

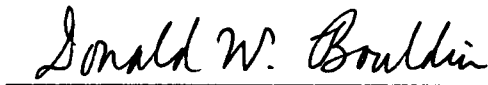
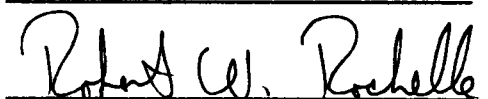
T124
24

To the Graduate Council:

I am submitting herewith a thesis written by Gary E. Underwood entitled "A Control Strategy for Communication on an Industrial Computer Network." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Electrical Engineering.


Robert E. Bodenhimer, Major Professor

We have read this thesis
and recommend its acceptance:

Accepted for the Council:


Vice Provost
and Dean of the Graduate School

STATEMENT OF PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a Master's degree at The University of Tennessee, Knoxville, I agree that the Library shall make it available to borrowers under rules of the Library. Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgment of the source is made.

Permission for extensive quotation from or reproduction of this thesis may be granted by my major professor, or in his absence, by the Head of Interlibrary Services when, in the opinion of either, the proposed use of the material is for scholarly purposes. Any copying or use of the material in this thesis for financial gain shall not be allowed without my written permission.

Signature Mary E. Underwood

Date August 1986

A CONTROL STRATEGY FOR COMMUNICATION
ON AN INDUSTRIAL COMPUTER NETWORK

A Thesis
Presented for the
Master of Science
Degree
The University of Tennessee, Knoxville

Gary E. Underwood
August 1986

Dedicated to my father
Clarence Edward Underwood
1923-1986

His inspiration, courage, and example
will always encourage me
to achieve.

ABSTRACT

The use of computers in the industrial world for data acquisition and control of the production process is rapidly increasing. These industrial automation systems are usually many distributed computers, each controlling a separate process. Using a communication network, a host computer transmits commands and collects data from each distributed controller. The organization of data flow on this network and the varying rate, amount, and type of data from each distributed controller require an efficient and intelligent communication strategy. Also, effective recovery from error conditions and the need to change the data requirements of a particular controller pose problems for the host communications control function. A communication control package to meet these requirements was developed during the Winter and Spring of 1986.

The host communications task and distributed computer system are applied to the automation of a tank farm storage area at a petroleum refinery. This system includes a gasoline blender controller, various process function controllers, and numerous tank gauging transmitters. The host communication control task gathers data continuously from these devices as well as transmitting commands and requesting data on operator command. Automatic error logging and recovery from network failures are designed into the system. Configuration of the data structures for each controller and tuning of the data flow on the network allow changing the host computer system to fit the expansion needs in the refinery. Test data is included to demonstrate the communication response of a typically loaded network.

TABLE OF CONTENTS

CHAPTER	PAGE
INTRODUCTION	1
I. SYSTEM CONFIGURATION	4
Host Computer Description	4
Distributed Controller Description	9
Petroleum Tank Farm Environment	14
Host Computer Software Description	29
II. NETWORK DESCRIPTION	40
Open Systems Interconnection Model	40
Protocol Description	45
Communication Primitive Description	52
III. HOST COMMUNICATIONS TASK FUNCTIONALITY	61
Task Interface Definition	61
Real Time Data Polling	79
Command Queue Message Handler	86
Service Queue Request Handler	87
Communication Flow Control Algorithm	91
Error Handling	101
IV. HOST COMMUNICATIONS TASK DESIGN	107
Executive Module	107
Initialization Module	111
Real Time Data Scanning Handler	120
Command Queue Request Handler	144
Service Queue Request Handler	146

CHAPTER	PAGE
V. TEST RESULTS	151
Real Time Data Scanning Performance	151
Command Queue Message Response	159
Service Queue Performance	160
Error Recovery Response	161
VI. CONCLUSIONS	164
BIBLIOGRAPHY	166
VITA	168

LIST OF FIGURES

FIGURE	PAGE
1. Host computer system configuration	5
2. Host computer chassis internal configuration	8
3. Network Communication board configuration	10
4. Tank configuration screen format	15
5. Single tank data screen format	17
6. Tank work area screen format	20
7. Discrete configuration screen format	21
8. Process graphic screen example	23
9. Analog configuration screen format	25
10. Single loop tuning screen format	27
11. Gasoline blender screen format	28
12. Software task hierarchy	33
13. Alarm list screen format	36
14. Tank history screen format	39
15. OSI reference model layers	41
16. HDLC network topology	44
17. HDLC frame structure	46
18. HDLC frame control field arrangement	47
19. Sample HDLC frame sequences	49
20. Primitive field structure	52
21. Network transaction example	59
22. Communications Control task interfaces	62
23. Dual-port memory data structure configuration	64
24. Request block format	66

FIGURE	PAGE
25. Transmit buffer format	67
26. Receive buffer format	68
27. Real time data area organization	71
28. PLC configuration screen format	80
29. Runtime polling scan sequence	82
30. Device mode state diagram	84
31. Communication flow control algorithm example	96
32. Executive module hierarchy	108
33. Executive program flow diagram	110
34. Initialization module hierarchy	112
35. Initialization module flow diagram	114
36. Real Time Data Initialization module flow diagram	116
37. Network Initialization module flow diagram	118
38. Scanning Initialization module flow diagram	121
39. Real Time Data Scanning module hierarchy	123
40. Real Time Data Scanning module flow diagram	125
41. Initialization Transmission module flow diagram	128
42. Initialization Reception module flow diagram	129
43. Scanning Transmission and Reception module flow diagrams	132
44. Self Test Transmission and Reception module flow diagrams	134
45. Error Recovery Transmission and Reception module flow diagrams	136
46. Command Queue Transmission and Reception module flow diagrams	138
47. Device Statistics module flow diagram	140
48. Error Detection module flow diagram	142

FIGURE	PAGE
49. Command Queue Request Handler module flow diagram	145
50. Service Queue Request Handler module flow diagram	147
51. Scanning performance test environment	154
52. System loading effects on scan rate performance	156
53. Scan tuning priority test results	158

INTRODUCTION

Distributed computer systems are often used for monitoring and control of industrial processes. Each distributed computer allows local control of an isolated process. These individual process controllers are then connected together to form the overall automation and data acquisition system for the industrial site. The connection between the controllers is implemented with a communication network which usually concentrates the automation data in a host computer system.

This distributed controller and host computer arrangement is used in the automation of the petroleum refinery tank farm. The wide geographic arrangement of the monitored and controlled processes makes practical the use of distributed controllers. Data is gathered and commands are sent from the host computer across an industrial local area network (LAN) to the individual controllers located at each process. In a refinery tank farm, the distributed devices can be divided into two categories. The first type of device is a programmable logic controller (PLC) which is used for process control applications. These applications include operation of a gasoline blender, control of transfer pumps and valves, and monitoring of custody transfer flow. The second type of device is the intelligent tank transmitter (ITT) which is used to monitor level, temperature, and status of the petroleum product tanks.

The host computer serves as a central point for the operators to view the processes and send commands for action. Data is gathered continuously from the remote devices and control commands are sent to these devices on operator demand. The control of the host communication

function is complicated by the varying rate, type, and amount of data which must be gathered and processed from each controller on the network. While gathering process data, the communication control function must be able to handle unique communication packets on operator demand. These communication messages are used to initiate actions by the controllers. Error conditions must be processed and recovery attempts made to insure data integrity on the network. Also, the flow of information on the network must be controlled to prevent congestion and provide an adequate data flow rate from each device.

A communication control package for the host computer was developed to address these problems. This communications package divides the communication activity into three areas. The first activity is the continuous acquisition of real time data (RTD) from each PLC and ITT on the network. Transmission, reception, and processing of command queue messages are the second area of activity. The command queue messages are generated by other functions in the host computer on operator demand. Typical command queue messages would turn on a pump, open a valve, or start a gasoline blend. Handling service queue requests is the final area of activity. The service queue provides overall network functions such as taking a particular controller offline, initiating the self diagnostics sequence on all network devices, or resetting the communication statistics.

The communication control task provides functions which are common to all three areas of activity. These functions include error logging, error recovery attempts, communication statistics gathering, and activity prioritization. The error logging and recovery activity

involves posting of system alarm messages to inform the operator of communication problems, identifying appropriate recovery action, and attempting periodic recovery to restore proper communication.

Statistical information is gathered to provide the system engineer with an idea of the performance of the communication link for each device on the network. The engineer can then adjust the activity prioritization parameters to increase or decrease the amount of communication resource devoted to each device.

Performance data and test results are compiled to demonstrate the communication response on a typically loaded network. The performance data includes the rate of RTD gathering using various priority levels for the devices. Various error conditions are simulated on the network. The system response to these errors is examined to study the error logging and error recovery performance.

CHAPTER I

SYSTEM CONFIGURATION

Host Computer Description

The host computer serves several purposes in a distributed automation system. The primary purpose is to concentrate the data from all the distributed controllers for easy viewing by the operators. Concentration of the data involves retrieving it from the remote controllers, organizing the data according to some scheme, and storing the data for immediate use by the operator. The host computer also serves as a central location for posting alarm conditions and maintaining a running history of alarms and system events. By acting as the master node for the communication network, the host computer connects all the controllers together to form a complete system. The master on the network initiates all requests for data from the controllers and receives all the responses.

The configuration of the automation system includes the host computer as well as support equipment. Figure 1 shows the equipment configuration for the tank farm automation (TFA) system. The TFA system has five operator stations for viewing the data and issuing commands. Each station consists of a 19 inch color display screen and a keyboard. The color terminals can display both alphanumeric characters and graphical drawings. Each keyboard is equipped with labeled function keys which simplify command entries to change the screen being viewed. The operator stations can be located remotely by using modems to communicate across phone lines to the host computer.

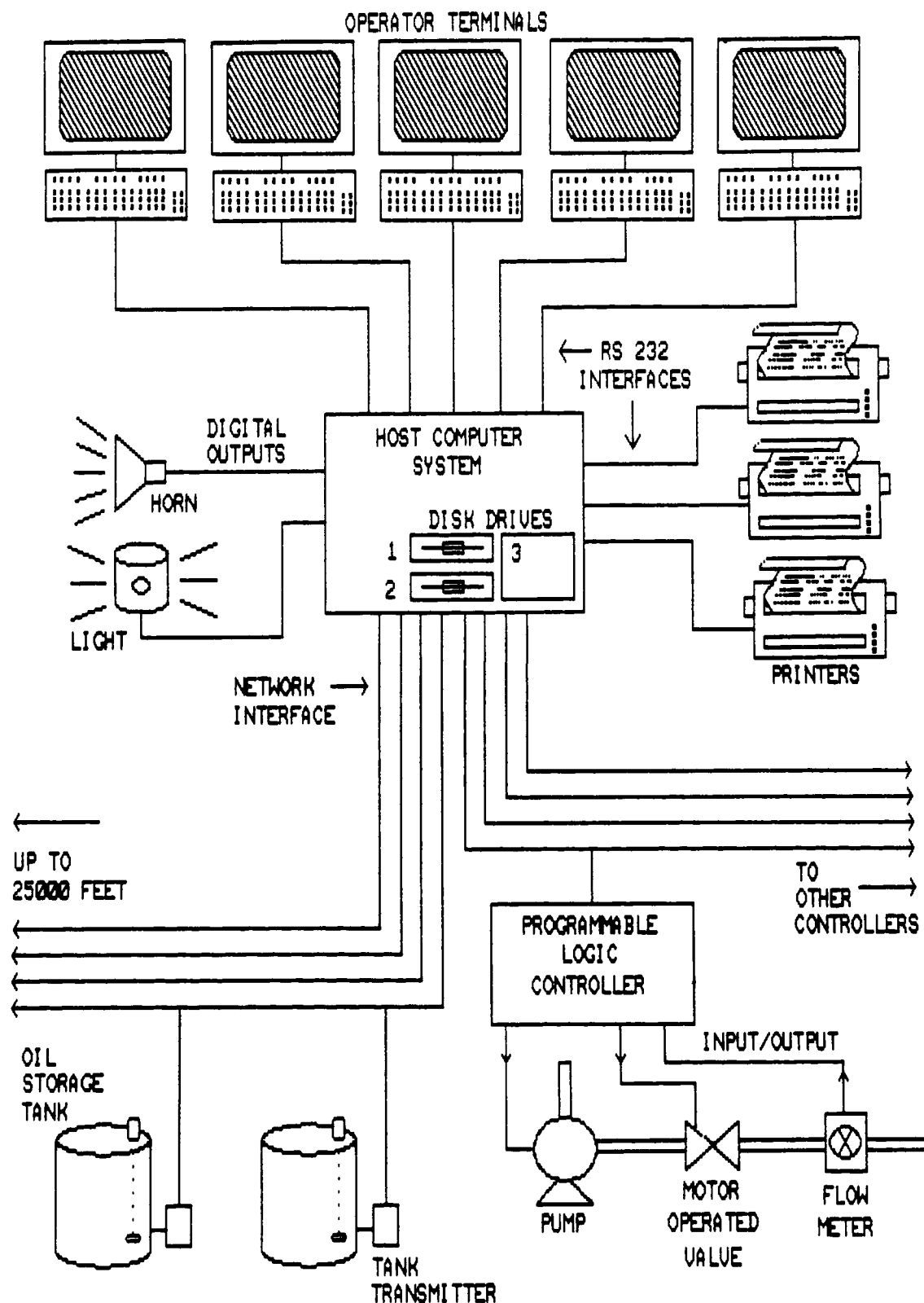


Figure 1. Host computer system configuration.

The TFA system can have up to three printers. These printers are used to provide hard copy output of the displayed data on the operator's screen. When the print key is depressed on the station keyboard, the operator is prompted for the destination printer number. The printer will then output a paper copy of the screen the operator is viewing. Certain reports can also be scheduled to print at any desired start time in the next 24 hours and at any repetition rate greater than once every 15 minutes.

The host computer is equipped with magnetic disk drives for mass storage of data and programs. Mass storage consists of two 5-1/4 inch floppy drives and a 5-1/4 inch winchester drive. The floppy drives can each store 640 kilobytes (Kbytes) of data and the winchester drive can store 20 megabytes of data.

Distributed controllers are connected to the host through eight communication channels. Each of these channels can support up to 254 separate devices. These devices can be located up to 25,000 feet from the host chassis using a direct wired connection. This distance can be increased drastically by using modems to connect to various transmission media such as phone lines, terrestrial microwave links, and satellite uplinks. Distances of over 1000 miles have been traversed with a single communication channel.

The host computer chassis features several digital outputs used to drive remote alarm signaling devices. These digital outputs are interfaced through a relay to drive alarm warning lights and sirens to alert field personnel to a potentially hazardous condition.

The host computer chassis actually contains four processors as

well as several support boards. Figure 2 shows the internal configuration of the host computer chassis. These boards are connected together on a bus using the Multibus standard established by Intel. The system memory can also be accessed across a secondary bus connection using the iLBX bus specification. This allows fast access to the system random access memory (RAM) without wait states for bus arbitration.

The main central processing unit (CPU) board utilizes a Motorola MC68010 microprocessor. This processor chip executes the main program in the TFA system and controls the actions of the other computers in the chassis. The CPU can address up to eight megabytes of system memory. This memory is made up of dynamic, error correcting RAM and battery backed-up static RAM. The battery backed-up RAM is used to hold critical system data in the event of a power failure.

The three disk drives are controlled by another computer board in the chassis. A local processor is used on the intelligent disk controller board for accessing data on the disk drives. The most recently used sectors of the disks are kept in cache memory by this board to speed up access to frequently used data. The formatting, reading, and writing of data to the disk drives is controlled by the local processor.

Another processor is used on the asynchronous communication (asynch comm) board. This board communicates with all the operator terminals and printers connected to the system. The board consists of a local processor which exchanges terminal and printer data with the central CPU through 64 Kbytes of dual ported RAM on Multibus. The dual ported RAM allows the CPU to insert a buffer of data to be sent to a

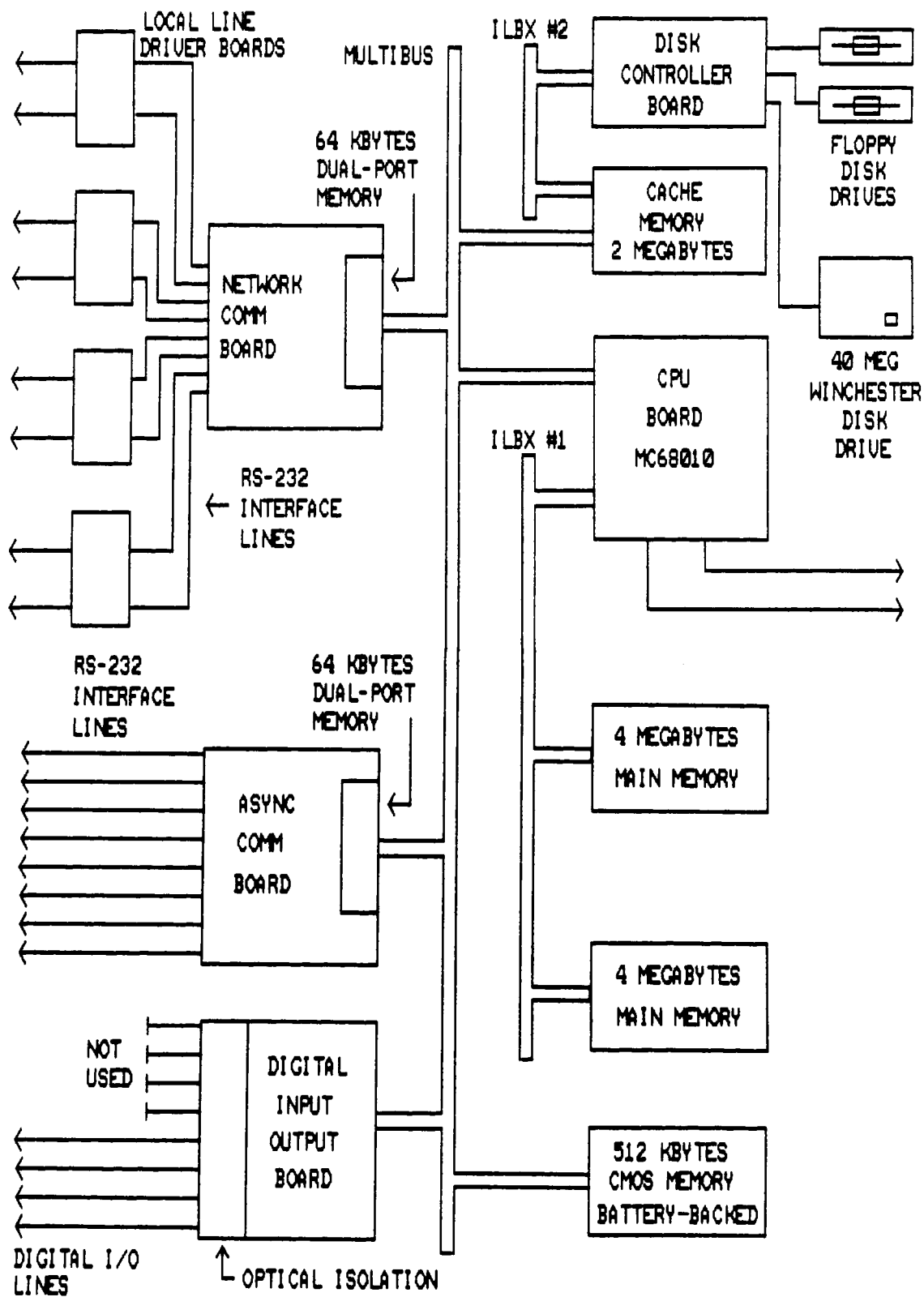


Figure 2. Host computer chassis internal configuration.

terminal for display into the Multibus side of the RAM. The local processor then retrieves the data from the other side of the RAM and sends it out the chosen channel. Up to eight communication channels are supported by the asynch comm board using the RS232 interface standard.

The digital outputs described previously are driven by a parallel input/output (I/O) board. This board will support up to 64 configurable input or output channels. Only four of the I/O channels are presently being used to drive the digital alarm outputs.

The fourth processor in the host computer is on the network communications board. This board is actually the same hardware as the asynch comm board with a different firmware program installed. Figure 3 shows the internal arrangement of this board. In this configuration, the CPU drives the board by inserting data and setting control flags in the dual-port RAM. This board utilizes an Intel 80186 microprocessor and contains 512 kbytes of local RAM. The eight communication channels are driven using four Zilog Z8530 universal synchronous/asynchronous receiver transmitters (USARTs). The Z8530 USART relieves the 80186 local processor from unnecessary computational burden by implementing a large portion of the communication protocol.

Distributed Controller Description

The distributed controllers are small computers which have been designed for the rugged field environment of industrial automation. These devices are used to automate small batch processes or collections of sensors and actuators in various locations around the industrial

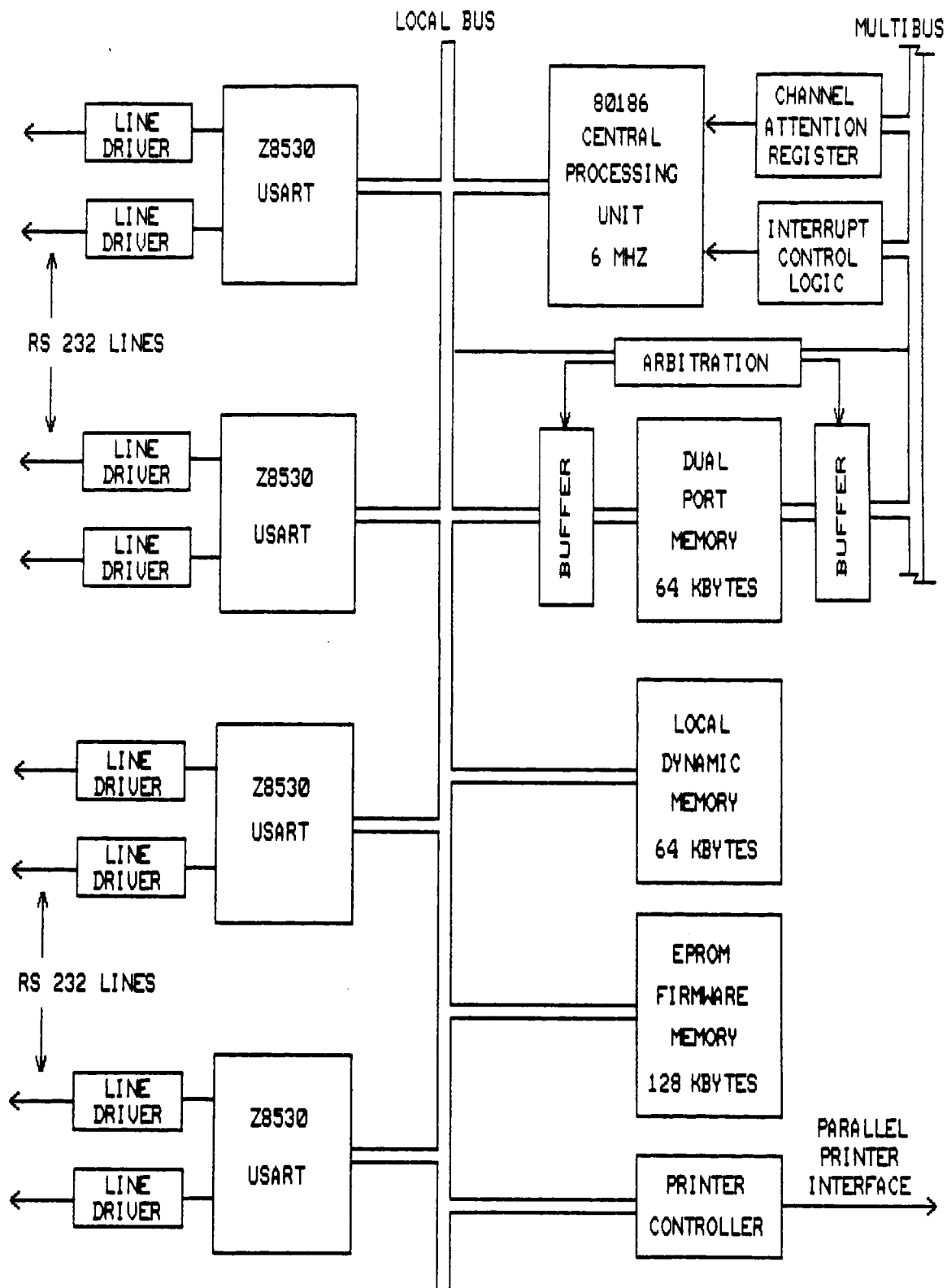


Figure 3. Network communication board configuration.

site. The distributed devices used in petroleum tank farm automation can be divided into two categories: (1) programmable logic controllers and (2) intelligent tank transmitters.

The PLC is a field programmable computer used to control discrete and analog variables associated with an industrial process. Each PLC is programmed in a symbolic language called relay ladder logic (RLL). The control program is constructed by drawing a schematic of various functional blocks on a programming computer screen and loading the numerical representation of this drawing into the controller. These functional elements consist of simulated relay contacts, relay coils, timers, drum step counters, and special function blocks. The special function blocks allow the programmer to read analog data, perform comparisons of analog values, and create mathematical equations for controlling analog outputs. These functional blocks can also read from and write to actual input and output points connected to the PLC.

Local input and output modules are used to connect the PLC to the industrial equipment. These modules are generic blocks which can be configured into a variety of arrangements. Typical module contain eight discrete input points, four 120 volt relay outputs, four 4 to 20 milliamp analog inputs, or 0 to 5 volt analog outputs. The input points are scanned by the RLL program, calculations are made, and the output points are driven to the appropriate value.

Certain models of the PLC also contain built-in, proportional-integral-derivative (PID) loops. Each loop compares the measured, analog process variable with the loop setpoint. The resultant error is entered into the PID algorithm along with the gain, rate, and reset

values to produce an analog output. Analog outputs are then used to control the process.

Network interconnection for the PLC is implemented with a network interface module (NIM). The NIM operates as a slave node on the network by transmitting only when the host addresses it by station number. By performing all the network interface activities, the NIM retrieves the requested data, formats the data for transmission, and implements the protocol necessary for transmission to the host. The NIM will also receive data to install into PLC memory, deformat the data, and gain access to the PLC memory to write the data. Through the NIM, the host computer can read and modify any data point within the PLC. This includes internal discrete points, analog values, loop gains, and RLL program memory. It is this capability which provides the host computer with the means to totally access and control the process connected to each distributed PLC.

Data associated with moving and storing petroleum products in a tank is acquired by the intelligent tank transmitter. The ITT is a small, dedicated computer which is housed in an explosion proof housing. By mounting to the side of a tank, the ITT connects directly to the level gauging device, temperature probes, and float switches mounted in the tank. The ITT is not a programmable device since it is dedicated to perform only tank data acquisition. However, a variety of parameters are configurable using a hand-held terminal. The hand-held terminal connects to the ITT with a fiber optic cable for explosion prevention in this hazardous atmosphere.

A variety of level gauging methods and hardware can be interfaced

to the ITT. Level is usually measured directly with a float suspended on a wire. Gauges are also available which calculate level from the hydrostatic pressure of the liquid. Another type of gauge measures the liquid surface position using radar mounted on the top of the tank. The ITT also measures the liquid temperature using resistive thermal devices located at various points on the tank. Usually, an emergency float switch is mounted to the top of the tank as a redundant check to prevent an overflow in the event of gauge failure. The position of this switch is monitored by the ITT and reported to the host. Constant measurements of the tank parameters are maintained in a table accessible by the host computer across the network. In addition to the tank data, the ITT periodically performs self diagnostics and reports the data in the table for inspection by the host computer.

The ITT implements a subset of the communications functionality available on a PLC. On the network, the ITT reacts like a PLC as it transmits the requested data table to the host. Only two write commands from the host are supported by the ITT. The first command initiates the self diagnostic sequence on the ITT. A second command is available on some models of the ITT to command the gauge to raise, lock, and lower the position of the level gauging float. This is done to check for hysteresis and stuck float conditions. The operator will also raise the float to the top of the tank during mixing operations to prevent damage to the gauge.

Petroleum Tank Farm Environment

Automation of a petroleum tank farm primarily involves three activities: (1) tank data acquisition, (2) pump and valve control, and (3) process control. The host computer receives data from the controllers running these activities, organizes and stores the data according to the process, and presents the data to the operator on the operator screen or on a printed report. Each activity has specific operator screens designed to present the application data in an organized manner. Through these screens, the operator can command the controllers at the process to perform specific actions as desired.

One of the primary activities in the petroleum tank farm is tank data acquisition. This involves retrieval of the level, temperature, and status information from the ITT. After this data is gathered, the host computer calculates volume, flow rate, and other parameters based on these measurements and configured data. This data is then compared with configurable alarm limits and the appropriate alarms are posted in the alarm list and event history. Several of the measured and calculated parameters are also logged into a file of historical data for trending purposes.

The first action necessary for tank data acquisition is to configure the tank on the system. Configuration parameters are entered on the tank configuration screen by the system engineer. Figure 4 shows the layout of a typical tank configuration screen. The engineer will establish a relationship between the eight character tank name and the actual ITT connected to the tank by entering the correct network cable number and device address. Next, the tank parameters are entered

NETWORK STATUS :
ALL LINKS OK

STRAPPING TABLE

NUMBER OF POINTS : 32 (MAX 50)
BREAK POINT LEVEL : 11.5 (FEET)

PT	FEET	GALLONS	PT	FEET	GALLONS
1	0.0	0000000	26	47.5	3163167
2	0.5	33296	27	50.0	3329650
3	1.0	66593	28	52.2	3496132
4	1.5	99889	29	52.75	3512780

5	2.0	133186	30
6	3.0	199779	31
7	4.0	266372	32
8	5.0	332965	33
9	6.0	399558	34
10	7.5	499447	35
11	10.0	665932	36
12	12.5	832412	37
13	15.0	998895	38
14	17.5	1165377	39
15	20.0	1331860	40
16	22.5	1498342	41
17	25.0	1664825	42
18	27.5	1831307	43
19	30.0	1997790	44
20	32.5	2164272	45
21	35.0	2330755	46
22	37.5	2497237	47
23	40.0	2663720	48
24	42.5	2830202	49
25	45.0	2996685	50

```
DEADBAND DATA
DLC DB (FEET)           : 0.175
MCT DB (FEET)           : 0.25
TNM LVL DB (FEET)       : 0.084
TNM TIME DB (MIN)       : 2.0
DIRECTION (FEET)        : 0.275
VOLUME DB (GAL)         : 1500
```

```
DEADBAND DATA
  DLC DB (FEET)           : 0.175
  MCT DB (FEET)           : 0.25
  TNM LVL DB (FEET)       : 0.084
  TNM TIME DB (MIN)       : 2.0
  DIRECTION (FEET)        : 0.275
  VOLUME DB (GAL)         : 1500
```

```
DEADBAND DATA
  DLC DB (FEET)           : 0.175
  MCT DB (FEET)           : 0.25
  TNM LVL DB (FEET)       : 0.084
  TNM TIME DB (MIN)       : 2.0
  DIRECTION (FEET)        : 0.275
  VOLUME DB (GAL)         : 1500
```

```
DEADBAND DATA
  DLC DB (FEET)      : 0.175
  MCT DB (FEET)      : 0.25
  TNM LVL DB (FEET)  : 0.084
  TNM TIME DB (MIN)  : 2.0
  DIRECTION (FEET)   : 0.275
  VOLUME DB (GAL)    : 1500
```

```
DEADBAND DATA
  DLC DB (FEET)           : 0.175
  MCT DB (FEET)           : 0.25
  TNM LVL DB (FEET)       : 0.084
  TNM TIME DB (MIN)       : 2.0
  DIRECTION (FEET)        : 0.275
  VOLUME DB (GAL)         : 1500
```

```
DEADBAND DATA
  DLC DB (FEET)           : 0.175
  MCT DB (FEET)           : 0.25
  TNM LVL DB (FEET)       : 0.084
  TNM TIME DB (MIN)       : 2.0
  DIRECTION (FEET)        : 0.275
  VOLUME DB (GAL)         : 1500
```

```
DEADBAND DATA
  DLC DB (FEET)      : 0.175
  MCT DB (FEET)      : 0.25
  TNM LVL DB (FEET)  : 0.084
  TNM TIME DB (MIN)  : 2.0
  DIRECTION (FEET)   : 0.275
  VOLUME DB (GAL)    : 1500
```

```
DEADBAND DATA
  DLC DB (FEET)      : 0.175
  MCT DB (FEET)      : 0.25
  TNM LVL DB (FEET)  : 0.084
  TNM TIME DB (MIN)  : 2.0
  DIRECTION (FEET)   : 0.275
  VOLUME DB (GAL)    : 1500
```

```
DEADBAND DATA
  DLC DB (FEET)      : 0.175
  MCT DB (FEET)      : 0.25
  TNM LVL DB (FEET)  : 0.084
  TNM TIME DB (MIN)  : 2.0
  DIRECTION (FEET)   : 0.275
  VOLUME DB (GAL)    : 1500
```

15

including dimensional data, alarm limits, and volumetric data. This data usually indicates maximum conditions for the tank and is password protected to prevent alteration. The 50 strapping points entered on the right side columns of the tank configuration screen are empirically determined by measuring the outside dimensions of the tank. From these points, the TFA system calculates a fifth order polynomial relating level and gross volume.

