicle. Since a broadcast communication is selected, the link must also be able to provide the necessary arbitration for efficient access to the communication channel.

2. Timing Requirement: Tii restrictions are placed on the data carried by the communication link: the link must deliver data within a predefined amount of time. This is a direct consequence of the real-time nature of the platoon experimeme. Thus, to ensure the generation of real-time control, an upper limit is placed on the amount of transport delay that is allowed by the system. The link is declared to be malfunctioning if the limit is exceeded.

3. Link Monitoring: While not required by the control algorithm, it was decided that some protocol should be implemented into the communication link to facilitate link monitoring, allowing vehicles in the platoon to monitor the status and maintain the robustness of the link. The protocol enables most link failures to be quickly localized.

4. Control Synchronization: In addition to communication functions, the link is also used to maintain a synchronization of the sampling times for control among vehicles participating in the platoon for effective platoon control. It is recognized that we can take advantage of the periodic transmissions from the lead vehicle at the beginning of each cycle of control generation. That is, upon reception of this message, the clocks running in following vehicles are synchronized to the clock running in the transmitting vehicle, and all vehicles can start their control cycle simultaneously.

2.3. Design of Communication Link

The design of the communication link follows an evolutionary approach. At first, only a minimal set of functionalities are designed for the two-vehicle platoon link, since this experimental setup enables the link to ignore many multiple access problems. For multi-vehicle platoon experiments, however, the link design must address the problem of multiple access. In addition, the multiple vehicle link is expected to be more robust and provides more services. Thus, an extended version of the two-vehicle communication link, which answers many of the newly risen problems, is designed.

2.3.1. Link Topology

The topology of the communication link designed for platoon experiments is based on the token-bus architecture (IEEE Standard 802.4) developed by General Motors. A token-bus network physically connects all stations to a common communication medium. All network traffic is broadcast on the bus to all the stations. To ensure that only one station is allowed to transmit at a rime, no station may transmit until it has obtained the token (the symbol of the right-to-transmit). The token is passed from station to station in a logical order, which may or may not be related to the physical location of the stations on the network.

The token-bus network was developed as an Ethernet (IEEE Standard 802.3) alternative to implement transmissions with bounded delays. This is because in Ethernet, all stations on the network compete for the same communication channel, and frame collision is a frequently occurring phenomenon. Unfortunately, collision recovery on Ethernet is performed probablisticallg. Thus, it is possible for a station to wait an arbitrarily long time before it is permitted to transmit. The probabilistic nature of Ethernet makes it unsuitable for real-time applications, since these applications require a hard upper limit on the delay of the network. The Token-bus network, on the other hand, is designed to correct this problem. Here, the network only allows one station to transmit at any time, thereby avoiding all collisions. Thus, it is possible to place a deterministic upper limit on the delay of a token-bus network.

It is not our intention to implement a complete set of the 802.4 protocol, since it is much too complex for our purposes. The communication link design only incorporates a subset of this protocol. However, if additional functionalities are desired, they can easily be added to the existing link design.

2.3.2. Two-Vehicle Platoon Link Design

The initial link design tends to be simplistic in terms of both features and structure. The state diagrams for the communication link can be found in figures 2a and 2b. The aim of this design is to acquaint ourselves with various

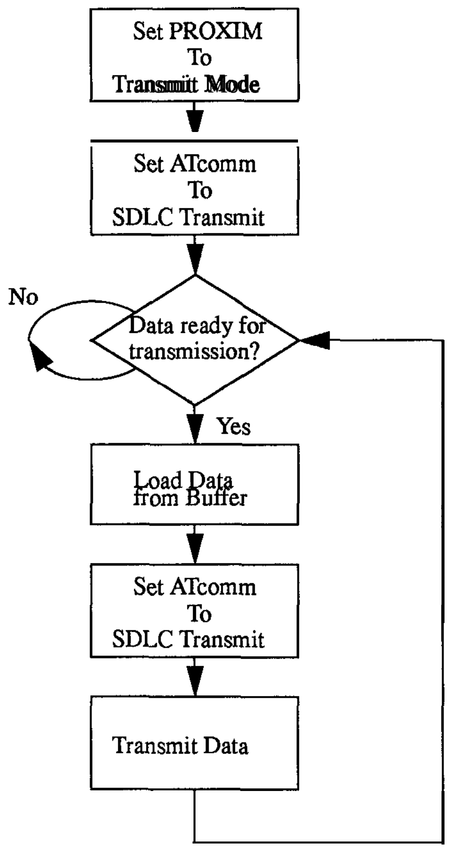

Set PROXIM To Transmit Mode
Set ATcomm To SDLC Transmit
No
Data ready for transmission?
Yes
Load Data from Buffer
Set ATcomm To SDLC Transmit
Transmit Data

Figure 2a. Communications State Flow Chart for Lead Vehicle in Two-Vehicle Platoon

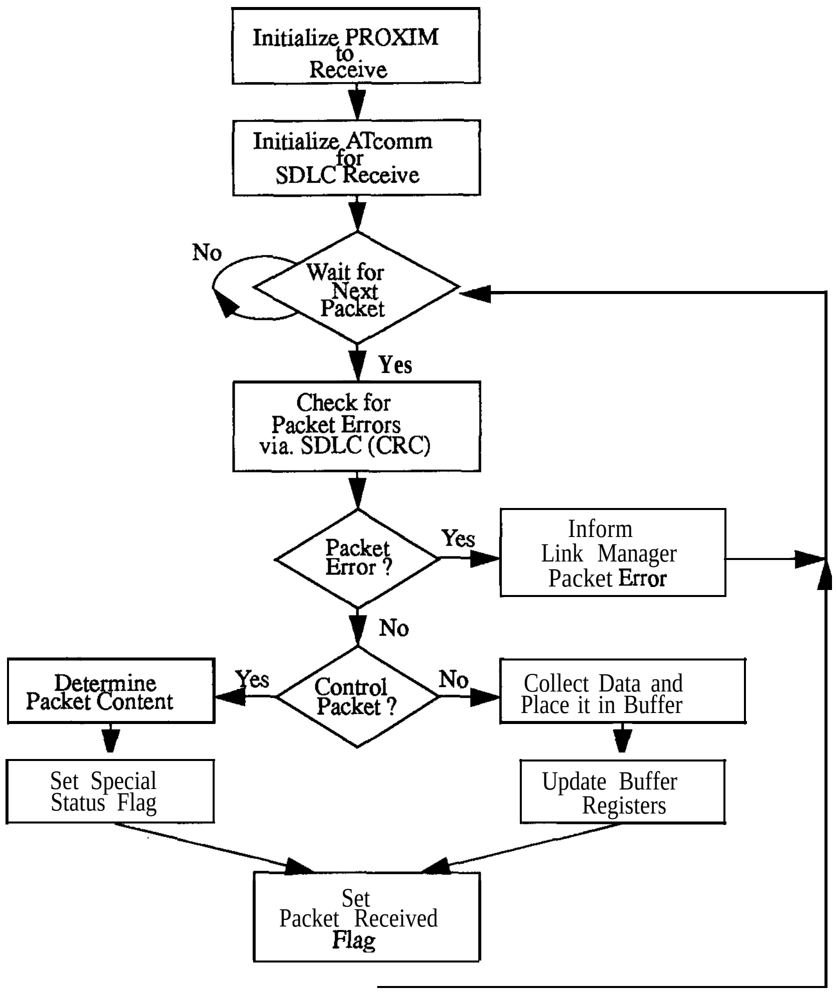

Initialize PROXIM to Receive
Initialize ATcomm for SDLC Receive
No
Wait for Next Packet
Yes
Check for Packet Errors via. SDLC (CRC)
Packet Error ?
Yes
Inform Link Manager Packet Error
No
Determine Packet Content
Yes
Control Packet ?
No
Collect Data and Place it in Buffer
Set Special Status Flag
Update Buffer Registers
Set Packet Received Flag

Figure 2b. Communications Flow Diagram for Following Vehicle in Two-Vehicle Platoon

hardware and software components that are used in this project, while providing the necessary data transport services for the experimental vehicles. Each station on the network is programmed to implement minimal functionality, enough only for the proper execution of the control algorithm.

For these experiments, the control mechanism requires the link to transport lead vehicle information to the following vehicle. The lead vehicle, however, does not require any data from the following vehicle, since it is human driven. Therefore, the lead vehicle is programmed to only perform transmissions and the following vehicle is programmed only to receive messages, establishing an unidirectional channel.

Although easier to implement, the unidirectional link design compromises the integrity of the network: it blinds the lead vehicle from any situation developing in the following vehicle. If, for some reason, the control mechanism fails on the following vehicle, the responsibility of relating this event to the lead vehicle resides solely on the driver in the following vehicle. As a consequence, network management is not implemented.

All transmissions on the link fall into one of the two categories: data and status messages, where a message denotes a complete transmission. In this experiment, all data messages are identical in format, where each message contains the lead vehicle's identification, speed, acceleration, and timestamp when the data is sampled relative to the beginning of the experiment. In addition to data, the network also transports status messages among the vehicles. The only status message the current design specified is the "Alert To End" message, which is sent from the lead vehicle to inform the following vehicle that the experiment is over. Other messages can be easily added into the design.

The link design also takes steps toward link failure management. In the case of a link failure, the subsequent break in timing synchronization activates a time-out checking routine in the following vehicle, where the routine keeps track of the time elapsed while the receiver is idling, without receiving any message. The link, meanwhile, attempts to reestablish the connection before the time-out signal is generated. If this is successful, the link resynchronizes itself with the next data messages from the lead vehicle, and the link is re-established. If the idle time exceeds a predefined quantity (the time-out threshold), a time-out signal is generated to alert the resident control mechanism. When the time-out alert is detected, the controller immediately recognizes that the link has been severed, and it should take the appropriate measures.

Error-detection procedures are not implemented in the two-vehicle communication link. The design assumes that all such operations are performed by the facilities included *on the* communication hardware. If an error is detected by the hardware, the entire message in which the error occurred is automatically discarded, enabling the link to provide errorless communications between the vehicles. A reliable link, where all the transmitted data are received and verified, can be implemented in software on top of the existing link design. In addition to error-detection, the design also omits implementing error-correction into the link's operations. However, extensions can be added to the existing software to enable elaborate error detection schemes to be integrated into the link.

In the platoon experiments, the control algorithm is designed in such a way that the control signals are generated periodically. Since each control calculation requires one set of transmitted data, the link is designed to provide the necessary data transport for all vehicles within each period of control calculation. From computer simulations, it is determined that a period of 55 ms is adequate for the link to accomplish its goals. Therefore, within a single 55 ms interval, the link is designed to perform the following: first, all relevant data are sampled using a bank of vehicle sensors on the lead vehicle. When the vehicle data is fully acquired, the controller on the lead vehicle polls the communication link manager for permission to transmit. If the communication subsystem is ready to transmit, the communication manager will accept the data and transmit it to the following vehicle. The following vehicle processor pends on this transmission. The reception of this message signals the start for a new cycle in following vehicle's control signal generation. When the message is received and parsed, the controller on the following vehicle activates the data sampling process to acquire data from the local vehicle sensors. Together with the lead vehicle data, the control process is able to compute and generate the corresponding actuation signals. Note that the following vehicle does not requite a loop timer, since its timing is synchronized with the lead vehicle. This sequence of processes is then repeated until the end of the experiment. The experiment terminates when either of the vehicles exits the experiment, or when the communication link is lost. The decision to terminate is made by the human drivers who are driving the test vehicles. Lastly, if there is any time left in the 55 ms period, the vehicles idle for the control loop timer to expire, at which time the lead vehicle starts the next loop.

In order for the control mechanism to function properly, all of the above must be completed within 55 ms. If any of this is not performed within the 55 ms loop time, a time-out is generated. The particular handler that is activated by the control process depends on the number of consecutive time-out detected. When the time-out condition lasts for more than 3 seconds, the experiment is automatically aborted. The 3 second duration is arbitrarily selected; further studies should be conducted to insure the safety of the platoon.

2.3.3. Multiple Vehicle Platoon Link Design

A more robust communication link is designed and implemented for the multiple vehicle platoon control experiments. The state diagrams for the multi-vehicle platoon communication link is shown in figures 3a and 3b. Here, additional vehicles are added to the existing two vehicle platoon structure, resulting with a platoon having a variable number of vehicles. Henceforth, the lead vehicle is designated as vehicle 1. This vehicle is operated by a human driver throughout the entire experiment, and no automated control is applied to that vehicle.

The addition of vehicles significantly increases the complexity of the link design. The increase in complexity is mostly due to various concerns dealing with multiple access over a shared medium. A significant amount of effort was spent in the development and the optimization of various anti-collision measures while maintaining real-time transport capabilities.

In the multiple vehicle platoon control algorithm, the communication link is given the task of transporting data messages from the lead vehicle to the following vehicles and between each vehicle and the vehicle that immediately precedes it. The contents of the data message in the multiple vehicle link design bears an almost identical format to that of the two-vehicle platoon link: the message contains the transmitting vehicle's speed, acceleration, and the time at which this data was acquired relative to the beginning of the experiment. In addition to this information, the multiple vehicle link design also calls for identification of the transmitted message which identifies the source of the message in a multiple access environment.

The link design calls for a token-bus structure, where the right-to-transmit token is passed sequentially along all vehicles in the platoon. The current iteration of transmission ends when the designated last vehicle in platoon completes its transmission. The token is then returned to the lead vehicle, thereby starting the next loop of transmissions.

In accordance with the token-bus design, the lead vehicle is given the right-to-transmit token at the start of the experiment. To prepare for transmission, the lead vehicle collects the necessary vehicle data and transmits it over the link. Due to the communication hardware used, the channel is broadcast in nature, and all vehicles receive this transmission. In addition, a successful broadcast from the lead vehicle also constitutes a token pass to the vehicle behind it, thereby giving the next vehicle the right to transmit.

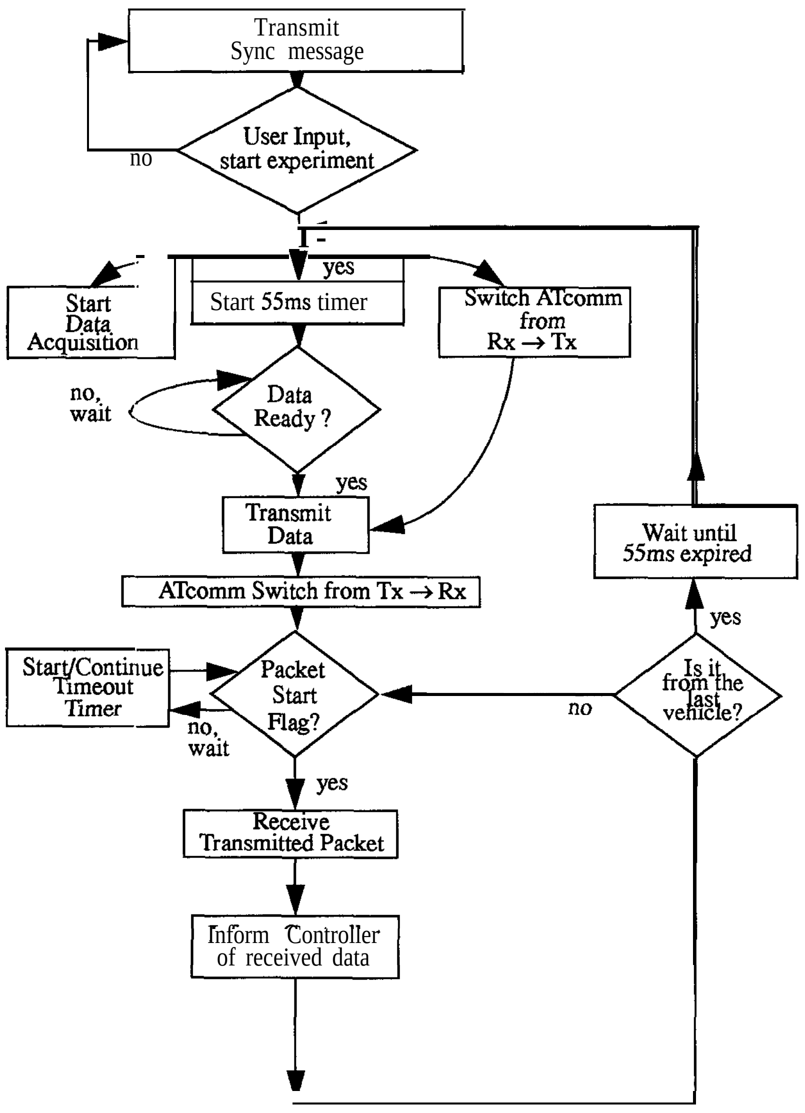

Transmit Sync message
User Input, start experiment
no
yes
Start Data Acquisition
Start 55ms timer
Switch ATcomm from Rx → Tx
no, wait
Data Ready ?
yes
Transmit Data
Wait until 55ms expired
ATcomm Switch from Tx → Rx
yes
Start/Continue Timeout Timer
Packet Start Flag?
no, wait
no
Is it from the last vehicle?
yes
Receive Transmitted Packet
Inform Controller of received data

Figure 3a. State Diagram for The Multivehicle Link: Lead Vehicle

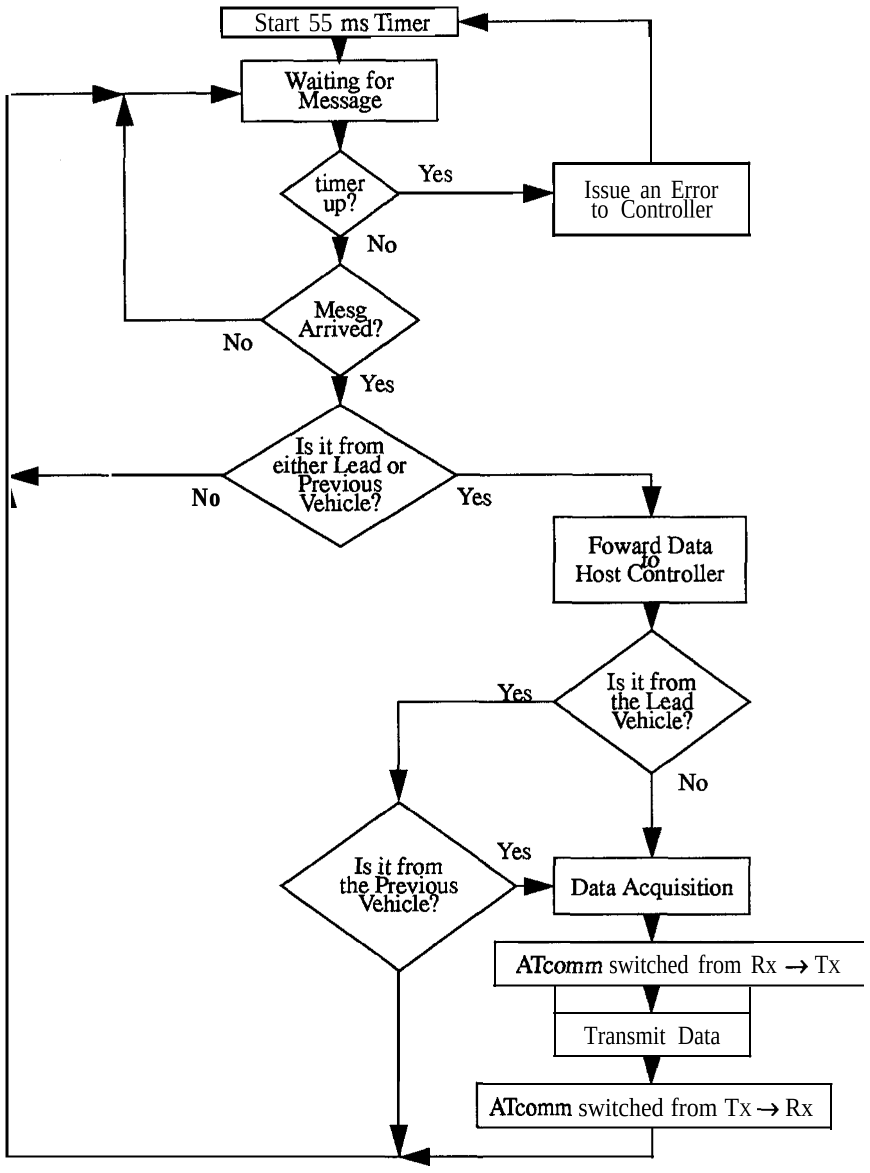

Start 55 ms Timer
Waiting for Message
timer up?
Yes
No
Issue an Error to Controller
Mesg Arrived?
No
Yes
Is it from either Lead or Previous Vehicle?
No
Yes
Foward Data to Host Controller
Is it from the Lead Vehicle?
Yes
No
Is it from the Previous Vehicle?
Yes
Data Acquisition
ATcomm switched from Rx → Tx
Transmit Data
ATcomm switched from Tx → Rx

Figure 3b. State Diagram for Following Vehicles (Multivehicle Link)

When each of the following vehicles receives the token from its preceding vehicle, it immediately searches for the data it needs to transmit. This transmission is intended for the lead vehicle or the vehicle behind the transmitter, depending on the location of the transmitter on the token-bus. If the transmitting vehicle is the last vehicle on the token-bus, its transmission is then received by the lead vehicle. While the lead vehicle does not need the received data, the reception of the message alerts that it now owns the right-to-transmit token, and it needs to restart the control loop when the current loop timer expires.

If the transmitting vehicle is not the last vehicle on the token-bus, then the link transfers the data to the vehicle behind the transmitter. Although all vehicles receive this broadcast, the link masks the message from control mechanisms. With the same broadcast, the link also registers the transfer of the right-to-transmit token. This process is repeated for each of the following vehicles.

The above design should satisfactorily support the data-transport needs of the control mechanism. In addition, it forces all vehicles to participate in the link, which enables the link to monitor all vehicles. Although not all vehicles are required to actively participate in the link, the lack of participation also removes it from the link's monitoring and its problem recovery mechanisms. However, this traffic becomes an additional burden the link needs to transport, and thereby reduces the amount of time available for data transport.

The link is designed to be monitored through the communication subsystem located in the lead vehicle. The lead vehicle is chosen mostly because it is able to allocate the greatest amount of computing time for this task. Further, since the lead vehicle is the control site for future platoon protocol extensions (which may include inter-platoon communication), locating link management in this vehicle centralizes network management to a single location. The primary responsibility for the link monitor is to detect communication time-outs: if any vehicle does not transmit its data/status message within a predefined duration, a time-out is generated by the link monitor. If the time-out condition does not correct itself within a reasonable amount of time, the link monitor attempts to warn the other vehicles by broadcasting a time-out alert message. This warning, in turn, is passed to the control mechanism, so that the appropriate action can be taken. In addition, the monitor can pinpoint the location of the problem from the last transmission it received: the problem must have occurred in the vehicle after this transmitter.

One flaw that the design didn't address is the link latencies between the lead vehicle and the last vehicle's transmissions. The sequential transmissions of the link introduces timing skew in the control process. It should be noted that the actual control signals can only he generated after data from the previous vehicle is received. Therefore, the more vehicles there are in the platoon, the larger the timing skew between the control signals generated by vehicles at the start of the transmission sequence than at the end. Since platoon control can only tolerate a certain amount of skews in control signal generation, it becomes a contributor in limiting the maximum number of vehicles the link can support.

As with the two vehicle platoon experiments, computer simulations determine that an interval of 55 ms is adequate for the platoon control. Within this 55 ms interval, each vehicle must perform the necessary communication and generate the appropriate control, which places the upper limit on the number of vehicles the link can support. By looking only at the specifications of the hardware, the 55 ms interval is long enough for the link to support four platooned vehicles.

3. Communication Link: Hardware

The communication link is composed of both hardware and software. Hardware provides the means to connect various stations (vehicles) in the network together while software provides intelligent control over the hardware components. The software also attempts to provide an efficient and error-less protocol for the exchange of information. Together, they provide a versatile communication link for the platoon experiments.

The communication link includes the following hardware components: radio transceivers, communication interface controllers, and host computers. Figure 4 illustrates the hardware setup of the communication link. One set of communication equipment, including one radio transceiver, one interface board, and one host computer, is installed in each of the vehicles used in the platoon experiment. Over-the-air broadcast-nature communication is made possible by using the spread-spectrum digital radio transceivers, model RXA-1000, developed by Proxim Inc. For robustness reasons, the transceiver is installed in the interior of the vehicle. To enhance the range of communication, an extension antenna is attached to the radio transceivers; this antenna reduces the attenuation of the received signal. The transceivers, in turn, are controlled by the ATcomm communication interface board for the IBM PC/AT, from Metacomp Inc. The communication interface controller board is installed in a host IBM PC/AT compatible personal computer which is based on the Intel 80386 microprocessor with the Industrial Standard Architecture (ISA) bus

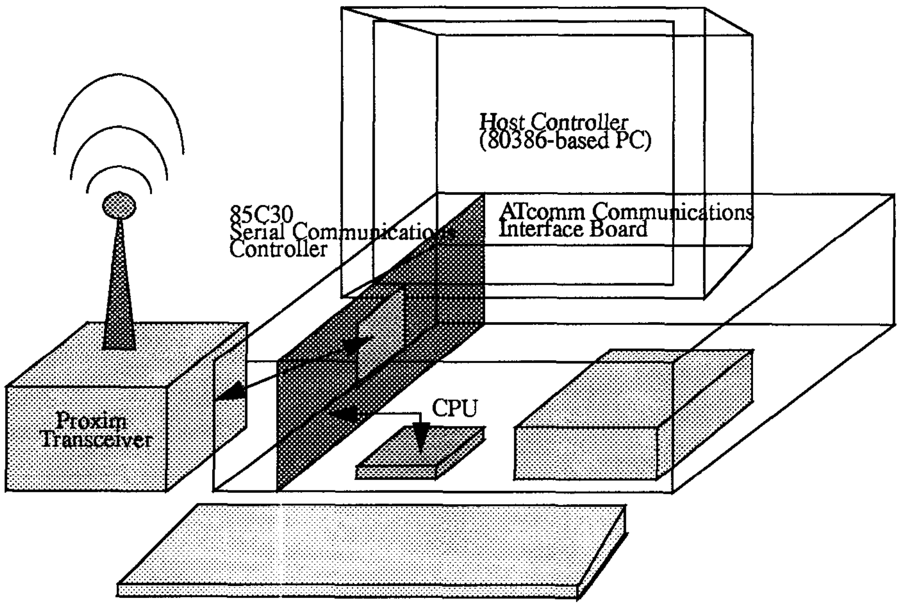


Figure 4. Communications Link Hardware Setup (Per Vehicle)

structure. The host computer provides intelligence of the network routing functions and performs computation on the control instructions using the data from the received messages and those from the sensors.

3.1. Proxim RXA-1000 Spread-Spectrum Digital Radio Transceiver

The Proxim RXA-1000 radio transceiver is used to provide over-the-air broadcast communication facilities for the platoon. A block representation of the transceiver is shown in figure 5 [5]. The transceiver is a VLSI surface-mounted single board radio, where both the transmitting and the receiving radio circuits are integrated onto the same radio (hence the name transceiver). Moreover, the surface-mount technology enables the transceiver to be of a diminutive size, thereby enabling it to be installed in tight spaces. The transceiver has a low power consumption of less than one watt. This allows the transceiver to tap its power directly from the host computer, thus reducing the complexity of the power hardware. The Proxim transceiver promises a communication range of approximately one-quarter mile. Thus. Proxim's small size and low power consumption makes it a good choice as radio for the vehicles.

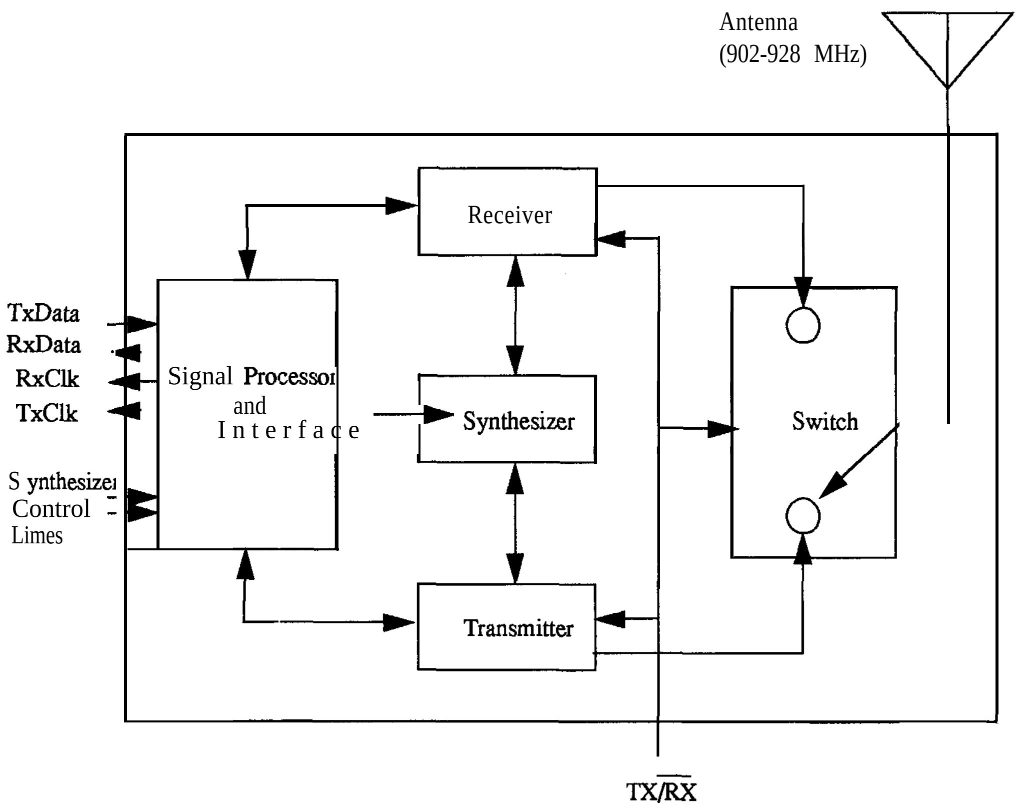


Figure 5. Block Diagram of PROXIM RXA-1000 Transceiver

In addition, the Proxim transceiver is chosen for its high data transfer rate, its spread-spectrum capabilities, its availability, and the ease in interfacing the transceiver to an external device, which in turn provides the control of the transceiver for the host computer. The transceiver has a maximum transfer rate of 121 Kbits/sec. This transfer rate can he achieved if the transceiver is used in synchronous mode. In synchronous mode, both the Proxim transceiver and its communication interface, the device that interfaces the transceiver to the host computer, share a common data clock. This clock information is used to synchronize the transceiver and the communication interface to the sampling of the data bit, thus reducing sampling phase jitter. The best sampling instant corresponds to the middle of the bit, thereby achieving minimum sampling error resulting from edge jitter, the uncertainty of sampling instant with respect to the data waveform. Proxim achieves this by generating the sampling clock for the communication interface for both incoming and outgoing data bit streams. Data transfer is then synchronized between the transceiver and the communication interface. In order to maximize the utilization of the communication link, all Proxims are used in the synchronous mode.

The Proxim transceiver is also capable of performing data transfers at 19.2 Kbits/sec when the transceiver is used in asynchronous mode. In the asynchronous mode, data is transferred by having the transceiver oversample the data stream at a rate of 121 Kbits/sec. The edge jitter limits the maximum practical transfer rate to 19.2 kbaud. The asynchronous mode is used to maintain compatibility with the numerous inexpensive serial communication devices usually used in personal computers. Since the communication interface that is used in this project can be programmed to perform transfers in synchronous mode and thus taking the advantage of the higher transfer rate, asynchronous mode is not used.

The Proxim transceiver operates in the 902-928 MHz frequency band. This band is approved by the Federal Communication Commission (FCC) for unlicensed use by commercial spread spectrum systems. To maximize its frequency selections, the transceiver divides the FCC approved frequency band into seven overlapping or four independent channels, depending on the need of the application. Thus, up to four independent communication links can be simultaneously established using the Proxim transceivers.

Proxim transceivers can easily be interfaced to the communication interface via the Roxim supplied Engineering Test (ET) boards [6]. The circuitry on the ET board is used to provide a unified interface for all I/O connections used by the Proxim transceiver, and to provide manual control over various transceiver settings. One such setting is the frequency selection of the transceiver. The reason for not incorporating computer-based frequency control in the communication link is that the communication software only needs one channel to provide full capabilities for the link. This decision is made as a direct consequence of the half-duplex operations of the transceiver: the overhead incurred with frequency switching is more significant than the savings it brings. This removes the burden and the overhead of controlling radio frequencies and simplifies the operations of the communication link. However, the ET board circuitry can be easily bypassed in applications that requires an active control on the frequency, Another aspect of the Proxim transceiver that can be controlled by the ET board is its mode of transmission. Transceivers can be switched between transmit and receive modes by either the communication controller or by manually setting the circuitry on the ET board. Both methods have been experimented. In the twovehicle platoon experiment, discussed in the previous section, the mode of operation is manually fixed via the ET board. For the four-car experiment, however, transmission mode switching is controlled automatically by the communication controller board. Since the transceiver provides its status through a set of control flags, intelligent control over the transceiver can be achieved. For example, the CTS (Clear to Send) flag produced by the Proxim transceiver notifies the communication controller that the transceiver is ready to transmit. If the CTS flag is reset, then the controller needs to switch the transceiver to the transmit mode. Thus, the existence of the CTS flag provides a flow control during the transmit operations of the transceiver.

The Proxim transceiver employs spread-spectrum modulation. Spread-spectrum is a method of transmission where the signals involved are transmitted using bandwidth in excess of the minimum bandwidth necessary for the transmission of the signal [6]. It has been demonstrated that spread-spectrum can be used to reduce the effects of intersymbol interference due to multipath distortions, and it provides immunity to external interference. Since all communications are performed in an urban environment where a multitude of sources of radio reflections is present, it is necessary to design the communication link to provide as much immunity to the multipath distortions. This and the immunity to nearby urban generated radio interferences made spread-spectrum an appealing choice. According to the specifications, the Proxim transceiver has a claimed bit error rate of 10^{-5}.

Proxim transceivers have two serious drawbacks when used in real-time applications. First, the transceiver can only operate in the half-duplex mode. That is, the transceiver can either transmit or receive, but not both, at one time. This is typical of a radio-based communication device. Although the transceiver is capable of switching between transmit and receive mode, a significant amount of overhead is required to achieve this. Also, additional overhead must be added to the switching delay since the transceivers must be allowed to synchronize with each other. The transceiver faces a similar overhead problem when it attempts to tune to different frequencies. Moreover, this delay is on the order of the switching delay. Therefore, if an application requires the transceiver to perform both switching and tuning, then the delays that would be incurred are compounded. In real-time applications such as the platoon experiment where it might be necessary for the transceiver to switch operation modes, the usefulness of the communication link is constrained by the slow reaction time of the transceiver. In addition, since there is no hardware mechanism on the transceivers that automatically detects for transmission errors, the transmitting transceiver cannot easily detect errors due to corrupted transmissions. Thus, Proxim’s inadequate real-time support and the lack of internal error detection reduce the usability of the transceivers.

Proxim’s second serious drawback is in its inability to report its operating status back to the host computer. For example, the transceiver is unable to indicate whether it is currently operating in the transmit or the receive mode. The application must guess the state of the transceiver. This is inadequate for platoon control applications. Without status reports, it would be impossible for the computer to accurately control the transceiver. The current version of the communication software estimates the mode of the transceiver by keeping track of the various commands issued to the transceiver. Thus, mode switching on transceivers is performed in two steps in order to insure the reliability of the transceiver. Fist, the switching command is sent to the transceiver. Then, the software waits a predefined interval for the transceiver to perform the switch and settles into its steady state, thus avoids operating the transceiver in its transient, unpredictable, state. This "waiting interval" is calculated based on the performance specifications of the transceiver and timed using the hardware timers available on the communication interface.

3.2. Metacomp ATcomm Intelligent Serial Communication Controller Board

A communication interface/controller, the ATcomm serial communication controller from Metacomp, is installed in each vehicle. Its primary function is to provide an interface and establish computer control over the transceivers. Its intelligence is used to implement most of the functionalities defined in the lower three layers of the OSI reference model.

3.2.1. Features of ATcomm Communication Controller Board

We chose the Metacomp ATcomm Communication Interface Board to provide high speed communication capabilities for the host computer. The ATcomm communication controller is a serial communication controller board designed specifically for the IBM PC/AI compatible personal computers with ISA or EISA (Industrial Standard Architecture and Extended ISA) bus architecture. It acts as the data interface for both serial and parallel data transfers between the host computer and the radio transceiver. See figure 6 for an overview of the ATcomm coprocessor board [8].

Serial communication capabilities are provided to the host computer through the onboard AMD 85C30 communication controller ICs (integrated circuits). The ATcomm interface board has room to accommodate two 85C30 controller chips. Each of the 85C30 controllers is used to provide two independent full-duplex serial communication channels, for a total of four independent serial I/O channels. All four channels can be programmed in either the asynchronous protocol or one of the supported synchronous protocols. Since multiple boards can be installed in the same system, the maximum capacity that the board supports is impressive. However, due to the single-channel (half-duplex) nature of the transceiver, the ATcomm controller boards used for this project are configured with only one 85C30 IC to support the single radio channel. This does not seriously affect the expandibility of the board, since the ATcomm boards can easily be upgraded to their maximal configurations.

The ATcomm interface also provides standard parallel data interface facilities through an onboard 8255 parallel port controller. Currently in this project, only the serial communication facilities provided by the Akomm interface is used. Future applications for the parallel data interface includes the control of display of communication link status.

In addition to parallel capabilities, the ATcomm board also supports accurate (up to a fraction of a millisecond) timing functions through an onboard 8254 timer IC. The 8254 supports three independent timers. The communication software uses one of these timers to provide the necessary delay needed to assure a successful mode switching at the transceiver. This is due to the fact that the Proxim transceiver does not provide facilities to provide a reliable feedback of its current mode of operation to its control interface controller.

The ATcomm communication controller can be programmed to support serial communication up to 2.23 Megabits/second using one of the synchronous protocols and up to 302 Kbits/second when using asynchronous pro tocol. High-speed communications are supported through direct memory access @MA) transfers; data is directly transfers between the onboard buffer memory and the serial port registers on the interface/controller, bypassing the central processing unit (CPU). For lower data rates, data transfers using polling and interrupt modes are also supported.