The single tank data screen shown in Figure 5 allows the operator to view the tank data, configure operator alarm stops, and control the ITT connected to the tank. After configuration, the first operator action is to enter the operator alarm stops and product density information. Next, the tank is put into an operational mode to begin data acquisition. The operational modes enable and disable certain directional alarms when the tank is filling, emptying, or closed. If the transmitter has failed, the tank is put in manual mode and the level and temperature are entered manually to obtain the volume calculations. Once the tank is in an operational mode, the items on the single tank screen are continuously updated.

Several volume calculations are performed on the tank as the level and temperature are retrieved from the ITT. The polynomial derived from the strapping data on the configuration screen is used to calculate the gross volume in the tank. This is compared with the maximum capacity of the tank to obtain the available room. The available inventory in the tank is the gross volume minus the water volume contained in the tank. Net volume represents the calculated volume in the tank corrected to 60 degrees Fahrenheit. The volume correction factor is calculated from the

DATE : 12/17
TIME : 19:30

S I N G L E T A N K
TAG ID TK-101 A

NETWORK STATUS :
ALL LINS OK

STATUS :>EMPTY

DATE : 12/14
TIME : 13:23

PRODUCT DATA

STATUS CODE LEGEND

O = OUT OF SERVICE F = FILL
M = MANUAL E = EMPTY
G = GPE R = RUN
T = TEST C = CLOSED

PRODUCT NUMBER : 7
PRODUCT NAME : NO4 BULK HTG
PRODUCT FORMULA : 2147-A1

DATE TIME

API AT 60 F : 47.9 11/16 15:49
BSW % : 7.39 12/02 8:47

LEVEL
FEET TEMP
DEG F

VOLUME DATA BBL

HIGH ALARM 54.5 125
HIGH STOP 50.75 90
CURRENT 27.46 75
LOW STOP 2.5 25
LOW ALARM 2.0 15

GROSS VOLUME : 1472974
NET VOLUME : 1471485
WORKING CAPACITY : 3258900
AVAILABLE ROOM : 1785926
VOLUME/FT : 53640

TANK CLOSED DATA

DATE TIME

FLOW RATE : 42017 BBL/HR

LEVEL (FEET) 50.72 12/15 13:28
GROSS VOLUME 3248724
NET VOLUME 3247787

TIME TO LEVEL STOP (HH:MM) : 31:52

WATER GAUGE LVL 1.789 10/23 17:51

COMMENTS : SHIPPING TO PIER 17 A

TIME : hh:mm hh:mm hh:mm hh:mm hh:mm hh:mm hh:mm hh:mm
LEVEL : hh:mm hh:mm hh:mm hh:mm hh:mm hh:mm hh:mm hh:mm
TEMP : hh:mm hh:mm hh:mm hh:mm hh:mm hh:mm hh:mm hh:mm
ALARM : hh:mm hh:mm hh:mm hh:mm hh:mm hh:mm hh:mm hh:mm

GRP#-ALARM GRP:

Figure 5. Single tank data screen format.

present temperature of the product and the temperature characteristics of the liquid such as specific gravity and thermal expansion. Net volume is also corrected for bottom sediment and water held in suspension in the product.

The volume data calculated for a tank is also used to approximate liquid flow rate and project a time-to-target for a operator entered target level. This data is used to monitor the progress of a tank filling or emptying operation remotely and to project when field personnel are needed to shut down a product movement operation.

Alarming on tank data consists of static and dynamic tests for alarm conditions. The static alarms are comparisons of level, temperature, and volume with configured limits. When the limit is violated, the alarm is posted on the alarm list, logged on the event history file, and the digital outputs are turned on to sound the warning siren. The operator acknowledges the alarm from the keyboard which will log another message in the event history and turn off the digital outputs. When a tank is in alarm, all the live and calculated data entries on the single tank screen will be shown in red to indicate a problem with the tank. The ITT also contains a status word which is used to post a variety of maintenance alarms. These include gauge failure, analog range failure, and other internal problems in the ITT.

The dynamic alarms are calculated on a specific time base and compared against the configured deadbands. These alarms are based on the rate of change of the variables. For example, the drastic level change alarm will be posted if the level changes more than the deadband amount in one minute. The counterpart of the drastic level change alarm

is the tank not moving alarm. If the tank is in a moving mode and the level does not change at least the deadband amount in one minute, the tank not moving alarm will be posted. The moving closed tank and leaking tank alarms are based on the movement of a closed tank with respect to the level and volume at the time of closing.

The operator also has the ability to view several tanks at once to monitor operations in various areas. As shown in Figure 6, a work area can be configured to hold 10 tanks on each screen with 10 pages available in each work area. The operator can configure up to 35 different work areas, assigning each a different name and description. If a tank goes into alarm while being viewed on the work area, the entry for that tank will flash red. After the operator acknowledges the alarm, the entry will be solid red until the condition clears.

Control and monitoring of the pumps and valves used for product movement operations is the second important automation task in the petroleum tank farm. The pumps and motor operated valves are connected locally to PLCs in the field. When the PLC receives a command from the host to turn on a pump or open a valve, it will drive the proper outputs to sequence the pump on or open the valve. The PLC also monitors the discharge pressure of the pump and the valve position and reports this data back to the host.

The first step in setting up the host to monitor a PLC in a pump and valve control application is to configure the data points. Figure 7 shows a typical discrete configuration page. The system engineer uses this screen to create an association between a tag identifier name and a particular discrete on a PLC. After entering the PLC number and

DATE : 12/17
TIME : 19:30

T A N K W O R K A R E A
WEST 1 WEST TANK FARM FUELS

NETWORK STATUS :
PAGE 03 ALL LINKS OK

TAG ID	LEVEL TEMP CURRENT	LEVEL LOW ALARM HIGH ALARM	AVA. ROOM AVA. VOL. BARRELS	NET VOLUME BARRELS	STOP LEVEL TIME TO LEVEL STOP	TANK STATUS PRODUCT NAME
>TK 217	54.397 97	2.500 58.500	2175 3175672	3158972	55.000 0:27	FILLING NO 4 AVGAS
>TK 423	37.245 72	2.700 58.500	32501 1875623	3218945	45.800 0.00	CLOSED WINTER UNLD
>TK 147	22.457 25	1.500 42.750	21354 1254952	3254841	14.000 4.25	EMPTY PENHEX
>TK 124	12.547 74	3.200 25.475	12548 752450	1254384	15.000 0.00	OUT OF SERVICE CRUDE ARAMCO
>TK 9841	54.397 45	5.500 58.500	12354 15654465	19457825	55.000 3.25	FILLING BUNKER OIL S14
>TK 47	45.797 86	0.500 58.500	13235 5455672	8155972	48.000 1:27	FILLING REFORMATE
>TK 858	74.397 128	2.300 88.500	5655 8575672	8458972	15.000 0.00	RUNNING PERIXYLENE
>TK 117	34.397 47	1.500 48.500	5475 6475672	7158972	45.000 2:27	FILLING ISOPROPANE
>TK 48	46.397 57	3.500 54.725	54541 6175672	7158972	34.000 0.00	CLOSEDG NO 5 HTG OIL

GRP#-ALARM GRP:

Figure 6. Tank work area screen format.

DATE : 12/17
TIME : 19:30

D I S C R E T E O U T P U T G R O U P
R F L O W R E F O R M E R P U M P P A D 2 A P A G E 0 1

NETWORK STATUS :
ALL LINKS OK

TAG ID	PLC NUM	DISC S/D	DISC NUM	PROCESS FUNCTION	ALARM MSG	STATUS MSG	COMMAND MSG
>MOV 147	1	S	1	PMP 12A DISCHARGE	26	129	14
MOV 149	1	S	2	PMP 12A INLET TK 47	26	129	14
MOV 150a	1	S	3	PMP 13 DISCHARGE	26	132	16
PMP 12A	1	D	4	REFORMER 17 PUMP	48	15	22
PMP 134	2	S	4	REFORMER 77 PUMP	48	15	22
PMP 145	2	S	17	REFORMER 19 PUMP	48	15	22
PMP 146	2	S	18	TK 147 OUTLET	48	15	22
PMP 147	2	S	19	TK 445 INLET	48	15	22
MOV 22	7	S	2	PMP 134 INLET	26	129	14
MOV 144	1	S	36	PMP 74B INLET SD 47	26	329	74
MOV 749	7	D	2	PMP 127 INLET TK 28	26	32	14
MOV 235	1	S	4	PMP 234 OUTLET	26	129	64
MOV 12C	3	S	22	PMP 12A INLET TK 47	26	129	14
MOV 544	12	S	2	PMP 16 INLET	26	23	14
MOV 55	12	S	2	PMP 322 INLET	26	45	22
MOV 24	12	S	2	PMP 348 INLET	26	123	14
MOV 78D	12	S	2	PMP 175 INLET	26	132	26
BV 1154	1	S	2	BLOCK VALVE TO PIER	0	129	22

GRP#-ALARM GRP:

Figure 7. Discrete configuration screen format.

the discrete number on that PLC, the engineer enters a description of the device, an alarm message number, status message number, and command message number. The alarm message number is used to identify a configurable message which is posted in the alarm description when a discrete alarm condition occurs. The status message provides a description of the state of the discrete such as open and closed for a valve. The command message number points to a configurable string which allows the operator to command the discrete using an english language instruction such as on, off, open, or close.

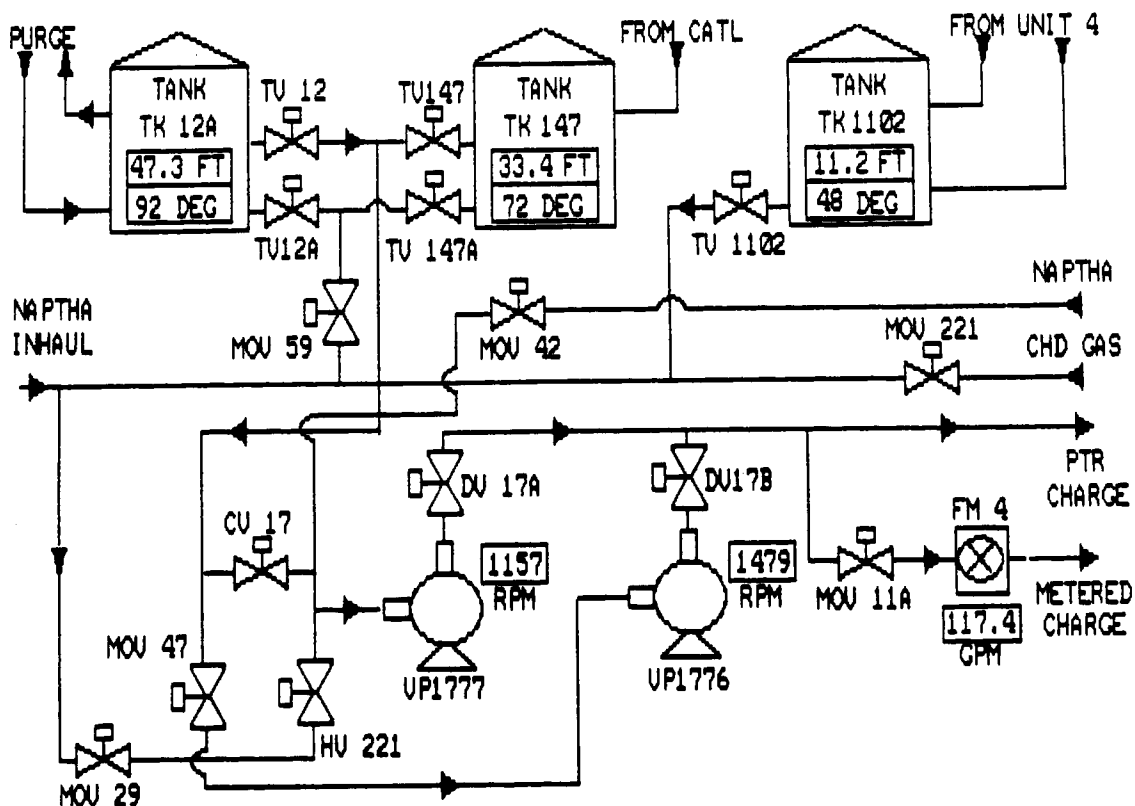
Each discrete output consists of a command bit location in the PLC and a status and alarm bit which is read into the host. When the operator sends a state change to the command bit, the PLC will drive the output to the desired state. When the device actually reaches the desired operating condition such as valve opened or pump on, the return status bit will change state to indicate the device state. If the device does not reach the specified state in an acceptable time or a failure occurs, the PLC will set the alarm bit and the host will post an alarm condition on that discrete. The operator commands the pump or valve by pressing the tag key on the keyboard, entering the device identifier name, and selecting the desired state.

The operator views the state of the pumps and valves on a process graphic screen. Figure 8 shows an example of a typical process graphic for a refinery tank farm. When a pump turns on or a valve opens, the graphical symbol will change color to indicate the state of the discrete. If a point goes into alarm, the symbol will flash red until acknowledged and then will turn solid red.

DATE : 12/17
TIME : 19:30

PROCESS GRAPHIC
PTR 1 PLATINUM REFORMER

NETWORK STATUS :
ALL LINKS OK



GRP#-ALARM GRP:

Figure 8. Process graphic screen example.

Process control is the third automation activity in the petroleum tank farm. This activity involves monitoring and supervisory control of continuous and batch processes. An example of a typical continuous process is a flow control loop used to maintain a specific flow rate through a pipeline, catalyzer, or product mixer. Operation of the gasoline blender is the primary batch control process in the tank farm. The blender consists of several flow control loops maintained at a precise ratio to the total output flow rate of the blender. Process control usually involves the control and monitoring of discrete points, analog values, and loops simultaneously.

Configuration of the analog values to be monitored is accomplished on the analog configuration screen shown in Figure 9. The engineer names the point to be configured and enters the proper PLC number and analog number. Next, a description is entered to identify the process being monitored. The scale factor, high range, and low range are used to scale the analog value from the PLC into engineering units on the host computer screen. Associated with each analog are four alarm limit values. The high-high and low-low alarm limits are configured by the engineer and should be considered emergency limits. The high and low alarm limits are configured by the operator and used as process limits during particular operations.

The loop configuration is similar to the analog point configuration. To configure a loop, the engineer enters the name, PLC number, loop number within the PLC, and description on the loop configuration screen. The process variable controlled by the loop is the analog of the same number on that PLC. Configuration of the

DATE : 12/17
TIME : 19:30

A N A L O G I N P U T G R O U P
CAT43 CATALYZER 43 FLOWS

PAGE 01 NETWORK STATUS :
ALL LINKS OK

TAG ID	PLC NUM	ANA NUM	UNIT	PROCESS FUNCTION	DEC SF	RANGES HIGH/LOW	ALARMS HHIGH/LLow	ALARMS HIGH/LOW
FMTR 120	4	1	GPM	STRM 12 FLOW	1 64	10000.0 0.0	9575.0 1.5	7500.0 10.0
FMTR 121	4	2	GPM	STRM 13 FLOW	1 64	10000.0 0.0	9575.0 1.5	7500.0 10.0
FMTR 121	4	3	GPM	STRM 14 FLOW	1 64	10000.0 0.0	9575.0 1.5	7500.0 10.0
FMTR 17	5	1	GPM	OUTLET FLOW TK 17	1 64	8000.0 0.0	7545.0 3.5	7200.0 10.0
FMTR 19	5	2	GPM	OUTLET FLOW TK 121	1 64	8000.0 0.0	7545.0 5.0	7200.0 15.0
FMTR 20	5	4	GPM	OUTLET FLOW TK 3	1 64	4500.0 0.0	4275.0 2.0	4000.0 10.0
TRBN 3	14	12	BPM	TURBINE MTR 3	2 32	2500.50 0.0	1575.30 10.25	1400.00 17.55
TRBN 2A	14	14	BPM	TURBINE MTR 2A	2 32	2500.50 0.0	1575.30 2.55	1400.00 10.25
CMPR 14	14	15	LPM	COMPRESSOR 14D	1 16	7500.0 0.0	6575.0 1.5	6500.0 10.0

GRP#-ALARM GRP:

Figure 9. Analog configuration screen format.

remaining parameters is performed on the single loop tuning screen. This allows tuning the loop for optimal performance. Figure 10 shows a typical single loop tuning screen. The engineer configures the gain, rate, reset, orange deviation, and yellow deviation on the screen. Gain, rate, and reset affect the rise time, overshoot, and stability of the loop during transient events and setpoint changes. The orange and yellow deviation values are alarm limits which compare the error between the setpoint and process variable. Changing the loop mode to manual allows the engineer to break the loop feedback path for testing. The automatic and cascade modes are used for normal operation. A trending plot on the right side of the screen is used to observe the transient response of the loop to step input changes in the setpoint.

The gasoline blender is a batch process controlled by several loops and discrete points. Blending gasoline involves combining several hydrocarbon products together in an exact ratio. This ratio is critical to produce the correct octane, vapor pressure, and additive level in the fuel. The operator configures a blend on the screen shown in Figure 11. First, the operator enters the blend demand flow rate and total blend volume. Next, the percentages of the individual components are entered. As these parameters are entered in the host, the values are transmitted across the network to the blender PLC. When the sum of the individual setpoints is 100%, the operator can command the PLC to begin the blend. The command is transmitted as a discrete output similar to a pump or valve command. The PLC will then sequence the blender pumps on and begin opening the valves to ramp up the flow rate. When the flow rate reaches the demand level configured, The blender PLC will maintain

DATE : 12/17
TIME : 19:30

S I N G L E L O O P T U N I N G
TAG ID PUMP 24C

NETWORK STATUS :
ALL LINKS OK

PROCESS FUNCTION : PUMP FLOW 24C BLNDER

PV : 524.05

1000.00

SP : >522.75

MODE : AUTO

GAIN : 13.254

RATE : 00.452

RESET : 3.2400

HH ALARM : 995.25

H ALARM : 990.00

L ALARM : 10.025

LL ALARM : 0.00

HH DEV. : 622.75

H DEV. : 572.75

L DEV. : 5482.75

LL DEV. : 5422.75

TREND FREQ. : 1 SEC

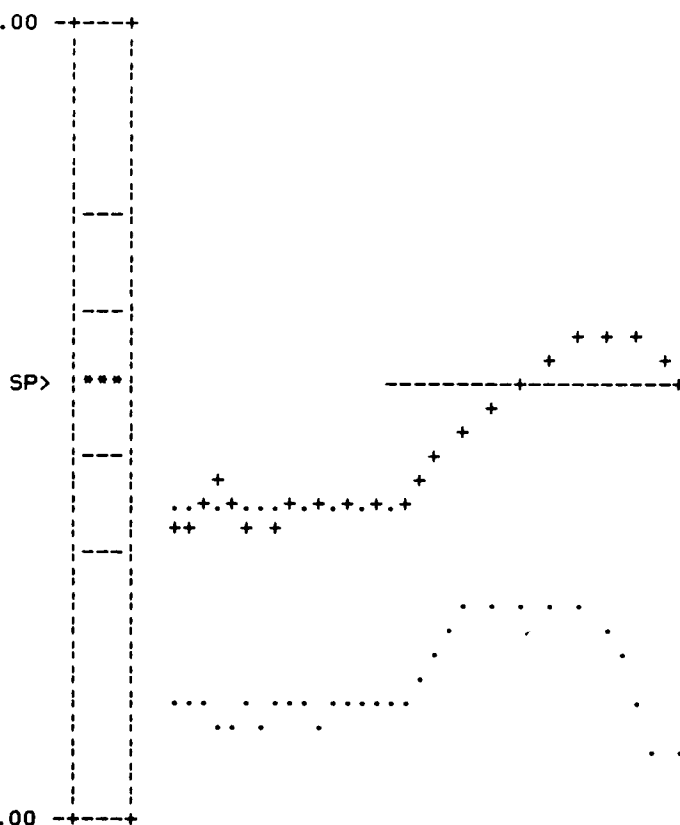
OUTPUT % : 47.38

+++++

0%

100%

0.00



GRP#-ALARM GRP:

Figure 10. Single loop tuning screen format.

DATE : 12/17
TIME : 19:30

MASTER BLEND

NETWORK STATUS :
ALL LINKS OK

LOOP TAG ID	SET PT. % TAG ID	SET PT. %	FLOW RATE	FLOW UNIT	VOLUME TOTAL	VOLUME ERROR	VOL. UNIT	LOOP IN PACING
TK 147	STPNT 1	14.75	127.4	GPM	12438.9	0.7	GAL	N
TK 148	STPNT 2	24.93	324.8	GPM	31452.0	14.3	GAL	N
TK 21	STPNT 3	7.50	22.7	GPM	2496.0	5.0	GAL	N
TK 2A	STPNT 4	8.29	25.4	GPM	2537.1	0.0	GAL	N
TK 453	STPNT 5	37.64	476.1	GPM	12438.9	146.8	GAL	N
TK 454	STPNT 6	0.11	7.2	GPM	32.7	0.2	GAL	N
TK 455	STPNT 7	4.00	14.9	GPM	1468.2	1.7	GAL	N
TK 147	STPNT 8	2.78	8.9	GPM	754.1	1.6	GAL	N
		0.00	0.0		0.0	0.0		N
TEL 12	ADDT 1	0.00	127.4	MLM	0.0	0.0	ML	N
TML 2	ADDT 2	1.25	127.4	MLM	5726.1	0.0	MLL	N
ADDT 1	ADDT 3	4.35	22.8	LPM	438.9	1.7	LTR	N
		0.00	0.0		0.0	0.0		N
		0.00	0.0		0.0	0.0		N
		0.00	0.0		0.0	0.0		N
		0.00	0.0		0.0	0.0		N

SET POINT % TOTAL = 100.00%

BLEND PROCESS

DEMAND TOTAL VOLUME : 652500.0 GAL DEMAND VOLUME TAG ID : DMND_VOL
ACTUAL TOTAL VOLUME : 63617.8 GAL

DEMAND RATE : 1050.0 GPM DEMAND RATE TAG ID : DMNDRATE
ACTUAL RATE : 1047.4 GPM

BLEND STATUS : ON BLEND COMMAND TAG : BLND_CMD
BLEND PACING STATUS : N RESET TOTALIZER TAG : BLND_RST

SET PT. 100% STATUS : Y

GRP#-ALARM GRP:

Figure 11. Gasoline blender screen format.

that rate until the actual blend volume nears the total blend volume requested. The PLC will then ramp down the flow rate to a dribble level until the blend is complete. At this point, the PLC will automatically close down all the valves and shut off the pumps.

Several alarm conditions are possible during the batch blend process. If the pumps do not sequence on properly when the blend is first started, the blend will automatically abort and alarms will be posted on the blend and the offending pump. During the blend, the status of the loop outputs is constantly monitored by the PLC. If a particular loop cannot supply the requested flow rate, that blend will go into pacing mode and an alarm will be posted in the host. During pacing, the flow rates of all the blend streams will be reduced to allow the paced loop to recover. If that loop does not recover in a specified amount of time, the blender will shut down and additional alarms will sound.

Host Computer Software Description

The software program which implements the TFA system functionality is organized as several different tasks running together in the host computer. Operation of these tasks is coordinated by a multi-tasking operating system (OS). When the host computer is powered on, the operating system is automatically loaded from the disk drive and put into operation. Thereafter, the OS organizes and controls the activities of the individual tasks running under it.

The OS controls the activity of the multiple tasks by imposing a priority structure. The priority of the individual tasks determines

which task is running at a given time. This priority structure is two dimensional. The first dimension is priority level. Each task is assigned to a particular priority level. A task on a given priority level will run only if the tasks on the higher priority levels are suspended waiting on a system resource. The second dimension of the priority structure is time slices within a priority level. If more than one task is on a given level and that level is active, the tasks are allowed to run according to a time slice scheme. Each task is assigned the number of time slices it is allowed to run. After the allotted time has expired, each of the other tasks are allowed to run in a round robin fashion.

The operating system also governs the shared resources within the computer. For example, if two different tasks desire to write to a file on the disk drive at the same time, the OS will suspend one of the tasks until the other has completed the disk transaction. When a higher priority task is suspended, the computing resources are devoted to the lower priority tasks. The OS also controls the interrupt service required by the other computer boards in the system. For instance, when a key is pressed on the keyboard, the terminal communications board will interrupt the system CPU with the request for service. The OS will retrieve the key value, place it in a buffer, and clear the interrupt. A task can suspend waiting on an interrupt to occur. In the above example, a task could suspend waiting on a key input. When the OS services this interrupt, it will enable the suspended task and allow it to run and perform the requested action indicated by the key.

Several system utilities are provided by the OS to aid inter-task

communication and synchronization. These utilities are the queue and semaphore package. The queuing package allows a task to construct a message or data table and enqueue it to another task. When the other task is running, it can dequeue the message and use the data. The queuing media is a first in, first out (FIFO) structure. Several tasks can enqueue to the same queue and the messages will stack up in FIFO order for the receiving task to dequeue. A receiving task also has the option of suspending operation while waiting on an enqueued message.

The semaphore package is used for task synchronization and concurrency locking of data structures. When a task performs a wait command on a semaphore, it will suspend operation until another task signals that semaphore. This will cause the tasks to operate when the signaling task desires service from the waiting task. A shared data structure can also be concurrency locked by performing a wait before accessing the area and issuing a signal when leaving the area. If a task already has the resource, the next task will suspend on the wait until the first task signals. The tasks will get the resource in the order in which they were suspended.

The database management system (DBMS), although not part of the operating system, is a global data management package used by every task in the TFA system. This package provides a method to organize data into records for easy modification and retrieval. The DBMS consists of an offline configurator which takes the data definition file and produces tables for use by the online DBMS utilities. The online utilities allow each task to find the key to a record, get data from the database, and modify data in the database.

The software tasks in the host computer are organized in a hierarchy from the highest priority tasks to the low priority background tasks. Figure 12 shows the structure of the software tasks in the TFA system. Each of the 10 different tasks (some tasks are repeated multiple times) runs as an individual program with connections to the other tasks in the system through queues, semaphores, global data structures, and the database.

The System Control task is the highest priority task in the system. System Control (SC) initiates and controls all the other tasks in the system. When the TFA software package is invoked by the operating system, System Control is the first task to run. After initializing global data structures, the next action by SC is to display the system configuration screens. The engineer configures the number of terminals, printers, and communication channels in service and selects the port connections. Using this information, SC will then initiate each task as required. If any task does not initialize properly, SC will abort the TFA software and run the self diagnostics package to ascertain the cause of the problem.

Once the system is running, System Control serves two purposes. The first activity is to keep track of the time based events in the system and either enqueue a message or signal a semaphore to inform a task to initiate the action. The second activity is to test the health of the system and take appropriate action. If a system resource such as a disk drive should fail to operate, SC will put the system into degraded mode. While in degraded mode, alarms are posted to tell the operator of the problem but the system is allowed to continue running

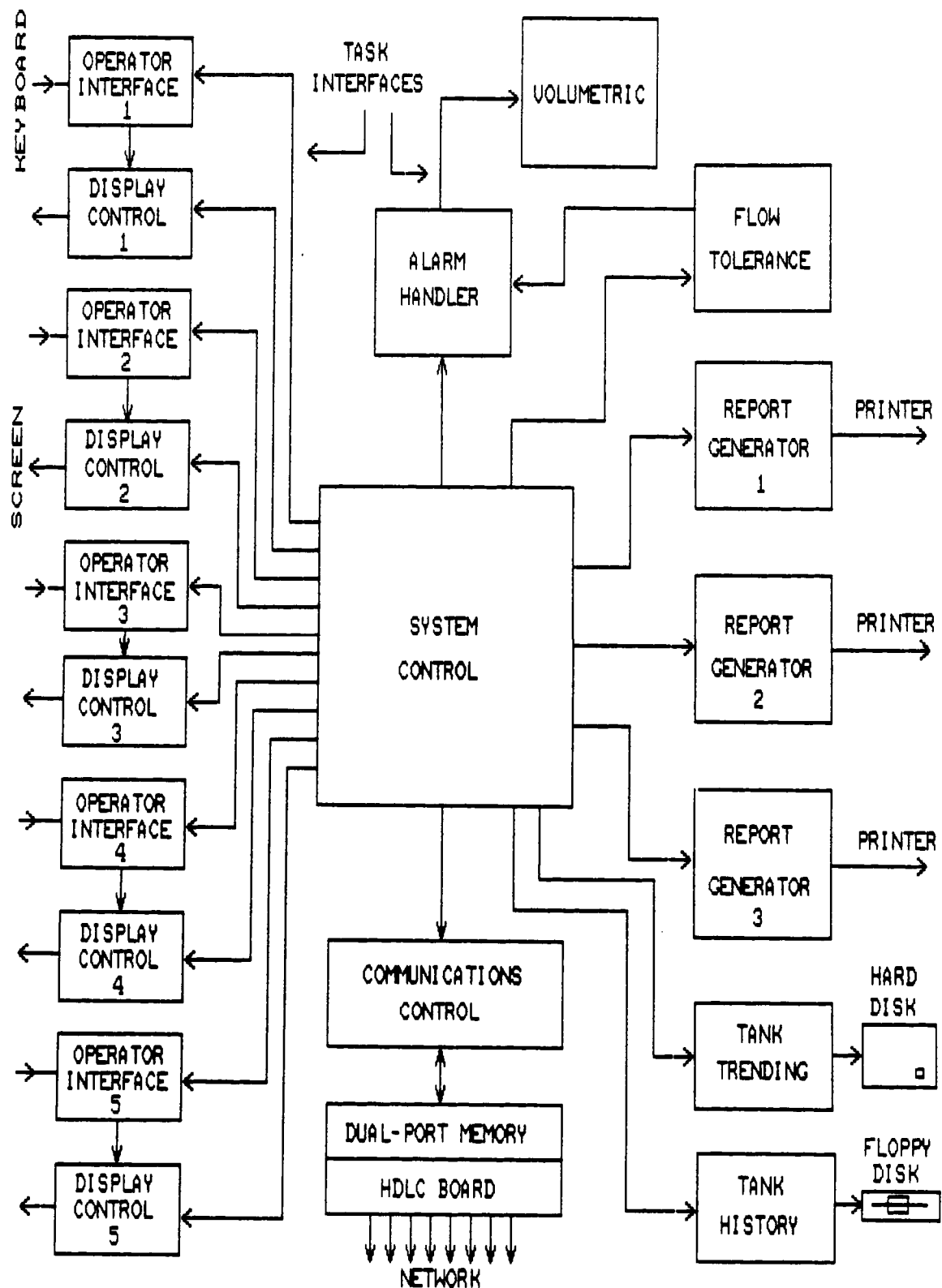


Figure 12. Software task hierarchy.

until it is convenient to turn the system off for repairs. If a task should quit running due to a system fault, SC will abort the TFA package and run the diagnostic software to ascertain the nature of the problem.