Both polling and interrupt modes of operations are supported by the ATcomm board. When used in the polling mode, the CPU repeatedly samples the port register; any data that appears in the register is immediately processed. When used in the interrupt mode, an interrupt signal is generated to alert the CPU on the interface board; the CPU then services the interrupt and processes the received data. Although slower than DMA transfers, both of these methods offer easier implementation and full control over the processing of the received data using the CPU on the ATcomm board. Unfortunately, the data transfer rate for this project is dictated by the Proxim transceiver to be

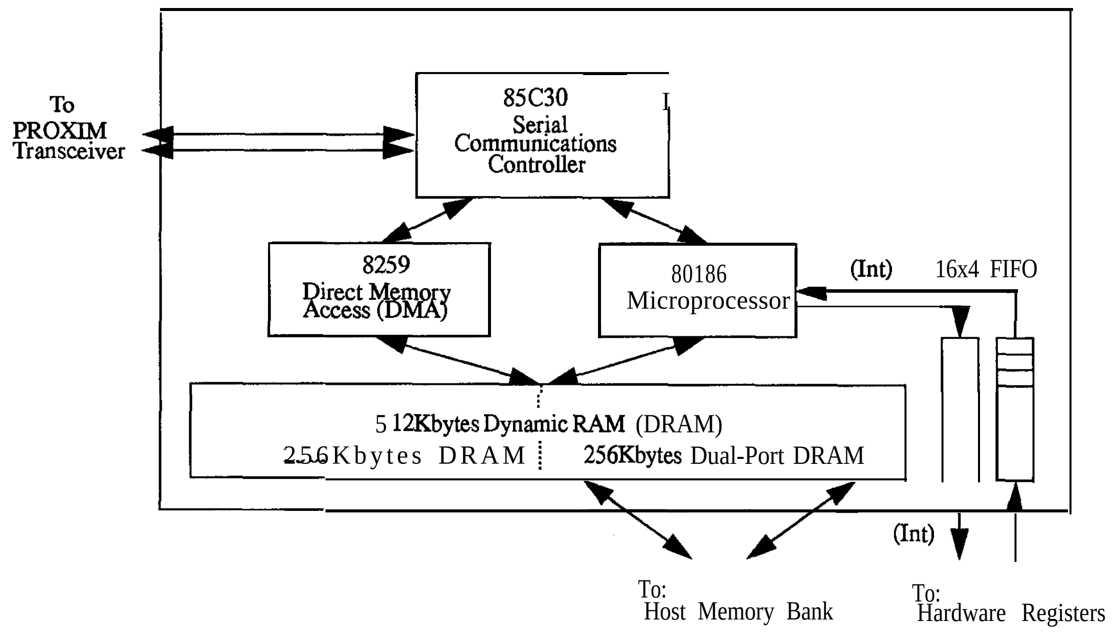


Figure 6. Block Diagram for ATcomm Communications Coprocessor Board

121 Kbits/sec. Since this is only 5% of the capacity of the ATcomm board, there is enough CPU power to handle the polling responsibilities. The tedious task of DMA and interrupt programming are avoided.

For future expansions and upgrades, the ATcomm interface can easily be reprogrammed to support communication applications that require higher transfer rate. An example of such application may be the next phase of this project, high-speed communication link. We can easily integrate the current software running on the ATcomm interface into high-speed links. A possible wide-bandwidth alternative to the transceivers, the infra-red optical link, is currently being developed by Professor Walrand's research group.

Numerous daughter boards (named the ELX boards) are available from Metacomp Inc. to provide connection between the ATcomm interface and other communication devices. The ELX board implements this connection using well established interfacing standards such as RS-232, RS-422, etc. However, since raw outputs from the Proxim transceivers have the same electrical properties as the signals on the ATcomm interface, the ELX interface board is not used for this project. All data and control connections are made directly between the transceiver and the ATcomm controller.

The ATcomm controller also has the capability to operate completely independent of the host computer. This is made possible through an onboard 9.63 MHz Intel 80186 microprocessor and its support chip set; the CPU and its peripherals make the ATcomm interface a single board computer. This setup promotes a multitasking environment under which the ATcomm controller is allowed to continuously monitor (poll) and control communications without any supervision from the host computer, and vice versa.

Although the ATcomm interface and the host computer are two independent entities, a number of facilities are provided on the ATcomm interface for interprocessor communication. First, facilities are provided on the interface board to support host-to-ATcomm and ATcomm-to-host interrupt activities. Interrupts are generated when the FIFO (First-In-First-Out) buffer on the interface board is being written. The interrupt is cleared when the FIFO is cleared. Two 16 x 4 bits FIFOs are provided where one is used as the buffer for data written by the host for the ATcomm interface and the other is provided as buffer for data from the interface controller for the host. If the FIFO is written by the host, an interrupt will be issued to the interface board. The value that is written into the FIFO is irrelevant. This value is usually intended as data, and the value must be cleared from the FIFO in order to clear the interrupt. Similarly, an interrupt is generated on the host computer when the ATcomm-to-host FIFO is written.

In both the two and four-vehicle platoon experiments, the ATcomm-based FIFO/interrupt is not used by the communication manager. Rather, the ATcomm board to host interface is implemented using polling operations. Under the polling scheme, changes in status of the observed equipment are detected through constant monitoring performed in software. Although polling is less efficient than its interrupt counterpart for monitoring system activities, we have plenty of computing resource, in both the ATcomm board and the host computer, to implement polling. Moreover, the polling-based system is far less difficult to implement than a interrupt-based system, which reduces the development time of the overall system. Therefore. the more complex interrupt-based system is not implemented for these network experiments.

The second ATcomm-host interfacing facility is the memory on the ATcomm board. ATcomm board provides a hefty 5 12K of DRAM for the storage of the onboard software and related run-time variables. Of this 5 12K RAM, 256K RAM is unaccessible from the host computer. This memory is primarily used for local software and temporary data storage. The remaining 256Ks are mapped as a bank of dual-port memory, thus providing block access to both the host computer and the ATcomm board. This dual-port memory is normally used as a buffer for user-desired data. It provides the capability for fast transfers of large blocks of data between the host computer and the ATcomm controller. In addition to data, the dual-port memory also performs the passing of status messages and commands; the polling algorithm monitors this memory area for possible status changes and receiving commands.

Although ATcomm interface is complete with advanced hardware, it is the software that gives intelligence to the ATcomm interface. This software takes on the responsibility to process, buffer, and notify the host computer when messages have been received. It also handles the necessary processing for outgoing messages and any requests issued by the host computer. Furthermore, it maintains the proper operation of the ATcomm interface and the transceiver. Since no operating system software was supplied with the ATcomm controller board, typical of an embedded controller, thus all codes executed on the ATcomm board must provide the necessary system maintenance functions. This increases the size and the time in the development of the code, but it also provides a greater degree of control for the software. All software written for the ATcomm controller are loaded from the host computer down into the ATcomm board and executed through an onboard debugger/monitor. The onboard ATcomm interface software is discussed in more detail in section 4.

3.2.2. 85C30 Serial Communication Controller

The serial communication functions supported by the ATcomm communication interface board are performed by the onboard 85C30 communication controller. The AMD 85C30 serial communication controller IC is responsible for control of the serial communication *between* the communication controller board and the Proxim transceiver [5]. See figure 7 for the block diagram of the 85C30 controller chip. The 85C30 controller chip is designed to work with the 8- and 16-bit microprocessors; thus it can be programmed to work closely with the other components on the ATcomm controller board.

Two 85C30 serial communication controllers can be installed on one ATcomm controller board. Each 85C30 controller is capable of controlling two full-duplex serial channels, where each channel can be programmed to provide independent communicatior. with the external world. However, since only one communication channel is needed for the platoon experiments, only one 85C30 IC is installed on the ATcomm board used for this project.

Each channel supports high speed serial communications of up to 4 Mbits/sec, according to 85C30 specification. However, due to the limitations of the ATcomm board, the maximum transfer rate supported is actually 2.23

Mbits/second. Although the maximum transfer rate for this project is only 121 Kbits/sec (dictated by the Proxim transceiver), the high speed capabilities of the 85C30 gives us the option of moving into more advanced (and faster) forms of communications while retaining the same communication interfacing hardware, thereby reducing hardware cost and minimizing software development for the new setup.

The 85C30 controller is capable of supporting a wide variety of communication protocols, in both asynchronous and synchronous flavors. This greatly extends the flexibility of the 85C30 controller for wide range of communication applications. In applications where the communication hardware only supports asynchronous communication (such as standard voice-band modems), the 85C30 can be programmed to provide the proper support, thus replacing a dedicated universal asynchronous receiver-transmitter (UART). The 85C30 also provides support for two popular synchronous protocols: the Binary Synchronous Communication (Bisync) protocol (a synchronous character-oriented protocol popularly used for polling remote terminals) and the SDLC (Synchronous Data Link Control)

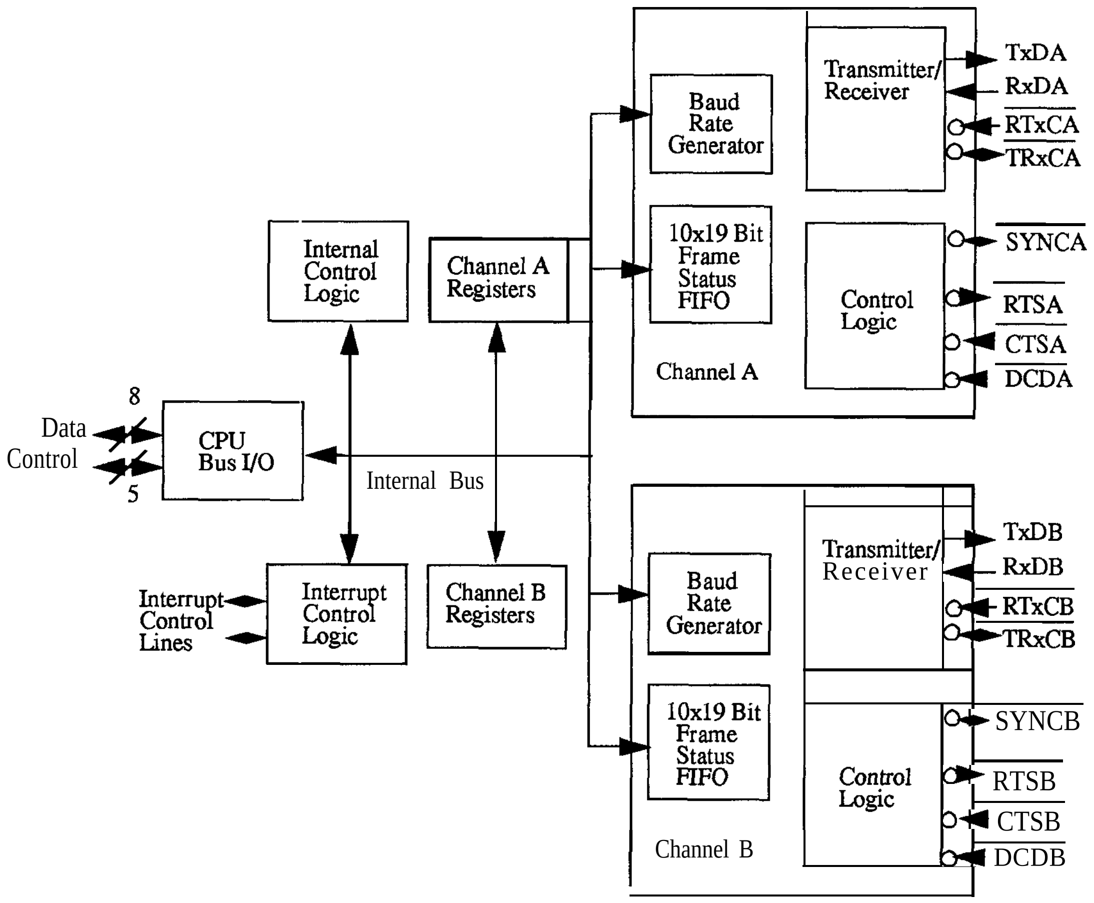


Figure 7. Block Representation of 85C30 SCC

protocol, which are synchronous bit-oriented protocols. In this project, a synchronous protocol is used in order to maximize the communication efficiency of the Proxim transceivers.

Of all the synchronous protocols supported by the 85C30, one protocol is selected for its richness in feature. The protocol selected is the SDLC synchronous protocol. The SDLC protocol is derived originally from the data link protocol used in the SNA (Systems Network Architecture), developed by IBM. IBM later submitted the SDLC protocol to ANSI (American National Standard Institute) and IS0 for acceptance as standards in the US and the international networking communities. ANSI modified SDLC protocol to become the ADCCP (Advanced Data Communication Control Procedure). Meanwhile, OSI modified SDLC to become the HDLC (High-level Data Link Control) protocol. CCITT (International Telephone and Telegraph Consultation Committee) later adopted and modified the HDLC protocol for the LAP (Link Access Procedure) portion of the X.25 network interface. [10]

The SDLC protocol is a widely used data-link protocol. It has a number of features that make communication a clearly defined task. First of all, SDLC protocol is a synchronous bit-oriented protocol (BOP). Like the synchronous character-oriented protocols (COP, an example of this is the IBM BISYNC), SDLC protocol can be used in both full- and half-duplex operations. SDLC protocol, however, is less dependent on special control characters that are used by COP for synchronization. Instead, SDLC protocol relies on the positioning of bits with specific fields for synchronization. Therefore, SDLC is more robust than COPs in resisting bit-length corruptions in communications since as synchronization is lost, SDLC can reestablish synchronization by scanning the bit-stream for a particular bit pattern. COPs, on the otherhand, may never reestablish synchronization if it suffers misalignment problems (that is, characters start from different bits for different end of the network). For radio systems where such problems are common, SDLC stands out as the synchronous protocol of choice.

SDLC is an in-band protocol: all SDLC information are stored in structures called frames, along with the data. Each frame follows a standard format: the boundaries of the frame are delimited by a pair of frame beginning/ending flag. The address field comes after the beginning flag. After the address field comes the control field. Then comes the information field. And at the end of each frame is the checksum field. See figure 8 for the format of a data frame under the SDLC protocol.

The boundary flag, which delimits two frames, is an unique 8-bit sequence “01111110.” To prevent confusion in distinguishing between the flag and the contents in the frame that happens to have the flag sequence in it, SDLC prevents the sequence of six consecutive one in both data or the control fields. SDLC accomplishes this by performing the so-called “zero-bit insertion” or bit-stuffing. Here, a zero bit is added by the SDLC at the transmitter after all five consecutive one sequences. Likewise, the SDLC at the receiver deletes the zero bit, if it exists, after every sequence of five consecutive one bits. This successfully distinguishes the flag bit pattern from any other bit patterns containing 5 consecutive ones.

Prior to the any SDLC-based communications, all receivers on the link must search (or “hunt,” as used by the 85C30 reference manual) continuously for the flag sequence. This enables the receivers to perform both the bitwise and framewise synchronization with the transmitter of the flag bit-pattern, which is usually the lead vehicle, and all data transmitted while the transceivers are not synchronized are discarded. There are at least two boundary flags between two consecutive messages. If for some reason the transmission is corrupted in such a way that one of the boundary flags is destroyed, SDLC will hunt for the next boundary flag, resynchronizing with its peer. Thus, SDLC can quickly recover, within one frame of the corrupted frame, from the loss of communication due to errors generated by interference for the communication channel.

SDLC protocol address feature enables selective reception by the transmitter: only the receiver that is named by the transmitter can receive data message. However, for this project, since every data message may contain important control information for all vehicles, the selective filtering feature is not used.

The information field contains all the non-SDLC generated data. For example, all vehicle information and link maintenance controls external to the SDLC protocol are placed in this field. This field can be of any number of bits. Usually, the contents in this field is an integral number of 8-bit characters. However, due to the "zero-bit insertion" scheme used by SDLC, this may not be true.

In addition, SDLC protocol provides a cyclic redundancy code (CRC) error checker. The CRC checker performs CRC checks on the entire frame, including the address field, the control field, and the information field. The CRC checker does not take in account of the bits that are inserted into the data stream (generated from the zero-bit insertion process), the bits in the SDLC framing flags, and the CRC bits themselves. The generated CRC is stored in the frame checking field as a sequence of 16 bits. The CRC polynomial, needed for CRC error-checking process, is the industry standard CCITT-CRC polynomial ($X^{16}+X^{12}+X^5+1$). Thus, CRC is used to provide the ability to detect errors in communication without the need to resort to manually adding an external (outside of the SDLC protocol) error coding. However, CRC is not used for error correction. Therefore, in the interest to provide a more robust com-

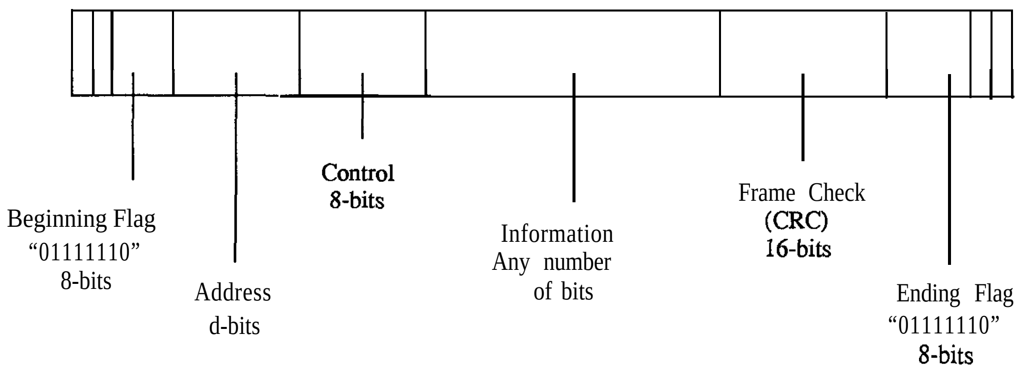


Figure 8. SDLC Frame Format

munication environment to improve immunity of the link against communication errors, a more sophisticated coding scheme should be implemented.

The beauty of CRC calculation is in its implementation: it can easily be implemented in hardware. The process of performing computation using the CCITT-CRC in hardware is displayed in figure 9a. The entire computation is preformed using sixteen single-bit shift registers and three exclusive OR (XOR) gates. The 16 registers are first initialized to 1 (this is defined by the SDLC protocol). Then, the data bits are circulated through the CRC computation, and the two CRC-octets are copied off from the output of the shift registers. To perform error-detection using CRC procedure, identical CRC computations are preformed at both the transmitter (on the output data stream) and at the receiver (on the received data stream). The CRC-octets that are generated by the transmitter and the receiver are compared. If the two quantities do not match, the 85C30 then announces that a CRC-error has been detected, and the packet is declared to be corrupted. The 16 bit CRC-CCIIT catches all single and double bit errors, all errors with an odd number of bits, all burst errors of length 16 bits or less, 99.997% of 17 bit error bursts, and 99.998% of 18-bit and longer bursts. This is the syndrome of the CRC code. It can be calculated using the circuit described in figure 9b.