The Operator Interface (OI) task receives all the commands from the keyboard and initiates the requested action. An OI task is created by System Control for each terminal configured on the system. Normally, the Operator Interface task is suspended waiting on an interrupt from the terminal interface board when a key is pressed on the keyboard. After the command is received, OI will decide on the appropriate action and enact it. For example, if the key is a request to view the single tank screen, OI will prompt the operator for the tank number. After checking the validity of the number, OI will enqueue a message to Display Control to display the chosen screen. Another example is a request to turn on a discrete output associated with a pump. After the request is entered, OI will ask the operator for the tag identifier for the discrete to turn on. Next, OI will check the validity of the tag and locate the PLC and address of the chosen discrete. Finally, OI will enqueue a message to the communication task requesting the discrete be turned on. Similar to the above examples, the OI task can request changes to any commandable point in any PLC on the network. OI can also change the values of alarm limits and other process parameters in the database or in a PLC on the network.

The Display Control (DC) task produces all the screen images of the data in the system. Like OI, a DC task is created for each terminal on the host system. After initially displaying a screen requested by OI

through the queue, DC will periodically refresh the data on the screen as it changes. The DC task retrieves the data from the database, disk files, global data structures, and from the real time data area. OI enqueues the messages to DC to display a screen and to change page number on the multiple page screens.

The Report Generator task (RG) operates similar to the DC task. System Control creates a report generator task for each printer configured on the system. Normally, RG is suspended waiting on a queue message for a report request. When the Operator Interface task enqueues a request to print a particular screen, the RG task will dequeue the request, retrieve the required data, format the report, and output the data to the asynchronous communication board for transmission to the printer. While printing the report, OI can enqueue a message requesting that the report be canceled. System Control task also enqueues requests to RG for printing the scheduled reports.

The Alarm Handler (AH) task is responsible for tracking any alarm conditions in the host computer or from a PLC on the network. When an alarm condition is recognized, AH must post the alarm on the alarm list, create an entry for the event history file on disk, and mark the alarm condition in the database record for that tag identifier. The AH task also updates the alarm group summary area at the bottom of each screen. Alarms conditions are divided into functional groups. When an alarm occurs, the group area for that alarm will flash red at the bottom of each screen. When acknowledged, the group area will be highlighted in the colors assigned to each alarm group. Figure 13 shows a typical alarm list screen along with the alarm summary area at the

DATE : 12/17
TIME : 19:30

A L A R M L I S T

PAGE 01 NETWORK STATUS :
ALL LINKS OK

TAG ID	DATE MM/DD	TIME HH:MM	ALARM DESCRIPTION	CURRENT VALUE	UNIT	GRP NUM	ERROR CODE
MOV 47	12/17	19:27	OPEN TIMEOUT ERROR	CLOSED		7	0024
PMP 234A	12/17	18:55	DISCHARGE PRESSURE LOW	OFF		7	0012
TK 123	12/17	18:24	DRASTIC LEVEL CHANGE	45.78	FEET	2	0000
TK 22	12/17	18:21	ENRAF GAUGE FAILURE	0.00	FEET	4	07A0
FLOW17	12/17	18:00	HIGH FLOW RATE	1879.2	GPM	3	0000

GRP#-ALARM GRP:

4-MAINTANCE
7-DISCRETE

2 -TOLERANCE

3-ANALOG

Figure 13. Alarm list screen format.

bottom of the screen. Alarm conditions can occur two ways. The AH task can receive a message through a queue requesting that an alarm condition be posted or cleared. For example, if a PLC ceases to communicate on the network, the Communication Control task will send a message through the alarm queue to produce an alarm on the device. The AH task also looks for alarm conditions in the PLCs by testing the live data being read across the network. After an alarm has been posted, the AH task checks for operator acknowledgment of the condition and clearance of the alarm. When an alarm is acknowledged and cleared, additional entries are created and logged in the event history file.

The Volumetric task (VL) calculates all the volume data on the tanks configured in the system. Normally, the volumetric task suspends waiting on a queue message when the tank data has changed. When the AH task recognizes a change in the tank data, a message is enqueued to VL requesting volume calculations. The gross volume is calculated using the polynomial relating level and volume described previously. Next, the available room and available inventory are calculated using the working capacity (volume at fill height) and the water gauge volume. The net volume is then calculated by finding the volume correction factor from the product characteristics and the temperature. All these volume parameters are stored in the database tank record. Volume calculations are also requested from OI when the operator changes a tank parameter which affects volume.

The Flow Tolerance (FT) task is designed to maintain a rolling average of the tank flow rate and check for occurrence of the tank rate alarms. This task suspends waiting on a semaphore signal from System

Control task which is sent every minute. When signaled, the FT task projects the flow rate on every configured tank in a moving mode by comparing the last level to the current level. The instantaneous flow rate is then combined into an average of the previous four flow rates. Next the volume remaining to a target level is divided by the flow rate average to obtain an approximate time-to-target. The FT task also checks the rate alarms and enqueues a message to AH if a violation is found. For instance, if the difference between the previous level and the current level exceeds the drastic level deadband, the drastic level change alarm will be sent to Alarm Handler.

The Trending and Tank History tasks produce historical records of the tank data. The Trending task maintains a 15 minute rolling record of the last two hours of tank level, temperature, and alarm activity. This record is displayed on the single tank screen in Figure 5 on page 17. A similar trending function is performed by the Tank History task which maintains hourly readings for the last 48 hours. The Tank History task maintains a record of tank level, temperature, volume, flow rate, tank status, and outstanding alarms. Figure 14 shows the format of the tank history screen. Both the Trending and Tank History tasks suspend on a semaphore which is signaled at the proper time by the System Control task. The trended data records are maintained in disk files.

The Communication Control (CC) task performs all the network communication functions in the host computer. The functionality and detailed design of this task are described in the remainder of this document.

DATE : 12/17 INDIVIDUAL TANK HISTORY NETWORK STATUS :
 TIME : 19:30 TAG ID TK 178A PAGE 01 ALL LINKS OK

DATE MM/DD	TIME HH:MM	LEVEL FEET	FLOW RATE GALLON/HR	NET VOL BARRELS	TEMP DEG F	TANK STATUS	ALARM GROUP
12/17	19:30	25.25	1200	5423566	77	FILLING	
12/17	19:15	25.54	1214	5423564	77	FILLING	
12/17	19:00	25.67	1225	5423651	77	FILLING	
12/17	18:45	25.78	1201	5423656	76	FILLING	
12/17	18:30	25.85	1175	5424654	77	FILLING	
12/17	18:15	25.92	1210	5424654	75	FILLING	LTA
12/17	18:00	26.01	1200	5424584	77	FILLING	
12/17	17:45	26.14	1200	5424584	76	FILLING	
12/17	17:30	26.24	1214	5425874	78	FILLING	
12/17	17:15	26.33	1203	5425954	79	FILLING	
12/17	17:00	26.37	1207	5426665	78	FILLING	
12/17	16:45	26.44	1214	5426888	77	FILLING	
12/17	16:30	26.57	1205	5427152	76	FILLING	
12/17	16:15	26.77	1198	5427563	76	FILLING	
12/17	16:00	26.89	1205	5427725	77	FILLING	
12/17	15:45	26.92	1204	5427954	78	FILLING	
12/17	15:30	27.25	1187	5428524	77	FILLING	
12/17	15:15	27.54	1250	5428354	80	FILLING	
12/17	15:00	27.88	1202	5428652	79	FILLING	
12/17	14:45	27.97	1204	5428820	78	FILLING	
12/17	14:30	28.12	1202	5428901	77	FILLING	
12/17	14:15	28.25	1240	5429244	77	FILLING	
12/17	14:00	28.34	1217	5429304	77	FILLING	
12/17	13:45	28.48	1205	5429473	77	FILLING	
12/17	13:30	28.57	1243	5429543	79	FILLING	
12/17	13:15	28.62	1201	5429651	77	FILLING	
12/17	13:00	28.71	1202	5429654	77	FILLING	
12/17	12:45	28.82	1186	5429704	78	FILLING	
12/17	12:30	28.90	1220	5429734	77	FILLING	
12/17	12:15	29.12	1230	5429814	77	FILLING	
12/17	12:00	29.24	1224	5429853	75	FILLING	LTA
12/17	11:45	29.45	1211	5429914	77	FILLING	
12/17	11:30	29.70	1205	5429975	77	FILLING	
12/17	11:15	29.91	1184	5432541	76	FILLING	
12/17	11:00	30.04	1205	5433654	77	FILLING	

GRP#--ALARM GRP:

Figure 14. Tank history screen format.

CHAPTER II

NETWORK DESCRIPTION

Open Systems Interconnection Model

The communication network provides a means to connect the various controllers and remote data sensing equipment to the host computer system. To control the flow and sequencing of data transmissions across the network, a set of standards and procedures is used in the development of the host computer and the remote device interfaces. The Open Systems Interconnection (OSI) model will be used to describe the functioning of this network. This model is a set of standards for describing and building a communication network which allows connection of a variety of equipment adhering to the network definition.

The OSI reference model describes the many functions used to communicate between computers in a distributed system.

Implementations of these functions consist of software written to span the gap between the application process, which starts the communication . . . and the physical medium over which the communication travels Often this software is embedded in special-purpose circuitry that is included in computers or other communications equipment.¹

The OSI model consists of seven layers, each describing a separate area of functionality in the network system. Figure 15 shows the sequence of the seven layers. Each layer builds upon the functionality of the layer below it. The following paragraphs describe each layer of the OSI model and the particular implementation used for the TFA system network.

¹ John Voelcker, "Helping computers communicate," IEEE Spectrum, March 1986, p. 62.

	OSI LAYER	DESCRIPTION	TFA SYSTEM IMPLEMENTATION
7	APPLICATION	APPLICATION ACCESS COMMAND INTERPRETATION	DEVICE INITIALIZATION REAL TIME DATA GATHERS COMMAND QUEUE TRANSMISSIONS SERVICE QUEUE DIAGNOSTICS
6	PRESENTATION	COMMUNICATION SYNTAX DATA FORMATTING FRAME CONSTRUCTION	NETWORK COMMUNICATION PRIMITIVES FRAME DEFINITIONS
5	SESSION	SESSION CREATION DIALOG MANAGEMENT SESSION TERMINATION	NOT USED
4	TRANSPORT	MESSAGE SEGMENTATION FLOW CONTROL MESSAGE REASSEMBLY	NOT USED
3	NETWORK	MESSAGE ROUTING STORE AND FORWARD	NOT USED
2	DATA LINK	PROTOCOL MESSAGE FRAMING STATION ADDRESSING ERROR CONTROL	HDLC PROTOCOL UNBALANCED TOPOLOGY NORMAL RESPONSE MODE
1	PHYSICAL	TRANSMISSION MEDIA CHANNEL ISOLATION STATION CONNECTION	LOCAL LINE RS 232 - RS 423 (NRZI)

Figure 15. OSI reference model layers.

The physical layer of the OSI model describes the actual physical channel which carries the data stream as well as the mechanisms for activating and deactivating the transmitting and receiving stations. This bottom layer of the model covers both the electrical and mechanical characteristics of the media. The host computer network is capable of supporting several different media types. EIA standard RS-232, RS-422, and RS-423 are the most familiar interfaces supported. Specifically, the RS-232 standard allows connection to modems for long distance transmission across phone lines, terrestrial microwave links, and satellite uplinks.

The physical media used in the industrial environment is called local line. This media consists of a dual twisted-pair line driven by differential line drivers. These line drivers are optically isolated from the host computer chassis to provide safety from lightning strikes and power surges up to 2500 volts. This media can span a distance of up to 25,000 feet running at 9600 baud. Easy connection of the remote nodes is accomplished by simply splicing into the twisted pair and connecting a node tap. This permits a multi-drop configuration of stations on the line.

The data link layer of the OSI model provides for control and synchronization of the data transmitted across the physical link layer. Control of the flow of information is achieved regardless of the contents of the data. The protocol used in the TFA system is called the High-Level Data Link Control (HDLC) procedure. HDLC is a bit oriented protocol which allows asynchronous communication over a channel. HDLC used on the local line network media is transmitted using zero

insertion. This allows automatic synchronization of the transmitter and receiver by inserting zeros into the bit stream after a certain number of one bits has passed.

Historically HDLC was developed for unbalanced link level operation with one master station on the link controlling one slave or a number of slaves. The master station is responsible for initiating all data transfers and for initialising and controlling the link. The master station in HDLC is called a primary and the slave a secondary.²

The unbalanced mode of operation is implemented by a single host computer serving as the primary controlling the communication of multiple secondaries. Figure 16 shows the topology of a single channel of the network. The host computer conducts transmission on up to eight of these lines simultaneously. Operation of the HDLC link in the normal response mode means that no secondary responds until directly addressed and commanded by the primary station. A detailed discussion of the HDLC protocol is presented later in the protocol description section of this chapter.

The network layer of the OSI model provides for inter-network routing of messages. This is usually done using a method called packet switching where data packets are assigned a destination address and each node performs a store and forward function. The devices in the TFA system have the capability of implementing the X.25 packet switching method. Because no need exists for inter-network communications in the TFA system, this function has been disabled in all devices and will not be described further.

² D.W. Davies et al., Computer Networks and Their Protocols (Chichester: John Wiley & Sons, 1981), p. 214.

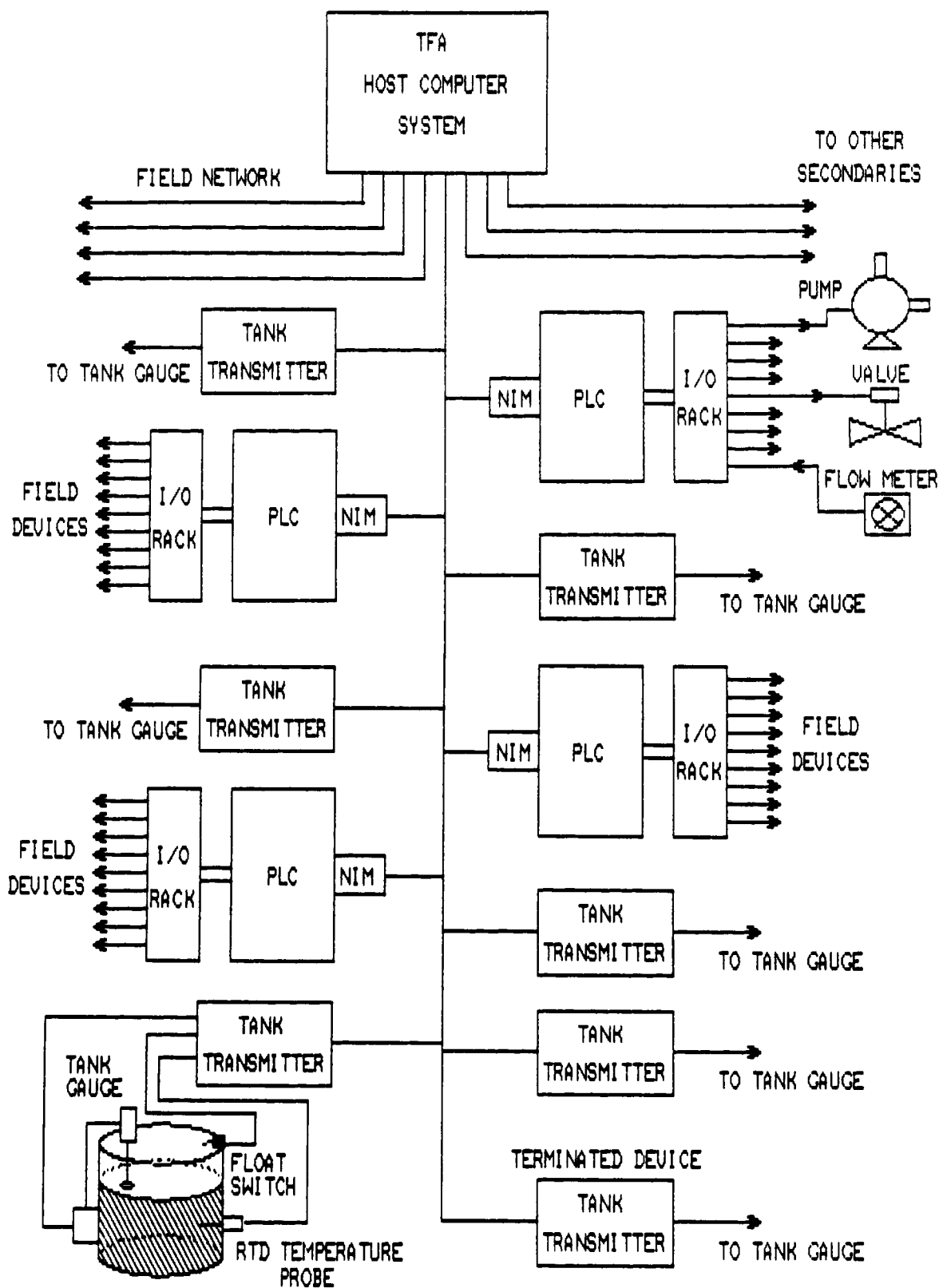


Figure 16. HDLC network topology.

The transport and session layers of the OSI model provide for message multiplexing, message segmentation, and conversation synchronization. Because this is an isolated network without provisions for redundant paths and inter-network traffic, these layers have not been implemented in the current communication scheme for the host computer.

The presentation layer of the OSI model takes care of the formatting and syntax of messages between parties. The presentation layer is defined by the communication primitives and frame structure used between the host computer and the network devices. The communication primitives and structure are described in detail in the section on communication primitive description later in this chapter.

The application layer of the OSI model describes the high level requests for data and commands sent across the network. This layer governs both the structure of the application data as well as the handling and storage of the data. The communication control task described in this document implements the high level interface between the application tasks described earlier and the network services which retrieve the data and issue commands on the network. The functions of the application layer are described extensively in Chapter III.

Protocol Description

The network protocol describes the set of procedures used for the orderly control of messages between entities on the network. These procedures not only describe the information exchange and response acknowledgment but also specify the error detection and recovery

procedures. The HDLC protocol provides for the control of information flow by processing messages as frames of data. Figure 17 shows the structure of an HDLC frame.

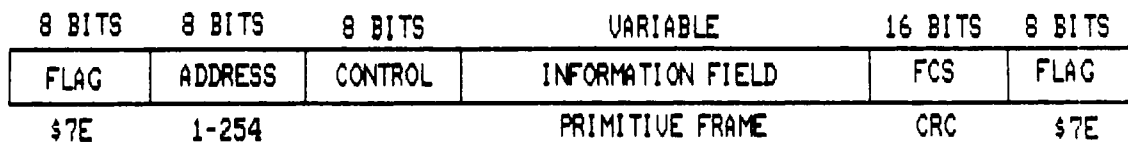


Figure 17. HDLC frame structure.

The flag field is the frame delimiter which indicates the beginning and end of a complete frame. This eight bit field contains a single zero bit followed by seven ones and ending with a zero. Numerically, this is 7E in hexadecimal notation. Because this must be a unique pattern, zero insertion must be used to prevent an occurrence of this number as a normal data pattern.

The address field contains the destination secondary address of the HDLC frame when transmitted from the host. When issued as a reply frame back to the host computer, this field contains the responding station address. The range of secondary addresses is 0 to 255. Each secondary on a channel is assigned a unique address. When a frame

is transmitted with that address, the appropriate secondary will accept the frame and respond back using the same address. Address 255 is reserved for broadcast transmissions which all devices will accept. No reply is issued by any secondary on a broadcast.

The control field specifies the nature of the command being sent to the secondary. Three classes of control commands are contained in the control field: (1) information frames, (2) numbered supervisory frames, and (3) unnumbered control frames. Figure 18 shows the bit assignments in the control field for each class of command.

	MSB				LSB
INFORMATION FRAME	N(R)	P/F	N(S)	0	
NUMBERED SUPERVISORY	N(R)	P/F	S	0	1
UNNUMBERED	N	P/F	M	1	1

Figure 18. HDLC frame control field arrangement.

The N(R) and N(S) segments are modulo 8 sequence counters which keep track of the number of received and sent messages. The poll/final (P/F) bit is used by the primary station to solicit a response from the secondary and is set to one by the secondary to indicate the end of a transmission. The S, N, and M fields are used to indicate the nature of the supervisory or control frame.

The information field of the HDLC frame contains the application

data and any additional control fields for other layers of the OSI model. The HDLC information frame (I-frame) transports the information field regardless of the contents.

The frame check sequence (FCS) field is used to ensure the correctness of the frame. A 16 bit cyclic redundancy check (CRC) is calculated by passing each byte of the frame through a polynomial equation. The CRC is calculated on the address, control, and information fields excluding the inserted zero bits.

The command structures used by the HDLC protocol allow for the orderly transfer and acknowledgment of messages. Normal response mode is used in the TFA system because the host computer is the central concentrator of data and the primary controller of the network.

The origins of HDLC lie in link level protocols for unbalanced multidrop configurations. For this reason, what is referred to as the normal response mode (NRM) of operation is that the secondary transmits responses to a poll from the primary. In a configuration operating in this mode a secondary obtains permission to transmit frames when it receives a command from its primary with the poll bit set.³

The first action on the network is to bring a secondary device online. This is accomplished when the host sends the set normal response mode (SNRM) command. The SNRM is an unnumbered control frame as shown in figure 18. The proper response from the secondary is an unnumbered acknowledge (UA) frame. Figure 19 shows the SNRM sequence between primary and secondary station. When the SNRM command is acknowledged by a secondary, the host will internally remember that the device is online.

³ Davies et al., p. 217.

A. Online mode setting sequence.

Primary: : SNRM,P :

Secondary: : UA,F :

B. Receiver ready sequence.

Primary: : RR 0,P :

Secondary: : RR 1,F :

C. Receiver not ready sequence.

Primary: : RR 0,P :

Secondary: : RNR 1,F :

D. Disconnect mode sequence,

Primary: : RR 0,P : : SNRM,P :

Secondary: : DM,F : : UA,F :

E. Information frame transfer sequence.

Primary: : RR 0,P : : I 0,0,P : ..delay.

Secondary: : RR 1,F : : RR 1,F :

Primary: ... : RR 1,P : : I 1,1,P :

Secondary: : I 0,1,F : : RR 2,F :

Figure 19. Sample HDLC frame sequences.

The receiver ready (RR) and receiver not ready (RNR) control commands are numbered supervisory frames used to determine if a secondary device can accept a new frame. The RR or RNR response from the secondary indicates whether that device can accept the frame. Figure 19 presents examples of a typical exchange.

The disconnect mode (DM) response from a secondary indicates that the device is not yet online. In response to a DM frame, the host will send a SNRM frame after which the secondary will respond with a UA frame. This sequence is also shown in Figure 19.

The information frame (I-frame) is used to transport the application data in the information field. When the frame is issued, the N(R) and N(S) sequence numbers are used to keep track of the message order to the secondary. The host will set the P/F bit to signal the secondary that a response frame is expected. Figure 19 shows the sequence for transmitting an I-frame. The first action is for the host to send an RR frame to ensure the secondary is available to receive an I-frame. If the response from the secondary is a response RR frame, the I-frame is sent. The secondary will respond with an RR or RNR depending on whether the I-frame buffer is filled. After an appropriate delay for processing the command, the host will poll the secondary with another RR frame to test for completion. If the frame is complete, the next response will be the return I-frame containing the requested data. If the secondary is not finished, the response would be an RNR frame. This sequence is continuously repeated with new sequence numbers and additional I-frames.

Several other commands and exchange sequences are used in the HDLC

protocol. Some of these include the reject command and the disconnect request command. These are rarely used and will not be discussed.

The HDLC protocol in the host computer is entirely implemented by the intelligent network communication board. To initiate a transmission, the communication task constructs the information field section of the I-frame and transfers it to the communications board along with the network address and channel number. The communications board appends the HDLC frame to this information field and controls the transmission sequence to the requested secondary. After the response I-frame returns from the secondary, the communications board notifies the host CPU of completion. If any errors occur during transmission, the communications card is programmed to take appropriate error recovery action such as retrying the frame or sending a SNRM frame. The HDLC board is capable of handling multiple outstanding frames on several secondaries simultaneously.

The network interface module (NIM) connected to each distributed PLC implements the HDLC protocol. When a request for data is received, the NIM translates the command into the language of the PLC and retrieves the requested data. This data is formatted into a reply I-frame and installed in a buffer in the NIM waiting on the host RR poll. When the host sends the RR poll, the NIM will respond with the return I-frame using the proper N(R) and N(S) sequence numbers. The NIM therefore serves to implement the data link layer of the OSI model for the PLC. The ITT connected to each tank operates similar to a PLC. Since no other processors are built into the device, the data link layer is implemented directly in the ITT firmware.

Communication Primitive Description

The presentation layer of the OSI model is implemented using a set of communication primitives. These primitives form a standardized language on the network for requesting and sending data between devices. Any device which communicates using these primitives can be connected to the network and operate properly. The presentation layer utilizing these primitives is contained entirely in the HDLC information field within an I-frame. After the communication task in the host computer builds up the primitive frame, it submits them to the network communications board to have the HDLC frame appended before transmission. By constructing the primitive frame, the host communication task implements the presentation layer of the OSI model

The primitive structure contains several fields in the command and reply frames. Figure 20 shows the field structure of the primitive frame for both a transmitted command frame and a received reply frame.

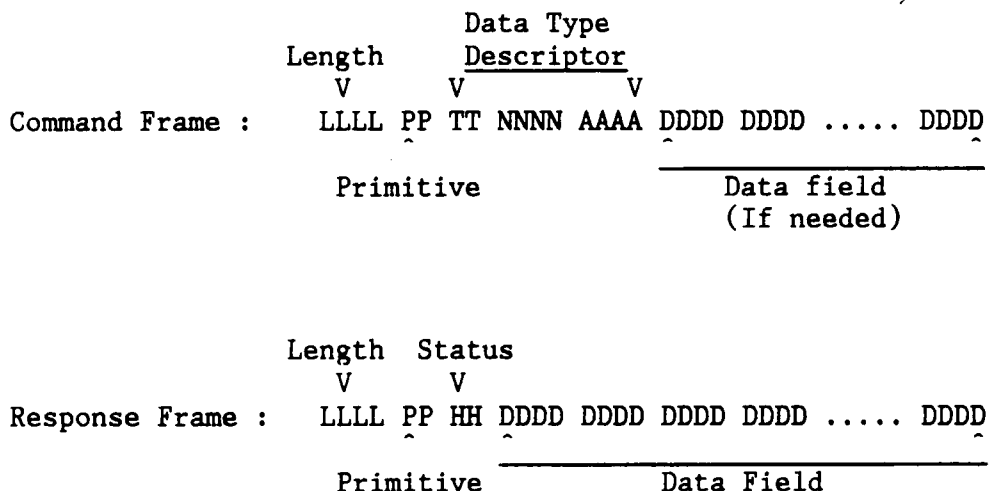


Figure 20. Primitive Field Structure.

The first word of both the command and response frame is the LLLL length field. This field is the total number of bytes in the primitive frame excluding the LLLL field.

The next field is the primitive command field. This one byte field indicates the nature of the command or request. Each primitive has a specific request and response format. The primitives are divided into functional groups: (1) exception, (2) status, (3) control, (4) read, (5) write, and (6) data acquisition. Each of these functional groups will be covered in depth.

In the response frame, the HH field after the returned primitive indicates the status of the attached controller. This field will indicate whether the device is executing the ladder logic program, if it has aborted operation due to an error, or if it has been manually shut down.

The next field is the data descriptor field. This field varies according to the type of primitive. The data fields also vary according to the primitive and data type chosen. Most of the primitives use the TT field in the data descriptor area. This field refers to the data type to be read or modified by the command. There are over 50 different data types which can be selected using the one byte TT field. The NNNN parameter specifies the number of locations affected and the AAAA field is the starting address of the first location. In the data field, the information is formatted as either integer words, packed bits, individual bits, or floating point numbers.

The exception primitive (00) is used to inform the primary station of a problem occurrence in the last transmission. This primitive only

appears in a response frame. The format for the exception primitive is:

Response Frame: LLLL 00 PP DDDD TT

The PP field is the primitive issued when the error occurred. A code indicting the nature of the problem is contained in the DDDD field. Typical examples of problems are: primitive not implemented, data type not defined on this device, and data address out of range. The TT field is used with some exceptions to indicate the requested data type.

The native primitive (01) is used to issue a device-specific command in the language of the particular PLC. This primitive essentially bypasses the NIM and sends the native command following the primitive directly into the PLC. This is typically used to invoke the device self diagnostic sequence on each PLC by sending the specific native directly to the internal CPU.

The status primitive (02) is used to ascertain the device operational status similar to the HH field in each response frame. The reply frame to the status command contains four fields as shown:

Response frame: LLLL 02 HH EE FF GG

The HH field is the same as in every primitive reply. The EE field indicates the status of the auxiliary power source if one is used. An indication of the status of the NIM is contained in the FF field. The GG field indicates whether the NIM is in local or remote mode.

The configuration read command (03) is used to determine the type

of PLC being used and the memory sizes for the device. The response frame format is as follows:

Response frame: LLLL 03 HH DDDD EEEE FFFF GGGG IIII JJJJ

The HH field is the status field as usual. Identification of the device type is contained in the DDDD field. The EEEE parameter contains the size of instruction memory containing the ladder logic program. The FFFF field corresponds to the size of variable memory, the GGGG field the size of constant memory, and the IIII and JJJJ fields contain the size of local and global input/output memory.

The change state primitive (10) allows the host computer to remotely change the device operational state indicated in the HH parameter. The format of the primitive is:

Command frame : LLLL 10 DD

The DD field requests the mode desired for the controller such as: (01) enter operational mode, (02) enter non-execution of ladder logic while running loops, or (03) enter non-execution of ladder logic and freeze loop state.

The read data primitive (20) is used to read contiguous data locations in a specific memory type. The command format is :

Command frame: LLLL 20 TT NNNN AAAA

In the read primitive, the TT field defines the type of memory to access. The NNNN field identifies the number of locations and the AAAA field is the starting address to read. The response frame format is:

Response frame: LLLL 20 HH DDDD DDDD DDDD

The HH field is the usual device status reply. In the response, the DDDD fields are the data locations requested starting at location AAAA through location (AAAA + NNNN) in the selected data area.

The write block primitive (30) is used to write new data into a group of contiguous memory locations. In the write primitive request frame, the format is:

Command frame: LLLL 30 TT AAAA DDDD DDDD DDDD

Similar to the read primitive, the TT field defines the data type and the AAAA field contains the starting address of the block to write. The DDDD field contains the data to be written and the number of DDDD words determines the number of locations to be modified.

The data acquisition group of primitives allow the host to define up to 32 blocks of data and the retrieve them by requesting the data by block number. To set up the data areas associated with each block, the define block primitive (50) contains these fields:

Command frame: LLLL 50 CC TT NNNN AAAA CC TT NNNN AAAA

The CC field in the define frame identifies the block number to be defined. Similar to the read primitive, the TT NNNN AAAA fields identify the memory type, number of locations, and starting location to be associated with this block. Block definition fields can be repeated for however many blocks are needed up to 32 times.

The gather block primitive (51) is used to retrieve the previously defined blocks. The format of the command is:

Command frame: LLLL 51 EEEEEEEE

The EEEEEEEE field is a bit mask with one bit for every block available. When the gather command is issued, the data blocks indicated by the corresponding bits being set to one are returned consecutively with no delimiters. The response frame format is:

Response frame: LLLL 51 HH EEEEEEEE DDDD DDDD ... DDDD DDDD ...

The HH status is returned as usual along with a copy of the requested block mask. It is the user's responsibility to ensure that the requested data in the desired blocks will fit in a single I-frame reply.

The final data acquisition primitive is the write and gather primitive (52). This primitive allows for the flow of data in both directions with one request and response frame. The format of the command frame is:

Command frame: LLLL 52 EEEEEEEE TT AAAA DDDD DDDD DDDD

The EEEEEEEE block mask is the same as the gather block mask previously described. Like the write primitive, the TT AAAA DDDD fields are the data type, starting address, and data words to be written to the controller. This structure is exactly like the write primitive 30 described earlier. The response format is:

Response frame: LLLL 52 HH EEEEEEEE DDDD DDDD DDDD DDDD

The response frame is exactly like the gather frame with a repeat of the block mask followed by the data defined by the blocks requested.

Now that a description of the network using the OSI model has been fully developed, an example of a complete transaction using each layer of the model will be presented. Figure 21 shows the OSI model layers in the host computer and a PLC along with the transaction at each layer during a typical data exchange.

The communication transaction starts at the application layer of the host computer with a request for the analog inputs in PLC number 3. As the transaction progresses to the next level, the presentation layer interprets this into a request to gather the previously defined block number 7 from the PLC on channel 4 address 27. This gather primitive frame is then passed to the communication board. The data link layer implemented in the communications board appends the HDLC control fields onto the primitive frame to create the an I-frame. This frame is then transmitted through the physical layer across the local line media.

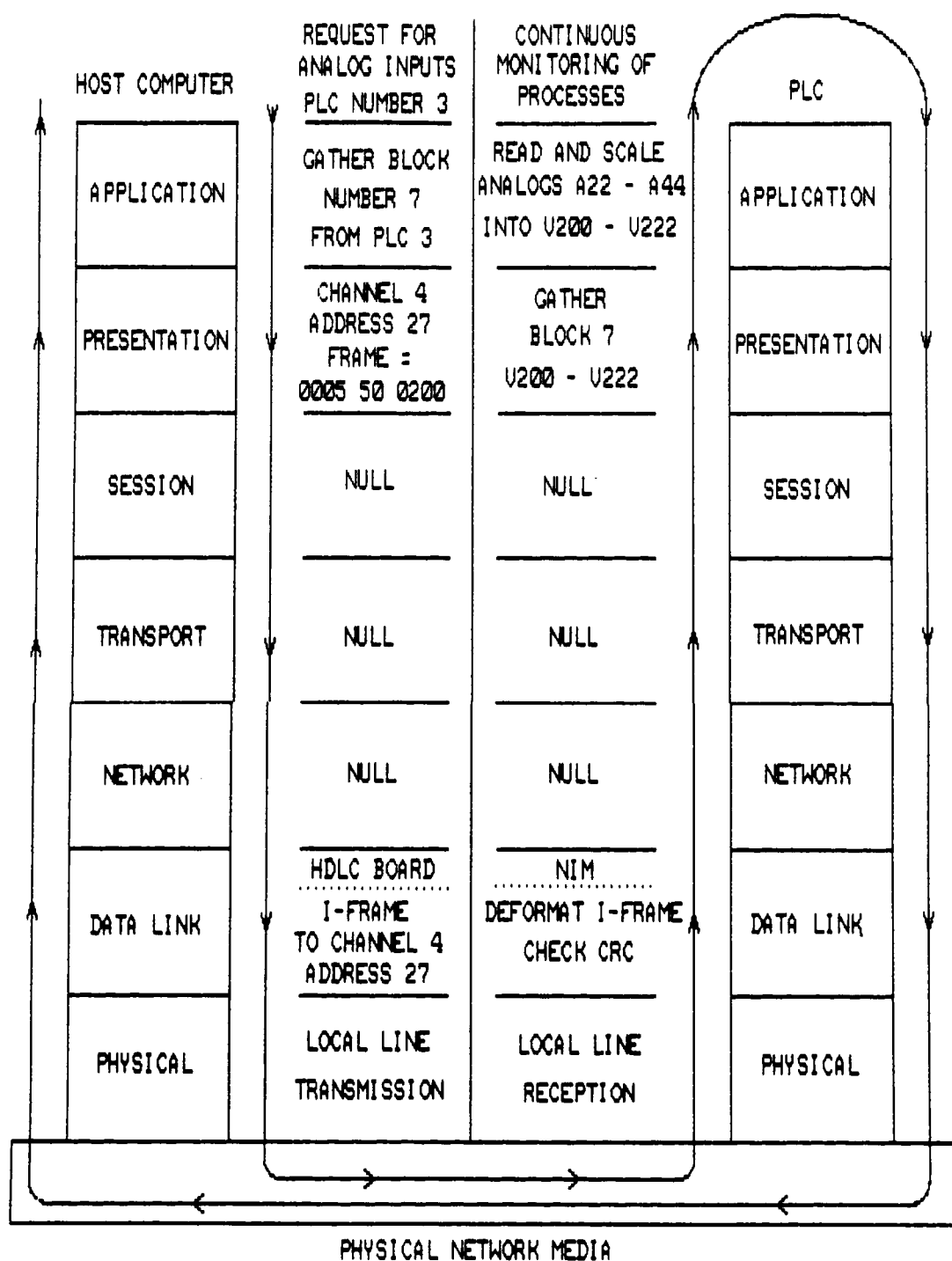


Figure 21. Network transaction example.

When the I-frame is received by the PLC, the NIM will strip off the HDLC control fields and check the CRC for correct transmission across the network. If no HDLC errors are detected, the frame is passed from the data link layer to the presentation layer in the NIM for interpretation. The gather block command is recognized and the NIM will pass a command to the PLC requesting the memory area specified by the block definition. In a PLC, the application layer consists of the ladder logic program which retrieves the analog data from the input module, scales the data, and installs it in the block defined by the host. When the data is retrieved from memory, the NIM will format the primitive response frame and pass it to the data link layer. This layer will then append the necessary HDLC I-frame response fields to the frame and wait on the RR poll from the host for the response. As soon as the RR is received, the I-frame is transmitted back across the network media to the host.

Once received by the host, the HDLC fields are stripped from the primitive frame and checked for correctness. The primitive frame is then passed back up to the presentation layer where a check is made for the exception primitive 00. If no errors are found, the data is then made available to the requesting task in the host computer for use within the application layer.

CHAPTER III

HOST COMMUNICATIONS TASK FUNCTIONALITY

Task Interface Definition

The Communications Control (CC) task in the host computer serves as an interface between the other application tasks and the devices connected to the network. The primary activity of the CC task is to coordinate the flow of data across the network between the tasks in the host computer which use the field data and issue commands and the devices in the field which generate the data and activate the outputs. This interface function is accomplished by implementing the presentation and application layer of the OSI model. These interface activities include interpreting the data requests, converting to the appropriate language (primitives) for the network, issuing the formatted messages, retrieving the responses, deformatting and error checking the returned data, and presenting the data to the application tasks.

The interfaces to the CC task are in two directions: (1) the interface to the network through the network communications board and (2) the interfaces to the applications tasks for data requests and command transmissions. Figure 22 shows the interface environment of the CC task.

Communication to the network is made through the network communications board by transferring the primitive frames and control signals through the dual-port memory buffer. This buffer is 64 Kbytes in size and is accessible to the host computer CPU on one

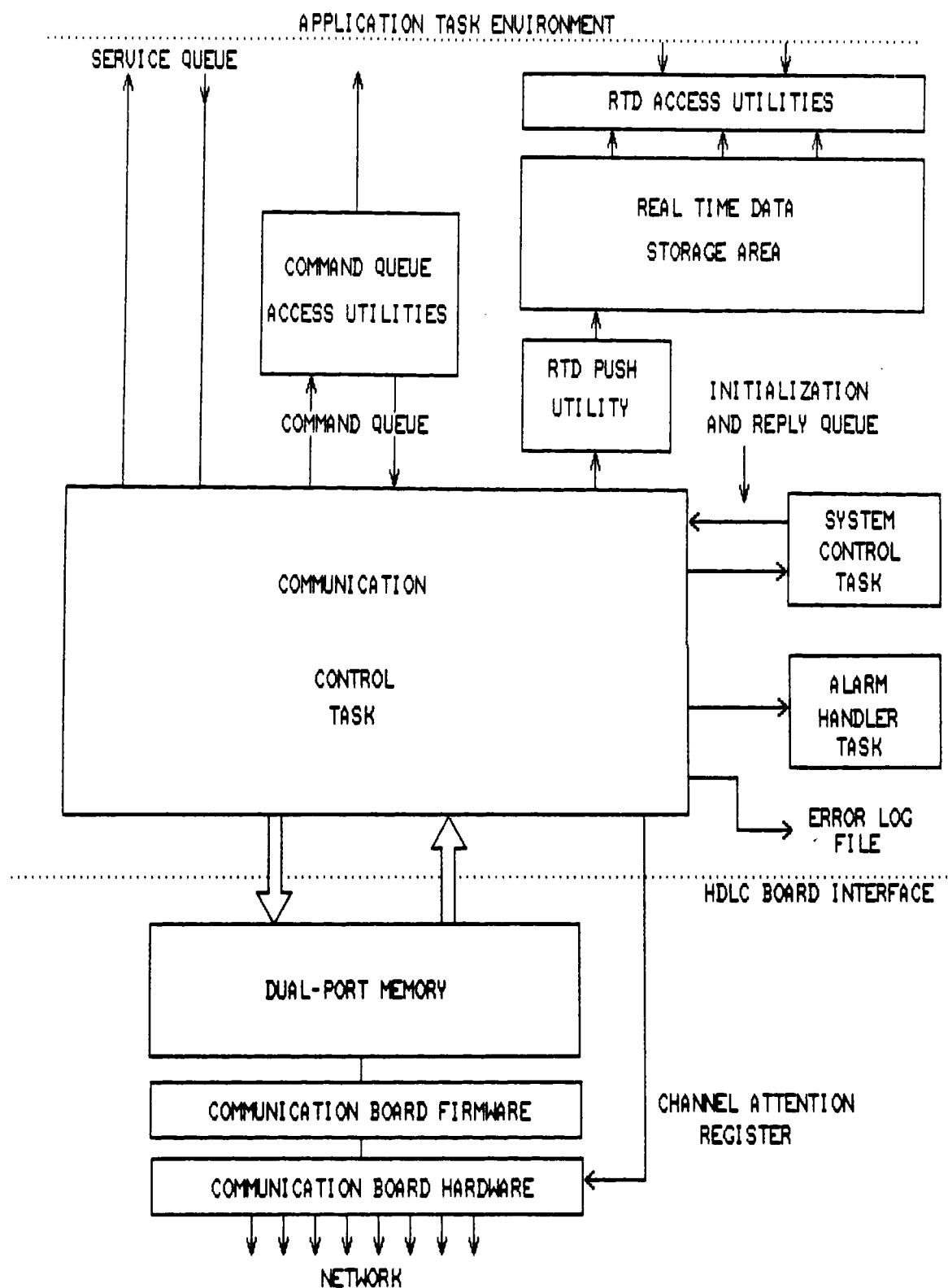


Figure 22. Communications Control task interfaces.

side and to the onboard processor on the network side. The frames and control signals placed in the host CPU side of the memory by the CC task are retrieved on the other side of the buffer by the onboard processor and transmitted. When the reply frame is received, it is placed back in the buffer by the onboard processor and flags in the buffer are set to notify the CC task. The data can then be retrieved and used by the host CPU.

Four data structures are maintained in the dual-port buffer for organizing the flow of data: (1) message mailboxes, (2) request blocks, (3) transmit frame buffers, and (4) receive frame buffers. Figure 23 shows the organization of these data structures within the dual-port memory. While the mailboxes only exist once in the buffer, multiple copies of the request blocks, transmit buffers, and receive buffers are created depending on the number of configured devices on the network and the amount of data transmitted by each device.

The mailboxes provide a method of initializing the network communications board as well as monitoring the running status of the board during normal operation. Initialization of the board by the CC task is done by writing board configuration parameters into the mailboxes and then sending a channel attention command through the command mailbox. These configuration parameters include channel baud rate, selection of synchronous or asynchronous operation, number of acceptable errors before the device goes offline, and the number of retries on each error condition before failure. Once the channels have been configured and initialized, the board firmware will begin servicing transmission requests to the network. When the board is in

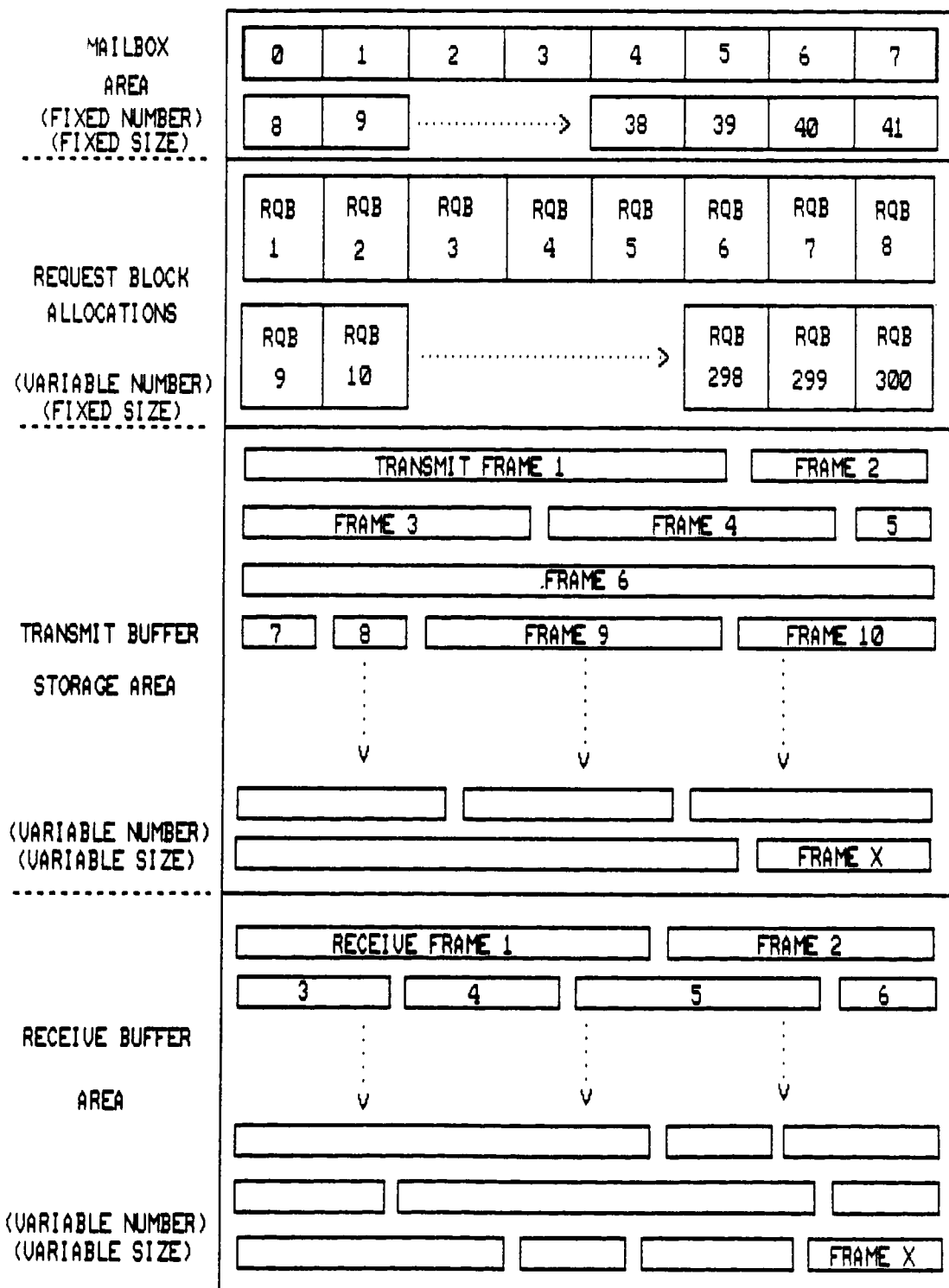


Figure 23. Dual-port memory data structure configuration.

full operation, the mailboxes are checked after every transaction for messages indicating internal failures in the network communications board. If this failure is detected, the system will automatically shut down for repairs and execute diagnostics.

The request blocks control the action of the network communication board with respect to each device on the network. Request blocks are allocated for the maximum number of devices (300) configurable on the network. Figure 24 shows the structure of a request block.

The flags in each request block are used as signaling devices between the network communication board and the CC task. When a frame is formatted and ready for transmission, the CC task will set the new request present flag. While the transmission is outstanding on the network, the request outstanding flag will be held true by the onboard processor. As soon as the received frame is installed in the receive buffer, the request outstanding flag will be cleared and the request complete flag set true by the onboard processor. The chain block flag is used to signal the end of the request block area by being set true on all blocks except the last block. To bring a device online, the single station reset flag is set by the CC task to request a device be sent a SNRM frame. After the request complete flag signals completion, the single station status flag indicates if the device responded correctly with a UA frame and came online. The block interrupt flag is used to connect several requests together for interrupt servicing. When set on several blocks, the CC task suspends waiting on the completion interrupt and then processes the data from several devices together. Finally, the disconnect flag is used to take a device offline.

NEW REQUEST PRESENT	REQUEST OUTSTANDING	REQUEST COMPLETE	CHAIN BLOCK FLAG	BYTE 1 - BIT 4
SINGLE STATION RESET (SNRM)	SINGLE STATION STATUS	BLOCK INTERRUPT REQUEST	UNNUMBERED INFORMATION FRAME	BYTE 1 - LSB
SINGLE STATION DISCONNECT	UNUSED	UNUSED	UNUSED	BYTE 2 - BIT 4
UNUSED	UNUSED	UNUSED	UNUSED	BYTE 2 - LSB
TRANSMIT BUFFER ADDRESS				WORD 2
RECEIVE BUFFER ADDRESS				WORD 3
ADDRESS SEGMENT		UNUSED		BYTE 7 AND 8
PORT NUMBER		SECONDARY ADDRESS		BYTE 9 AND 10
RETRANSMISSION COUNT		CRC ERROR COUNT		BYTE 11 AND 12
TRANSMITTED MESSAGE COUNT		RECEIVED MESSAGE COUNT		BYTE 13 AND 14
POLL TIME FOR LAST TRANSMISSION				WORD 8

Figure 24. Request block format.

The remaining parameters in the request block define the memory ranges and operational data for the device. The address segment location is used to define the memory page where the transmit and receive buffer are located. If the dual-port memory size is exceeded, the network communication board has the ability to use any memory on the host CPU bus. If this segment register is set to zero, the board defaults to usage of the onboard dual-port memory. The transmit and receive buffer address parameters are the lower 16 bits of the selected address locations. The network address and cable number define the location of the device on the network. Statistical data is recorded in a record of the number of transmitted messages, received messages, CRC errors, and error retries. The poll timer location is the amount of time, in milliseconds, between the transmitted frame and the response frame.

The transmit buffer is used to pass the transmit primitive frame to the communication board. Before transmitting a primitive frame, the communication board will add the HDLC I-frame data. Figure 25 shows the structure of the transmit buffer. The first item is the number of bytes

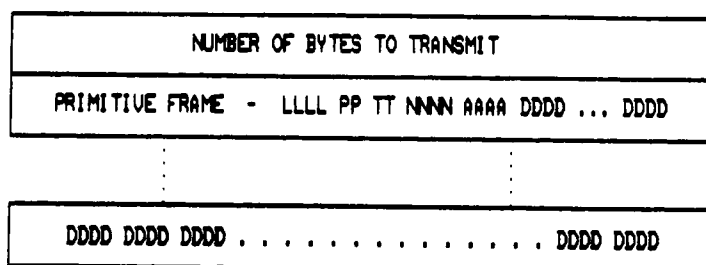


Figure 25. Transmit buffer format.

to transmit. Next, the primitive frame is installed in the buffer beginning with the length LLLL, primitive PP, and data descriptors.

The receive buffer is used to pass the reply frame of data back to the host computer by the communications board. Figure 26 shows the format of this buffer. The first location is an error code used to indicate any transmission or reception problems on the network. If this code is zero, the transmission was successful. The next item is the number of received bytes. This is followed by the primitive frame with the HDLC overhead information removed.

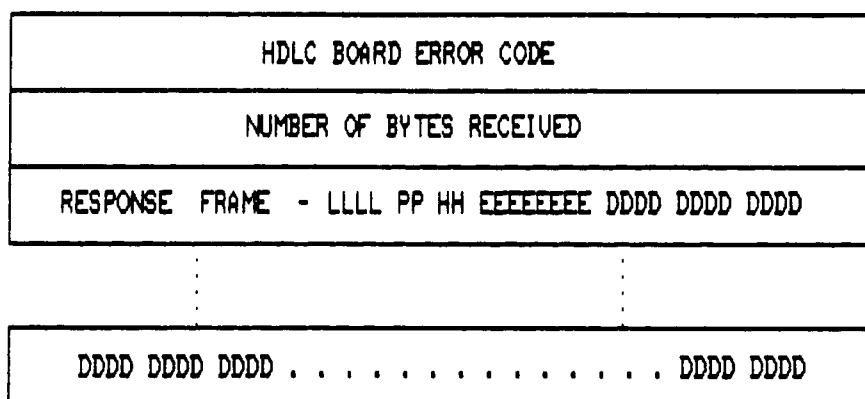


Figure 26. Receive buffer format.

The transmit and receive buffers are allocated for the exact size needed to hold the largest I-frames to be sent to or received from a particular device. Instead of reusing the transmit buffers by loading in a new I-frame between each transmission, the CC task creates a separate transmit buffer for each unique I-frame to be sent and maintains a record of the addresses of these buffers. When a new

I-frame is sent, the CC task merely changes the buffer address in the request block to point to the next buffer. Only one receive buffer is allocated for each device and it is sized for the maximum expected response frame from the device. Several other transmit and receive buffers are allocated at maximum size for use when an operator entered I-frame is to be sent since the size of this frame cannot be predicted. When the operator command frames are sent, these command buffers are used by changing the buffer addresses in the request block. After the transmission is completed, the addresses in the request block are pointed back to the original frames.

The communication interfaces to the application tasks are designed for two types of network data transactions: continuous retrieval of prearranged data structures from the network devices and operator commanded communication frames. Continuously retrieved data structures are configured in the device using the define block primitive. Each block is defined for a different type of data such as discrete inputs, analog inputs, or loop setpoints. This data is put into a table in the controller memory by the ladder logic program for retrieval by the host computer. After the blocks are defined, the host computer continuously retrieves the data using the gather block primitive. This data is processed by the CC task into a global structure called the real time data (RTD) area.

The RTD area is a dynamic global area designed to store and forward the process data to the application tasks. This area is dynamic because it is designed to assume the size exactly needed to hold the data retrieved with no extra space. Any task in the system can access

this data by using the global pointer. Figure 27 shows the organization of the RTD area. The RTD area is arranged by PLC number and then by each data type within a PLC. If a particular data type is not configured for a device, no space is allocated for that data type. The data for one device is immediately followed by the data for the next device. Access to the RTD area is controlled by a map which contains index pointers to each data block within RTD. When a particular data point is desired, the index from the map is used.

The RTD area contains both process data which is read continuously and initialization data which is read only when a device is brought online. Eighteen different data types are presently supported by the RTD area. The discrete inputs are single bit items used to indicate the state of a discrete input point on the PLC. These are packed eight to a byte in the discrete input table and loaded into RTD when read across the network. The discrete outputs are actually the indication of the state of an output on the PLC. These are packed as two bits per output with four to a byte. One bit is the state of the output and the other bit is an alarm status. The outputs are commanded by sending an operator formatted frame to turn the output bit on or off. After the command is sent, the discrete output status bit in RTD will reflect the change in the output state.

The analog inputs are 16 bit scaled integers used to monitor the value of an analog input on the PLC. Each analog has a corresponding alarm status byte to indicate if the analog has failed or violated preset limits contained in the PLC. The analog ranges and alarm limits are also maintained in RTD. Each PLC maintains these values in a table

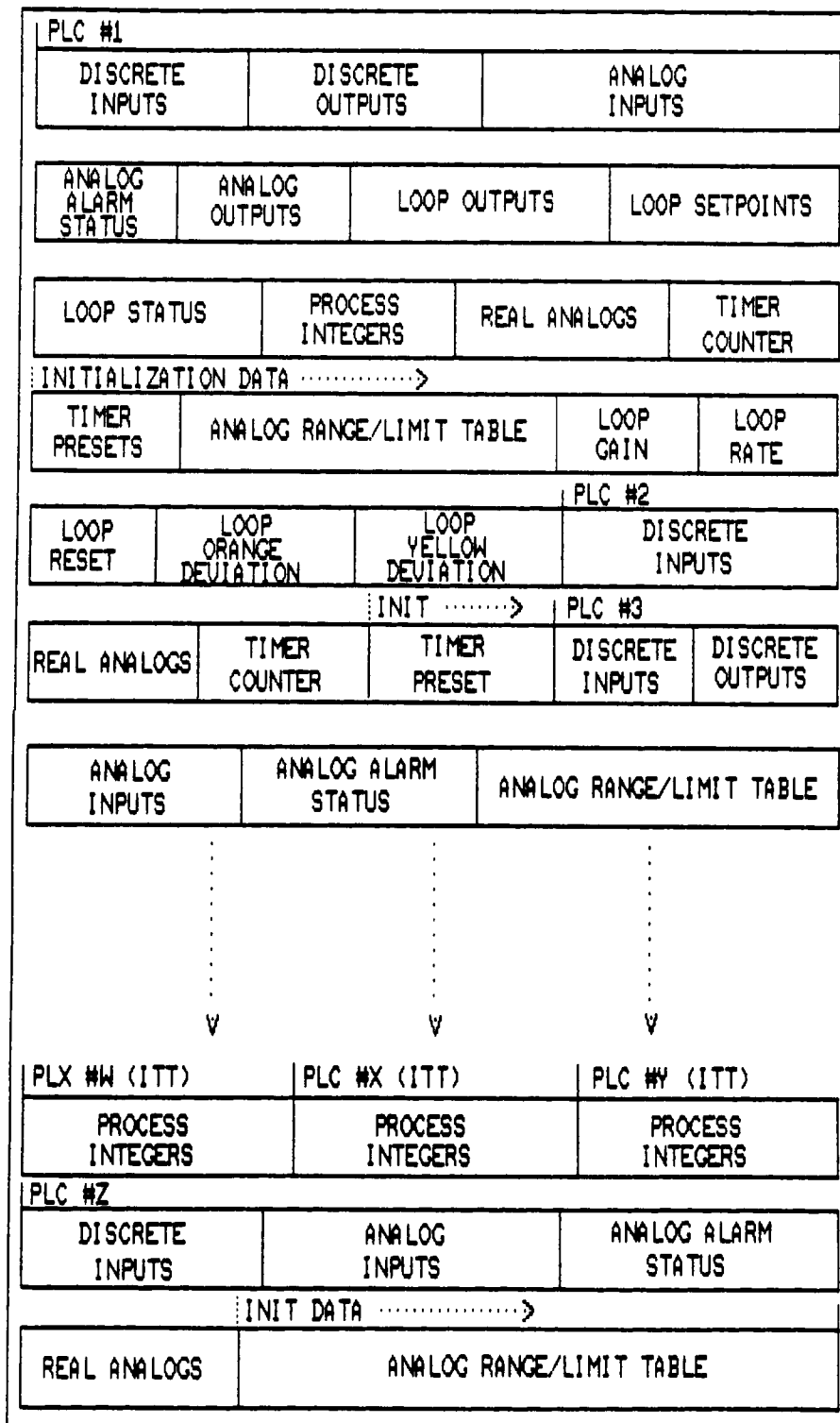


Figure 27. Real time data area organization.

for all the analogs configured on a device. These values are only read when the device is brought online or a value is changed from the host computer. The ranges and limits are also changed using an operator commanded I-frame.

PID loop data is also formatted into tables in the PLC memory by the ladder logic program. Loop data is organized as loop output, setpoint, and status in separate tables. The loop process variables are maintained as the first analog inputs in the analog data area. Loop output and setpoint are kept in the table as scaled 16 bit values and the status is 16 individual status bits packed into a word. The loop initialization data is also held in RTD and read only once. This data consists of the gain, rate, reset, orange deviation, and yellow deviation. The gain, rate, and reset values correspond to the proportional, integral, and derivative portions of the PID loop. Orange deviation and yellow deviation are alarm limits based on the percentage variation between the loop setpoint and the process variable being controlled.

To retrieve various other data points within the PLC which do not fall into the previous categories, the process integer and real analog data types are used. The only difference between these structures is that the process integer is a fixed point 16 bit number and the real analog is a 32 bit floating point number.

The timer/counter data type is used to read the values of process timers and event counters within the PLC. Similar to the analog inputs, the timer/counter current value is continuously updated from the PLC while the preset value is only read when the device is brought online.

The tasks in the TFA system access the RTD area using a set of global utilities. These utilities allow the task to get an individual data point or a block of values. Utilities are also provided to allow the CC task to put the data into RTD and allow an application task to update an initialization value which has been changed in the PLC. A task can check if a data point is configured in RTD by calling another utility. All the utilities use the indices in the map for access.

The `RTD_GET` utility allows an application task to get a particular data point from RTD. The procedure call to the utility is:

```
RTD_GET (PLC_NUMBER, DATA_TYPE, DATA_INDEX, DATA_BUFFER, ERROR);
```

A calling task specifies the PLC, data type, and data index point desired and the utility returns the data buffer containing the data and an error code. The error will be zero if no problems occurred and nonzero if the requested data point is out of range for that device. An example call would be to get analog input number four on PLC number 27. The return data buffer would contain the scaled value and status byte.

The `RTD_BLOCK` routine is used by the alarm handler task to retrieve all the status bits on a given PLC in a large array for alarm testing. The call to the procedure is:

```
RTD_BLOCK (PLC_NUMBER, BLOCK_TYPE, DATA_ARRAY, DATA_SIZE, ERROR);
```

To retrieve a block of alarm bits, the AH task specifies the PLC desired and the data type desired such as the discrete output alarm

bits or the loop status bits. The data is returned in an array of bits and the size is returned. Again, an error code is returned if the data requested is not available on that PLC.

The final RTD utility to be described is the `RTD_PUSH` routine. This utility is used by the CC task to install data into RTD as it is retrieved from the PLCs using the gather frames. The utility call is:

```
RTD_PUSH (DATA_ADDRESS, RTD_INDEX, DATA_SIZE, ERROR);
```

To install a block of data, the CC task passes the address of the data portion of the receive buffer frame in dual-port memory. Also, an index into the RTD area for that data type is maintained by the CC task and passed into `RTD_PUSH` along with the data size in bytes. An error is returned if the data specified will write over the data for an adjacent device because of an improper index or size.

The second method of data retrieval and transmission from the host computer is using operator commanded I-frames. These I-frames are formatted in the application task and passed to the CC task through the command queue interface. The CC task sends the frame to the desired PLC, retrieves the response frame, and checks it for errors. If no errors are present, the reply frame is enqueued back to the originating task. If an error is encountered during transmission, an error code is sent back to the task indicating the problem. The command queue provides a method for the application tasks to change data point values in the controllers and retrieve data on an exception basis.

The command queue interface is serviced by a set of utilities

designed to implement the details of formatting the primitive frames. An application task calls the utility and passes in the desired parameters and data. The utility will then format the data into a primitive frame, enqueue the frame to the CC task, wait on the return queue response frame, deformat the data from the return frame, and pass the requested data back to the task. A utility is implemented for each of the primitives described in chapter II. Several of the most used utilities will be discussed in detail as an example.

The CC_READ utility allows the application task to read any contiguous data structure in a PLC. The call to the utility is:

```
CC_READ (COMMAND, PLC_NUM, DATA_TYPE, NUM_LOCATIONS, START_LOCATION,  
         READ_DATA, STATUS, ERROR, COMPLETE);
```

The command parameter is entered as either wait, send, or receive. This provides the application task with the option of waiting on the reply or sending the frame, performing some other activity, and retrieving the response later. The PLC number, data type, number of locations, and start location define the area to be read. Data type is selected as any valid data structure supported by the PLC such as V memory, C memory, packed control registers (CR), or loop parameters. Response frame data is returned in the read data structure. This structure can be organized as bytes, words, booleans, or floating point numbers. It is the application task's responsibility to request no more than the maximum amount of data which can be retrieved in one I-frame and to request valid memory areas in the device. The status parameter is the HH parameter from the response primitive frame which indicates the running

status of the PLC. If the device has shut down on an error condition, the requested data will not be valid. An error code is returned nonzero if an HDLC problem was encountered during the transmission. The complete parameter is a boolean which is true if the response frame has returned. This is used primarily if the application task performs a send command and returns later to retrieve the data.