In addition to a good protocol support, the 85C30 communication controller also provides different line coding schemes. These data encoding/decoding mechanisms provides the 85C30 the control over the low frequency components of the serial data stream. Examples of line coding supported by the 85C30 includes the NRZ (Non-Return to Zero), NRZI (Non-Return to Zero Inverted), Biphase Mark, and Biphase Space (FM encoding). In addition to the ability to decode the above, the 85C30 can also perform Manchester Decoding. Therefore, for communication

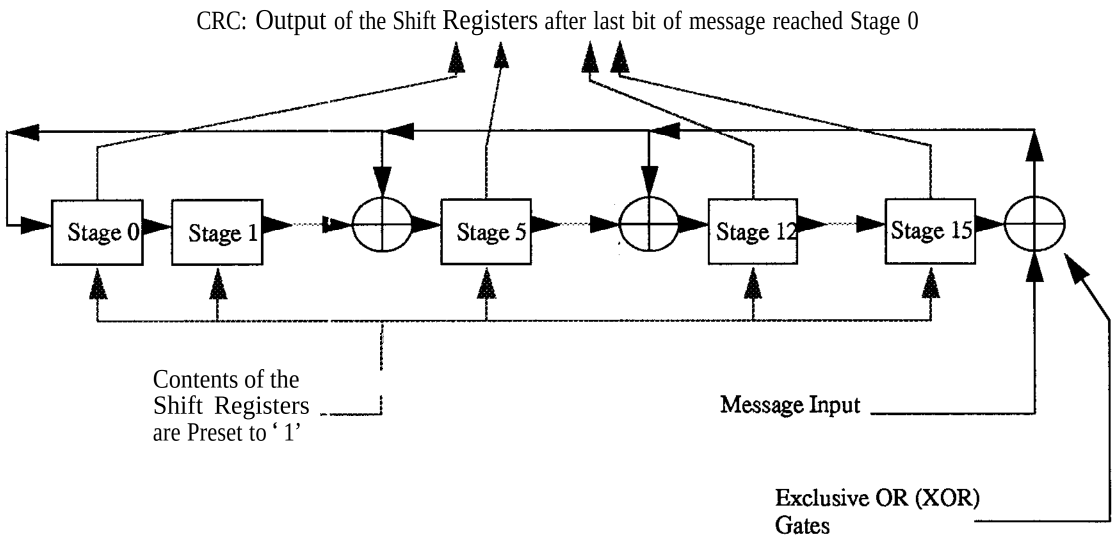

CRC: Output of the Shift Registers after last bit of message reached Stage 0
Stage 0
Stage 1
Stage 5
Stage 12
Stage 15
Contents of the Shift Registers are Preset to ‘1’
Message Input
Exclusive OR (XOR) Gates

Figure 9a. Cylindrical Redundancy Checksum Encoder
(For CCITT CRC Polynomial: $X^{16} + X^{12} + X^5 + 1$)

hardware that does not provide the facilities to transmit the clock along with the data, the clock can be encoded into the data stream, transmitted, and recovered by a phase-locked-loop, another built-in feature of the 85C30. Moreover, with the availability of FM encoding schemes, the 85C30 can be used with communication mediums in which the medium acts as a high frequency filter. An example of such medium would be the telephone channel. For this experiment, transmissions made from the transceivers are already encoded (scrambled) by the transceiver using the spread-spectrum encoding scheme. Thus, low frequency components in the original data signal are effectively modulated into higher frequency components, which reduce the effects caused by the communication medium. Moreover, in applications (such as the infra-red optic link) where line encoder/decoders are not yet available, the 85C30 controller can then be programmed to provide line coding features without implementing additional hardware.

In addition to line coding, the 85C30 can also be programmed to perform clock recovery from transmissions using the on-chip digital phased-locked loop (DPLL). Since the transceiver performs the clock recovery while decoding the spread-spectrum data, this 85C30 feature is not utilized. For communication links with which a clock recovery mechanism is not inherently available, the 85C30 controller can be programmed to provide the necessary timing support.

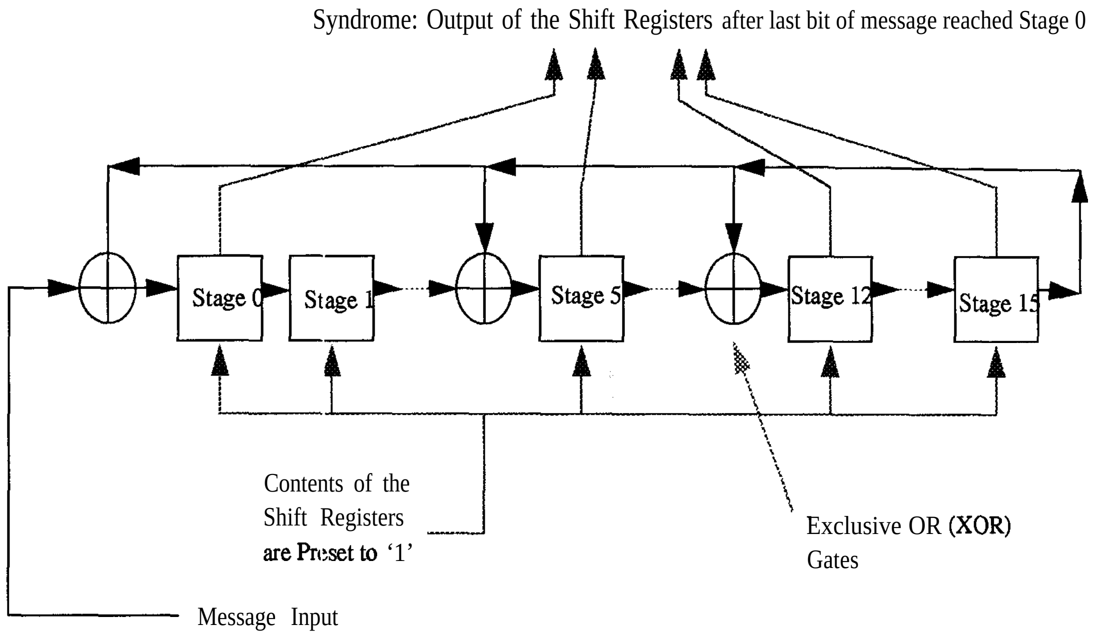

Syndrome: Output of the Shift Registers after last bit of message reached Stage 0
Stage 0
Stage 1
Stage 5
Stage 12
Stage 15
Contents of the Shift Registers are Preset to '1'
Exclusive OR (XOR) Gates
Message Input

Figure 9b. Cylindrical Redundancy Checksum Decoder
(For CCITT CRC Polynomial: $X^{16} + X^{12} + X^5 + 1$)

4. Communication Link: Software

The intelligence of the communication link lies within the software. A successful implementation of the link software must maximize its utilization of given hardware capabilities and overcome their shortcoming. In this project, the software is used to provide control over a variety of communication hardware and to establish arbitration for communication among vehicles on the link.

Due to the modularity in the hardware, autonomous software modules are constructed for each of the two major communication hardware: the ATcomm interface board and the host computer. For the ATcomm interface, the link ***manager*** is written to provide necessary handling of the hardware and maintaining the integrity of the link. The link manager provides the data-link functions, such as error-checking, data framing, and data buffering, to the communication process, and the driver is completely residing in the memory provided by the ATcomm communications board. For the host computer, ***the communication manager*** is used to provide protocol **functionalities** for the network. The communication manager, riding on top of the link manager, implements the protocol used for the communication link. The manager also acts as an interface between the low-level device control and the control mechanism. The interrelationship of each communication software module is shown in figure 10. All code modules for the communication software are written in either standard ANSI C and/or in 8086 assembly language.

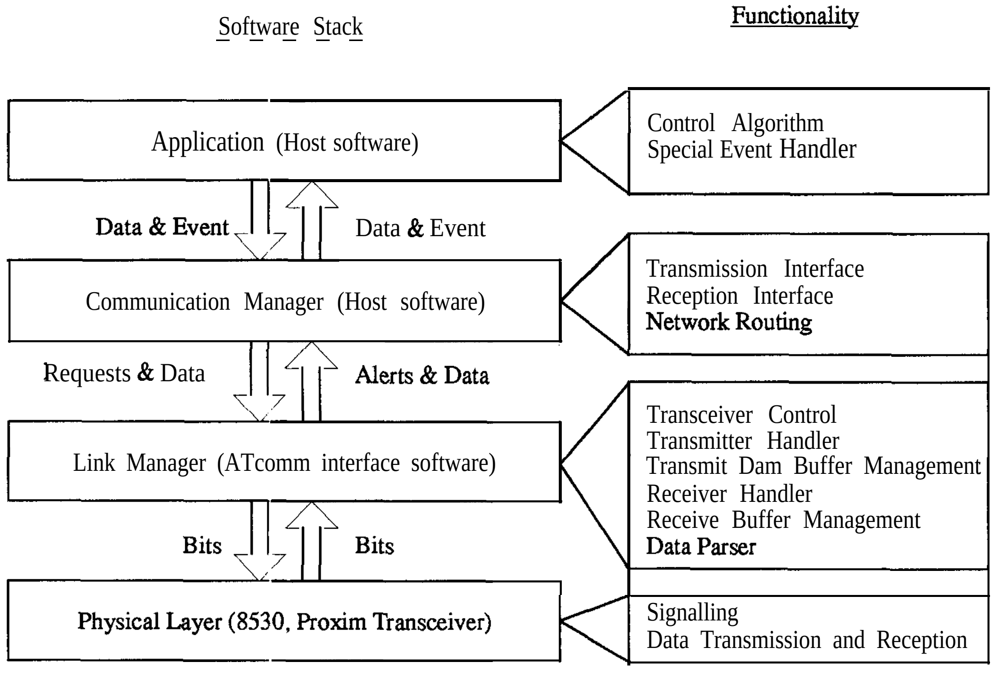


Figure 10. Communication Link Software Hierarchy

Besides modularity, uniformity in the software is also strived for in the software development. That is, the same software must be able to run on all vehicles participating on the communication link; there is no need to develop one version of software for the lead vehicle and another for the following vehicles. This eliminates statically designating vehicles for specific positions on the platoon. It also eliminates potential mix-up in software execution during experiments. Therefore, link and the communication managers are developed to be able to fulfill the different requirements demanded by each vehicle.

4.1. Components of the Software

For these experiments, the communication process is programmed to alert the host computer only when a broadcast message is completely received and processed. This allows independent operations between the communication processes and the control processes, enabling control to be performed simultaneously with communication process. In addition, due to the modular design of the hardware, data acquisition processes can be performed concurrently with the control and communication processes.

4.1.1. Device Software Development: ATcomm Software Development

The link manager software is written to be executed on the ATcomm interface board for the purpose of controlling the environment on the ATcomm interface controller and Proxim radio transceiver. The link manager utilizes the 8530 controller's SDLC protocol and other related hardware peripherals to provide the link with data link layer facilities as defined by the OSI reference model [10]. Thus, the link manager attempts to establish a seemingly errorless communication link.

The system software provided by the ATcomm interface is an EPROM-resident debugger. Its main function is to initialize processor-associated components on the ATcomm interface after each power cycle. Therefore, the link manager must provide all the necessary routines to obtain direct control over the components on the ATcomm interface. In addition, all test software must be downloaded to the debugger for execution. Thus, all current software developed for the ATcomm interface suffer an overhead penalty for the use of the debugger. Although the link manager software can be placed into EPROM to eliminate the debugger and its overhead, the gain is not significant enough to warrant the additional development effort.

Software for the Akomm interface is written in the Absolute Object Code format (also called the 8086 Object Module Format, or OMF) by Intel, since it is the only language understood by the debugger on the ATcomm interface board. This object code format is ideal for the development of software for systems with embedded controller because it specifies the absolute memory location for each of the routines in the code. This enables OMF-based code to be loaded into a static environment (such as an EPROM) where code relocation is an unnecessary luxury. However, OMF-based development is the biggest software obstacle we faced in the development of the link manager: little support is available for OMF-related software development. For instance, software debugging can only be performed in either assembly code or the OMF equivalent code. Not surprisingly, OMF is a pseudo-machine code format where the actual machine code is intermixed with the memory location information. Therefore, debugging with OMF code is just as troublesome as with debugging straight assembly code. A good amount of time was spent on familiarizing myself with the OMF formatted code.

Standard assemblers and compilers do not provide direct support to generate OMF format code. Luckily, commercial post-processors exist that convert C compiler/assembler output into the OMF format. This greatly reduces the time for code development. In addition, at the completion of the development of the ATcomm board software, another post-processor can be used to convert the OMF code into the 80186 machine code. This enables the ATcomm software to replace the resident debugger as the system software.

Majority of the source code for the ATcomm software was written in standard ANSI C and compiled using Microsoft C version 6.00 for MSDOS-based personal computers. It provides limited management for chores that are normally performed by the operating system, such as initializing the components on the ATcomm interface to suit our application. An assembly code header, written in the Microsoft Macro Assembler, version 5.10, is attached to the object code produced by the C compiler to define specific code locating information. The combined code is processed into the OMF format using C6toPROM, a C-object file to OMF (and other code format) post-processor, written by Systems & Software, Inc [11]. The OMF codes are then downloaded into the RAM on the ATcomm interface board and executed through the onboard firmware debugger.

Two stages of communication software development are planned. In the first stage, the ATcomm software is written specifically for the two-vehicle platoon experiments. Here, the link manager is specialized to different vehicles. The manager on the lead vehicle is capable of only transmitting data while the manager on the following vehicle is capable of only data reception. The later stage involves the development of software to support four or more vehicle platoon communication. The link manager implemented for this stage does not differential between vehicles, and it is capable of both data transmission and reception.

4.1.2. Host Software Development: VRTX Multitasking Environment

Platoon control is a real-time application. Unfortunately, normal operating systems are not equipped to handle the requirements of the real-time applications since these operating system are designed to operate with less restrictive timing constraints, and it is unlikely that these operating systems are able to respond to events rapidly enough for platoon control. In addition, real-world events rarely occur in strict sequences, and any software that deals extensively with the external world must be able to handle simultaneous occurrences of multiple events. Again, regular operating systems are not equipped to handle this situation. Therefore, a real-time multitasking operating system is necessary for this experiment. The operating system chosen for this experiment is ***VRTX*** [12] by Ready Systems.

VRTX (Versatile Real-Time Executive) is a high-performance operating system for embedded microprocessors and real-time applications. It provides a set of basic mechanisms that application software can utilize in order to support real-time operations. The mechanisms supported by VRTX include multitasking support, CPU scheduling, communication, and memory allocation.

VRTX provides a healthy set of functions to implement multitasking and CPU scheduling. For instance, it supports the creation, execution, and removal of tasks independent of each other. Each event or task can be created, executed, and deleted without affecting any other currently running tasks. In addition, priority can be assigned to tasks so that one can order their schedule of task executions. For example, one can program VRTX in such that a way that many tasks can be executed sequentially, in a round-robbin fashion, or just suspend some tasks and activate the rest. The programmer controls the choice.

VRTX also provides a generous set of intertask communication capabilities. However, these capabilities are not applicable for the actual communication with the transceiver. Four standard multitasking operation system communication tools are provided by VRTX. They are the mailboxes, queues, event flags, and semaphores.

In addition to the above, VRTX supplies mechanisms to provide a systematic method to allocate memory resources. Unfortunately, the amount of memory that is supported by VRTX memory allocation is too small to meet the requirements of this experiment. Therefore, a separate set of functions are written to provide VRTX access to the expanded memories using the expanded memory manager, which have a maximum addressing capacity of 16 Mbytes.

Lastly, VRTX supplies the necessary input/output components that interface the VRTX software with the rest of the world. First, a MSDOS compatible file I/O executive (IFX) for permanent storage of experimental data, enabling off-line data analysis, was supplied. Currently, this enables VRTX to access 3.5” floppy magnetic media, which provides a 1 .44 Mbyte of storage per diskette. Unfortunately, high capacity hard drive support is provided in a future version, and is not used in these experiments. Second, VRTX supplies a windowing executive (WIX) that displays windowed output on standard MSDOS displays. Thus, WIX gives the user a good text-based visual interface for the VRTX application. This is needed for the control program to display, in an orderly fashion, the status of various components of the system.

Like the link manager for the ATcomm interface, separate versions of the communication manager are written for the two and the multi-vehicle platoon experiments. For each version, separate and independent modules are implemented to perform the data transmission and reception functions in conjunction with the link manager. Furthermore, these modules are combined to provide a multiple access protocol for the communication channel (similar to those in the token-bus network) and time synchronization for the IPCS (Integrated Platoon Control System), the main control mechanism.

4.2. Two-Vehicle Platoon Communication Link

The link software developed here provides the skeletal structure for the more elaborate multi-vehicle link software. It is used to examine various assumptions about the characteristics of the hardware and the environment during the design phase of the experiment. As mentioned previously, the modularity in hardware forces the implementation of two separate modules, the link and the communication module, for the software. The link manager establishes low-level device control, whereas the communication manager implements data interfacing between the IPCS and the link manager.

4.2.1. The Link Manager

As designed, the two-vehicle communication link manager aims to implement a half-duplex unidirectional link between the vehicles. In these experiments, the F'roxim transceiver installed in the lead vehicle is configured to operate continuously in the transmit mode. The transceiver in the following vehicle, on the other hand, is configured to operate only in the receiving mode. While this severely limits the capabilities of the transceivers, it also eliminates much of the burden and the complexity of mode switching. Further, the link manager does not need to have control over the transceivers since the transceivers are configured to operate in a single mode. The only responsibility it has is to transfer the data from the memory buffers in the ATcomm interface to the 85C30 communication controller for transmission, and to store the data received by the 85C30 controller and alert the host computer of the arrival of data.

The link manager itself is divided into two communication tasks: a transmission handler and a reception handler, where the transmission handler only supports transmission services while the reception handler only supports reception services. Due to the unidirectional nature of the link, only the transmission manager is installed in the link manager at the lead vehicle. Refer to figure 11 for the flow charts describing the commumcation link manager for the two-vehicle platoon experiment.

Before the link manager activates the handlers, the link manager initializes the ATcomm interface and the Proxim radio. In addition to initializing relevant aspects of the ATcomm communication board, the initialization routine also configures the 85C30 controller into the SDLC mode. SDLC protocol parameters that need to be specified includes the size of the character (number of bits per character), the size of CRC polynomial, and the function of the link manager (that is, the manager in the lead vehicle initializes the 85C30 in transmit mode under the SDLC protocol, and a similar setting for the following vehicle). It also defines the memory locations for data exchanges and shared status flags between the link and the communication manager. For this project, all transmissions are made with 8 bits/character and employs the standard CCITT CRC for error-detection.

To start the transmission process, the lead vehicle's 85C30 sends out continuous boundary flag sequences to establish frame synchronization with the receiving vehicle. Meanwhile, the lead vehicle's link manager/transmission handler continuously polls for the presence of the Request-To-Transmit (RTS) flag, set by the communication manager, on the ATcomm board. If the flag is not set, the transmission handler is idled. If the flag is detected, the link manager informs the 8530 to prepare itself for data transmission. To signal the beginning of the packet, one final boundary flag is transmitted. The 8530 then sends the SDLC address field and the control field data after the boundary flag. Then the data that is present in the 8530 input buffer, which are placed there by the transmission handler, are transmitted. When the data is exhausted the 8530 then transmits the CRC check sequence generated from the transmitted data. To complete the packet, the 8530 transmits another boundary flag. At this time, the transmission handler is idled and the RTS flag is resetted. The entire process, starting after the initialization, is cycled over.