The CC_WRITE utility implements the write block primitive(30). The call to the procedure is:

```
CC_WRITE(COMMAND, PLC_NUMBER, DATA_TYPE, NUM_LOCATIONS, START_LOCATION,  
         WRITE_DATA, STATUS, ERROR, COMPLETE);
```

The call to this utility is much like the call to the read utility. To modify a PLC memory area, the user passes in the write buffer filled with the data to be written. Again, it is the application task's responsibility to write only the amount of data possible into a valid data structure in the PLC. If a mistake is made, the error parameter will indicate the PLC error code to the invalid action committed.

For data structures which are frequently retrieved through the command queue, the user can define acquisition blocks in the PLC. The CC_DEFINE utility allows the user to set up the block definition and send it to the controller. Since the other blocks are used by CC task to retrieve the RTD area data, the user is restricted to defining only blocks 28 to 32. The application task can then use the CC_GATHER and CC_WRGATHER commands to retrieve the blocks. The call to the write and gather command is:

```
CC_WRGATHER (COMMAND, PLC_NUMBER, DATA_TYPE, NUM_LOCATIONS,  
             START_LOCATION, WRITE_DATA, BLOCK_MASK, GATHER_DATA,  
             STATUS, ERROR, COMPLETE);
```

Command, status, error, and the complete boolean are used exactly as with the other commands. The PLC number, data type, number of locations, start location, and write data specify the data to be written into the controller. A block mask is specified for the blocks to retrieve and the block data will be gathered and returned in the gather data structure. The application task maintains a record of the amount of data defined in a block and must only request the number of blocks which will not overflow the reply I-frame.

The service queue interface to the CC task provides a method to change the operational status of a device, perform diagnostics, and control the statistics acquisition. To send a service queue request, the application task formats a queue message containing the PLC number, the desired action, and a reply queue number for the response. Possible actions are to take a device offline, put a device online, turn the statistics logger on, turn the statistics off, reset the statistics values, and request self diagnostics on the device. Taking a station offline will cause the device to be removed from the active list and no more RTD polling or command queue activity will be permitted. This is usually done when a PLC is going to be repaired. After the device is ready for service, a service queue request is made to put the device back online. When this is done, normal RTD polling and command queue activity will resume after the initialization primitive frames have been sent to define the blocks and gather the range/limit data.

Several other interfaces are used by the CC task during normal operation. The alarm handler input queue is used by the CC task to post and clear various communication alarms when a device on the network experiences a failure. To send an alarm message, the CC task formats a message containing the device number and alarm type and enqueues the message to the AH task. The AH task will then post the alarm on the alarm list and log the occurrence in the event history file. During system initialization, the CC task waits on a queue from the System Control task to coordinate the system startup activity. The SC task will enqueue a message to CC to start the initialization process. After the CC task starts up the communications board and brings the online devices in service, a response message is enqueued back to the SC task indicating success or failure of the initial actions. If the HDLC board fails initialization, the SC task will take the system down and execute the computer self diagnostics to ascertain the cause of the problem. The CC task also logs message in the system error log file when internal problems occur. Examples of problems are when a queue overflows with too many messages, when the database does not respond with the correct data, or when a system fatal error such as HDLC board failure occurs. The operator can view the error log screen to determine the cause of the failure and take corrective action.

The interfaces to the CC task are designed to minimize the complexity of the communication interface for the application environment. By utilizing utilities and menu driven screens to coordinate continuous and exception driven communication transactions, the process data can be easily accessed and modified by any task.

Real Time Data Polling

The real time data polling activity within the CC task is the primary communication transaction. The CC task continuously gathers blocks defined for selected data areas within the controller memory. This data is then installed in the RTD area previously described and made available for the application tasks. This has the effect of making the RTD area a window into the data storage areas in the controllers.

The system engineer configures the RTD areas in each device by filling in the PLC configuration screen on a host computer terminal. Figure 28 shows the format of this screen. This screen is requested from the system menu and the engineer enters the desired PLC number. All changes to this screen are made with the device offline. The changes to the RTD location parameters are used only after the system has been reinitialized since the RTD area is dynamically allocated. This configuration data is stored in the database.

The engineer begins configuration by entering the cable number and network address of the device. To ensure the proper device is being accessed, an encoded ID number is placed in a reserved V memory location and compared against the device identification code in the host. This location is read when the device is brought online and checked for correctness. The device polling priority determines the amount of communication resources to devote to this controller. Also, the device type and model number are used to ensure the correct device is at this location on the network. These fields are compared to the result of a configuration read primitive command (02) issued when the device is first brought online.

DATE : 12/17
TIME : 19:30

P L C C O N F I G U R A T I O N
PLC NUMBER 124

NETWORK STATUS :
ALL LINKS OK

PLC DEVICE DATA

PLC PARAMETERS MODIFIED : N
PLC ON LINE ? (Y/N) : >Y
CABLE NUMBER (0-8) : 3
TIWAY ADDRESS (1-254) : 17
DEVICE ID LOCATION (V) : 970
SCAN PRIORITY (0-20) : 4
DEVICE TYPE : 550
DEVICE MODEL : 16

DISCRETE INPUT DATA

NUMBER OF INPUTS : 58
INPUT TYPE ? (V/C) : V
INPUT START : 100

DISCRETE OUTPUT DATA

NUMBER OF OUTPUTS : 24
OUTPUT TYPE ? (V/C) : C
OUTPUT START : 528
COMMAND TYPE? (Y/CR) : CR
COMMAND START : 220

PROCESS INTEGER DATA

NUMBER OF INTEGERS : 0
INTEGER TYPE? (V/C) : V
INTEGER START : 0

LOOP DATA

NUMBER OF LOOPS : 16
MEMORY TYPE? (V/C) : V
OUTPUT START : 200
SET POINT START : 232
STATUS START : 264

ANALOG INPUT DATA

NUMBER OF INPUTS : 37
INPUT TYPE? (V/C) : V
VALUE START : 296
STATUS START : 323
LIMIT TABLE TYPE? (V/C) : C
ALARM LIMIT TABLE START : 500

ANALOG OUTPUT DATA

NUMBER OF OUTPUTS : 0
OUTPUT TYPE? (V/C) : V
OUTPUT START : 0

REAL ANALOG DATA

NUMBER OF REAL ANALOGS : 10
MEMORY TYPE? (V/C) : C
REAL ANALOG START : 740

TIMER COUNTER DATA

NUMBER OF TIMER/CNTRS : 5
MEMORY TYPE? (V/C) : C
TC CURRENT VALUE START : 450
TC PRESET VALUE START : 455

NOTE : ANY CHANGES TO THIS SCREEN (EXCEPT PLC DEVICE DATA) ARE EFFECTIVE
ONLY AFTER SYSTEM RECOVERY.

GRP#-ALARM GRP:

Figure 28. PLC configuration screen format.

The next configuration step is to enter the number of points, memory type, and starting location of each data type available on the controller. If the number of values parameter is zero, no memory is allocated in RTD and no block will be defined for that data type. The discrete output data type also requires the engineer to enter the starting address and memory type of the discrete output command bits. These command bits can be located sequentially in either CR memory or written directly to a Y output point. This information is used by the command queue utilities when a task formats a write primitive to change the state of an output. The analog input and timer/counter data types include parameters for locating the initialization data to be loaded into RTD when the device is brought online.

When the system is powered up and initialized, the System Control task reads the PLC configuration record for each device from the database and space is allocated in RTD for the described data. The RTD map record is filled out with the index to each data type within each controller. Before the initialization sequence, the CC task reads the PLC configuration record. This data is then used to define the various blocks for RTD retrieval. As these blocks are defined, the CC task stores the indices and data block sizes in the PLC control record for use in the RTD_PUSH utility.

After the initialization actions have completed, the CC task begins runtime polling. Runtime polling consists of cycling through each controller and sending the gather frames necessary to retrieve the RTD areas defined in that controller. Figure 29 shows a diagram of the scanning sequence through the control records for each device. Within a

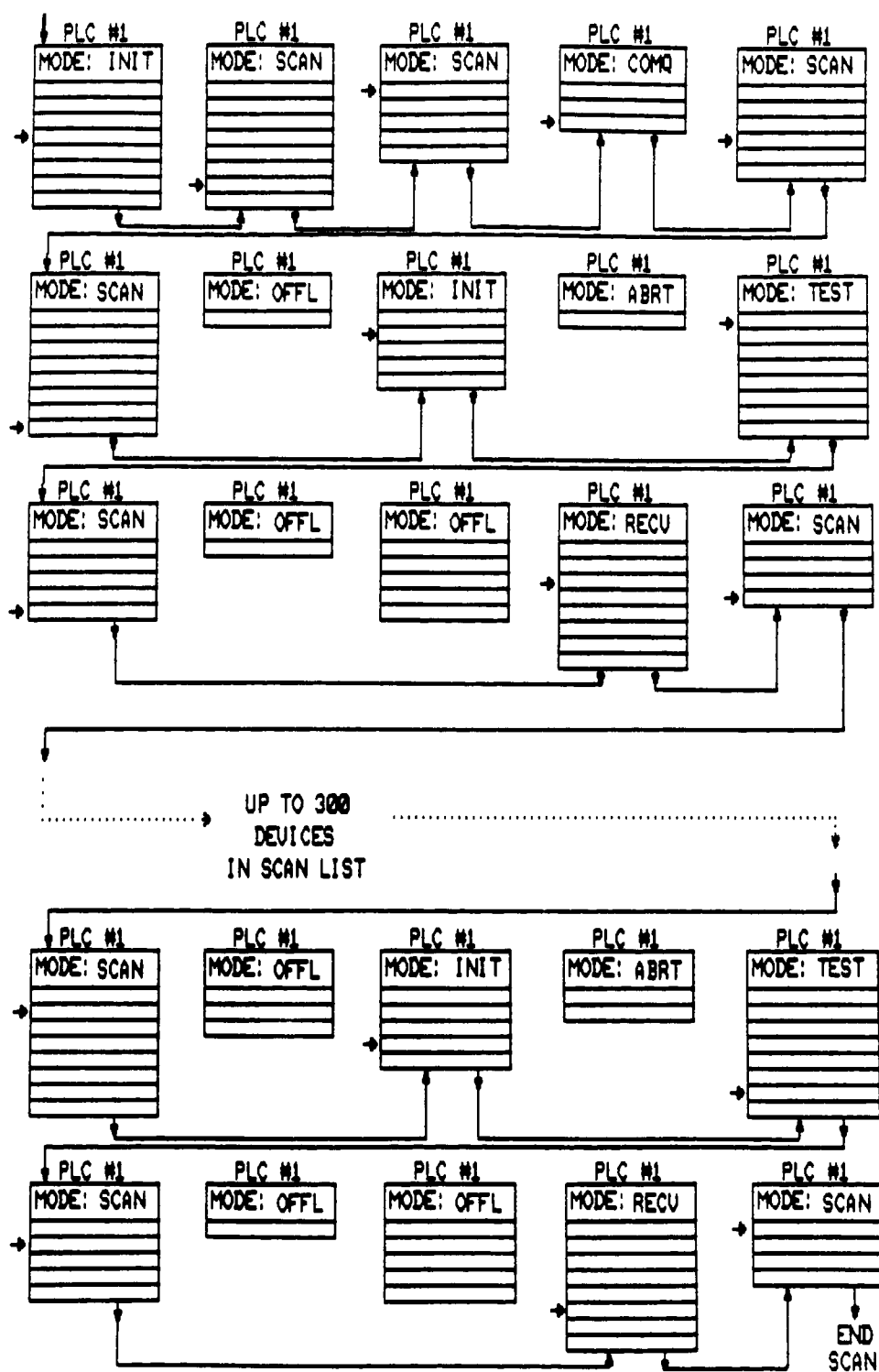


Figure 29. Runtime polling scan sequence.

particular device, the CC task sends each of the gather block commands from the first to the last command and then starts over again. The CC task uses the PLC control record to keep track of which block is presently being retrieved and the index into RTD for the return data. To execute a scan, the CC task searches through the PLC control records and the request blocks from the first online device to the last device. The PLC control record contains a pointer to the next online device to prevent having to examine each record to find out if it is online. A poll sequence begins with the first online device. If the previous command has finished, the return data is pushed into RTD. If the previous gather frame is still outstanding, CC will go on to the next online device. After a gather frame is retrieved, the next available frame in the sequence is transmitted and CC will go to the next device until all the devices have been serviced.

Several other types of activity occur during runtime scanning in addition to RTD polling. Other activities include sending and receiving the initialization frames when a device is first brought online, handling command queue frames from other tasks, sending self test sequences, and performing error recovery activity. The present activity on a device is determined by the mode parameter in the PLC control record. Figure 30 shows a state diagram of the various modes and the transition paths between them. When a device is brought online, the mode selected is recover. When all the initialization steps have completed successfully, the mode transitions to scan mode. If a command queue request is made during scanning, the mode will change to command queue mode when the present frames returns. After the command queue

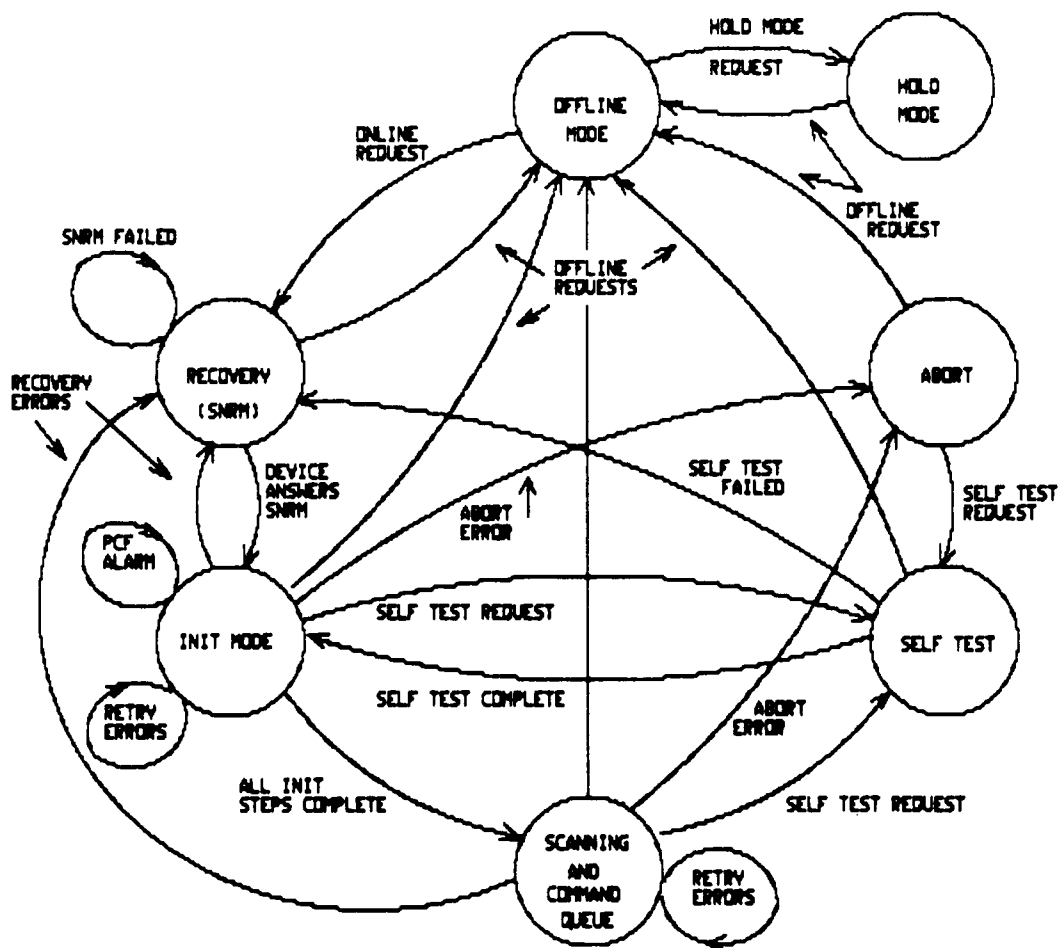


Figure 30. Device mode state diagram.

transaction completes, the mode will return to scan mode. If an error occurs during a transmission, the recovery mode is selected. In the recovery mode, the device is sent a SNRM frame in an attempt to clear the problem. If this completes successfully, the mode then goes to init mode then back to scan mode. The self test mode is requested from the service queue. In this mode, the device self diagnostic frame is transmitted. After completion, the mode goes to init mode and then goes back to scan mode. The hold mode is used for enabling command queue service while not performing RTD polling. This is used for uploading and downloading ladder logic programs while a PLC is in program mode. The hold mode can only be entered from the offline mode.

The CC task implements a flow control algorithm to determine when each device is granted service during a scan through the list. This flow control algorithm is needed to control how many times each device is scanned relative to the other devices on the network. A controller with only one gather frame will be serviced much faster than a device with five gather frames. To prevent this, the devices with less data are not serviced every pass through the scan list but are delayed a certain number of scans before being serviced. This flow control algorithm will be described in detail in a later section.

The CC task performs not only runtime polling but also services the command queue and service queue. After a number of passes are made through the scan list, the CC task checks on the command queue and service queue for entries. The amount of time devoted to RTD polling, checking on the command queue, and checking on the service queue is configurable.

During runtime polling, several statistical parameters are calculated and stored for each device. This service is enabled and disabled through the service queue for each device. The stored parameters include the number of I-frames transmitted and received, the number of complete scan cycles, the minimum scan time, the maximum scan time, the average scan time, the number of HDLC errors, the number of retry cycles, and so on. This data is used by the system engineer to determine the performance of each device on the network.

Command Queue Message Handler

The command queue message handler dequeues the command queue primitive frames from the application tasks and prepares them for transmission. This activity is run between RTD poll cycles as described previously. The actual transmission, reception, and enqueueing of the reply frame is performed by the polling handler during subsequent poll cycles.

The first action in the command queue handler is to check the command queue for frames. If no frame exists, the CC task will call the service queue handler. If a frame is present, the command queue handler will dequeue it. The PLC control record for the device will be checked for the proper mode. A command queue frame can only be sent if a device is in scan mode or hold mode. If it is not in the correct mode, a message is enqueued back to the task to indicate the problem. If the device is in the correct mode, the frame is then put into a transmit buffer in the dual-port memory holding area. The desired PLC control record is marked for command queue mode and the reply queue number is

stored along with the address of the transmit holding buffer where the command queue frame was loaded. Finally, the command queue handler will check the queue for other frames and then proceed to the service queue handler.

The next time the RTD polling handler performs a scan through the PLC control records, the PLC with a command queue frame ready will be in command queue mode. After the previous gather frame has been put into RTD, the command queue frame will be sent out. This is done by changing the transmit and receive buffer address pointers in the request block to point to the command queue holding buffer. The new request present flag is set in the request block and CC will then go on to the next online device.

During the next scan cycle, the request complete flag on that device will be checked. If the command queue reply frame has returned, the command queue receive buffer in the holding area will be deformatted and checked for errors. If no error exists, the frame will be enqueued back to the application task in the queue number stored in the PLC control record. If an error is found, an error code will be enqueued to inform the application task of the failure.

Service Queue Request Handler

The service queue request handler is designed to dequeue requests from the service queue to perform actions on the controllers. These actions are associated with changing the operational mode of a device or performing service actions on a device. The actions include taking a controller online and offline, invoking self diagnostics on a

particular controller, running self diagnostics on all devices on the network, enabling the statistics logger, disabling statistics, and resetting the statistics values on one or all devices.

Similar to the command queue, the first action of the service queue handler is to check if a service request is present in the queue. If no message is present, the CC task will leave the routine and call the RTD polling procedure. If a request is present, it is dequeued and serviced. The nature of the request is determined and the appropriate handler is called.

The offline request handler is designed to terminate RTD polling and command queue activity on a device. To remove a device from scanning, the chain index of the previous online device is pointed to the next online device. This will cause the RTD polling routine to skip this device on the next scan cycle. The device mode in the PLC control record will be marked for offline mode so command queue frames will be rejected by the command queue handler. To take the offline device into account, the flow control algorithm must recalculate the scan sequence. Finally, a message is enqueued back to the calling task indicating the action has completed.

The online request handler performs the opposite action of the offline request handler. First, the PLC configuration record is read from the database to determine if the port number, network address, device ID location, or device priority has changed. The device is placed back into service by finding the last online controller before the desired PLC and connecting the chain index to the present device. Next, the new PLC chain index is pointed to the next online device.

The next RTD scan execution will now service this device. Device mode is then marked for init mode. The first action when a controller is brought online is to redefine the blocks and to retrieve the initialization data into RTD. To determine when the device will be serviced relative to the other devices, the service count from the flow control algorithm must be calculated. The final action is to enqueue a message back to the application task indicating the action has completed.

When a self diagnostics request is received in the service queue, the service queue handler will see if the device is online and in either the init, scan, or command queue mode. If the correct mode is found, the device is marked for self test mode. When the RTD polling handler runs next, the marked device will be issued the self test command sequence after the previous outstanding I-frame has been moved into RTD . On successive poll cycles, the response frames will be checked for correctness and the next frame of the sequence issued. When the final frame completes, a response message will be enqueued back to the application task indicating success. If a failure occurs at any step in the self diagnostic sequence, a failure message will be enqueued back to the task and the error recovery process will be invoked.

The self test sequence involves four steps. First, the controller must be issued a change state I-frame requesting remote mode which stops execution of the ladder logic program. Next, the self diagnostics frame is sent. After the self test command is sent, the CC task will delay a configurable amount of time to allow the device to complete

execution of the self test. Finally, another change state I-frame is sent requesting run mode to restart execution of the ladder logic program. The device is then marked for init mode to send the define block command and gather the initial data.

The broadcast self test request allows the operator to issue the self test sequence to every online device on the network simultaneously. This service queue command can be initiated by the system engineer or can be scheduled to occur on a regular time base. The broadcast self test initiates the same self test sequence as the single unit self test. When the command is received for a broadcast self test, the service queue handler scans through the device control records and marks all the online device for self test mode. The next time the RTD polling handler runs, the self test sequence is sent to these devices after the previous frame has been retrieved and processed into RTD. As the devices complete the sequence, a count is kept and compared against the number of online devices. When all devices have finished, a reply queue message is formatted and sent to the application task.

The statistics on, off, and reset commands give the system engineer control of the online statistics and frame logging functions. When the statistics are enabled on a particular device, the RTD polling handler will gather performance data on the device. The engineer also has the option of enabling the frame logger. This function stores the first 10 bytes of each transmitted and received frame for the last 20 transactions. The frame logger can only be run on one device at a time because only one storage buffer exists. This allows the engineer to

track the communications activity on a device. The reset command clears the statistical information and frame storage buffer. Statistics commands are executable on a single device by sending the PLC number or on all devices by setting the PLC number to zero.

Communication Flow Control Algorithm

The communication flow control and tuning algorithm is designed to give the system engineer the ability to tune the rate of data flow to and from individual network devices consistent with the amount of data to be retrieved from these controllers. Data amounts can vary between the small data blocks from the intelligent tank transmitters to large multiple blocks of data required to scan a PLC. Critical data can also be retrieved at a higher rate by raising the priority of a device. Network congestion problems arise when the numerous requests for data from the ITTs degrade the scan rate for data from the large PLC devices. The communications control task, HDLC board, and network interconnections can be viewed as a fixed resource with a finite capacity for frame transmission and reception. The priority tuning algorithm is designed to control the utilization of this resource for RTD polling and command queue frame transmission. This algorithm can be divided into two parts: (1) the priority tuning algorithm and (2) the flow control algorithm.

The priority tuning algorithm allows the engineer to adjust the amount of communication resource devoted to each device to achieve a desired scan rate. As the priority of a particular device is increased, the lower priority devices are effectively scanned at a lower rate to

devote more communication time to the high priority device. Assuming that the network is operating at optimal speed, the only recourse for improving the communication speed of one device is to lower the scan rate of a lower priority device.

The priority tuning algorithm operates by assigning a priority level to each device on the network and using this to allocate scan time. Priority is entered on the PLC configuration screen while the device is offline. Higher priority levels devote more communication resources to a device. The algorithm first determines the highest priority device on the network. This device will be granted service on every pass through the scan list as explained before. The priority level of the highest priority device is used as an offset to reduce the scanning of the lower priority devices. For example, a device of priority one less than the highest priority level will be scanned on every other pass through the scan list. This means this device is running at one half the communication rate of the highest device. The next lower priority level will be scanned every third pass through the scan list effectively communicating at one third the rate of the highest device. Since there are 20 priority levels not including zero, a device at priority level 21 would run twenty-one times faster than an equivalent device at priority level zero.

The scan tuning priority is used to control the scan rate during RTD polling for devices of different data requirements. Consider two devices with different amounts of data for RTD. Device number one requires only one frame to perform a complete retrieval of data. Device number two requires five frames to retrieve all the data

available. To scan these devices at the same rate, device number two would be assigned a priority of four while device number one would be left at priority zero. This would cause device number two to be scanned five times for every scan of device number one resulting in an equivalent scan rate for all data from each device.

The actual mechanism for tracking the tuning algorithm requires the storage of two variables for each device on the network: (1) the service count and (2) the reset count. First, the tuning algorithm calculates the reset count by taking the device priority and subtracting the highest priority level resulting in a negative reset count as shown:

$$\text{Reset_count} = \text{device_priority} - \text{highest_priority.}$$

This reset count is loaded into the service count and incremented on every pass through the scan list. When the service count reaches zero, the device is granted service and the service count is loaded with the reset count again. In the above example, the highest priority level was four and the lowest zero. The reset count for the highest device is zero and the reset count for the lowest device is -4. This will cause the highest device to be serviced every scan and the lowest priority device to be scanned every fifth pass.

The flow control algorithm results from the need to evenly distribute the communication load across all polling cycles. If in the above example there were 25 devices at priority zero and the one device at priority four, the poll cycles would consist of 4 scans on

the highest priority device followed by one scan of all devices. This would result in a sporadic data flow to the highest priority device since it would have to wait for 25 devices to be scanned in rapid succession before it was serviced again. The flow control algorithm is designed to evenly distribute the communication load across all scan cycles. In the example, if the low priority devices were to be scanned in groups of 5 devices per cycle, this would result in a consistent communication rate for every device on the network.

The flow control algorithm is implemented by servicing only a portion of the delayed devices on each scan cycle. This is accomplished by starting up the scanning of these devices by a variable offset count which distributes the scanning evenly. Thereafter, the devices will reschedule consistently in a distributed manner. This distribution is achieved by creating an offset counter for each reset count level and using the offset counter to determine the startup service count of the devices. In the above example, the lowest priority devices had a reset counter value of -4. When the first device at priority zero is encountered, the service count is set to zero and the offset counter decrements to -1. The next device at priority zero will receive an initial service count of -1 and the offset counter will decrement to -2. Thereafter, the service count for priority zero devices (reset counts of -4) will be initialized to -2, -3, -4, 0, -1, -2, repeating for the number of devices at that level. Similarly, the offset counters are kept for every reset count level utilized. This has the effect of invoking service immediately for the first device, delaying service for one cycle on the second device, delaying service for two cycles on the

second device, and so on until all devices are allocated the initial service count. After scanning has begun, the devices will maintain this distribution on each priority level by rescheduling at their respective reset counts.

The startup cadence established by the priority offset counters, reset counters, and service counters produce the distributed communication flow. Figure 31 shows a chart of 13 devices as an example of the flow control and scan tuning algorithm. Similar to the above example, device number one is at priority four. Devices two and three are at priority three and the remaining 10 devices are at priority zero. The example illustrates the distributed scanning of the various devices caused by the startup service count offset. This has the effect of producing an even four devices scanned per cycle while maintaining the overall priority structure.

The number of devices serviced per cycle will only be evenly distributed if the number of devices at each priority level is an even multiple of the number of cycles per scan at that priority. For instance, if three device were at priority three in the above example, every other scan would service one extra device resulting in a scan pattern of 5, 4, 5, 4, and so on. Since each priority level can only add or subtract one device from the total number serviced per scan, the maximum deviation in the number of devices serviced per cycle is equal to the number of unbalanced priority levels.

The flow control algorithm must be adaptive to compensate for command queue frames, device going online and offline, and other perturbations to the normal scanning cadence. Once the flow control

DEVICE PRIOR	SCAN CYCLE NUMBER																			
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1/4	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
2/3	X		X		X		X		X		X		X		X		X		X	
3/3		X		X		X		X		X		X		X		X		X		X
4/0	X					X					X					X				
5/0		X					X					X					X			
6/0			X					X					X					X		
7/0				X					X					X					X	
8/0					X					X					X					X
9/0	X					X					X					X				
10/0		X					X					X					X			
11/0			X					X					X					X		
12/0				X					X					X					X	
13/0					X					X					X					X
SCAN TOTAL	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4

Figure 31. Communication flow control algorithm example.

cadence has been established at startup time, any deviations in the number of devices due for service during a particular interval will cause the scanning cadence to become unbalanced. The scanning cadence is maintained by adjusting the individual device service counts when a disturbance occurs. Several factors which disturb the cadence are: (1) devices coming online, (2) devices going offline, (3) outstanding frames when a device is due for service again, (4) device transitions to other than scanning mode. A special case of devices going online and offline exists for the highest priority device.

If a device is put online after the scanning cadence has been set, it must be scheduled to start scanning in cadence with the other devices at that same priority. This requires information about the number of the scan just conducted, the priority of the new device, the maximum priority level, and the number of devices already online at that priority. The first action in the service queue handler is to determine the priority of the new device by reading the PLC configuration record. Next, the reset count is calculated based on the maximum priority device. The initial service count is then calculated using the equation:

$$\text{Service_count} = \text{Offset_counter}[\text{priority}] + (\text{Scan_counter} \text{ MOD } (1 - \text{Reset_count})) ;$$

If Service_count > 0 then
 Service_count = Service_count + Reset_count - 1 ;

The offset counter in the equation contains the value remaining after the startup allocations and is an indicator of the number of devices

on that priority level. If the remaining value in the offset counter is zero, then the number of devices on that priority is an integer multiple of the scanning rate and that priority level is balanced. This value therefore is the scanning slot which this device would have been allocated to if it had been online at system startup time. Since many scans have occurred since then, the service count must be offset to allow for this. The $(\text{Scan_count} \bmod (1 - \text{Reset_count}))$ term creates this scan count offset. The MOD function returns the remainder of the integer division of the Scan counter and the frequency of polling for this priority. If this number is zero, then the present scan is an integer multiple of the scan rate for this device. If the service count calculated is positive, then one of the scans which just occurred is the scan which the new device should have received service. In this case, the device will be delayed until the next occurrence of this scanning slot.

A special case of bringing a device online occurs when the new device is of higher priority than the present high priority device. In this case, the only recourse is to resequence all the service counts and reset counts to the level of the new device. This is accomplished using the same procedure which determined the startup cadence. This may produce a momentary lapse in scanning for some devices which may have been due for service but might be additionally delayed when the new cadence is calculated. However, it is felt that this delay will be insignificant in the overall scan time of the device.

Taking a device offline also disturbs the scanning cadence and will invalidate the offset counter used to determine the service

count when a device is put online. For instance, if the devices taken offline all fall into the same scan slot, significantly fewer devices will be serviced during that interval resulting in an unbalanced scanning situation. This problem is averted by adjusting another device at that same priority to take the place of the device going offline. The device to adjust is the device whose service count is equivalent to the last device put online at that priority level during startup. Before the device is adjusted, the offset counter is incremented as if one less device had been put online at startup. The determination of the device to adjust requires compensation for the priority of the offline device, the scan cycle presently active, and the remaining priority offset. The equation for determining the service count of the adjust device is:

$$\text{Adjust_count} = \text{Offset_counter}[\text{priority}] + (\text{Scan_count} \bmod (1 - \text{Reset_count})) ;$$

If $\text{Adjust_count} > 0$ then
 $\text{Adjust_count} = \text{Adjust_count} + \text{Reset_count} - 1 ;$

Similar to the service count calculation for putting a device online, the adjust count is determined by modifying the offset counter by the modulus of the present scan. If the adjust count is positive, then it is made negative using the period of the scan rate for that priority. Once the adjust count is determined, a search for a device whose service count is equal to the adjust count is conducted. The search starts at the offline device and goes up until a device is found. If none is found then the search is started back at the new

device and goes down. This has the effect of adjusting the device nearest the device going offline. The service count of the adjusted device is set equal to the service count of the device going offline. This effectively replaces the scan slot of the device going offline and preserves the integrity of the offset counter.