The receiving process on the following vehicle follows a similar procedure. After 8530 and transceiver are initialized, the reception handler waits for the incoming data to appear in the 8530 receive buffer. Meanwhile, the 8530 searches for boundary flags transmitted from the lead vehicle in order to establish inter-vehicle frame synchronization. After the synchronization is achieved, the first non-flag character received is designated by SDLC as the

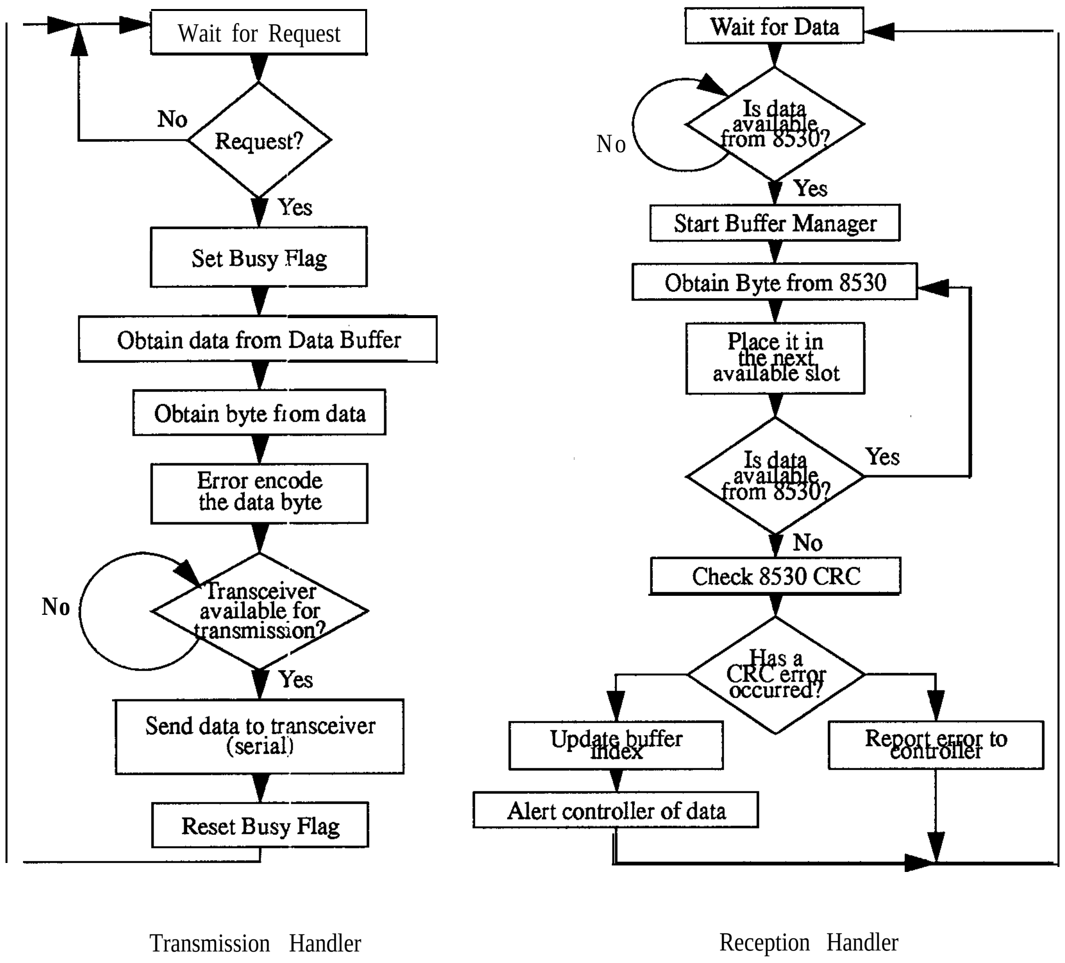

Wait for Request
No
Request?
Yes
Set Busy Flag
Obtain data from Data Buffer
Obtain byte from data
Error encode the data byte
No
Transceiver available for transmission?
Yes
Send data to transceiver (serial)
Reset Busy Flag
Transmission Handler
Wait for Data
No
Is data available from 8530?
Yes
Start Buffer Manager
Obtain Byte from 8530
Place it in the next available slot
Is data available from 8530?
Yes
No
Check 8530 CRC
Has a CRC error occurred?
Update buffer index
Report error to controller
Alert controller of data
Reception Handler

Figure 11. Flow Diagram for Link Manager (Two Vehicle Link)

address of the transmission. Since the address filter is not used in this project, the data in the address field is ignored and all data are received by all vehicles. After the control is received, the 85C30 processes the subsequent received information, consisting of the data and CRC sequence, and places them on the on-chip output buffer. At the same time, the 85C30 alerts the reception handler to handle the processed data. The reception manager transfers all data appearing at the 8530 output to the next available buffer storage, which is created in the main memory bank. This is continued until the reception handler detects an end-of-frame alert generated when the 8530 receives the closing boundary flag. The 8530 computes its own CRC sequence from the received data and compares it to the CRC that is extracted from the last 16-bits of received data. The CRC status is then processed. If the two CRC sequences do not match with each other, then it is assumed that error has occurred in the received data. The 8530 then sets the corresponding error flag to alert the communication manager of the error. Furthermore, all data in the corrupted packet is ignored. If no error was detected in the received packet, the reception handler places the packet into the circular reception data buffer. The sole purpose of the data buffer is to reduce the chance the communication manager missed the received data. To achieve this, the data buffer is constructed with the capacity to store and handle a number of fixed size packets. This increases the amount of time the communication manager has for data processing before the data is overwritten with newer data. The entire process is then repeated for the next data arrival.

It should be noted that the format in the packet transmitted in this phase of the experiment, and therefore the packet size, is constant. The exact content of the packet is defined in the next section, the Communication Manager. The link manager takes advantage of this property by fixing the size of the data message. Two advantages appear from this situation. First, fixed packet size can be used as an error-checking device, supplementing that of the CRC error checker. Although the SDLC CRC provides error-checking capabilities, the actual checking is performed at the end of the frame. If there is a transmission loss or if the framing flags are corrupted, then there might never be an end to the frame in question. If this occurs, the CRC error checker can never complete its computation, and a time-out alert would be issued. However, time-out alerts can cause undesired delays to the control algorithm. This must be avoided. It is noticed that in some circumstances, the lost of connection is accompanied by a burst of noisy data, causing the packet to contain more data than the size of the message. Since the size of the message is fixed, a larger than normal packet can only indicate that an error had occurred. Thus, another error-checking scheme is conceived.

Second, by fixing the size of the transmission data, a less complex transmission and reception handler software could be implemented. With a fixed sized data message, the communication link did not need to have the size handler routines implemented. In addition, this reduces the complexity in the buffer storage scheme implemented in the reception handler. This is due to the fact that the reception manager is designed to store the more recent past transmissions, and fixed data size simplifies the indexing scheme for the stored packets.

Provisions have also been made in the link manager to recognize status messages in addition to data messages. The status message is distinguished from the data message in that it only has a packet length of two bytes, and the first byte is a known escape sequence. This allows the link to provide status services to the control process, enabling it to react to special situations. When a status message is detected, the link manager alerts the communication manager via one of the many flags established in the dual-port memory. The only message implemented is the end-of-experiment message, which is transmitted from one vehicle to the other vehicle to signal the end the experiment. Since the link is half duplex, only the lead vehicle can announce the end-of-experiment message. Hence it is up to the lead vehicle to determine the duration of the two-vehicle platoon experiment.

4.2.2. The Communication Manager

The communication manager provides an interface between control processes running on the host computer and the link manager running on the ATcomm board. The communication manager is made up of two tasks, the transmission task and the reception task, to provide independent transmission and reception processing. Although the same manager software is used on both the lead and the following vehicles, the half-duplex nature of the link enables the manager to suspend the reception task in the lead vehicle and the transmission task in the following vehicle. The overall flowchart for the host communication manager implemented in the two-vehicle platoon experiment is shown in figure 12.

Both transmission and reception tasks are executed in the VRTX multitasking environment on the host computer. Similar to the link manager for the ATcomm interface board, the manager software is first developed and tested in a laboratory environment, where two fixed computers and their transceivers are placed at close proximity of each other. The software is later relocated to actual test vehicles to examine their abilities to handle real communication errors.

4.2.2.1. At The Lead Vehicle

The communication manager is integrated as two tasks that run under the main IPCS application. Although only the lead vehicle is required to transmit in this design, both the transmission and reception tasks are included in the communication manager. Moreover, the same manager software is used to provide receiving functions to the IPCS in the following vehicle. However, the execution of the receiving task is suspended in the lead vehicle throughout this experiment.

Transmissions from the lead vehicle contain both the vehicle data and the status of the vehicle. The vehicle data are obtained from the sensors on-board the vehicle through the data acquisition task integrated into the PCS. The data acquisition task also acquires the vehicle status flags (for example, collision alert flag). In addition, a timestamp, which identifies the time at which the data is sampled, is attached to each set of the acquired data/status. The result is organized into a message of data acceptable by the communication link manager. At this time, the transmission task is activated whenever the new message of data is presented for transmission.

Since there is an overhead in the transmission interface task (due to data processing for transmission, and the finite transfer rate of the communication link), it is obvious that the elapsed time between two transmissions should take into account of these overheads. Otherwise, new data will be presented to the communication link for transmission faster than the link can process, and cause eventual data loss in the link. For the two-vehicle experiment, the time between two transmissions coincides with the sampling time, which is controlled by a fixed interval timer. The reason for fixing the timing interval is to preserve a constant sampling and communication period. The data acquisition module is activated when an end-of-interval signal is generated by this timer. It seems intuitive that in order for the control calculations to provide the best performance, the sampling interval should be minimized. Unfortunately, due to limitations inherent in the current version of the VRTX kernel, the available timer is only capable of counting in 55 ms intervals. One consideration for this interval is immediately apparent: 55 ms may be too long an interval to be used in the real-time applications. The later version of the VRTX operating system, however, promises an interval timer capable of counting using smaller intervals. This should provide greater flexibility in the available timing options. For experiments performed in this project, however, 55 ms interval was adequate. The two-vehicle link timing diagram is shown in figure 13.

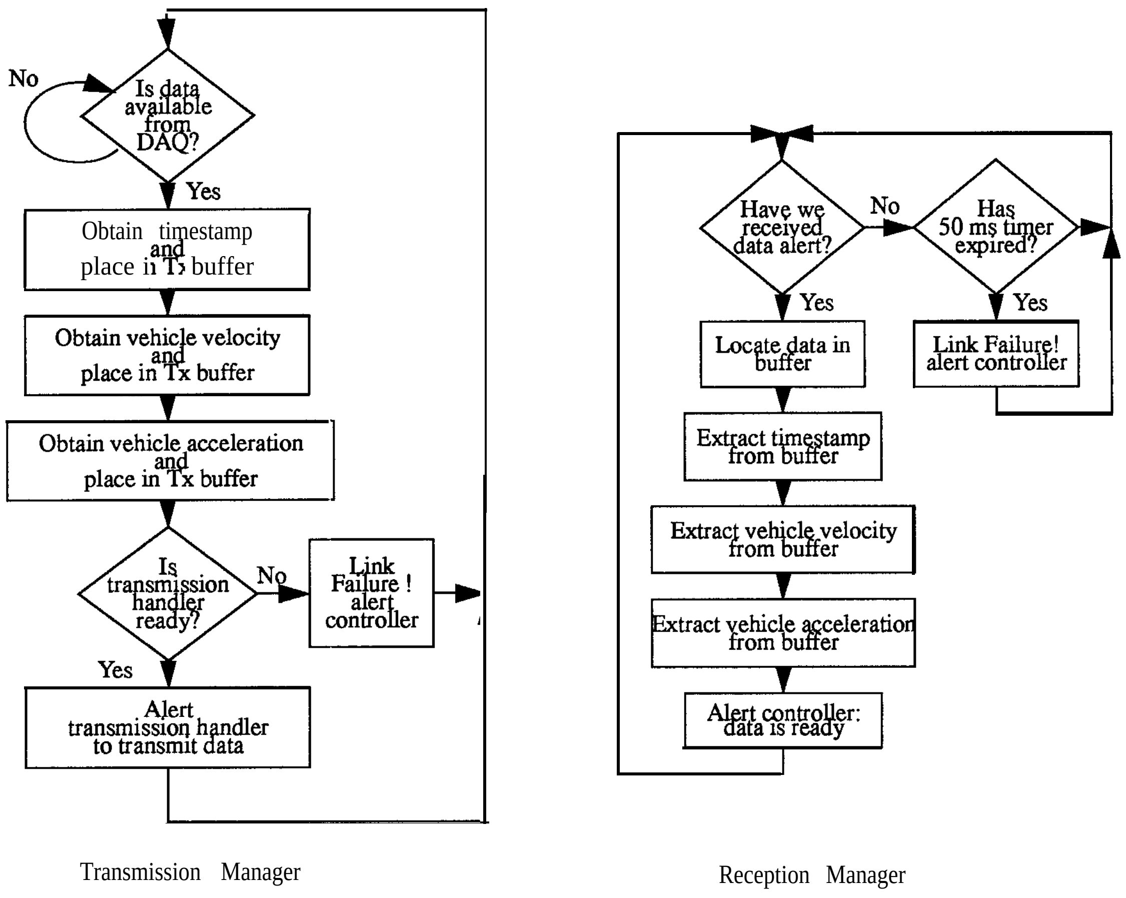

No
Is data available from DAQ?
Yes
Obtain timestamp and place in Tx buffer
Obtain vehicle velocity and place in Tx buffer
Obtain vehicle acceleration and place in Tx buffer
Is transmission handler ready?
No
Link Failure ! alert controller
Yes
Alert transmission handler to transmit data
Transmission Manager
Have we received data alert?
No
Has 50 ms timer expired?
Yes
Locate data in buffer
Yes
Link Failure! alert controller
Extract timestamp from buffer
Extract vehicle velocity from buffer
Extract vehicle acceleration from buffer
Alert controller: data is ready
Reception Manager

Figure 12. How Diagram for Communication Manager (Two-Vehicle Link)

When activated, the transmission task gathers and formats all relevant data into a single data message for transmission. The data message defined for this phase of the experiment has a fixed size of 8 bytes: the first 4 bytes are used to represent the timestamp (a long unsigned integer). The next two bytes carry the vehicle's velocity (a 2 byte integer). The last two bytes are used to represent the vehicle's acceleration (another 2 byte integer). At the same time, the transmission task looks for any special signals, generated by the IPCS, that needs to be transmitted. The signal is actually a two byte escape sequence, designated by an escape character(0x1B) and a sequence identifier. The only signal currently implemented is the End-Of-Experiment signal, which signals the receiving vehicle to end the experiment. If a signal is found, its transmission takes precedence over the transmission of the data message, and the data message is transmitted after the signal message is transmitted. The packet structure is shown in figure 14.

The completed message is placed in to the transmission buffer, which is implemented in a part of the dual-port memory on the ATcomm interface board. This buffer is primarily used for block data transfer between the link ATcomm interface and the host computer. In addition, the dual-memory port memory offers a flexible method to provide access to control flags for both the ATcomm interface board and the host computer. All status offered by the link manager are reflected through different flags located in this dual port memory.

Upon the completion of the message, the transmission task sets the Request-To-Send (RTS) flag in the dual-port memory. By setting this flag, the communication manager requests the link manager to process the message for transmission. The request is granted when the link manager detects that the transmitter is idle, and the link manager immediately processes and transmits the message. When all of the data has been transmitted, the transmission task portion of the manager suspends itself, waiting for another set of data to transmit. The request is denied if a prior data

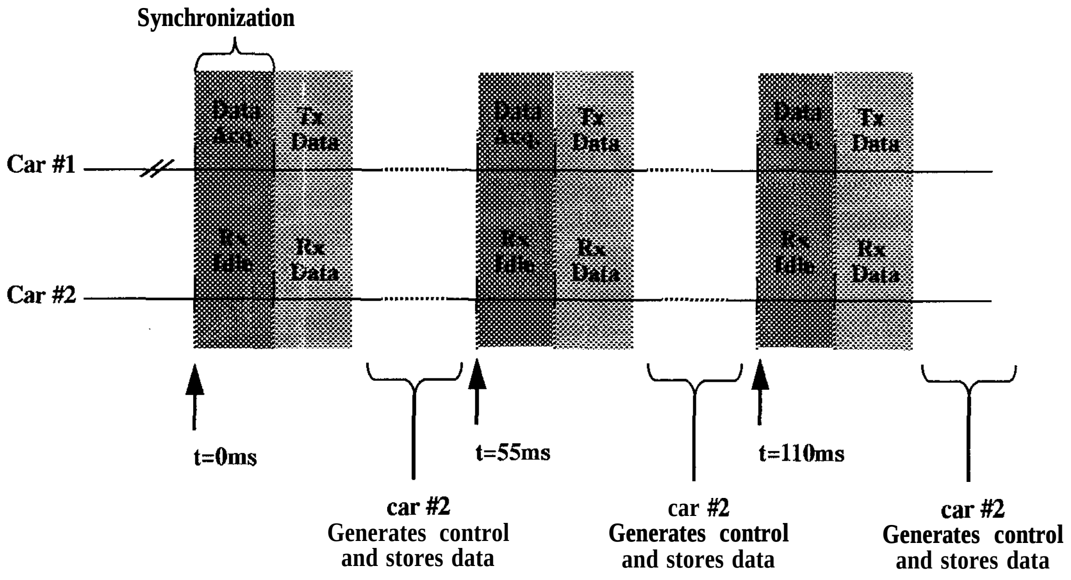

Synchronization
Data Acq.
Tx Data
Rx Idle
Rx Data
Car #1
Car #2
t=0ms
t=55ms
t=110ms
car #2 Generates control and stores data
car #2 Generates control and stores data
car #2 Generates control and stores data

Figure 13. Timing Diagram for Two-Vehicle Communication Link

message has not been completely transmitted. This alerts the communication manager to monitor the amount of time the request is denied. If the request is denied for an interval longer than two control cycles(100 ms), a time-out alert is generated to the IPCS. The link is then considered as severed and the experiment is aborted.

4.2.2.2. At The Following Vehicle

The communication manager at the receiving vehicle monitors for messages transmitted from the lead vehicle. The manager operates only with the reception task and suspends the transmission task. The control calculation and generation task within IPCS is suspended until the arrival of a new data message, since the control task requires lead vehicle data to complete its calculation. In order to avoid the situation in which the receiver endlessly waits for new messages, a time-out condition is implemented in the IPCS where if the following vehicle does not receive any new transmission within a predefined interval, a time-out condition is generated by the IPCS and a communication error alert is issued to the user.

When the reception manager on the ATcomm interface board receives a complete message of data transmission, it informs the main control task in the IPCS. The IPCS, in turn, activates the reception task within the communication manager. The communication manager for the following vehicle is identical to the one used in the lead vehicle; it is written in VRTX and integrated with the control algorithm within the IPCS. The primary responsibility of the reception interface task is to convert the received data message transmitted from the lead vehicle, which is stored in the dual-port memory buffer managed by the reception manager, into a format which can be processed by the control algorithm in IPCS. To achieve this, the reception interface task polls the message completion flag on the ATcomm board and the message error flag, both provided by the reception manager. When the message-completion flag is set, the reception task identifies the location of the data in the buffer. After the data is located, the reception task proceeds

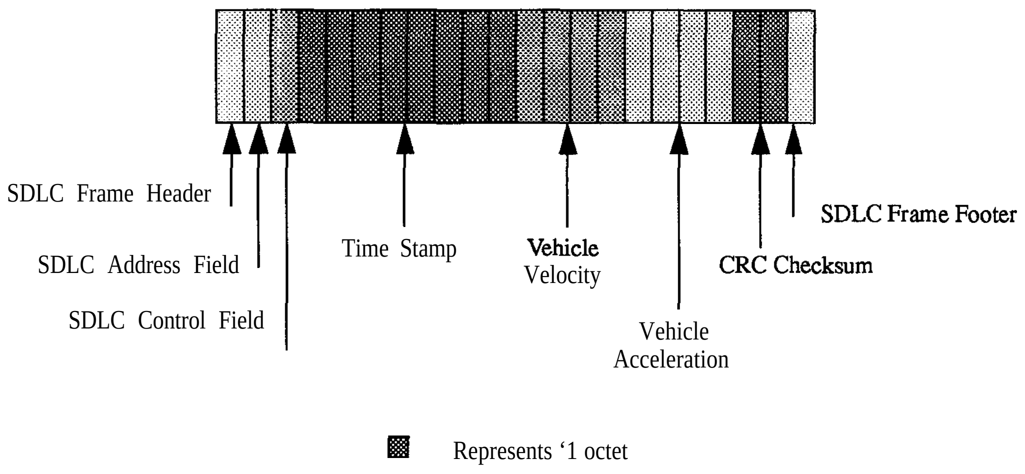


Figure 14. Packet Design for the Two-Vehicle Communication Link

to remove the control information coded onto the data message by the transmission task, and converts the resultant data into a format acceptable to the host control system. At the completion of data conversion, it sends the converted data to the control algorithm, activates the data acquisition task to gather the latest following vehicle information, and resets the buffer location variables, which informs the reception manager to reinitialize the buffer. At this time, the reception task returns to polling the message-completion flag. The process is repeated when the flag is again set.

If the message-error flag is set instead, however, the communication manager immediately determines that an error has occurred in the received message, and the message is discarded by the reception task. The communication manager then sends an alert to the main control task, to notify the control tasks that the message has been lost. Eventually, when the number of failed messages becomes intolerable, a time-out alert is issued by the control task, and the experiment is aborted. Before then, if the condition on the communication link improves, the error status is reset and the reception task returns to accept the next message of data.