A special case of taking a device offline occurs when the offline device is the maximum priority device. If no other maximum priority device exists at that priority level, then a new high priority device must be found. Once this device is found, all the service counts are recadenced using this new maximum priority.

The final factors which influence the scanning cadence are mode transitions and disturbances caused by late frames. When a device is in init mode, self test mode, or command queue mode, the device is serviced on every scan cycle as if it were the highest priority. During this time, the service count is maintained just as if the device were in scan mode. When the device transitions to scan mode, the service count will determine when service is granted thereafter. If the device transitions to recovery mode, the service count is still maintained and the device is serviced at that rate. Finally, if an outstanding frame has not completed when the device is due for service again, the service count will continue and this cycle will be skipped. This has the effect of halving the service rate if this occurs consistently.

The scan tuning and flow control algorithm provide a means to control the rate at which devices are scanned relative to the amount of data and the rate at which data updates are desired. The service

count is used to determine the times when frames are transmitted from the host to the secondary stations. Thereafter, the device is checked on every scan for the outstanding frame to complete and be processed. The next frame is sent when the service count again reaches zero. Command queue frames are transmitted immediately without waiting on the service count due to the urgency of the request from an application task. These operator commanded frames represent only a small statistical portion of the total communication load and will not affect the overall scan rate of the devices.

Error Handling

The Communication Control task must be able to identify error conditions which occur during communication transactions, classify the error according to appropriate response, and attempt to recover from the condition and continue normal operation. If the error persists, the CC task must notify the Alarm Handler to post an alarm on the controller and inform the operator to take appropriate action to repair the device. The communication errors can occur in several different places on the network and in the devices. Various errors can occur locally in the HDLC board such as transmitter underrun, receiver overflow, USART failure, local bus errors, and HDLC board memory parity errors. Communication errors are also possible while the frame is transmitted across the network physical media. The most common source of network problem is induced noise. This usually results in a CRC error, frame reject, or frame abort when a bit value is changed. Frame timing can also be a source of problems when a device answers

the network request too soon or too late. If a device experiences a hardware failure, it may transmit on the channel at the same time the host tries to transmit resulting in a collision. This should not be a continuous problem since most network devices are equipped with a jabber stop circuit which cuts the network connection when a device transmits abnormally. The network transmission problems are identified by the HDLC board on received frames and by the NIM or ITT on transmitted frames. When the error is found, an error code is reported to the CC task for appropriate action.

The controllers on the network can produce communication errors internally. Since the NIM translates the network frames for the controller, communication errors can result between the NIM and controller. Errors such as link failure, unrecognized request frames, and unsupported memory types are possible. These problems are reported back to the host as an exception primitive. When the exception occurs, an error code is sent identifying the problem so the host can take corrective action. The CC task will also inform the Alarm Handler task to post a communication alarm on the device.

Another type of network failure occurs when a controller fails internally and ceases running the ladder logic program. With every response I-frame, the controller status is included to report the health of the controller. When this status indicates that the controller is no longer operating, the CC task will stop polling for data, post a programmable controller failed (PCF) alarm to report the condition, and implement recovery action waiting for the device to return to normal operation.

Configuration problems will also cause the error handler to be invoked. This can result from improper configuration of the network address in the device or entering the wrong network address or cable number on the PLC configuration screen. To verify that the device answering a particular network address is the desired unit, the CC task reads the unique device identifier and checks for correctness. If the result indicates the wrong device answered, the CC task will post an invalid identification (INI) alarm and cease communication to the device until the problem is corrected. A read of the device configuration is also performed and the controller model is checked against the model indicated in the PLC configuration record.

The CC task also tracks the error performance of each of the eight channels. If any channel shows an excessive number of network errors and all the devices on that channel are in communication alarm state, the CC task will signal the alarm handler to post a link failure alarm. This may indicate that a common problem on the channel is the source of the problem. When any device on that channel resumes normal operation, the link alarm is cleared.

The error handler in the CC task is designed to check for the various error conditions and identify the desired action. When a response I-frame is received, the error handler is called to check for errors. First, the receive buffer error code location is checked followed by a check primitive location for the exception primitive. Next, the device status in the reply frame is checked to see if the controller is still operating normally. Finally, the length and gather block mask are checked for correctness.

When an error is discovered, the CC task will choose one of five actions in response to the condition: (1) retry the frame, (2) reset the device by sending a disconnect and SNRM frame, (3) abort further communications, (4) select init mode, or (5) take the TFA system down for maintenance. In each case except the retry frame action, the Alarm Handler will be notified to post a communication, PCF, or INI alarm.

The retry action is selected on the errors which are transient and might cease with subsequent transmissions. These errors include CRC, frame reject, and timeout errors. The retry simply sends the same frame again and keeps a retry counter. If the frame continues to fail until the retry counter exceeds a maximum number of retries, the reset mode is selected and the communication alarm is posted.

The reset action is selected by changing the device mode to recovery mode. In the recovery mode, the device is first sent a disconnect frame if it is still online. This will cause the NIM to reset and perform self diagnostics. Next, the single station reset is requested in the request block and a SNRM frame is sent to the device. If the device responds correctly and comes online, then the init mode is selected to perform the initial tests, set up the define blocks, and scan for the range/limit data again. If the device does not return online, the SNRM frame is retransmitted periodically.

Some communication error conditions are fatal to further communication activity and cannot be recovered from. The abort condition is requested on these conditions. This action will post the communication alarm and mark the device off the scanning list. The

operator has to take the device offline, repair the fault, and put the device back online to bring the device back to normal service. An invalid ID check also aborts further communication since the controller answering may not be the correct device.

If the controller status in the response frame indicates that the device is no longer operating, the PCF alarm will be posted and the device will go into init mode. The first step of the init mode is a status read command which checks for device operation mode. This frame will be sent periodically until the device returns to an operating state. The remaining init frames will then be sent and the device will return to scan mode and resume normal operation.

The final error condition action is to take the TFA system down for maintenance and execute the system diagnostics. This action is only requested if the HDLC board experiences a fatal error condition or hardware failure. After every transaction through the dual-port memory, the CC task reads a mailbox for indications of board failure such as a memory parity failure or local bus error. The CC task also checks for HDLC board lock-up by tracking the number of complete responses per pass through the request blocks. If no requests complete for a specified number of scan cycles, then the assumption is made that the board has ceased to operate. If this occurs, CC task makes a request for system control to shut down the system for diagnostics and maintenance.

The error handling routines provide a method to identify a communication problem and take appropriate action to correct the failure. The operator and engineer are informed of the problem with

a variety of alarms and diagnostic codes. This will help the operator identify corrective action in the case of abort errors or recurring failure conditions. A typical action is to request the self diagnostic sequence on the device. Finally, recovery actions attempt to return the device to normal operation if possible and clear the alarm condition.

CHAPTER IV

HOST COMMUNICATIONS TASK DESIGN

Executive Module

The Communications Control task consists of functional modules of software which, when grouped together, perform the actions necessary to control the network communications and real time data scanning. These modules are organized in levels with the higher level modules calling the lower level modules to perform individual actions. The activity of all the software modules is controlled by the top level module called the executive.

The Executive module organizes the four primary activities of the CC task: (1) initialization of the network and data structures, (2) real time data polling, (3) command queue request frames, and (4) service queue request handling. Figure 32 shows the hierarchy of the modules called by the Executive. The Executive calls each module in turn to execute that area of functionality. To start operation of the task and network, the Initialization module is called once when the system is powered up. The RTD Polling module, Command Queue Request Handler, and Service Queue Request Handler are called repeatedly thereafter to perform those functions.

After initialization has been performed, the primary purpose of the Executive module is to control the amount of higher level resource devoted to each of the three communication activities. Sharing of resource between these functions is governed by tuning parameters which are configured by the system engineer and read from a database

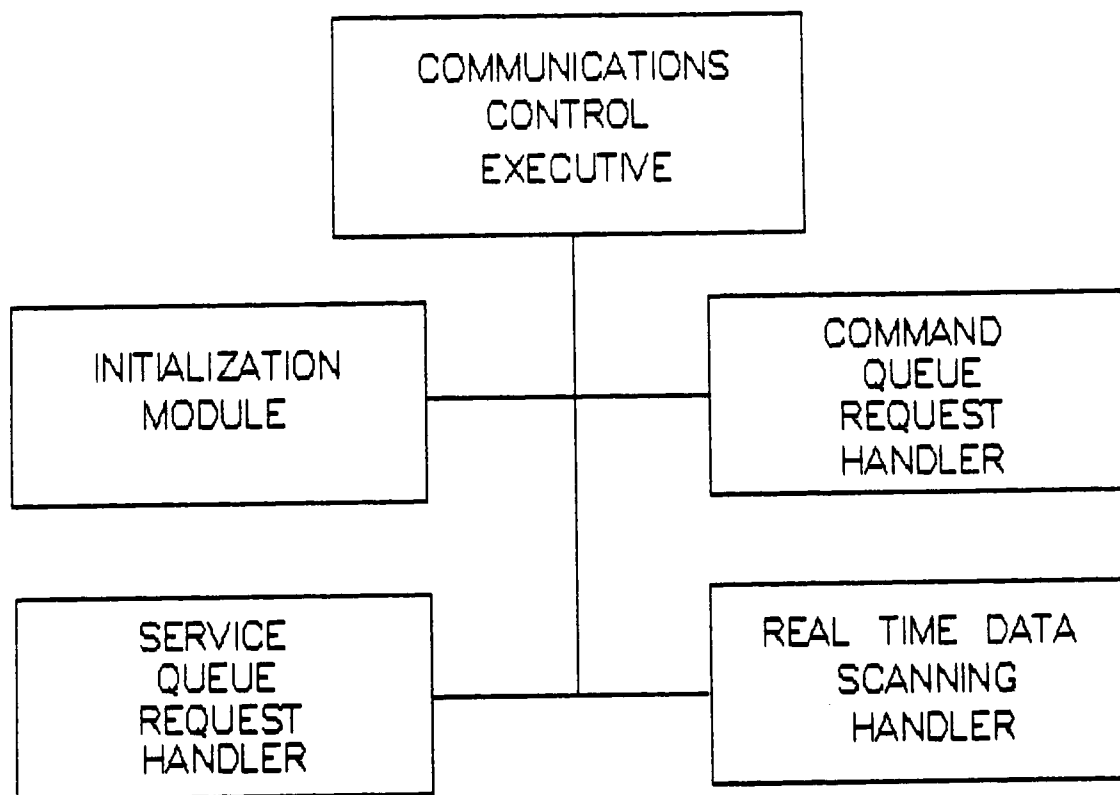


Figure 32. Executive module hierarchy.

configuration record during initialization. Figure 33 shows a flow chart of the Executive software program. The first action invoked when System Control starts the CC task is the initialization process. After this completes, the Executive program goes into a loop which executes the three communication activities. The Executive will execute this loop continuously until it detects that the system has left the run state and is preparing to shut down for maintenance or fault diagnostics. When this happens, the CC task will abort operation by suspending on a trap instruction.

Within the main loop, the RTD Polling Handler, Command Queue Request Handler, and Service Queue Request Handler are executed within individual loops whose duration is governed by the configuration parameters from the database. Each activity will be invoked the selected number of times and then the next activity will be given a chance to activate. If, for example, the engineer has chosen tuning parameters of three RTD polling cycles followed by one cycle of command queue handling and one cycle of service queue handling, the CC task would be devoting three times the resource for RTD polling as for each of the other activities. If the command queue or service queue is empty, the Executive will bypass that activity even if more than one cycle has been configured since no request is present. Any time the RTD Polling Handler is called, a scan through the PLC request blocks is made for possible activity on the network. The final step in the main loop is a delay module which can be configured with any amount of delay from one to 32 seconds. This is used to slow down the CC task and devote system resources to other system activities.

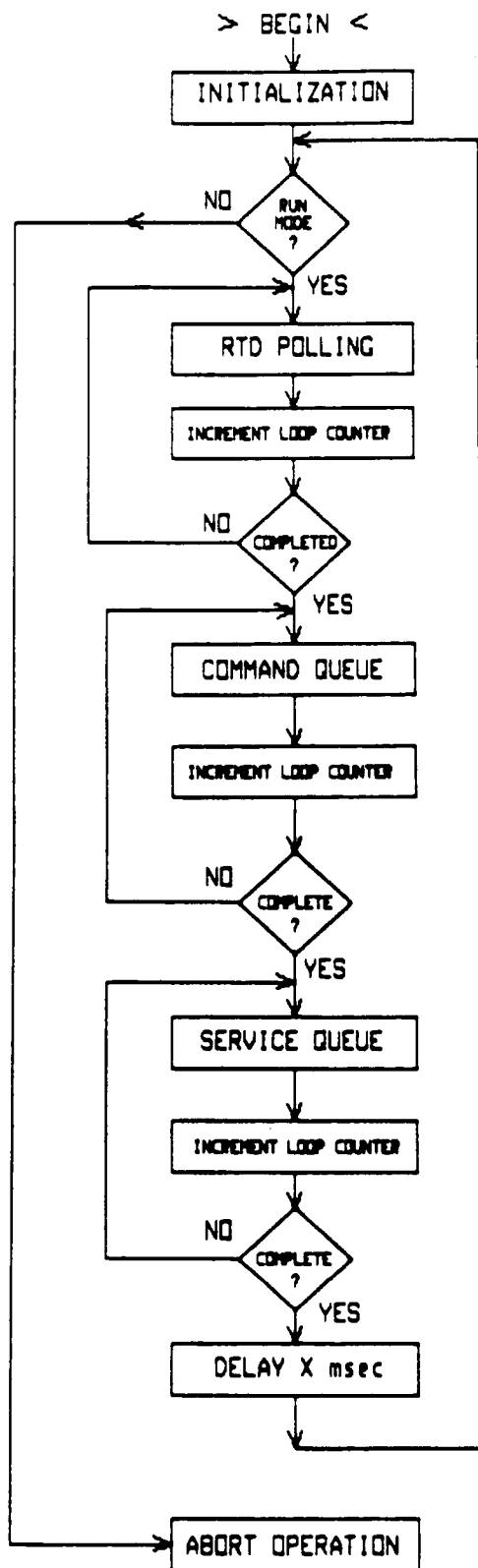


Figure 33. Executive program flow diagram.

Initialization Module

The Initialization module is designed to start operation of the communication network by calling lower level modules to perform the initialization actions. These actions include building and initializing the control data structures, allocating the real time data storage area, initializing the HDLC board, bringing the controllers online, performing the initial device configuration tests, downloading the RTD block definitions to each device, scanning for the range limit data, and setting up the flow control cadence. After these actions have completed, program control is returned to the Executive where runtime communications servicing begins.

The Initialization module calls three modules to perform the initialization activities: (1) the RTD Scanning Initializer, (2) the Network Initializer, and (3) the Scanning Cadence Initializer. These modules in turn call other modules to set up the scanning communication frames, initialize the HDLC board, and perform the initial communications for bringing the controllers online. Figure 34 shows the hierarchy of modules called by the Initialization subprogram. In addition to the task modules, global utilities are called to perform enqueues, dequeues, and get records from the database. These global utilities exist once in the system and are callable by any task in the TFA package.

The Initialization module is designed to call each of the functional modules in sequence and check for errors at each step. If an error occurs during initialization, a failure message is sent to the System Control task and system operation is aborted for diagnostics

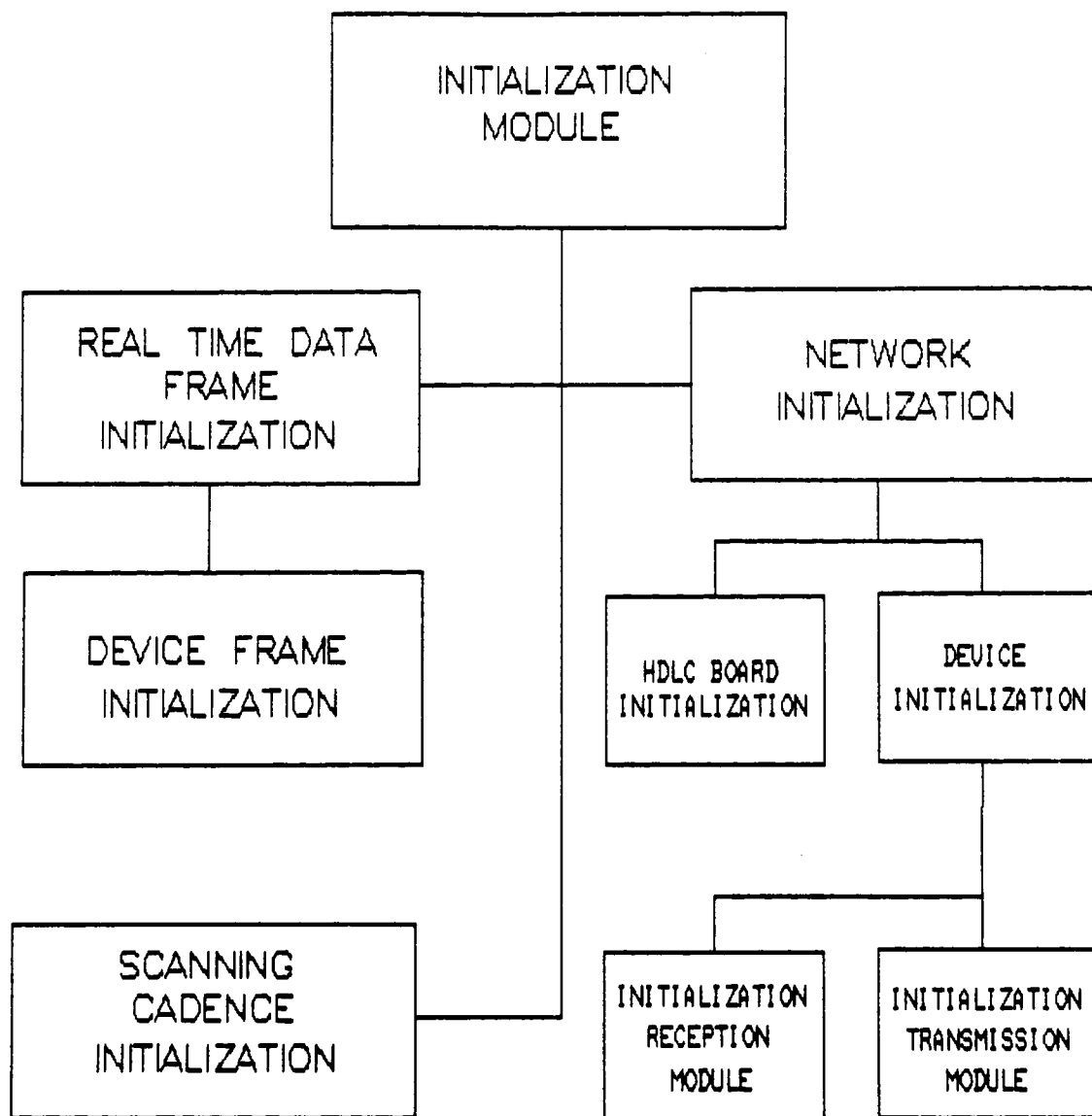


Figure 34. Initialization module hierarchy.

and repair of the condition. This action is necessary because the primary function of the TFA system involves retrieval and display of the controller data. If the communications network will not function correctly, the system will not be operable.

The program flow of the Initialization module is shown in Figure 35. First, initialization is performed on the queues and semaphores used by the CC task to communicate to the other system tasks. Next, the Initialization module waits on a queue message from System Control to signal the task to begin initial activity. When the system is powered up, the HDLC board automatically performs internal self diagnostics and reports the health of the board in a mailbox location. The next action of the Initialization module is to read the self diagnostic result from this mailbox and report any failure to System Control task.

The Initialization module next calls the RTD Initialization module to set up the frames necessary for defining and gathering the live data from the controllers. Before the CC task is released to initialize the local data structures and the network, the RTD area is allocated by the System Control task. This allocation involves building a map into the RTD area for each controller and each data type within the controllers based on the PLC configuration records in the database. The RTD Initialization module reads the map records to build up the communication frames for the controllers. If the database records are damaged and the RTD area could not be allocated, the RTD Initialization module will stop processing and report an error. Again, any error found and reported to System Control will suspend operation and invoke system diagnostics.

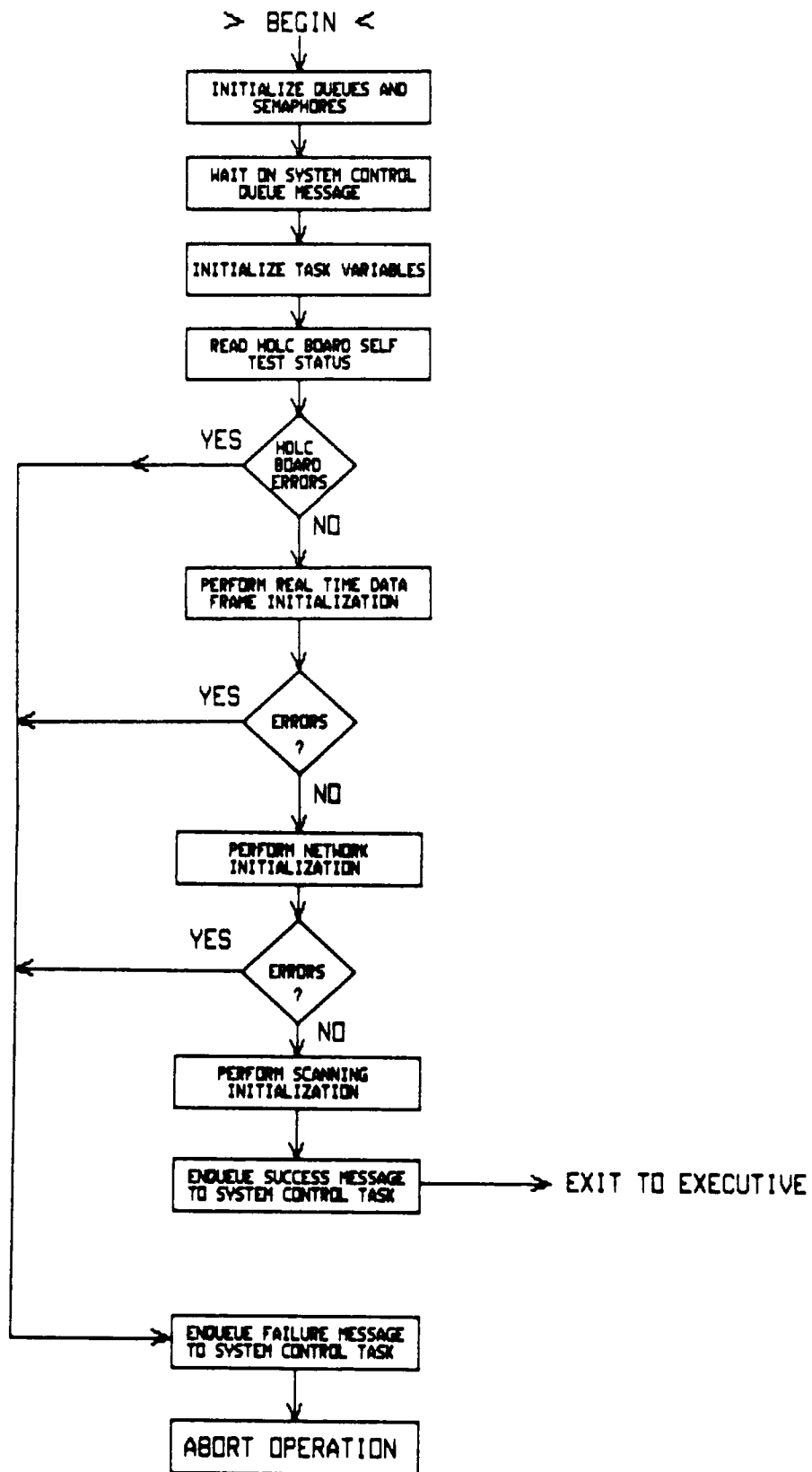


Figure 35. Initialization module flow diagram.

The next action of the Initialization module is to call the Network Initialization module. This module starts up the HDLC board and brings all the controllers online by sending the initial group of I-frames to each device. If any problems are encountered with the HDLC board during this process, the failure notification is enqueued to System Control and operation suspended. After the network is initialized successfully, the Scanning Initializer module is called to set up the scanning cadence for the flow control algorithm. After all these steps have completed, a success message is enqueued to System Control and the CC task will wait until the system enters run mode to begin operation.

The RTD Scanning Initialization module is designed to build up the I-frames necessary to perform the RTD definition and scanning for each configured device on the network. Figure 36 shows the flow diagram of the RTD Initialization module. The first action of this module is to read the CC task configuration record from this database. These parameters are used to tune the resource sharing discussed in the Executive and identify the number of configured devices on the network. The next action is to set up and initialize the communication frames common to all devices such as the status read frame and configuration read frame. Also, the command queue holding areas are allocated and the addresses stored. Next, the RTD Initialization module loops through all the configured devices and performs the initialization activities unique to each device. First, the PLC configuration data is read and used to allocate the request block. Next, the Frame Initialization routine is called. In this module,

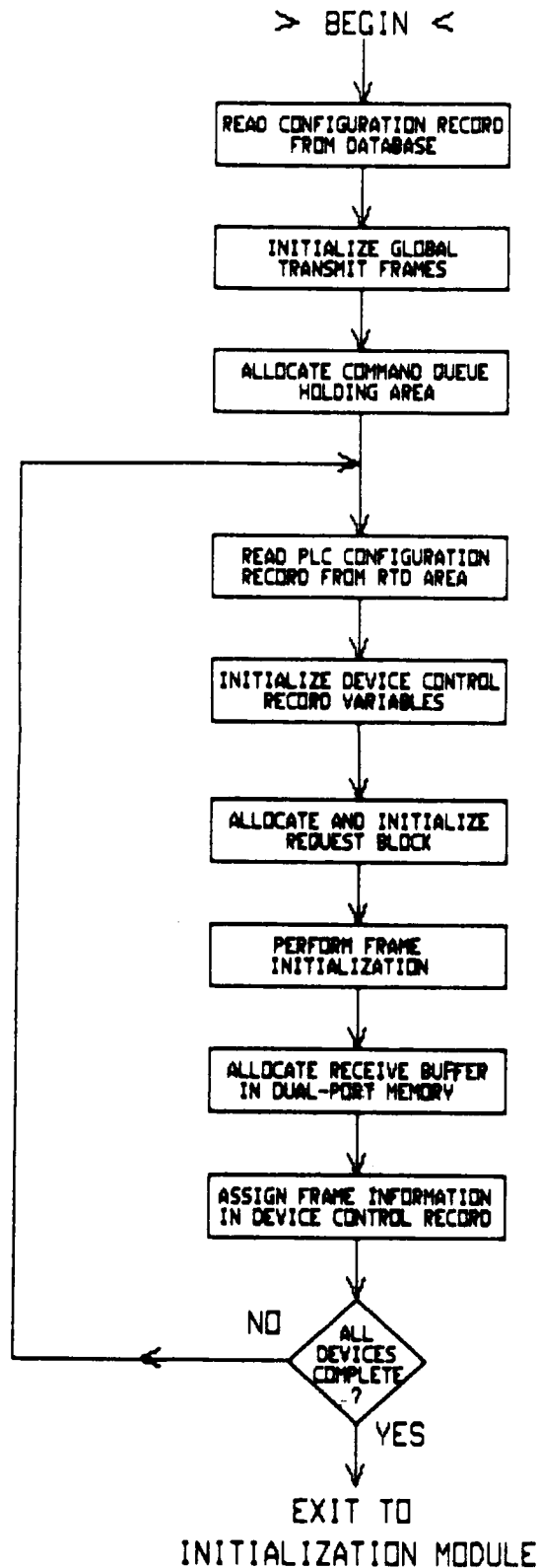


Figure 36. Real Time Data Initialization module flow diagram.

the transmitted I-frames are built up based on the data types and locations described in the record. Along with the gather frame, the initial define frame is built up which sets up the blocks in the controller which define the RTD acquisition areas. This module implements a compaction algorithm which optimizes the amount of information retrieved in each block. After the Frame Initialization module completes, the next action is to allocate and size the receive buffer based on the largest requested return data frame for RTD. Finally, the PLC control record for the device is filled out containing information on the size and location for the data to be installed into the RTD area. When all the devices have completed, control returns to the Initialization module.

The Network Initialization module is designed to start up operation of the HDLC board and initialize all the online devices attached to the network. Figure 37 shows the program flow for this module. The first action is to call the HDLC Board Initialization routine. This module reads the configured baud rates for each of the eight network channels and writes the channel baud parameters to the mailboxes in dual-port memory. If this action fails, then operation is suspended and the system will be taken down for repair by the System Control task as explained previously. The next action is to start up any devices defined as being online in the configuration record. If no devices are online, the initialization actions are bypassed. If devices are available, the first action is to transmit a SNRM frame to logically connect them to the network. This is done by setting the reset request and block interrupt flags in the individual

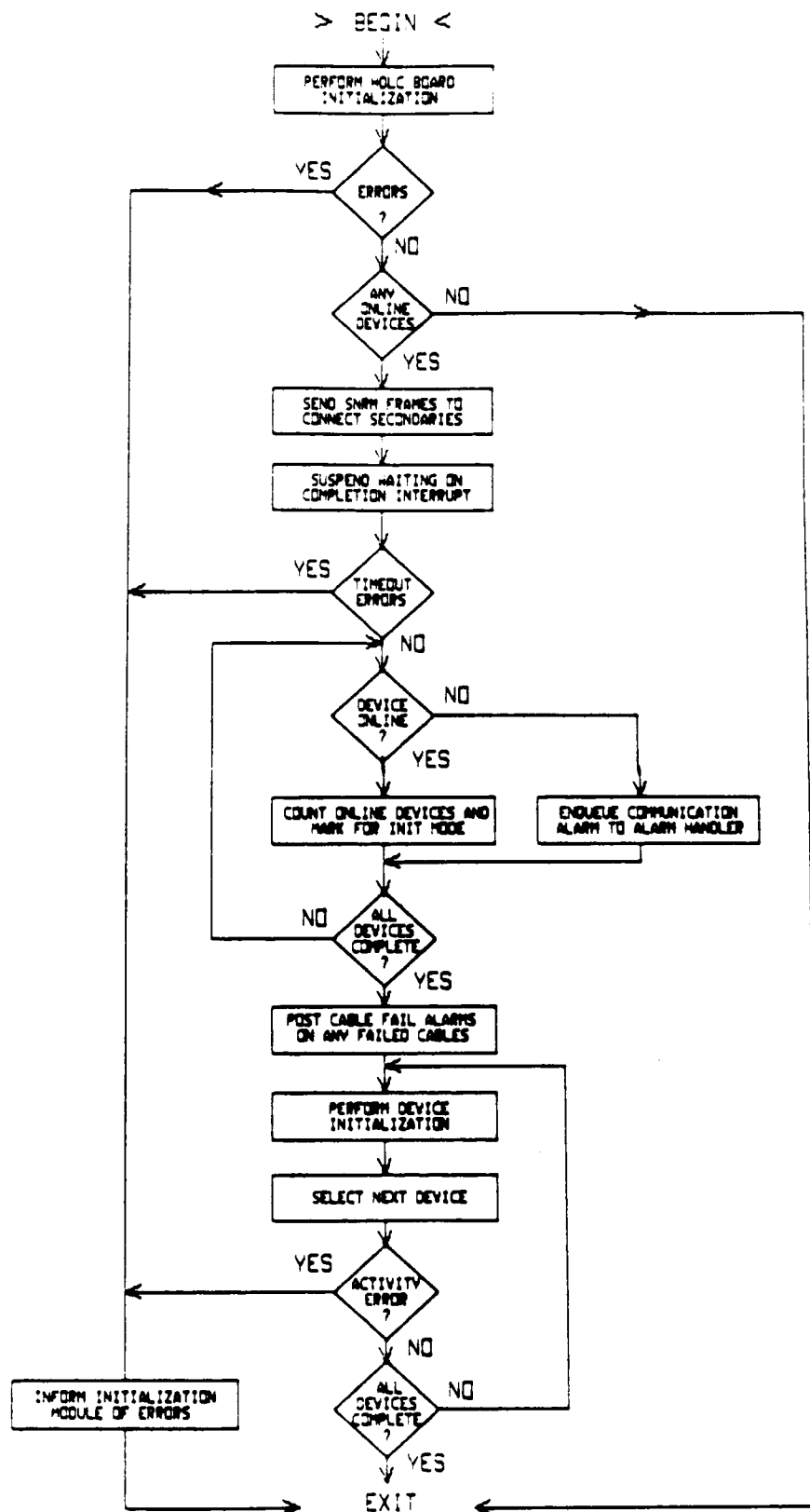


Figure 37. Network Initialization module flow diagram.

request blocks and then suspending operation to wait on the completion interrupt. If a configurable timeout expires before completion, the system will again be shut down. When complete, the devices which did not respond will have communication alarms posted. If any channel has online devices defined and no devices reported back, the channel is assumed to have failed and a channel fail alarm is posted by the Alarm Handler task.

The final network initialization activity is to send the initial frames to each online device. This is done by calling the Device Initialization module in a loop until all the devices have completed. As each frame is sent in the Device Initialization module, the response is checked and any errors result in alarm activity to post the communication, PCF, or INI alarms as appropriate. The sequence of initialization frames sent is: (1) a status read frame to check if the PLC is in run mode, (2) a configuration frame to test for the correct device model type, (3) a device identifier read to test for the correct device, (4) the define frame to set up the RTD definition, and (5) a series of gather block frames to retrieve the range/limit data for analog inputs and loops. While these frames are being sent, an activity counter tracks the number of devices completing during each pass through the scan list. If the HDLC board fails to service any devices for several loops through the list, operation is suspended on the assumption that the HDLC board has failed. Any occurrence of HDLC board failure will cause the system to be shut down.

The final module called by the Initialization module is the Scanning Initializer. This module sets up the initial scanning cadence

which evens the flow of communication frames on the network during a given scan. Figure 38 shows the program flow for this module. The first action is to zero the priority offset and device counters. Next, the highest priority device is found and used to set the scanning structure. This priority level is then used to calculate the reset counts for each device. The reset counts determine the rate at which a device is serviced during RTD polling. Finally the initial service counts are set up for each device. This causes devices at each priority level to be delayed and evenly spread out. As the service counts are calculated, the priority offset counter and device counters are updated for each priority level. When this routine completes, control returns to the Initialization module where the success reply is enqueued back to System Control and CC waits on the run mode.

Real Time Data Scanning Handler

The Real Time Data Scanning module performs all the communication activities after initialization completes. This includes both RTD scanning as well as command queue frames and service queue requests for self diagnostic frames. This module controls the transmission, reception, error checking, and distribution of all communications transactions on the network. The command queue frames are handled under this module because these frames are interjected into the normal flow of RTD scanning frames. Therefore, it is necessary to retrieve all outstanding RTD scanning frames before issuing a command queue frame. After the command queue frame completes, RTD scanning resumes where it was interrupted.

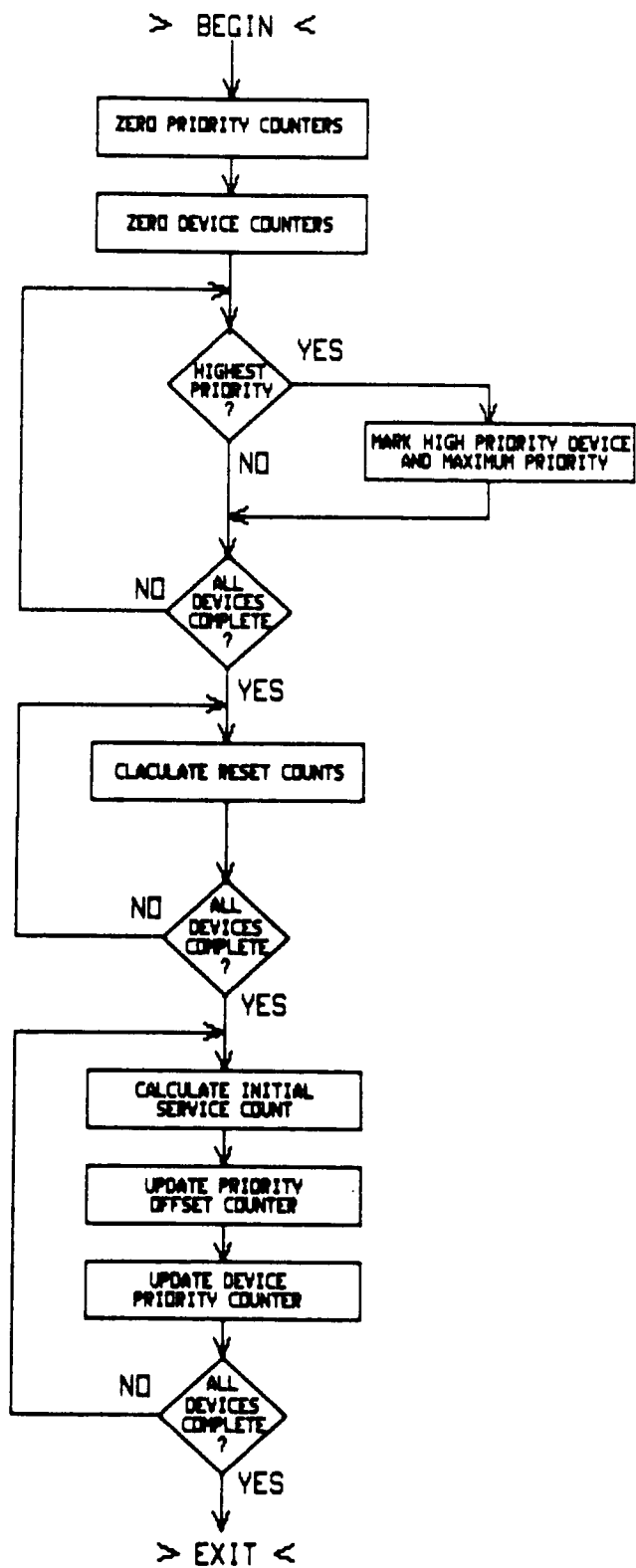


Figure 38. Scanning Initialization module flow diagram.

The RTD Scanning Handler also controls the error detection and recovery activities during communications. When an error is detected, the mode of the device will change to reflect the desired recovery action. If the device successfully recovers, the device mode will return to initialize the device and resume normal scanning.

The RTD Scanning Handler calls several lower level modules to perform the various activities associated with the modes and communication transactions. Figure 39 shows the hierarchy of modules called by the RTD Scanning Handler. Each of the online operating modes are serviced by a separate transmission and reception module. The online operating modes are those which engage in direct communication transactions. These modes are: (1) initialization mode, (2) scanning mode, (3) self test mode, (4) error recovery mode, and (5) command queue mode. The offline modes include those where communication transactions are terminated such as offline mode, error abort mode, and hold mode. When these modes are selected, no support modules are called and the RTD Scanning Handler skips to the next online device. The transmission and reception modules in turn call other lower level support modules. These include the Error Detection module, the Error Problem Action module, the Error Logging module, and the Device Statistics module.

The RTD Scanning module is designed to scan through the control records and request blocks for each online device and grant service to those with a completed communication transaction or those devices where the service count indicates it is due for a new transmission. RTD scanning begins with the first online device and then proceeds

to the next online device by using a forward index pointer maintained in the PLC control record. Figure 40 shows the program flow diagram for this module.

The RTD Scanning module consists of a loop which cycles through each online device. Before entering the scan loop, the first online device is selected. The first action within the scanning loop is to check if any previous transaction has completed by examining the control flags in the request block. If the transaction has completed, the active service mode at transmission is determined and the appropriate reception handler is called. These modules check for errors, handle the return data, and select the next applicable mode for transmission. The next action is to update the service counter for the device. If a previous frame is still outstanding, the RTD Scanning handler will bypass the transmission modules and select the next device. If no outstanding frame exists and the service counter indicates the device is ready for service, then the transmission modules are called. Again, the chosen device mode dictates which transmission module is called. The transmission modules then select the appropriate frame to send, put the transmission buffer address into the request block, and mark the new request present flag. Finally, the next online device is selected by using the forward pointer contained in the present device control record.

The RTD Scanning module implements the state transitions shown in Figure 30 on page 84. During each reception module, the Error Detection module can change the next state for transmission depending on the recovery action chosen. If no error is found, the next state

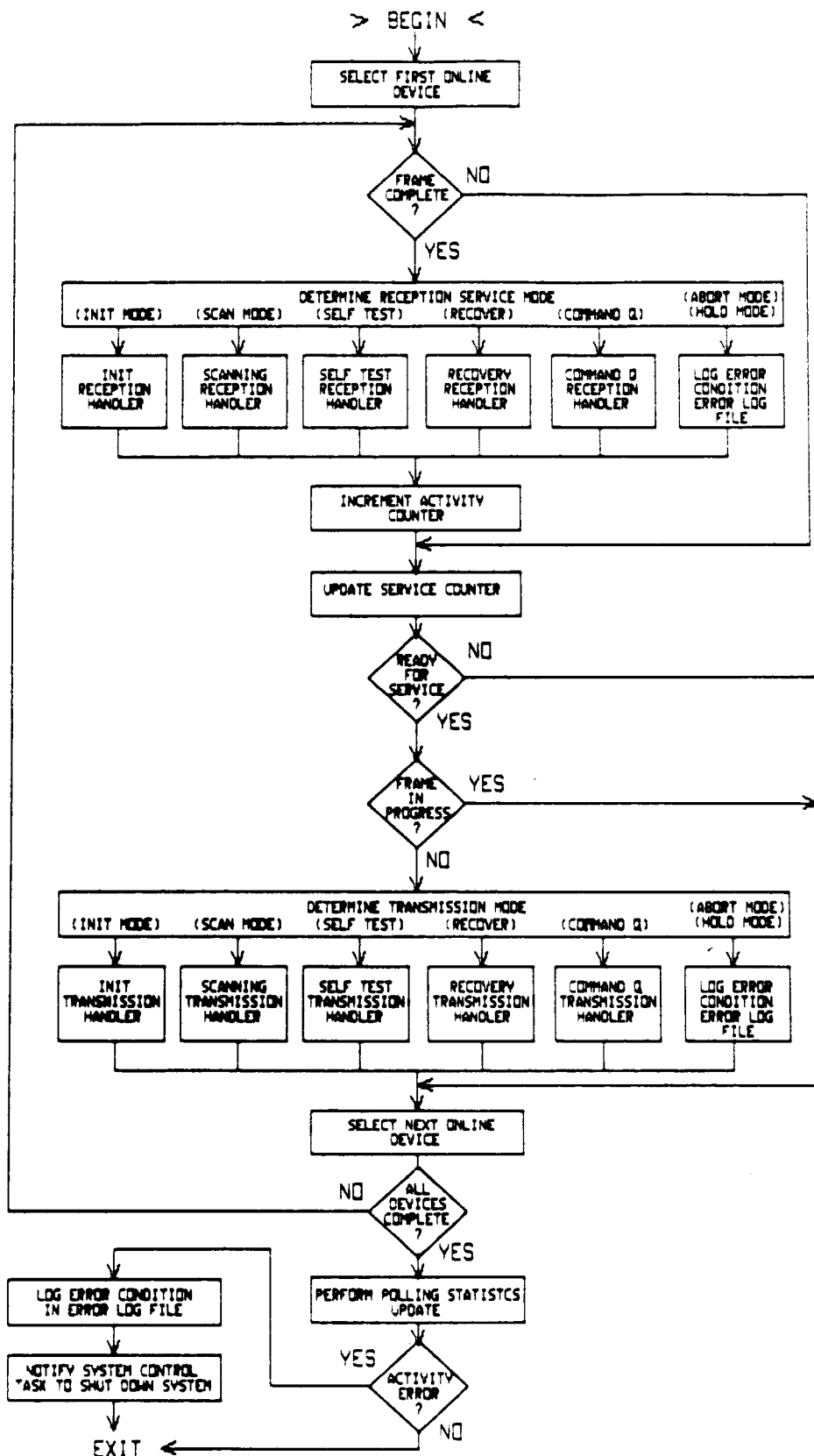


Figure 40. Real Time Data Scanning module flow diagram.

can be determined by the reception module. If the initialization activity has completed, the next chosen transmission state will be the scan mode. When a device is first brought online, the first chosen mode is the recovery mode where the device is sent a SNRM frame. If the device answers correctly, then the next state transition is to the initialization state. After initialization completes, then the scan mode is selected. The self test mode also goes to initialization then scan mode. The command queue mode is selected only when the device is in scan mode and then returns to scan mode when the command queue response frame is complete.

After all the devices have been serviced, the RTD Scanning module calls the Polling Statistics module. This module tracks various statistical parameters used to determine the performance of the CC task. The final action in the RTD Scanning module is to check for activity on the HDLC board. Similar to the Network Initialization module, the number of serviced devices per scan cycle is tracked by the module. If too many cycles pass with no devices indicating completion, the assumption is made that the HDLC board has ceased to operate. The action resulting from this is that System Control task will be notified to shut down the system and perform diagnostics on the board to determine the nature of the failure. After the RTD Scanning module completes, then control is returned to the Executive where the command queue and service queue handlers will be called.

The initialization mode is serviced by the Initialization Transmission module and the Initialization Reception module. These modules perform the same initialization steps on a device as those

described in the section on the Network Initialization module. Figure 41 shows the program flow of the Initialization Transmission module and Figure 42 shows the flow diagram for the Initialization Reception module. At step one, the transmission mode selects the device status read frame by putting the transmit buffer address into the request block for this device. The new request present flag is marked and the transmit statistics updated. On the next pass through the RTD Scanning routine, the Initialization Reception module will be called if the return frame has completed.

The first action in the reception module is to check for communication errors by calling the Detect Error module. If no errors are found, then step one will result in a status check on the return I-frame. The PCF alarm will be logged through the Alarm Handler if the return frame indicates that the controller is not operating. If this condition exists, the step counter will not change and the status frame will be periodically retransmitted to determine when the device enters an operating state. If the device is operational, the next initialization step is selected, any previous PCF alarms are cleared through the AH task, and the reception statistics are updated.

The transmission module will send a configuration read frame at step two. Again, when the next scan occurs the error detector is called and then the step two handler is called if no errors exist. If the configuration frame indicates a different device type than the device selection in the PLC configuration record, an invalid device (INI) alarm is posted. The device is then put into abort mode and no further communications are allowed with the device since it

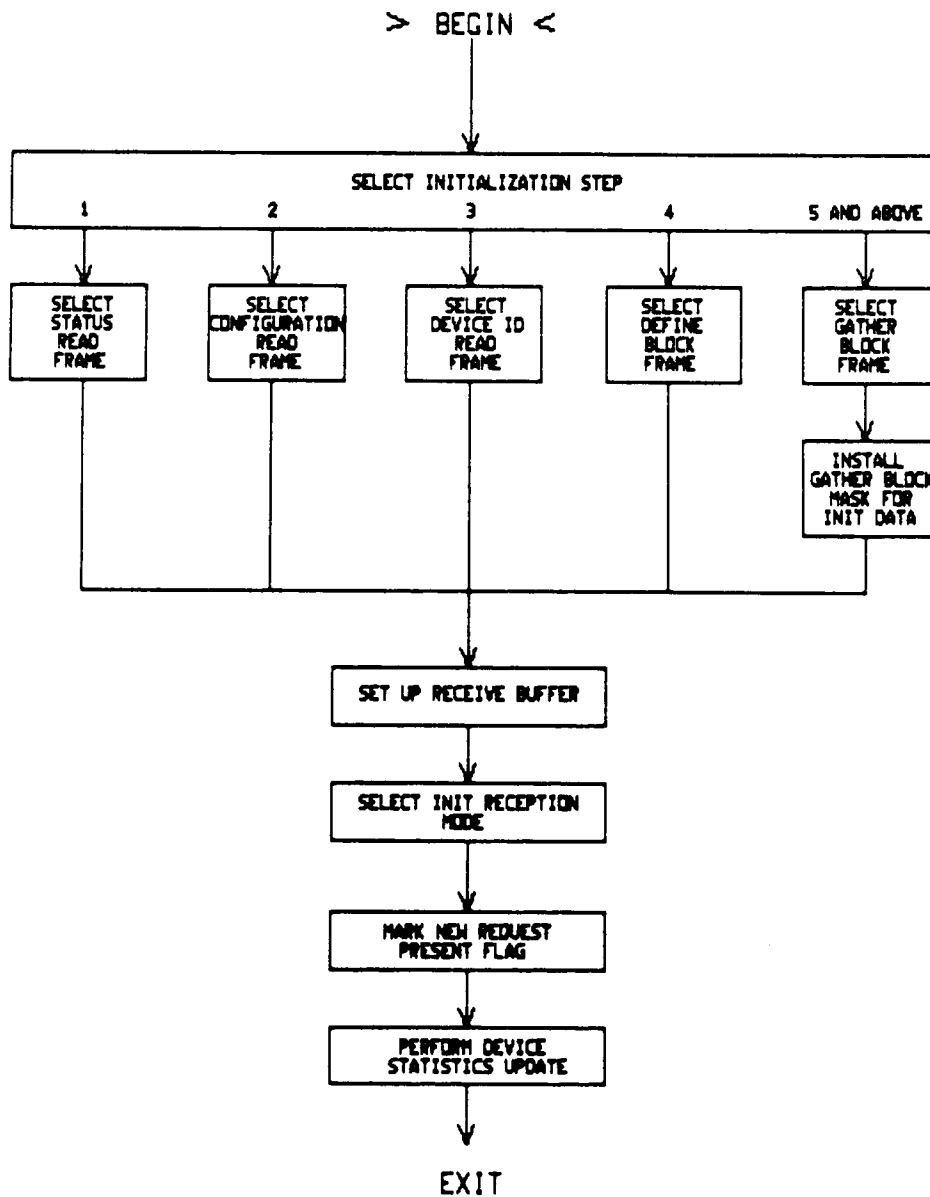


Figure 41. Initialization Transmission module flow diagram.

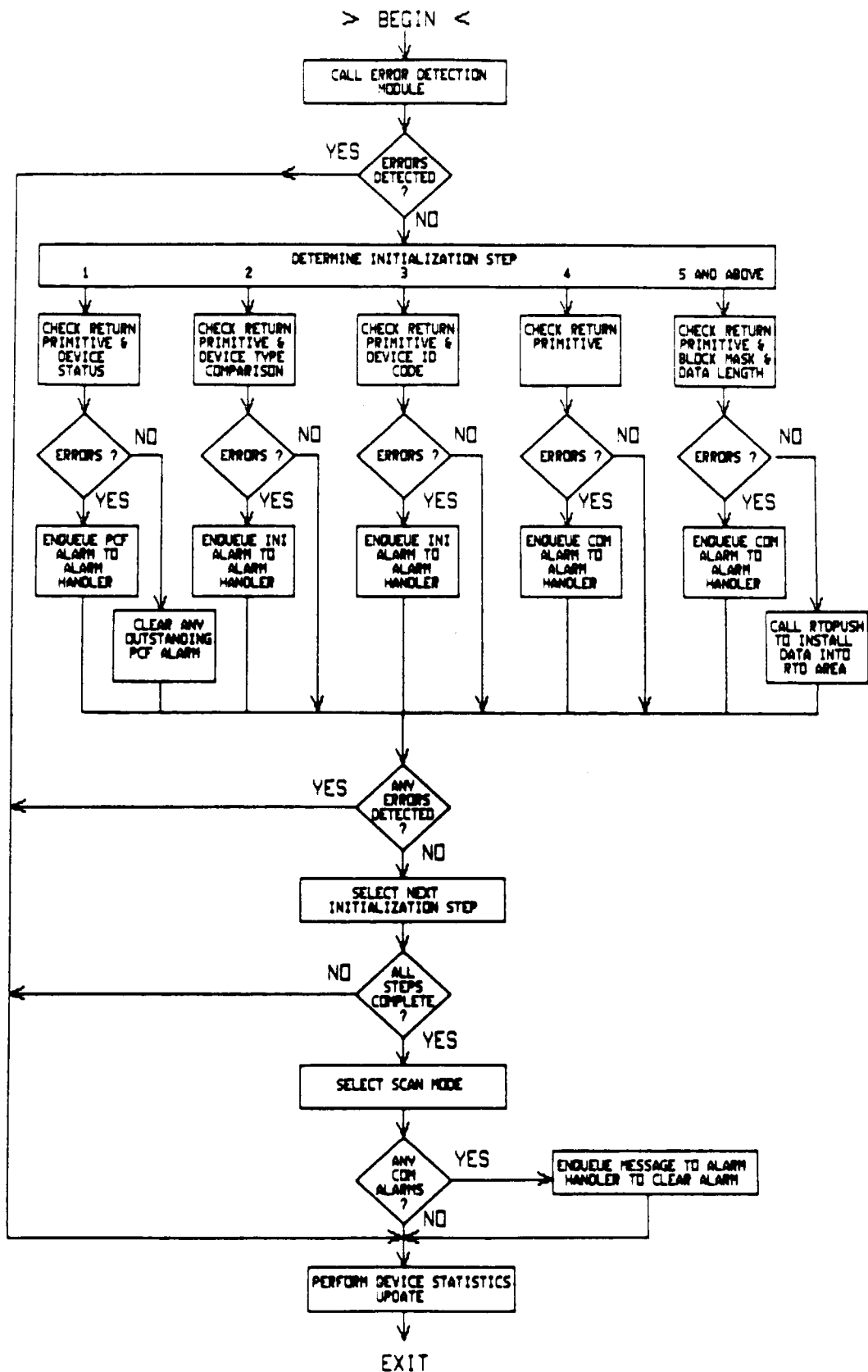


Figure 42. Initialization Reception module flow diagram.

may not be the correct unit. To clear the INI alarm, the operator must take the device offline and determine the source of the problem. Step three of the initialization sequence performs a similar test as step two. This frame reads the configured device ID number stored in V memory in the controller. In the reception module, the result of the read frame is compared to the configured identification code from the PLC configuration record. Again, if these do not match the INI alarm will be posted and the device marked for abort mode. After these steps complete successfully, the define block step is chosen.

To set up the RTD areas for data acquisition during scan mode, the define block frame is sent to the PLC. The transmission module sends this frame by putting the address of the transmit buffer configured with the frame into the request block for that device. The new request present flag is marked and statistics are logged. In the reception handler, a check is made for transmission errors and for the correct reply primitive. No data is returned on the define block primitive.

The final steps consist of sending the number of gather frames necessary to retrieve the range/limit data for the analog inputs and loops. Some devices may not have any of these gather frames to transmit. The transmission module installs the appropriate block mask in the gather frame and puts the address of the gather frame buffer into the request block. When the response completes, the reception module performs error checking and then processes the returned data into RTD. The install location and block size for RTD are maintained along with the block mask in the device control records.

When all the initialization steps have completed successfully,

the scan mode is selected. Also, if a previous error condition was present and the initialization mode was selected as the recovery action, an alarm clear request is enqueued to Alarm Handler.

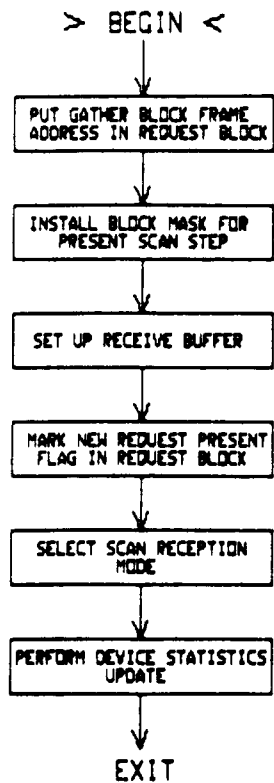
The scanning mode is serviced by the Scanning Transmission and Scanning Reception modules. Figure 43 shows the program flow for both of these modules. The scanning action consists of cycling through however many gather frames are necessary to retrieve all the defined RTD area in each device. A fully loaded controller can require up to 15 gather frames to retrieve all the runtime data.

The Scanning Transmission module selects the gather frame by putting the transmit buffer address into the request block. As each step of the scanning process is selected, the block mask is installed into the gather frame. After the new request present flag is marked in the request block, the statistics are logged as usual.

The Scanning Reception module calls the error detection module and checks for communications problems. If no error is present, the data is put into RTD using the index and block size kept in the control record with the gather mask. The next gather step is then selected. If all the gather steps have been issued, then step one is chosen and the process repeats continuously. The final action is to log the reception statistics.

The Self Test Transmission module is designed to initiate the device self diagnostic sequence by sending the appropriate self test frame. After the self test frame is sent and received, the next step implemented by the transmission module is to delay a configurable number of scan cycles before requesting the initialization mode. This

TRANSMISSION MODULE



RECEPTION MODULE

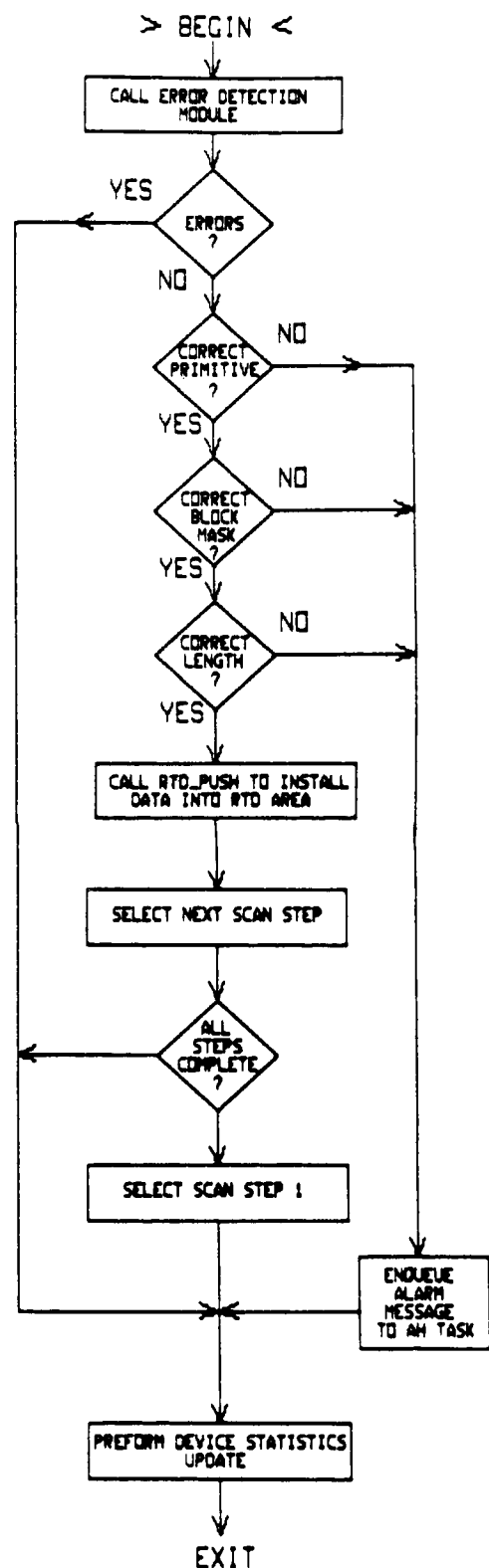


Figure 43. Scanning Transmission and Reception module flow diagrams.

delay is necessary to allow the device to complete internal diagnostics before performing the initial status tests in initialization mode.

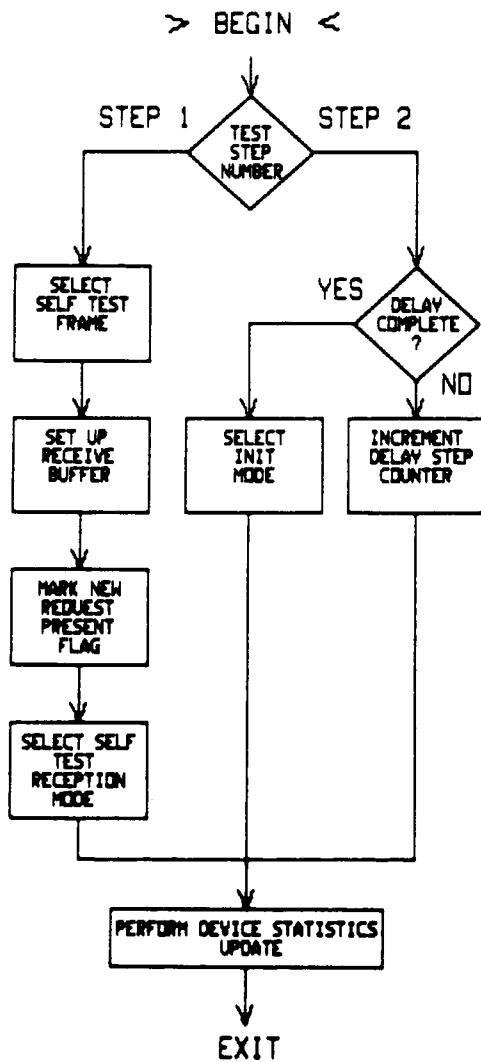
Figure 44 shows the program flow for the Self Test Transmission module.

During step one, the self test frame is transmitted by putting the transmit buffer address into the request block and setting the new request present flag as usual. The delay cycles in step 2 are counted by updating a counter and comparing it with the limit. When the limit is reached, the next mode is selected to initialization mode. The Device Statistics module is called as usual.

The Self Test Reception module program flow is also shown in Figure 44. When the response completes, the Detect Error module is called and the response to the self test transmission is checked for correctness. If the frame completed correctly, step two is selected for the delay in the transmission module. The next action is to determine if this self test was a single device request or part of a broadcast request. If a single device self test is indicated, a completion message is enqueued to the requesting task. If a broadcast self test is active, a counter is incremented to determine when all devices have completed. The final action is to update statistics.

The Error Recovery Transmission and Reception modules are designed to attempt corrective action on certain classes of communication failures. These failures include devices which enter a logically disconnected state where repetitive attempts to reconnect the device are needed. The most common cause of disconnection is physical media failure and power loss at the controller. This module is also used to contact devices which are brought online after the system is running.

TRANSMISSION MODULE



RECEPTION MODULE

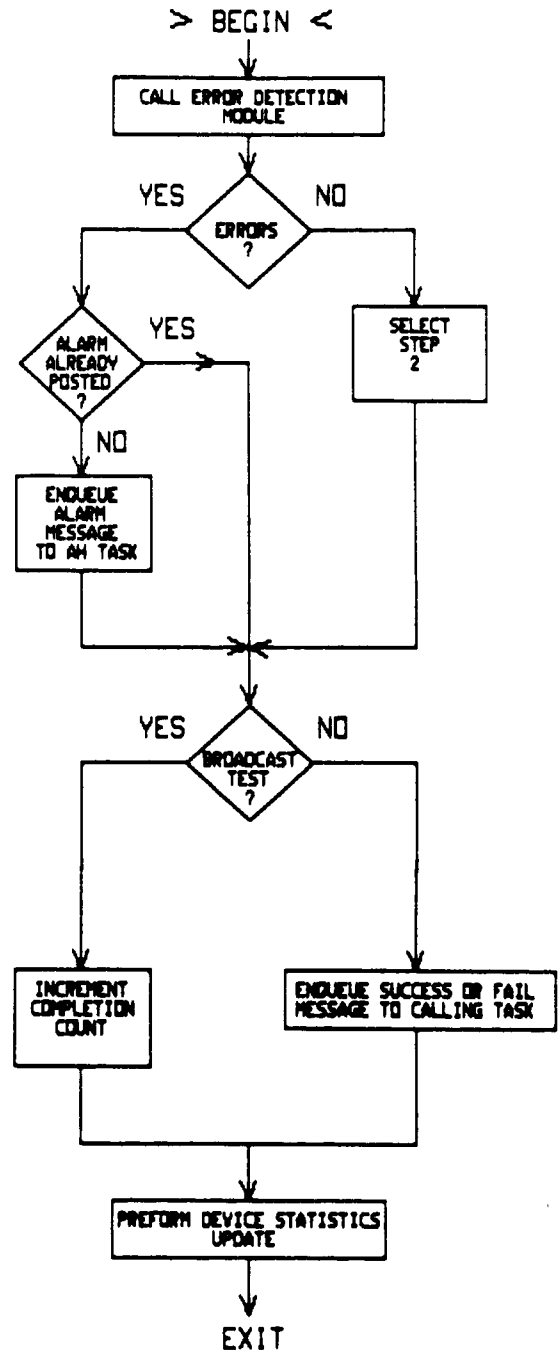


Figure 44. Self Test Transmission and Reception module flow diagrams.