The received message may also be a status message. Similar to the data message, the communication manager is alerted to the arrival of status messages via a set of status flags controlled by the link manager. The communication manager is able to distinguish different messages by polling these flags: each message is represented by unique combination of flags. The message is then related to the IPCS for processing. Currently, the communication link has only implemented the End-Of-Test status, which alerts the IPCS for au immediate termination.

4.3. Multiple Vehicle Communication Link

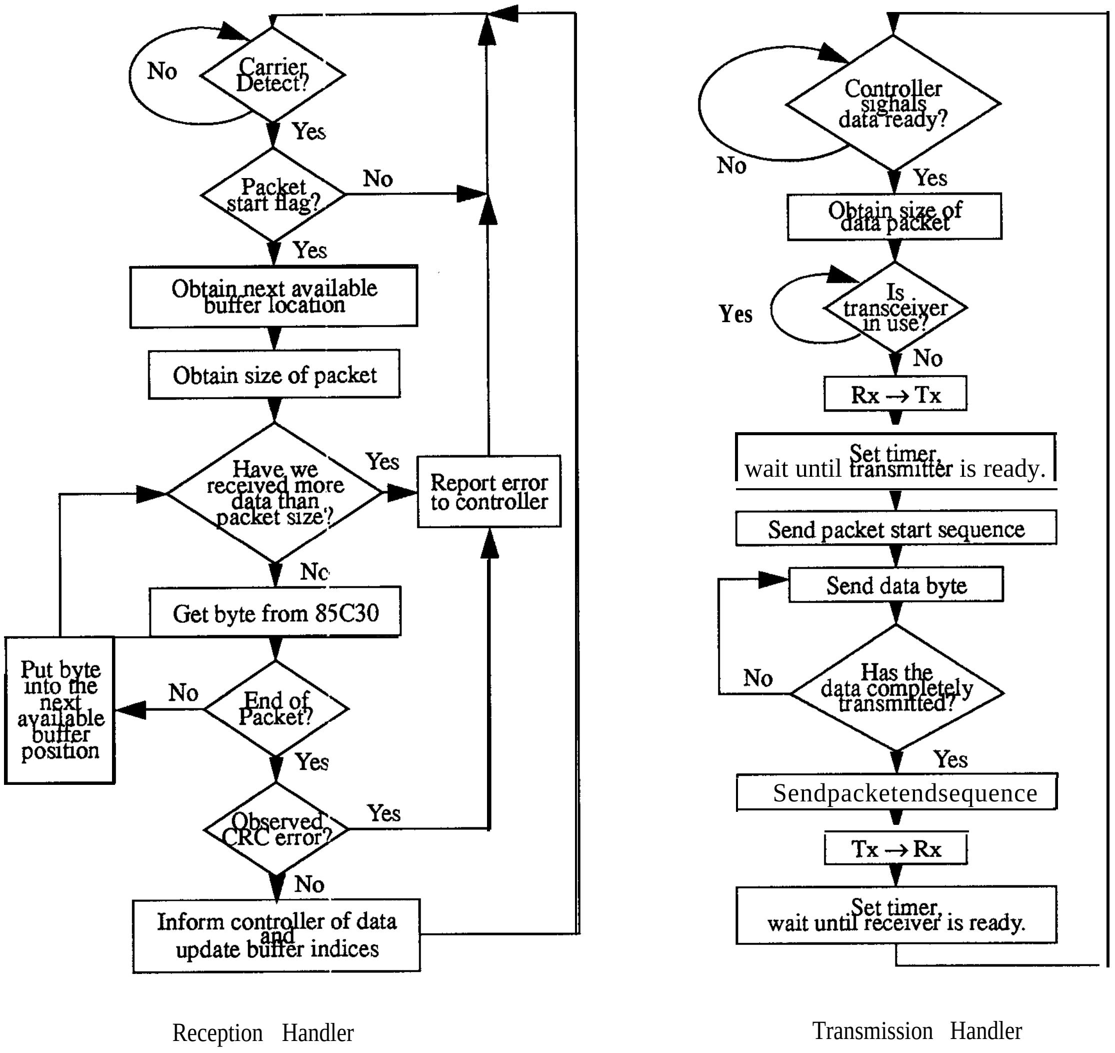
For these experiments, the communication link needs to support a varying number of vehicles participating in the link. It is also expected to provides link monitoring functionalities for the communication link. As an extension of the two-vehicle link, the multiple-vehicle software is implemented similarly: the software is functionally divided into two modules: the communication manager and the link manager, with the communication manager implementing network related responsibilities and the link manager handling link layer responsibilities.

Unlike the two-vehicle software, where different link managers are needed to run in different vehicles, the same multi-vehicle software is designed to be able to run in all vehicles. All functionalities of the communication link are build into this software, and any vehicle-specific functions are activated depending on the parameters specified by the application software running on that vehicle. For example, although all vehicles run the same set of software, only the lead vehicle runs the monitoring functions provided by the communication link.

4.3.1. Link Manager

The multi-vehicle **link** manager provides link layer **functionalities** and low-level communication hardware control. Here, link manager is further divided **into** two modules: the initialization manager and the mode manager. An overview of the structure of the link manager is shown in figure 15.

The link manager **first** initializes all necessary peripherals on the **ATcomm** controller board used for the experiment via the initialization module. This module performs the initialization of the 80186 microprocessor **and** its



No
Carrier Detect?
Yes
Packet start flag?
No
Yes
Obtain next available buffer location
Obtain size of packet
Have we received more data than packet size?
Yes
Report error to controller
No
Get byte from 85C30
Put byte into the next available buffer position
No
End of Packet?
Yes
Observed CRC error?
Yes
No
Inform controller of data and update buffer indices
Reception Handler
Controller signals data ready?
No
Yes
Obtain size of data packet
Yes
Is transceiver in use?
No
Rx → Tx
Set timer, wait until transmitter is ready.
Send packet start sequence
Send data byte
No
Has the data completely transmitted?
Yes
Sendpacketendsequence
Tx → Rx
Set timer, wait until receiver is ready.
Transmission Handler

Figure 1.5. Flow Diagram for Link Manager (Multi-Vehicle Link)

supporting chips. It also programs the 85C30 serial communication controller to perform SDLC communication. Lastly, it clears and sets up the data buffers that the link manager uses in the process of communication.

When the ATcomm board is initialized, its control is given over to the mode manager. The primary responsibility of the mode manager is to provide the scheduling for the two data handlers (the transmission and the reception handlers) over the duration of the experiment. The execution of these data handlers depends on the input received by the mode manager. If the mode manager detects a valid message header from the communication link, it immediately activates the reception handler to handle the rest of the data transmission. On the other hand, the mode manager activates the transmission handler when the manager is informed that a data message is ready for transmission.

The mode manager activates the transmission handler when the host communication manager requests a transmission. Thereafter, all exchanges between it and the host commtmication manager are made within a selected block of 64 KByte dual-port RAM. These exchanges can be made in one of the two forms: byte-size flags and data buffer. Currently, the transmission handler is configured so that all alert flags are kept within the first 16 bytes of this memory while the transmission buffer is designated as the next 256 bytes of memory. In order to achieve the fastest responses, both the flags and the data buffer are constantly monitored by both managers.

As of now, the flags implemented for this interface include the data alert, status message and message size flags. The data alert flag is a two-way information flag. It is primarily used to alert the link manager to transmit the data message stored in its buffer. When set, it indicates that the mode manager is beii idled. When it is reset by the communication manager, it indicates to the mode manager that a new data message has been placed in the transmission buffer and it should be sent immediately. Lastly, if the communication manager detects the flag to be reset, it indicates that the link manager is unable to process further data messages for transmission, due to its processing of a newly received message or transmitting the past message. The mode manager constantly polls this flag to determine if there is new data for transmission. This flag is also used as a test for the integrity of the Proxim transceiver: the transceiver has probably malfunctioned if the flag is detected in its reset state for an excessive amount of time. The appropriate action can be taken when this time-out occurs.

The link manager also implements a status message flag. This flag is usually set by the communication manager. Its value indicates the particular status message that the communication manager is desired to send. This flag is used in conjunction with the data alert flag. If the data alert flag is reset and the status flag is of a non-zero value, an appropriate status message would be composed in the transmission buffer and transmitted by the transmission handler. Currently. the only status message implemented is the end-of-experiment message. This flag serves to alert receiving vehicles to end the experiment.

Lastly, one of the flag bytes is assigned to represent the size for the transmitted message. Unlike the two-vehicle software, the multivehicle link manager is capable of handling messages in variable sizes. Thus, the message size flag is used to specify the exact size of each message. Since all of the messages encountered in this experiment have message sizes less then 256 bytes, a single byte is sufficient to describe the sizes of those messages.

To provide a temporary buffer for the of storage of the next to-be-transmitted data message, the transmission handler implements a transmission buffer. The size of the buffer is chosen to provide the minimum amount of storage to accommodate all messages used throughout this experiment, but can also be conveniently represented as a one byte quantity. This enables the sire of the message to be stored with the rest of the single byte flags. However, the buffer is currently configured for storage of a single message. All further storage requests are denied until the previous buffered message has been transmitted.

When the mode manager detects that the communication manager has set the data alert flag, it immediately starts the transmission handler. As its first action, the handler waits for the channel to be devoid of the F’roxim transceiver carrier; if a transceiver is activated while another is transmitting, all data transmitted after this moment are corrupted. Although this is unlikely to occur in this configuration since all transmissions take place after the previous message has be completely received, this carrier detect/collision avoidance is implemented as a precautionary measure.

The transmission handler then reconfigures both the 85C30 SCC and the Proxim transceiver for the transmit mode. Unlike the transceiver, the 85C30 is easily switched into the transmission mode. Due to its half-duplex nature, the transceiver suffers performance degradation when it is forced to switch between its operating modes. The ***transmit*** ready signal which signals the readiness of the transceiver, is found to be defective and unusable. Since all data sent to the transceiver will be corrupted if it is not in the transmission mode, it becomes necessary for the transmission handler to be certain that data is sent to the transceiver only after it is operating in the desired mode. Since the transceiver provides no feedback regarding its state of operation, the handler can only estimate the transceiver’s readiness. To do this, the handler takes the time listed in the specification for the transceiver to switch from one mode to another, which is about 1 ms, and adds a safety margin of 2 ms to derive the time the handler needs to wait after it commands the transceiver to switch from reception to transmission mode. The 2 ms safety margin is divided into two 1 ms intervals, one of which is used to allow the transceiver to settle into the transmission mode and the other to allow the reception handlers to detect the transmitter’s carrier. The 1 ms for carrier detection is mostly due to the latency in the execution of the CD routine. Thus, the handler needs to sit idle for approximately 3 ms before it is able to transmit the actual data. This is unfortunate since the switching latency between operating modes made implementation of the interfacing software more difficult. Moreover, it reduces the amount of time available to an individual vehicle to handle their communication duties.

While the Proxim transceiver is switching into the transmission mode, the transmission handler attempts to determine if the host wants to transmit a status message or a data message through the status message flag. If a status message is signaled for, the corresponding transmission sequence is composed and becomes the next to-be-transmitted message. Otherwise, the transmission handler notes that the data message stored in the transmission buffer is the next transmitted message.

When the transceiver is ready for transmission, the handler quickly transfers the composed message to 85C30. The 85C30 then places the message together with the other relevant parameters into a SDLC frame and sends it to the transceiver. When the transmission is completed, the handler releases the control of the ATcomm interface back to the mode manager. The transceiver is also switched back to the reception mode, and this transition takes an additional 2 ms to account for the actual transition and settling. At this time, the mode manager returns to its idling state, monitoring transmissions in the communication channel. The entire transmission lasts approximately 6 ms for a data message of 8 bytes.

The data messages are designed as to reduce the chance for corruption and to maximize the message capacity without introducing complex schemes. Each data message is enclosed by a pair of flag control characters. These control characters are different from the SDLC flag sequence, which delimits the entire frame.Thus, a higher level of error-checks supplements the CRC checker provided by the SDLC protocol. The boundary flags and status messages are implemented as escape sequences of two or more byte sequences that starts with the byte '0x1B.' In order to avoid confusion between the data byte '0x1B' and the escape sequence, a technique similar to the SDLC's "zero-bit insertion" is implemented: all '0x1B' data bytes are converted into 2 '0x1B' bytes at the transmitter and are converted back to a single '0x1B' byte at the receiver. Any other sequences that start with the byte '0x1B' are interpreted as escape sequences.

The Proxim transceiver is configured for reception while it is not transmitting; this allows the resident mode manager to monitor the network traffic and react quickly to any broadcasted messages. While in the reception mode, the transceiver alerts the mode manager of an incoming transmission by setting the transceiver's carrier detect (CD) flag, indicating it has detected the transmission carrier from another transceiver. As it turns out, the CD flag is not a reliable indication of the carrier: the transceiver often sets the CD flag even when the carrier is absent. This forces us to add a round-about method of carrier detection: the presence of the carrier can be inferred from the CD flag by counting the number of samples the CD flag is set over an interval: carrier is considered to be detected if the CD flag is discovered to be set for more than an upper threshold. On the other hand, if the number of samples that are set falls below a lower threshold, then it is assumed we have not detected a valid carrier. Upon the request for the status of CD, the threshold test is repeatedly performed until the test is passed, and the CD flag is *debounced.*

Currently, the mode manager performs 40 CD flag sampling operations per interval. In addition, it sets the upper threshold to 92% of the samples in the interval and the lower threshold to 8%. Therefore, the CD is considered to be valid if the CD flag is set more than 36 of the 40 samples. On the otherhand, carrier is considered to be absent if the CD flag is set for less than 4 of the 40 samples. The number of samples per interval and the thresholds are determined through trial-and-error. These quantities are chosen to allow reliable determination of the carrier in the shortest interval.

It is necessary to accurately determine the presence of the carrier. If the 85C30 SCC is set to the reception mode while the carrier is not present, the SCC would receive the artifacts produced by the automatic gain control (AGC) mechanism on the transceiver. Since the carrier is not present, the AGC assumes that the received signal is too weak for processing and attempts to amplify the signal. The resulting data is interpreted as a bit stream of1s. Unfortunately, this happens to be the SDLC 'break' sequence, where the SCC needs to be reset to restore its functions. Unfortunately, resetting the SCC also causes the transceiver to operate in the transmission mode. Without the corresponding data transmission, this inflicts a similar idling condition on the other SCCs used in the experiment. Thus, it is necessary to have an accurate detection of the carrier.

When the carrier is detected, the mode manager switches the SCC into the reception mode and waits for the message-start flag delimiter. The mode manager executes the reception handler when it detects the delimiter flag sequence. The reception handler continues to process all information received after the flag and transfers them to the reception buffer. After a second delimiter flag sequence is received by the handler, it is assumed that the message is completely received and the reception manager ignores rest of the transmission. However, if the CRC-error flag is detected or the SDLC end-of-frame flag is set before the reception handler detects the second delimiter sequence, the message is considered to be corrupt and an error flag is set to alert the communication manager.

With the message completely in the buffer, the handler then parses the content of the message to determine the nature of the message. If it detects the escape sequence '0x1B' as the first byte and a non-'0x1B' as the second byte of the message, the received message is interpreted as a status message. The handler sets the appropriate status flag and alerts the communication manager of the message. Otherwise, the message is considered to contain data only and the appropriate flag is set to alert the communication manager. At this time, the reception handler returns the control of the ATcomm interface back to the mode manager.

All exchanges between the mode manager/reception handler and the communication manager are made through a bank of reception flags and reception buffer located in the dual-port memory on the ATcomm interface. Although both the transmission and the reception flags are located in dual-port memory, they reside in two different 16 Kbyte memory pages. Thus, it is very unlikely for transmission-related exchanges to interfere with the data and flags in the reception memory page, and vice versa.

Flags and data buffer for the reception interface are implemented in a structure similar to the transmission data interface. The first 16 bytes of the reception memory page are used to represent the flags used for the information exchange while the next 256 bytes are used as a reception data buffer. To provide both the current and recent data messages, the 256 bytes are structured into a circular buffer. As with the two vehicle software, this buffer is essentially a series of storage spaces managed by the buffer manager residing in the receive handler. Each storage space can be of variable number of bytes depending on the size of the received data message. When a message is accepted by the reception handler, the buffer manager stores it in the first available buffer space. Moreover, the buffer manager stores subsequent messages directly after the previous stored message. This is continued until the entire buffer is filled, at which time the buffer manager releases the space storing the oldest message for new receptions. Thus, the buffer manager creates a circular buffer for the received messages.

To assure access to the latest correct message, the buffer manager updates the current location of this space only after the entire message has been successfully stored in the circular buffer. Two indicators are used to identify the latest received message: the first indicates the relative memory location of the message within the buffer block while the second indicates the length of that message. Together, they pinpoint the location and the length of the message, allowing it to be retrieved from the buffer. Moreover, since all messages are expected to be less than 256 bytes long and only 256 bytes are allocated to the buffer, the two indicators are conveniently represented by two bytes.

An important advantage of using a circular structure for temporary data storage is that it relaxes the constraint on the time available to the communication manager to access the received message without losing the message. Although not present in the current implementation of the buffer manager, an indexing system can be added to keep track of all the messages stored in the circular buffer, regardless of the size and the contents of the message. This enables the communication manager to miss a number of received messages but still have the capability to retrieve the missed data. However, if the communication manager neglects to process the buffered messages for an extended amount of time, they could be overwritten by newer messages. Although it is unlikely for the communication manager to miss that many messages, any occurrence of this would clearly indicate that the communication manager needs to streamline its operations. For applications where a larger data storage is required, the overall size of the buffer can be easily expanded from 256 bytes to 16 Kbytes.

In addition to the reception data buffer, the mode manager/reception handler also implements some alert flags for the communication handler to describe status of the reception handler. These flags includes: data alert flag, status message flag, message location indicator, message size indicator, and message error flag. The data alert flag indicates that a new message has been successfully received, waiting to be transferred to the application. The message size indicator shows the size of the received message while the location describes the whereabouts of the message in the buffer. Lastly, the message error flag alerts the communication manager that it has received a corrupted message.

Numerous avenues are left in the link manager for future implementation of the functions not yet designed. For instance, a more sophisticated error-checking and correction algorithm can be easily integrated with the link manager. However, it should be noted that the primary responsibility of the link manager is to provide control over the transceiver and to perform the necessary transmission and reception tasks. Therefore, the lii manager should not be burdened with tasks that are superfluous to the primary functions of the manager. This is especially true for wide bandwidth applications, where the link manager only has enough time to provide rudimentary data processing functions. Thus, any new communication functions, including error-checking and corrections, should be placed in the communication manager, where more system resources are readily available.

4.3.2. The Communication Manager

The communication manager software is implemented in the upper level protocol-related functions as opposed to the device level controls which are implemented by link manager. The separation between the communication and the lii manager allows them to operate independently from each other, which enables implementation of different network protocols over the same low level link control software. Furthermore, the communication manager is further divided into a transmission manager and a reception manager, each of which is an independent VRTX task, providing their respective network services to the host controller.

A polling-based scheme is selected for the communication manager. In this scheme, whenever the manager is activated by the real-time control, the manager continuously monitors the status of the link manager, enabling it to quickly respond to incoming alerts. Unlike an interrupt-driven scheme, polling reduces the complexity of the software by eliminating the interrupt service routine, ISR, but at the expense of having a a heavier system overhead due to the constant monitoring. Thanks to the amount of system resources available, however, the amount of overhead is negligible. The flow diagram for the communication software is shown in figure 16.

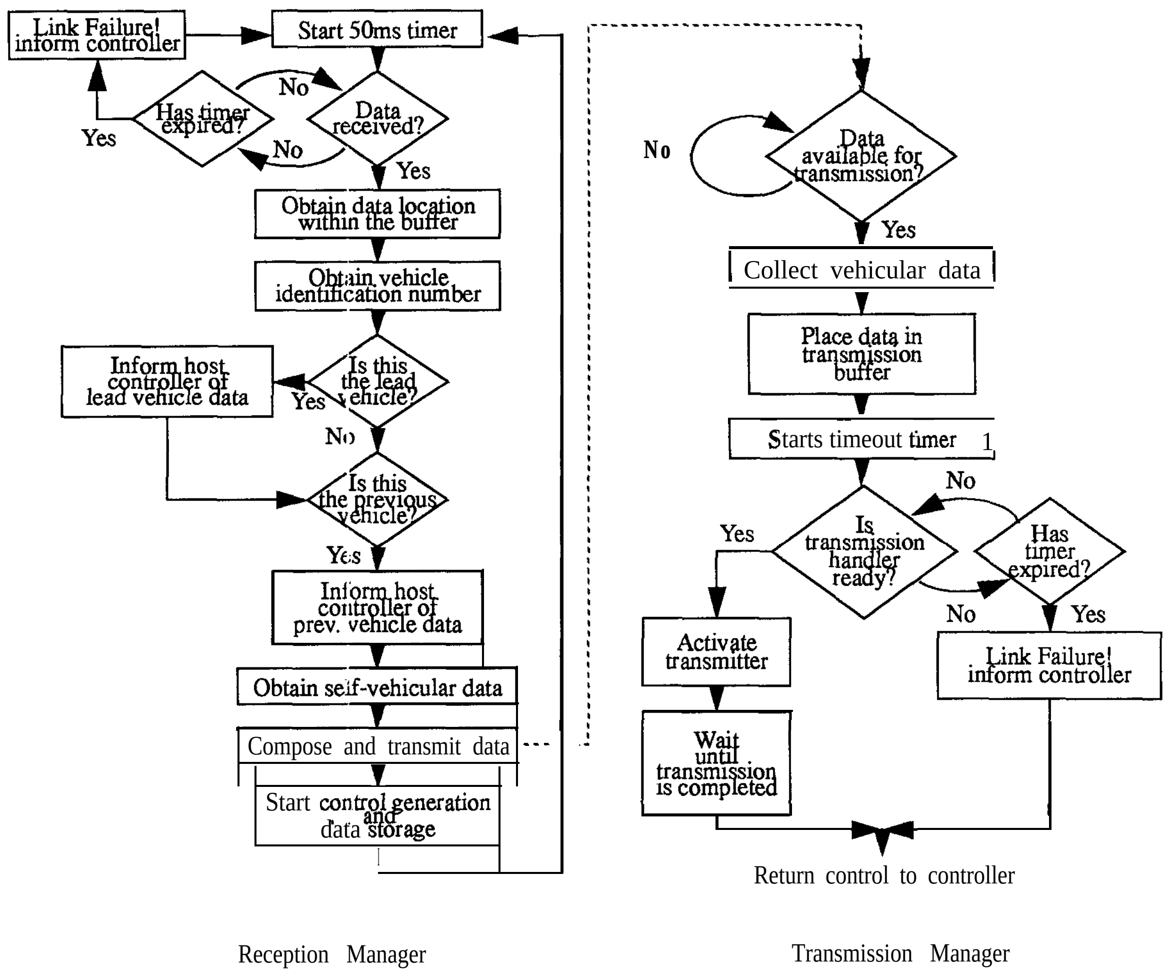

Link Failure! inform controller
Start 50ms timer
Has timer expired?
Data received?
No
No
Yes
Yes
Obtain data location within the buffer
Obtain vehicle identification number
Inform host controller of lead vehicle data
Is this the lead vehicle?
Yes
No
Is this the previous vehicle?
Yes
Inform host controller of prev. vehicle data
Obtain self-vehicular data
Compose and transmit data
Start control generation and data storage
Data available for transmission?
No
Yes
Collect vehicular data
Place data in transmission buffer
Starts timeout timer
Is transmission handler ready?
Has timer expired?
Yes
No
No
Yes
Activate transmitter
Wait until transmission is completed
Link Failure! inform controller
Return control to controller
Reception Manager
Transmission Manager

Figure 16. Flow Diagram of Communication Manager (Multivehicle Link)

The communication manager provides an ordering of transmissions, like that of a token-bus network. This is necessary to avoid collisions from multiple active transmitters, which may result in a non-deterministic link recovery time. Therefore, transmissions are made using the following order: the right to transmit is passed from one vehicle to another in the form of a token. whenever the token is returned to the lead vehicle, a new round of transmissions is started. The ordering of transmissions is designed according to the physical location of the vehicle: vehicles in the front of the platoon transmits before. those in the rear. Since the vehicle identifications are given according to their location, this enables vehicles with lower vehicle identification to transmit and generate control before those with higher identification. The timing diagram for the token passing is shown in figure 17.

The manager in the lead vehicle takes on the additional management task of monitoring the integrity of the link. For this project, link integrity is indicated by the existence of traffic over the link. The link is considered to be severed if the monitoring station does not detect any transmission in an interval of two control loop time. The resulting timeout restarts the current control loop and initiates a new round of transmission. Furthermore, an error message is displayed on the system console, thereby allowing the user to respond to the timeout. In the current experiments, the user aborts the experiment when a timeout signal is encountered. Note that all vehicles performs channel monitor-

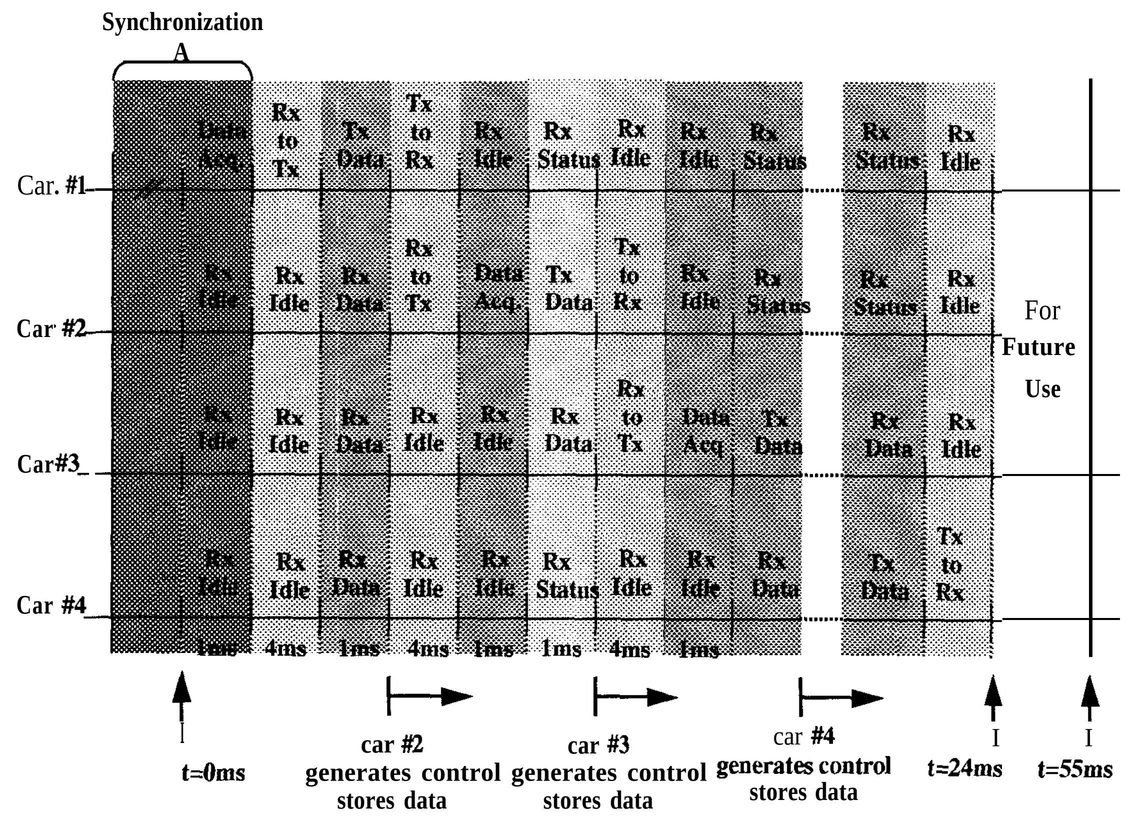


Figure 17. Timing Diagram for Multivehicle Communication Link

ing while they participate in the network. However, only the lead vehicle is selected as the monitoring station. This selection is made for the following reasons: first, more system resources available for performing this task on the lead vehicle than on any other vehicles since it performs no control generation. Moreover, since a timeout may re-start a round of transmission, the lead vehicle must be notified of this re-start. Thus, using the lead vehicle to monitor the link effectively eliminates any communication overhead as a consequence of the transmission of timeout message to the lead vehicle. Finally, since the link protocol for platoon merging is planned to be implemented in the lead vehicle, this consolidates link management to a single vehicle.

4.3.2.1. At The Lead Vehicle

The lead vehicle starts each round of transmission by sending the data it collected to the communication manager, which in turn activates the transmission manager. The transmission manager assembles the necessary information to form the data message. For this experiment, the vehicle data is composed of the vehicle's velocity and acceleration in the form of two two-byte integer. Next, the manager combines these data with the current timestamp (another two byte integer) and the vehicle identification number (1 byte integer). Finally, the message is prepended with an one byte quantity to indicate the length of the entire message, forming the desired message body of 8 bytes. The body is then enclosed with message start and end flags, defined by two-two byte escape sequences, to form the complete message. An escape sequence is defined as a sequence of bytes that starts with the escape character, 0x1B (hexadecimal); the sequence is used to distinguish transmission control codes from the actual data. To avoid any confusion between a data byte "0x1B" and a control byte "0x1B," byte stuffing is employed so that a data byte is now represented as an escape sequence of 2 "0x1B" bytes. The original data is returned as byte stuffing is reversed at the reception manager. The complete SDLC packet is formed when the SDLC protocol controller adds an additional6 bytes to the packet, for a total length of 18 bytes per packet. The structure of a data packet is shown in figure 18.

After the data message is formed and downloaded to the transmission buffer on the ATcomm board, the transmission manager alerts the link manager to transmit the stored message. At the same time, the control mechanism on the lead vehicle activates a fixed interval timer to maintain a constant loop time and for timeout alerts generation. From various simulations, the loop time has been determined to be 55 ms. At this point, the transmission manager suspends itself, waiting to be called again for the next transmission.

The transmission manager may also transmit a status message. A status message is broadcast from one vehicle to all others on the link to inform them the occurrence of a condition. The message body is composed by the controller, describing the status it wants to broadcast. The transmission manager then adds an escape character to the message, to inform the reception manager to forward the message to the controller. Since the message can be of variable length, a message size byte is prepended to the message. Lastly, the message is enclosed in the message start and end flags, and sent to the link manager for transmission.

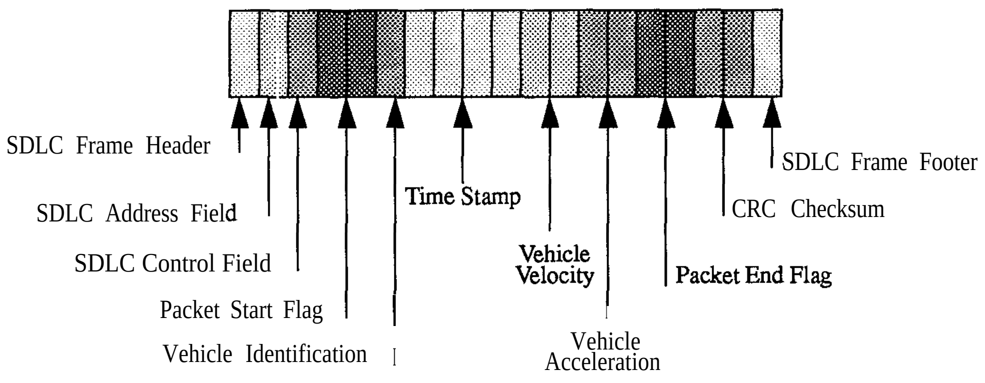
Between data transmission and the expiration of the loop timer, the manager monitors all messages the link carries through the reception manager. None of the data messages is passed to the controller, since the lead controller does not need to use data from other vehicles. If the transmission from the last vehicle is not detected by the reception manager within one loop interval, a timeout signal is generated. The loop, however, is restarted as the lead vehicle transmits its latest sampled data. If the timeout signal persists over several loops, the manager then declares that the link has been severed and requests the host controller to take the appropriate actions. Current implementation calls for the termination of the experiment when the timeout signal exists for 20 consecutive loops.

The only instance at which the reception manager forwards a received message to its host controller is when it receives a status message. As noted previously, a status message is represented by an escape sequence and can easily be recognized by the reception manager. This message is passed to the controller so the appropriate actions can be taken at its earliest convenience.

4.3.2.2. At The Following Vehicle

As with the lead vehicle, the communication manager for the following vehicles is divided into two complementary VRTX tasks: the transmission and reception manager. The transmission manager is activated only when the controller needs to perform transmission. Otherwise, it is suspended to conserve system resources. The reception manager, on the other hand, implements a polling-based scheme, which monitors the link manager for newly received messages.

The communication manager starts by activating the reception manager, which monitors transmission from the other vehicles in the platoon. Meanwhile, it suspends the transmission manager, since this vehicle has not received the right-to-transmit token. Of all the transmissions the reception manager receives, the manager is only



▨ Represents 1 octet (8 bits)

Figure 18. Packet Design for the Multi-Vehicle Communication Link

interested in the data messages transmitted from either the lead vehicle or the vehicle immediately preceding the current vehicle (which the control algorithm needed), and status messages from any vehicle. Data transmission from vehicles other than the two listed above are uniformly masked from the controller.

When the reception manager receives the lead vehicle's data message (by examining the vehicle identification included in the message), it immediately informs the host controller to start a new control loop, and resets its 55 ms control loop timer. It also attempts to extract the data from the received message and forwards it to the controller. Since the transmission is broadcast in nature, this transmission synchronizes the control activities in all of the following vehicles. When the data is completely extracted, the reception manager returns back to its link monitoring activities.

If the data transmission is determined to have been transmitted from the preceding vehicle, then the data within the packet is extracted and passed on to the controller. It also informs the controller that it has received the right-to-transmit token, and it is ready to perform its next transmission. At this time, the controller is expected to collect its own vehicle data and generate the appropriate controls. In addition, the controller needs to transmit a status message instead of a data message, which is subject to the vehicle's status. In order to allow the maximum number of vehicles to participate in the communication, each vehicle is allowed to make only one transmission in each transmission loop. Therefore, the right-to-transmit token is transferred with the vehicle's transmission. If a status message is needed to he transmitted, then the composed data message for this particular loop is removed from the memory.

At this point the transmission manager is activated to compose the desired message. Both data and messages are formatted in an identical layout as described in the previous section for the lead vehicle's wmmunication manager. The composed message is transferred to the lii manager, which performs the actual transmission, and the passing of the right-to-transmit token.

If the received message is a status message, its content is immediately forwarded to the controller for processing regardless of its origin. Currently, the only status message implemented is the TERMINATE message. The reception of this message indicates that its sender has terminated its automatic control, and any controller that receives this message should take the same action.

Each of the communication manager on the following vehicles implements a local timeout detector. This is used to detect the lack of activity from the preceding vehicle. This timeout signal is generated when the manager fails to receive any transmission from the preceding vehicle before the control loop timer expires. When this occurs, the manager immediately informs the controller that the link has timed out. The controller then takes whatever action is appropriate for this situation.

5. Link Performance

The wmmunication link is first tested in laboratory conditions and then in real-world situations. While not spectacular, the results obtained do give us insights toward this implementation of the wmmunication link.

5.1. Under Laboratory Conditions

The link is first implemented and tested indoors, since this is where all the development and testing tools are located. The tests took place in an office-sized (about 15 feet by 20 feet) room in Richmond Field Station, which is probably constructed out of wood. The 386 PC-compatible computers and its communication equipments are placed next to each other, with the computer powering both the ATcomm interface board and the Proxim radio. Furthermore, there are no substantial obstacles that would block the line-of-sight transmissions from the cellular antennas connected to the Proxim radios.

Fist, the two vehicle communications lii is tested. This series of experiments are performed in the Decemher of 1990. While two computers are used in place of the actual vehicles, the tests are performed using the completed two vehicle control software. The data acquisition mechanism is manipulated in such a way that it always returns a constant value back to the control application. The control application is run on each of the computers, with one configured to be the lead vehicle and the other to be the following vehicle. As with the design, a new control/transmission loop is started every 55 ms. Within this 55 ms, we expects the lead vehicle to transmit one data transmission while the following vehicle to generate one set of controls. Further, ah transmissions and any data corrupted flags are saved onto a magnetic media for off-line analysis.

We have performed a number of the above tests over a range of durations, varying from a minute to a few minutes. From viewing the results obtained from the testing, it is apparent that the radio-based wmmunication link is adequate to support the demands of the a two-vehicle control application. We have not experienced any communication errors over the duration of the test, which seems to indicate the reliability of the radios operating at close quarters and over a duration of a few minutes.

We have also experimented with how the link would recover once it detected the link is being severed. To simulate a severed link, we start the experiment with the above setup, and then we suspend the transmitting wmputer. Thus, as far as the receiving vehicle is concerned, the link has been severed. We observed that the communication manager at the receiving vehicle alerts the controller after the timeout condition has occurred. However, as soon as it detects the next transmission from the transmitting computer, the manager cancels the timeout signal and reconstructs the link.

The multiple-vehicle wmmunication link is next tested. This took place in May of 1991. Here, four computers are used in place of the proposed four vehicle platoon, and the appropriate software is loaded to all of these computers. As with the two-vehicle experiments, the wmputers are placed quite close to each other. Therefore, it is expected that there should not be any data corrupted messages in the transmission log. In addition, the link is be used to support a variable number of stations, ranging from two stations to four. This tests the link's ability to support platoons of different sizes.

The results from the multi-vehicle experiments using the above setup are as expected. No data transmission errors are observed, and the link is able to support the communication needs for platoons with different sizes (up to four vehicles) within 55 ms. Further, we examine how the computers that are using the link react when the link is lost. To simulate this, we shut one of the computers off, and proceed to observe the link's reaction.

Our observations are as expected. If the lead computer has not been shut off, then we observed that each of the computers ***behind*** the malfunctioning computer exhibits the timeout signal, at which point the controller on those vehicles reuses the previous successfully received data for its wntrol calculation. In addition, the ***lead*** vehicle also noted that the link has been severed, and it also generates a timeout alert to its controller. However, as the loop timer expired on the ***lead*** computer, a new loop is started and a new data message is transmitted, which in turn resets the wntrol loop at the ***following*** computers. When the malfunctioning computer is put back into operation, the link is recovered with the next round of transmissions, starting with the one from the ***lead*** computer.

From calculations and observations, we concluded that it takes approximately 8 ms on the average for each vehicle to perform its wmmunication duties. This is consistent with our theoretical estimate where 3 ms each are used to wait for the Proxim radio to switch from receive operation mode to transmit operation mode, and vice versa, 1 ms for the data acquisition, and 1 ms for the actual data transmission.

5.2. In the Real-World

With the success of the link operating under the laboratory conditions, we are quite puzzled by the poor performance of the communication liis when placed in real vehicles. The vehicles we used for testing the communication links are all donated to us by Ford. They consist ***of*** three Ford *Towncars* and one Ford Crown ***Victoria. The*** power for the computer is obtained from an DC-to-AC power converter installed in each of the vehicles. The radio transceivers are all equipped with an antenna extension, to allow the transceivers to receive signals without attenuation caused by the vehicle's metal body.

We first performed the two vehicle communication tests over various stretches of roadways in Richmond Field Station (see figure 19). This is experimented in March of 1991. The vehicles are driven in such a way that the antennas always have a line-of-sight between them. Lastly, we varied the separation between the vehicles, to observe the effective range of the transceiver. Later, the test is repeated on a stretch of highway in San Diego. We observed some interesting results.

With the separation of antennas ranging from 50 to 100 feet, the lii is still able to support the traffic needed by the control algorithm. The link, however, exhibits a significant number of corrupted packets (approximately 1%)

out of all packets that were transmitted. Given that a typical experiment lasts approximately 45 seconds, at 55 ms per loop and 16 bytes (the complete SDLC frame) per transmission per loop, the link would have carried approximately 115,200 bits (45 sec * 20 loops / sec * 16 bytes/loop * 8 bits/byte) of information over this experiment. Assuming that each corrupted packet contains exactly one corrupted bit (the best scenario), this gives a bit error rate (BER) of 1 in 12800 bits. This is a factor of 10 worse than the specified BER of $1e^{-5}$, and is a drastic change from its behavior in the laboratory. We are unable to explain discrepancy in the high BER experienced by the link when the equipment is used in a non-laboratory environment. It had been suggested that the high BER may be caused by radio-frequency interferences generated from various electrical components on the vehicles. Another explanation brought forth accuses the noises created by the DC-to-AC converter, which made the delicate electronics on the transceivers much less reliable. Neither of these explanations has been verified, however.

Next, we increased the separation between the two antennas, attempting to find the range of the Proxim transceivers. By gradually increasing the distance from the lead vehicle to the following vehicle, we are able to observe the percentage of successful transmissions versus the total number of transmissions. We found that the transceiver has an effective range of approximately 700 feet, or about 1/8 of a mile. The link is unable to successfully transport data between the vehicles if the vehicles are separated by more than 700 feet. The closer the two transceivers, the better the link performs. This observed range is, again, much worse than the quarter mile range specified by

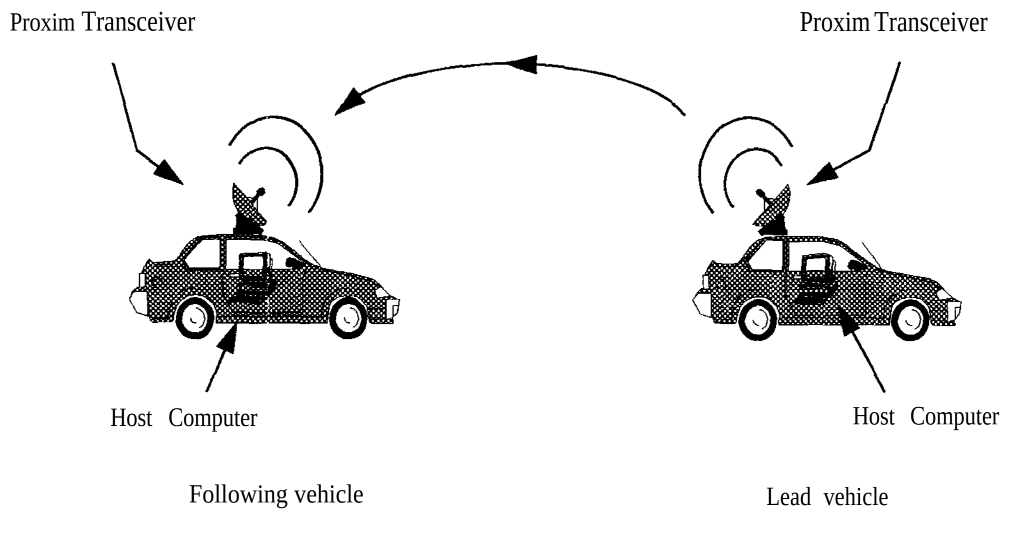


Figure 19. Two Vehicle Communication Setup

the transceiver. Furthermore, the range seems to be reduced while the vehicles are in motion. We observe that if one vehicle is set in motion while the other remains stationary, the average number of corrupted packets is much greater than when both vehicles remain stationary. Several suggestions have been offered in an attempt to explain the low range measurements. For instance, this may be caused by using the suboptimal antenna for the Proxim transceivers, or we have not supplied enough power to the transceiver. The latter may be disproved, when we are able to measure the desired voltage drop across the transceiver's power supply. As of the time of this writing, we have not discovered a way to increase the range of the transceiver.

The above findings have proven to be much less than desired. With a higher than expected bit-error-rate, the desired performance of the link may be placed in serious jeopardy, as the link may have a hard time to maintain the desired rate of throughput. With a shorter than expected range, this limits the size of the platoon and the separation between vehicles that can be supported by the platoon. (for example: from the previous paragraph, the distance between the lead and the last vehicle must be under 700 feet. If two consecutive vehicles are separated by 100 feet, then the platoon can only support 7 vehicles). As a consequence, these limitations may significantly contribute to setting a lower upper limit on the number of vehicles that can be supported by the link.

We then proceeded to the multiple-vehicle control experiments (see figure 20). Due to a problem in obtaining the vehicles, experiments involving three vehicles were delayed to around February of 1992. The four-vehicle experiments didn't take place until the end of May, 1992.

As with the two-vehicle experiments, the multi-vehicle experiments are conducted in two locations: a straight roadway in the Richmond Field Station and a stretch of highway near San Diego. One communication unit is installed in each of the vehicles. Since four vehicles are allocated for this experiment, it is also the maximum number of vehicles the link needs to support. Since the range and the BER information have been observed in the two-vehicle experiments, the primary goal for these experiments is to determine if the link is capable of supporting the volume of data traffic that is demanded by the wntrol algorithm. Specifically, we are interested in observing how the link behaves in a real-world multiple-access environment, under real-time wnstraints.

Unlike the two-vehicle communication link, the multiple-vehicle link needs to address the problem of multiple access. Here, multiple transmitters are given access to the shared medium. To avoid collisions, the link is designed to regulate the order of transmission. Although the design worked successfully in the laboratory, we are unsure of how it behaves in a real-world environment. We are also looking for the amount of time that is needed for the receivers to detect the carrier from any particular vehicle. Since the vehicles are now much farther than they were in the laboratory, we expect that it will take longer for the vehicles to detect a valid carrier. We would like to see how this affects the performance of the link.

The results are quite interesting. From our observations, it takes almost the entire 55 ms for all four vehicles to perform one loop worth of data transmissions. If accurate timing is not strictly maintained, it is very likely that the last vehicle won't be able to transmit within the 55 ms interval. Although a degradation in link's performance is expected, we did not expect that the link would use almost the entire 55 ms for data transmission. This is a distressing result, since it eliminates the possibility of adding more vehicles to the platoon. Moreover, it also renders adding network extensions (such as providing transport between two or more platoons) much more difficult. More time must be made available before the link is able to be extended.

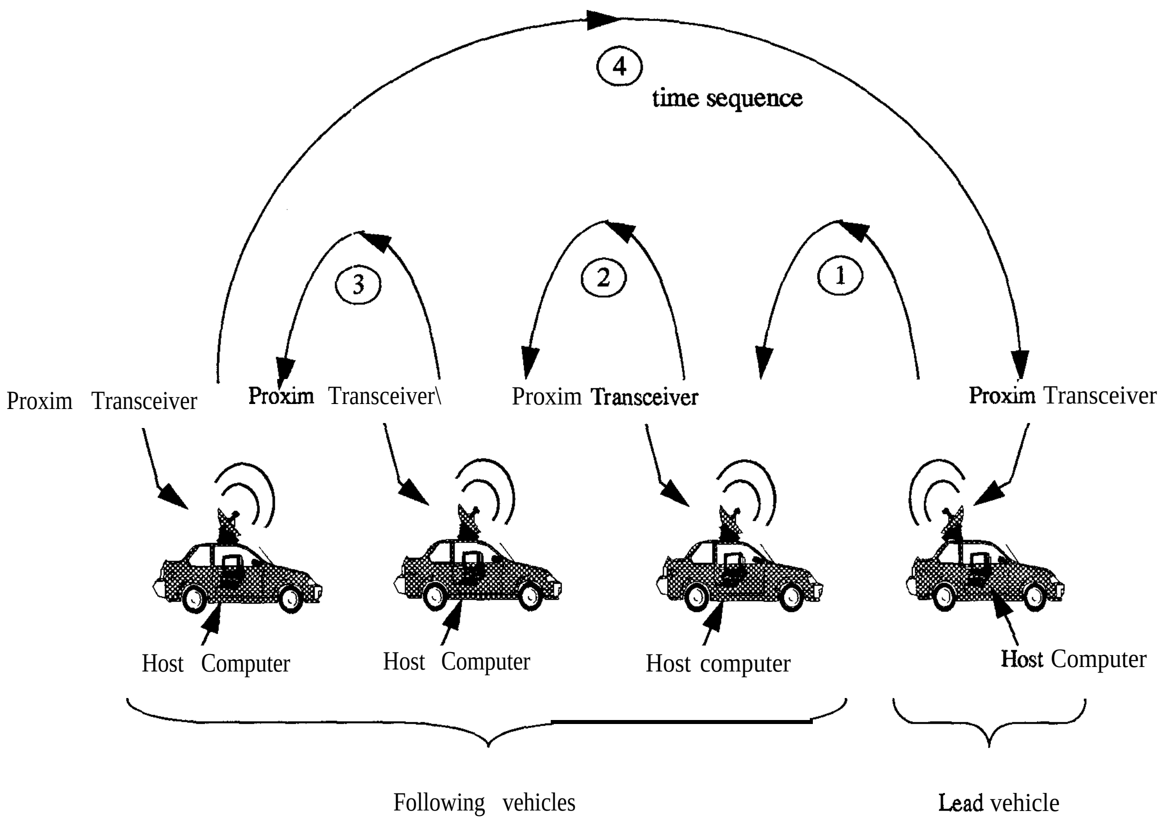


Figure 20. Multiple-Vehicles Platoon Communication Setup

In spite of these results, however, the current implementation of the communication link has proven to be capable of supporting a four-vehicle communication platoon. We have conducted numerous four vehicle platoon experiments, and in each of these experiments, the link is able to satisfy the data transport capabilities required by the controller.

6. Limitations of the Communication Link

6.1. Hardware Limitations

The wmmunication link, as implemented in this experiment, has several problems. The most crucial limitation that exists in the communication link lies in the transceiver hardware. It was observed that it can only operate in half-duplex communication mode. The half-duplex operations of the transceiver force us to take into consideration the transmission switching delay of the transceiver, which turns out to be the slowest of all of the communication equipment (on the order of milliseconds). Therefore, a significant amount of time must be allocated to allow the transceiver to switch between the two operating modes in order to avoid corrupting any data that is presently being transmitted or received. This is a major concern in this real-time platoon experiments. A full-duplex transceiver is much better suited for real-time sensitive tasks like this.

With the hardware used in this experiment, it is predicted that the link can only supports four vehicles in each platoon. In addition, more agile transceivers are necessary for the communication link to support more than four vehicles. Furthermore, the transceiver must be able to provide satisfactory performance when used under real-world environments. The Proxim transceiver, in its current state, is lacking in both of these characteristics. It fails to provide the necessary mechanism for reliable transceiver control. Furthermore, its performance appears to have degraded when it is used in a mobile environment. The transceiver that is used for the next phase of this experiment should prove to be satisfactory in both of these operational characteristics.

Lastly, the current setup suffers from lack of wmmunication between the two pieces of communication control hardware: the host computer and the ATcomm interface board. It is almost impossible to provide a complete control to the host computer without seriously burdening the software running on the ATcomm interface. Although the ATcomm interface board enables independent data processing from the host computer, it is believed that the host computer able to handle the additional ATcomm's communication processing without major diiculties, and a simple 85C30 serial controller card without ATcomm's intelligence is better suited in this experiment.

6.2. Software Limitations

A serious limitation in the current communication software is in its error-processing capabilities at both the transmitting and the receiving vehicle. At the transmitting vehicle, the Proxim transceiver's half-duplex operations made transmitter error-checking practically impossible: it is too time-wnsuming (due to all the associated switching delays) for the communication link to provide feedback to the transmitter from the receiver. The transmitter is, therefore, left with almost no self error-checking capability. Currently, there is no error recovery performed when an error has occurred in the data transmissions.

Since the transmitter is unable to make any error-checking (except for the CRC generation), the burden of error-checking is placed on the receiving software. Currently, only a crude error-checking routine is implemented. More effective error-checking and correction algorithms should be implemented.

The current wmmunication software cannot accommodate communication hardware that has a significantly higher data transfer rate than the Proxim transceiver. The current data transfer rate, which is limited by the Proxim transceivers, is a respectable 121 Kbits/second. However, in order for the link to provide better services to the vehicles, faster means of data transfer must be used. The current ATcomm software possesses too much overhead, mostly due to the polling of the 85C30, to provide reliable handling of faster communication. To accommodate communication at higher rates, a more efficient device link manager must be written. More importantly, the new driver must abandon the 85C30 polling scheme and take the advantage of the high speed DMA transfer available. For now, with wmmunications performed with the Proxim transceivers, the polling technique is an adequate solution.

7. Future Improvements of Radio-Based Communication Link

As this report repeatedly stressed, the weakest component used in the wmmunication link is the Proxim transceiver. Its sluggish response to transmission mode changes limits its utilization in this project. Therefore, the performance of the communication link can be dramatically improved when the Proxim transceivers are either improved or replaced by a full-duplex capable wmmunication transceiver. This eliminates the requirement for the transceiver to be switched between different modes of operations. It is likely that the best improvement the current communication scheme can obtain is to dedicate an independent full-duplex wmmunication channel between every vehicles in the platoon experiments. Thus, a full-duplex, low overhead wmmunication link can be realized. For instance, a simultaneous full-duplex radio-based wmmunication link between two vehicles can be established by simply installing two Proxim transceivers, preset to different frequencies, into each of the vehicles. The two transceivers are configured in such a way that the function of one transceiver is always the opposite of the other. Therefore, since each vehicle on the link can receive and transmit informations at the same time, a full-duplex communication link has been set up.

This setup is obviously ludicrous, since the additional equipment (twice as many transceivers) and additional software complexity (managing two transceivers on each ATcomm interface) are not worth the small increase in the throughput in the communication link. At more than one thousand dollars per board, notcounting the supporting hardware, the Proxim transceiver is an expensive investment. Moreover, since only four independent channels are available for the Proxim transceivers, the a radio-based communication link can only support a maximum four parallel data channels. This limits the number of vehicles that can be supported by the communication link, Thus, a Proxim radio-based communication scheme is inadequate to handle the amount of traffic existing in larger platoons. A different communication transceiver must be used.

Less dramatic improvements can also be obtained for the current communication hardware setup. By opting for more complex link protocols, it is possible to increase the throughput of the communication system. The complexity of the protocol is limited by the processing power of the ATcomm interface board.

An example of a more complex protocol can be described as follows: it is recognized that the maximum capacity of the link can accommodate network traffic for platoons of several vehicles. When the link is used to support smaller platoons, the link can be underutilized with intervals where the link rests in idle. In order to maximize the utility of the link, this idling interval can be used for broadcasting information in addition to that used for the control process. For example, this interval can be used to receive transmission from other platoons and/or vehicles, requesting to merge with the receiving platoon. All vehicles are given the right-to-transmit token when the wmmunication loop has progressed into this idling state. In this ***contention interval, the*** vehicles compete with each other for the right to transmit. If a vehicle has successfully begin its transmission in this interval, the other vehicles restrain themselves from further transmission until the channel is released by the transmitting vehicle. When two or more vehicles attempt to transmit simultaneously, the transmissions are corrupted, and the vehicles must retransmit this corrupted information when the channel is cleared. Thus, this network combines ethernet-like CDMA structure together with token-bus architecture in order to maximize the utilization of the communication link.

For the static multi-vehicle platoon control experiments, where the size of the platoon remains constant, only the network management data is transmitted in the contention interval. For example, if an abort condition was developed after the vehicle passed its transmission token to another, the aborting vehicle can use this interval to transmit a request-to-end experiment alert. Thus, it can send a priority message before the token is returned to that vehicle. For large platoons, this can save a significant amount of time. For dynamic multivehicle platoon control experiments, like the platoon merge experiments, the contention interval can be used for inter-platoon communication. [13]

8. Acknowledgment

I would like to take this time to acknowledge those who have supported me throughout this project. In particular, I'd like to thank:

- Both Professor Varaiya and Professor Walrand for their guidance, without which this project could never have been realized;
- Professor K. S. Chang, who consistently showered me with good advice and ideas;
- Isaac Porche, who refined my crude communication system into an actual working system;
- Fari Assad and Ahmad Bahai, who morally supported me throughout;
- Bret Foreman, who provided the timely advice enabling the completion of this project, and
- The PATH group, who intellectually and financially supported this project.

References

1. Drew, **D.R.,** ***Traffic Flow Theory and Control,*** McGraw-Hill Inc., 1968, pp. 298-310.

2. Drew, D.R., "A Study of Freeway Traffic Congestion," doctoral dissertation, Texas A&M University, College Station, 1966.

3. Chang, K.S., Li, W., Shaikhbahai, A., Assaderaghi, F., and Varaiya, P., "A Preliminary Implementation for Vehicle Platoon Control System," ***1991 American Control Conference,*** Vol. 3, pp 3078-3082, 1991.

4. Hedrick, J.K., McMahon, D., Narendran, V., and Swaroop, D., "Longitudinal Vehicle Controller Design for IVHS Systems," ***1991 American Control Conference,*** Vol. 3, pp 3107-3112, 1991.

5. ***PROXIM RXA Series Reference Manual, Proxim Inc.,*** 1990.

6. ***PROXIM RXA Series Radio Test and Evaluation Manual and Engineering Test Fixture (ET) Reference Man-***ual, Proxim Inc., 1990.

7. Lee, E.A., Messerschmitt, D.G., ***Digital Communication,*** Kluwer Academic Publishers, 1988, pp. 445-450.

8. ***ATcomm2 Intelligent Serial Communications Controller User Reference Manual,*** Metacomp, Inc., 1989.

9. ***Z85C30 Serial Communications Controller Technical Manual,*** Advanced Micro Devices, Inc., 1988.

10. Tanenbaum. ***Andrew S., Computer Networks,*** Prentice-Hall International, Inc., 1989.

11. ***Writing ROMable Code in Microsoft C,*** Systems and Software, Inc., 1990.

12. ***VRTX32 C Versatile Real-Time Executive User's Guide,*** Ready Systems, 1988.

13. Porche, Isaac, "Communication Protocols to Implement Coordinated Maneuvers of Automatically Controlled Vehicles," ***Master Report,*** Department of Electrical Engineering and Computer Science, University of California at Berkeley, 1992.

14. Shladover, S.E., "Longitudinal Control of Automated Guideway Transit Vehicles Within Platoons," ***ASME Journal of Dynamic Systems, Measurement and Control,*** Vol. 100, pp. 302-310, December 1978.

15. Shladover, S.E.. Desoer, C., Hedrick, J.K., McMahon, D., "Automated Vehicle Control Developments in the PATH program," ***IEEE Transactions on Vehicular Technology,*** Vol. 40, Feburary 1991.

16. Streisand, S.S., and Walrand, J., "Survey of Communications Needs and Possible Solutions in PATH," Department of Electrical Engineering and Computer Sciences, UC Berkeley, 1992.

17. Porche. I., Chang, KS., Li, W., and Varaiya, P., "Real-Time Task Manager for Communications and Control in Multicar Platoons," ***Intelligent Vehicles '92 Symposium,*** pp. 409-414, June 1992.

18. Porche, I., Chang, K.S., Li, W., and Varaiya, P., "Implementation of a Real-Time Communication Strategy for Automatically Controlled Vehicles," being submitted for ***IEEE Transactions on Industrial Electronics Technology.***