In the recovery mode, the primary corrective action executed is to issue a disconnect request frame followed by a SNRM frame to reconnect the unit. The disconnect frame reinitializes and tests the network interface module in the device. When a device is in disconnect mode, the SNRM frame attempts to logically reconnect the device to the network. Figure 45 shows the program flow diagram for the Error Recovery Transmission and Reception modules. The first action in the transmission module is to determine if the device is already disconnected. If it is not, the disconnect request frame is transmitted and step one is selected for reception. In the reception module, a check is made for any error codes and then the device is checked for a proper disconnect. If any errors are detected or if the device did not disconnect, a communication alarm is posted if the device is not already in alarm. If this step completes correctly, step two is requested. The transmission module will issue a SNRM frame when step two is requested or whenever the device is already offline because of errors. After the SNRM is transmitted, the reception module checks if the device has connected. If it has not, the SNRM frame is retransmitted periodically until the device connects. The SNRM and disconnect frames are sent by setting the appropriate flags in the request block. When the device does connect, the reception module will select initialization mode and the device will continue the progression until normal scanning is established. A counter is implemented at the first of the transmission module which causes the SNRM frame to be transmitted only after a certain number of cycles has passed. The first attempt is transmitted immediately and further attempts are delayed.

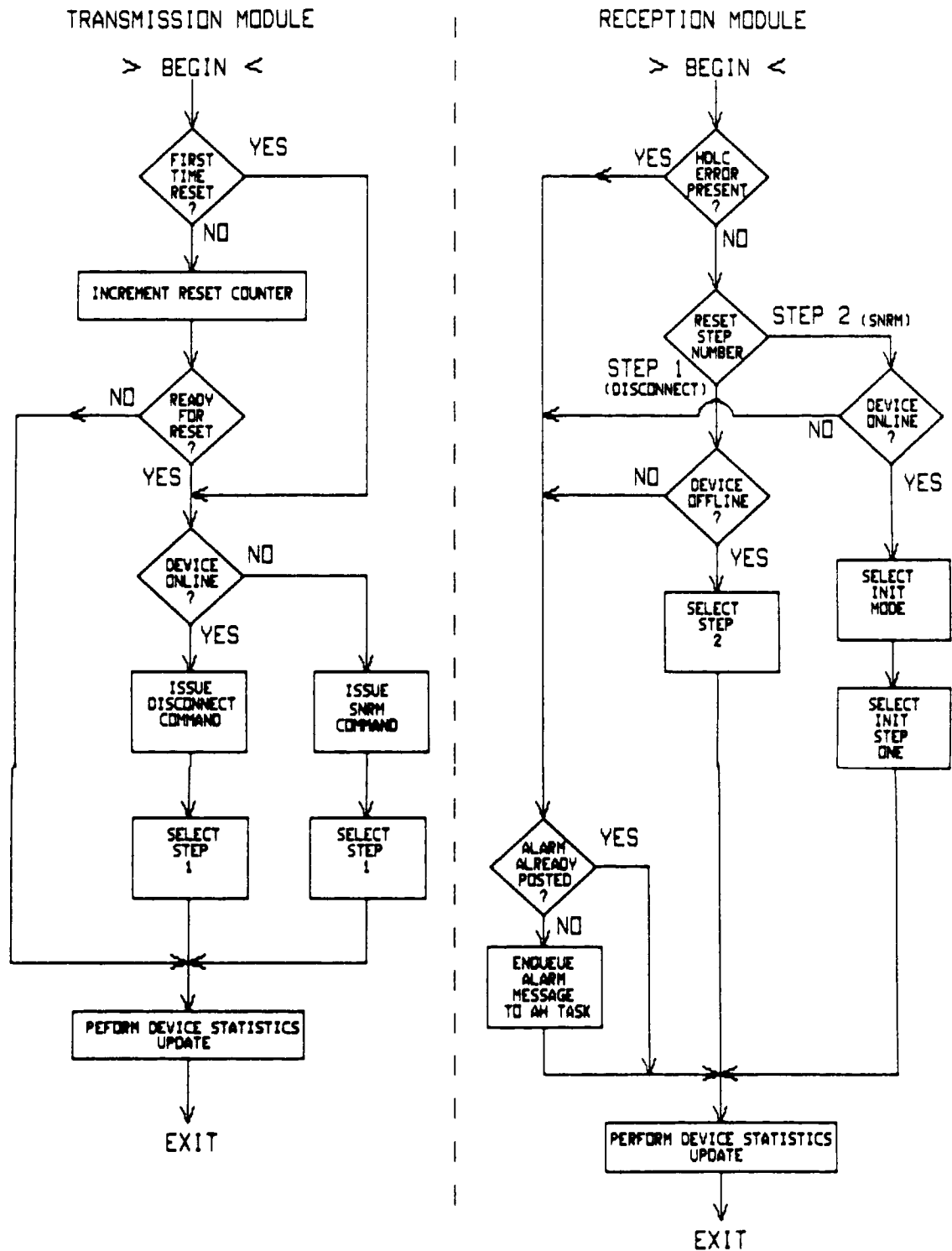


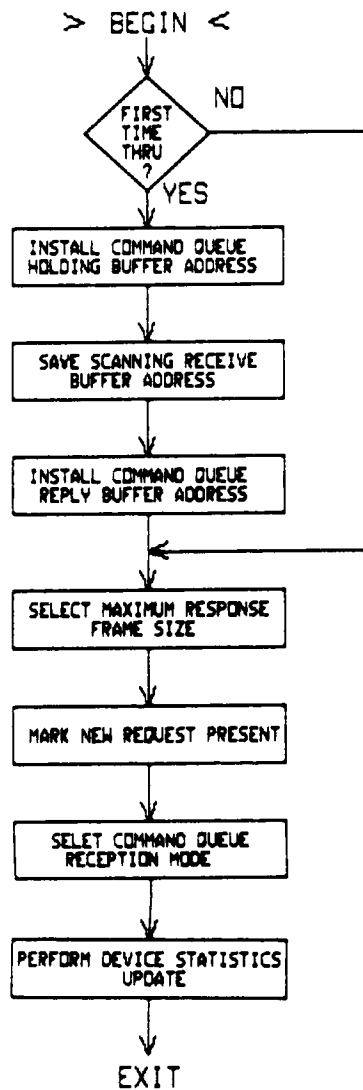
Figure 45. Error Recovery Transmission and Reception module flow diagrams.

The Command Queue Transmission and Reception modules are designed to transmit, receive, and enqueue a command queue frame back to the requesting task. When the command queue frames are received by the CC task, they are put into a holding buffer until the next available chance for transmission. As soon as the previous outstanding frame has been retrieved, the command queue frame is transmitted. Thereafter, the reception module checks for the response frame. When the response returns, it is tested for errors and then enqueued back to the requesting task. When the cycle completes, the device resumes normal scanning where it left off. Command queue frames can also be sent when the device is in hold mode. At completion, the device returns to the dormant hold mode waiting for another command queue frame.

The program flow for the Command Queue Transmission and Reception modules is shown in Figure 46. First, the transmission module installs the address of the holding buffer into the request block. A command queue reception buffer is also installed in the receive buffer address in the request block and the scanning buffer address is saved for reuse when scanning is selected. The special reception buffer is necessary since the length of the response frame may be longer than the response frame for normal RTD scanning. Finally, the new request present flag is set and statistics are logged.

When the response returns, the Scanning module calls the Command Queue Reception handler in Figure 46. The first action is to call the Detect Error routine to check for communication failure. If an error is detected which causes a recovery action, a failure message is enqueued back to the requesting task. If the error action is retry,

TRANSMISSION MODULE



RECEPTION MODULE

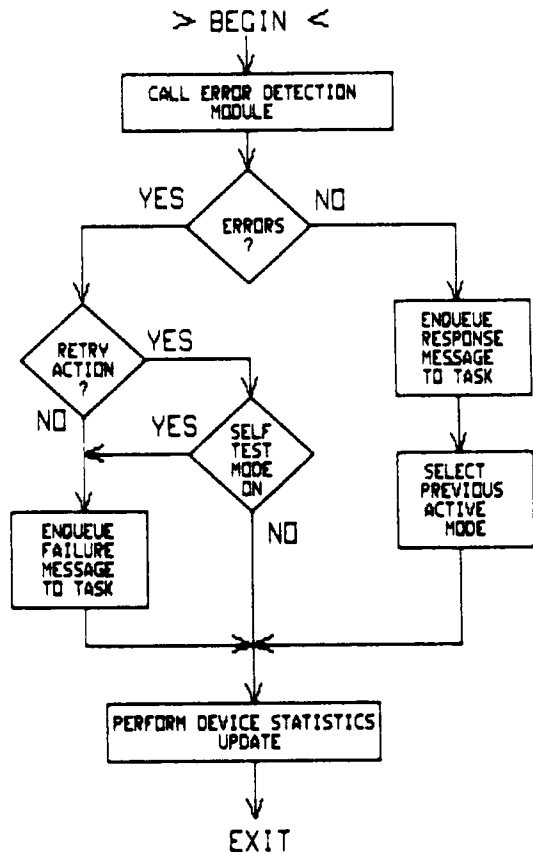


Figure 46. Command Queue Transmission and Reception module flow diagrams.

the transmission module will send the module again. If no error is detected, the response frame is enqueued back to the requesting task. The device mode is then set back to the scan mode or hold mode as before.

Statistical data is gathered during RTD scanning by the Polling Statistics module called by the RTD Scanning handler and by the Device Statistics module called in each transmission and reception module. This statistical data is used to assess the performance of the scanning handler and to adjust the tuning priority of the individual devices to achieve the desired scanning rate. The Polling Statistics module gathers performance information on the number of devices serviced during a scan through the list and the time it took to perform the scan. This includes the number of transmitted requests, the number of received frames, and the time elapsed since the completion of the last scan. To determine long term performance, the minimum, maximum, last, and average values are maintained in the statistical records. The Polling Statistics module also tracks the progress of a broadcast self test command and sends a queue message to System Control when all the devices complete the self test request and enter the initialization mode.

The Device Statistics module gathers performance information on each individual online device as a transmission or reception action occurs. Figure 47 shows the program flow for this module. The first action is to determine if statistics are enabled for this device. Statistics can be disabled through the service queue to reduce the overhead during normal operation. If statistics gathering is enabled,

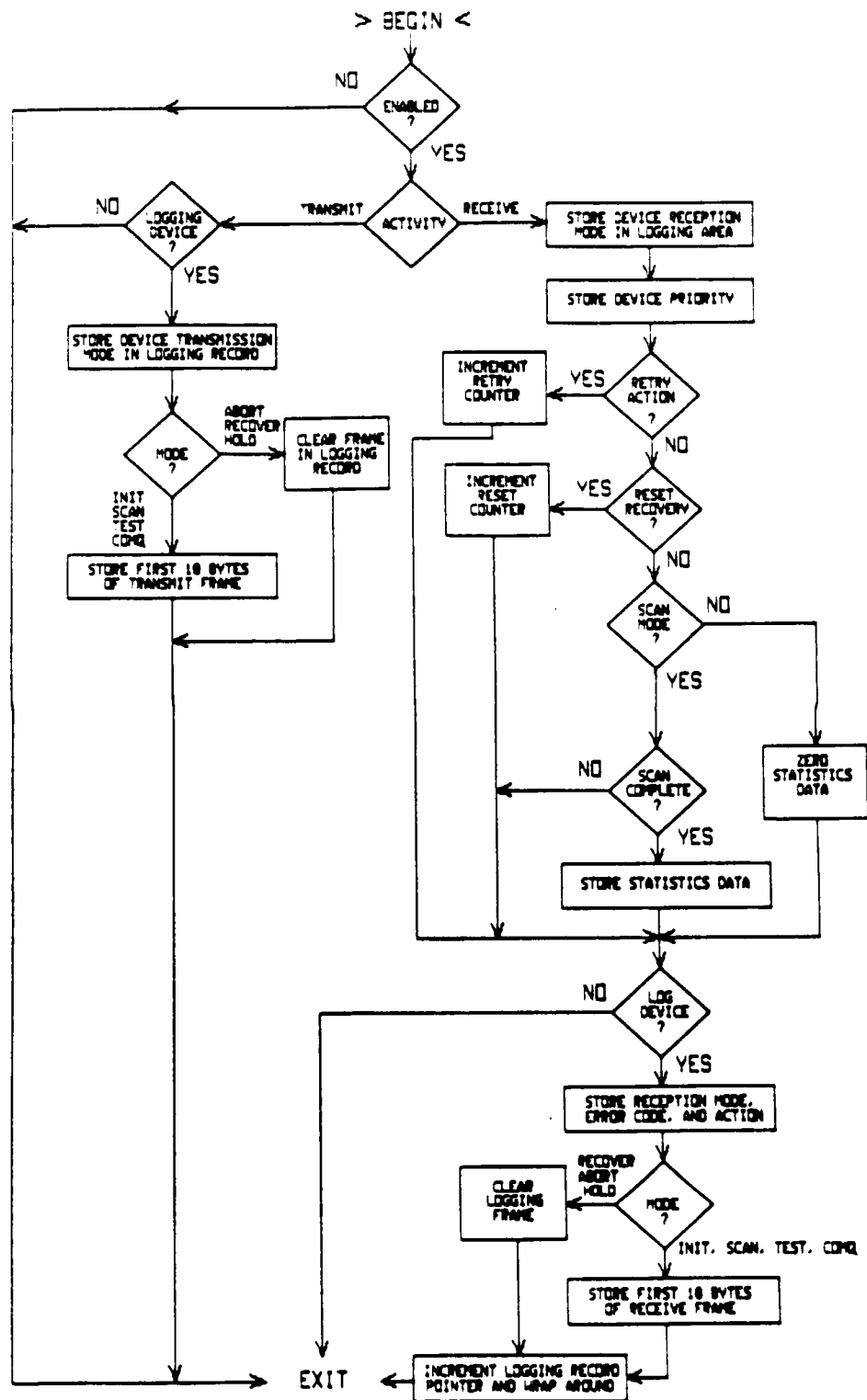


Figure 47. Device Statistics module flow diagram.

the next action is to determine if this is a transmission or reception call to the module. During a transmission action, the device is checked to see if frame logging is enabled. Frame logging consists of storing the first 10 bytes of each frame transmitted and received. A rolling buffer of the last 35 transactions is maintained. This is useful for debugging a faulty device by examining the sequence of events leading to a failure condition. Since only one buffer exists, only one device can be logged at a time. The frame logger maintains a record of the device mode at transmission and only logs frames if the device is in an active communication mode.

A reception module call will store several statistical parameters on the device. The reception parameters stored include the number of complete scan cycles (all RTD gather frames), the average scan time, the minimum scan cycle time, the maximum scan cycle time, and the last scan time. If the logging function is enabled, the first 10 bytes of the response frame is stored along with any error found and the recovery action assigned to the error. The reply frame is logged along with the transmit frame in the 35 entry buffer area. This information gives the system engineer a method of tuning the device priority to achieve the desired performance from each device.

Communication error detection is handled by the Error Detection module. This module examines a returned primitive frame and looks for various error conditions in the response data. Each frame reception module calls Detect Error at the start of processing. Figure 48 shows the flow diagram for the Error Detection module. The first error check is performed on the receive buffer error code location.

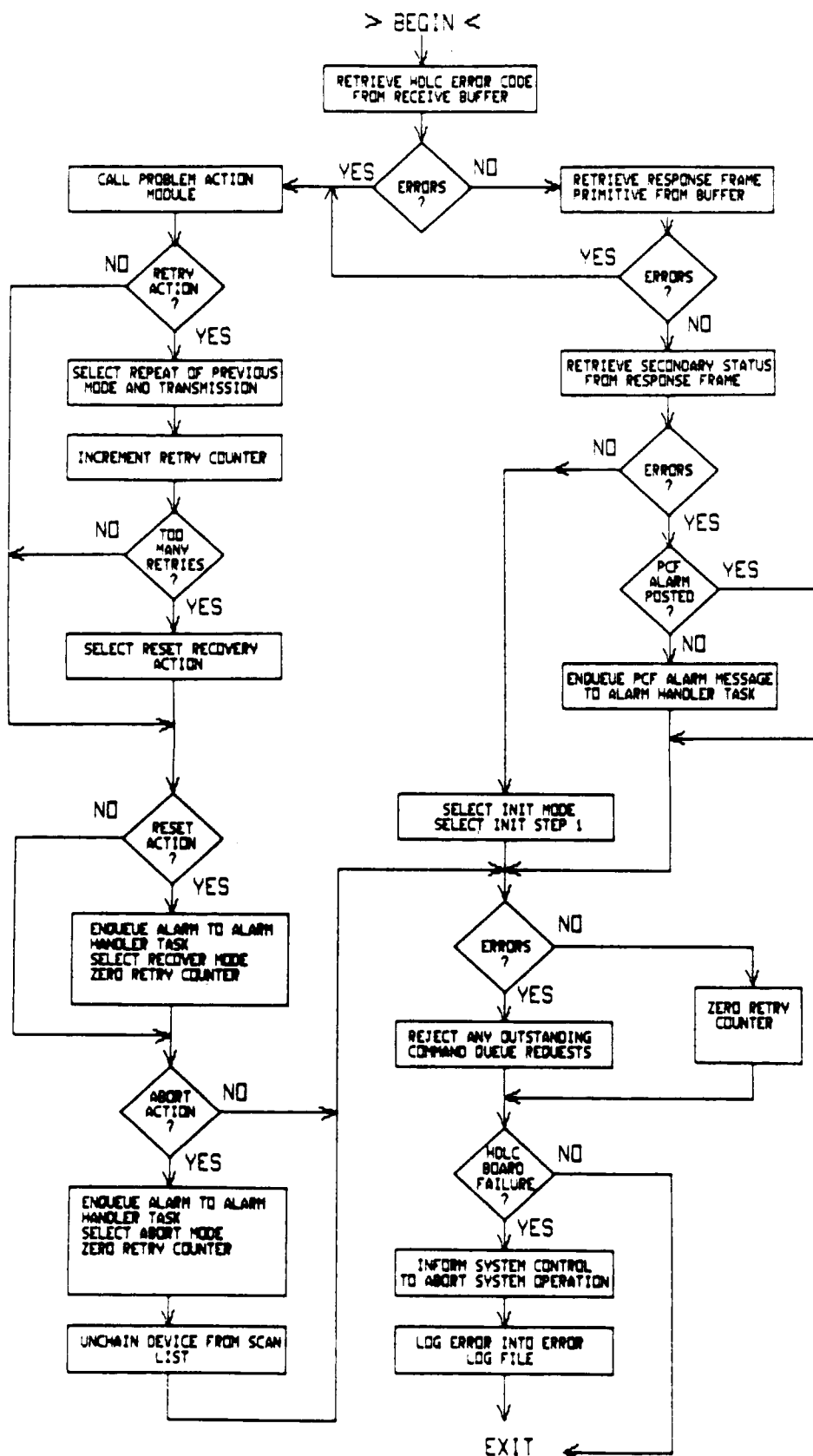


Figure 48. Error Detection module flow diagram.

If an error condition is found, the Problem routine is called to identify the recovery action. The Problem routine performs a binary search on a table of error codes and actions. When the proper code is found, the recovery action is passed back to the Error Detection module. If the error recovery action is abort mode or recover mode, the Error Logging module is called to inform Alarm Handler to post a communication alarm. A check is performed in the Error Logging module on the number of devices in alarm on each channel. If all devices are in alarm, the channel failure alarm is also posted. After the error code is tested, the primitive location is checked for an exception primitive. The exception primitive indicates a failure between the network interface and the controller. Again, the Problem module is called to determine an appropriate action. Abort mode and recovery mode result in a call to the logging module to post the alarm. The retry action will cause the same frame to be retransmitted until a correct response is obtained or the retry count is exceeded. Expiration of the retry count results in selection of the reset recovery mode. A check on the device operational status is performed by examining the value of the HH parameter which follows the response primitive in the frame. If a problem is found, the Error Logging module is called and a PCF alarm is posted. Selection of the initialization mode results in periodic reads of the device status until the device is operational. The final action in the Detect Error module is to check the status mailbox on the HDLC board for an internal parity or bus error notification. When this occurs, System Control task is informed to abort system operation and perform diagnostics.

Command Queue Request Handler

The Command Queue Request Handler module is called by the Executive when a command queue communication frame is detected. This module is designed to retrieve the frame from the queue, place it in the first available command frame holding buffer, and mark the appropriate PLC control record to service the frame. Figure 49 shows the flow diagram of the Command Queue Request Handler module.

The first action of the module is to dequeue the frame from the command queue. If the controller is in scan or hold mode, the frame can be transmitted. Any other device mode results in a failure message back to the calling task. If the frame can be sent, the command frame holding buffers in dual-port memory are searched for the first available buffer. That buffer is then allocated to the desired controller and the command frame is installed into the buffer. The response buffer in the holding area is also allocated for the response frame. Finally, the device mode is selected for command queue message mode and the routine exits back to the Executive.

When the RTD Scanning module is next called by the Executive, the command frame will be sent. The command queue transmit module will send the frame only after the Scanning Reception module has retrieved the outstanding gather frame. If an error occurs on the receive frame, a failure message will be enqueued back to the calling task and the command queue frame aborted. After the command queue frame has been sent, the Command Queue Reception module will test for completion on every cycle. When completed, the device will return to the previous mode and resume normal RTD polling.

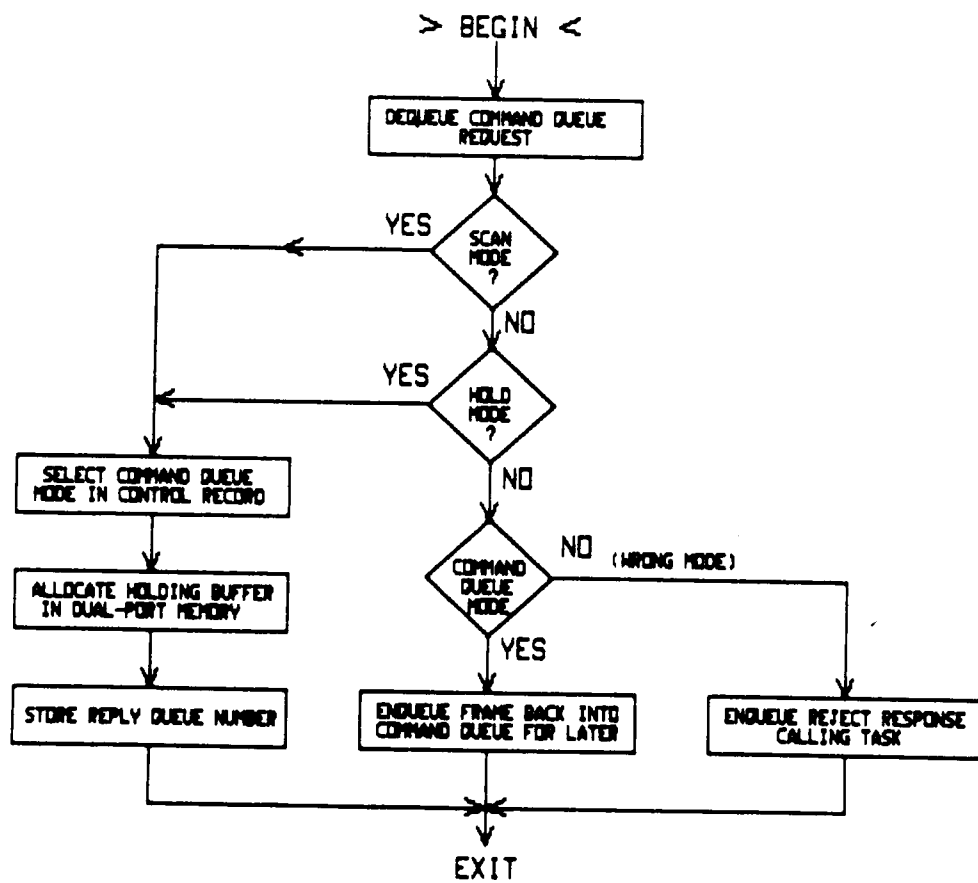


Figure 49. Command Queue Request Handler module flow diagram.

Service Queue Request Handler

The Service Queue Request Handler is designed to retrieve and service commands from other tasks to perform operational actions on a single controller or all devices on the network. These actions include taking devices offline and online, putting devices in hold mode, requesting self diagnostics on a single device or all devices, enabling and disabling statistics and frame logging, and resetting statistics and logging data. Figure 50 shows the program flow of the Service Queue Request Handler module.

The first action in the module is to dequeue the service command and to decide which device the action concerns. On commands affecting all devices, the device number zero is sent. The next activity is to select the appropriate service handler for the requested action. The 11 service handlers available are: (1) the offline handler, (2) the online handler, (3) statistics enable, (4) statistics disable, (5) statistics reset, (6) frame logging enable, (7) frame logging disable, (8) frame logging area clear, (9) device self diagnostics, (10) broadcast self diagnostics, and (11) hold mode request.

The offline handler is designed to remove a device from the active scan list, clear selected data structures, and correct the flow control cadence. Removing the device from the active scan list requires pointing the chain index of the previous online device to the online device after the chosen controller. This causes the RTD Scanning handler to skip over the offline device. Next, the service mode is set to offline and the online flag is cleared to prevent other service queue or command queue activity on the device. All

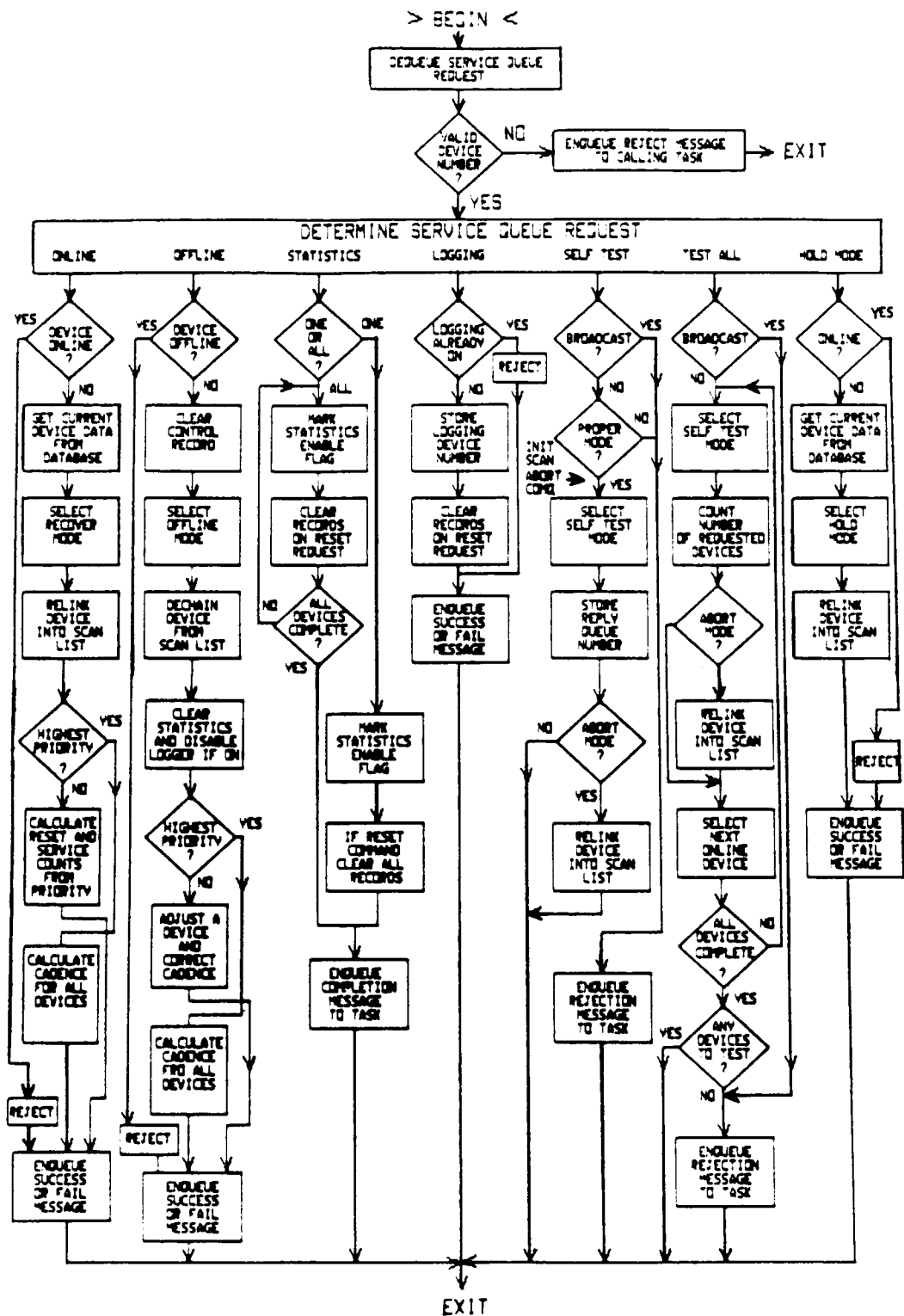


Figure 50. Service Queue Request Handler module flow diagram.

statistical information is cleared. The frame logger is disabled if currently active of the offline device. Next, the scanning cadence is corrected as described in Chapter III by replacing the offline device with another device of equal priority. Finally, a completion message is enqueued back to the requesting task.

The online handler performs the reverse operation of the offline handler by inserting the chosen device back into the scanning list. First, the PLC configuration record in the database is read to update any parameters which have changed such as network address, cable number, or priority. Next, the device is inserted into the scanning list by finding the previous online device and changing the chain index to point to the new device. The controller pointed to by the previous device is now pointed to by the new device. When the RTD Scanning handler runs, it will now go from the previous device to the new online device then to the next device. Service mode will then be marked for recover to issue the initial SNRM frame and connect the device. If the controller comes online, the device will proceed through the initialization mode to scanning mode. Next, the initial service count is calculated using the equation on page 97. This service count takes into account the device priority, the present scan cycle, and the number of devices on that priority level. Finally, the completion message is enqueued back to the requesting task.

Device statistics gathering is controlled by the statistics enable, disable, and reset handlers. The statistics enable handler checks the device mode for online mode and then marks the statistics enable flag on that device true. A message is then enqueued back to the

calling task indicating success. A failure message is sent if the chosen device is offline. If the requested device is zero, then all the online devices are enabled. Similarly, the statistics disable handler marks the enable flag on the chosen device false and the enqueues a completion response. Again, if the device number is zero, all device statistics gathering is disabled. The statistics reset request clears all statistical data on a device or all devices and responds with a queue completion message.

The logging enable, disable, and reset functions operate similar to the statistics gather handler except the logging function can only be enabled on one device at a time. Enabling logging is accomplished by putting the desired PLC number into the logging device number. If another device is already enabled, a failure message is enqueued back to the requesting task. To disable frame logging, the logging device number is set to zero. The logging frame storage area is reset by clearing each of the 35 frame records.

Single device self diagnostics can be requested through the service queue. The first step is to test the service mode of the device. Self diagnostics cannot be requested on an offline device, a device in hold mode, or a device in recovery mode which has lost contact with the network. If the desired device is in these modes, a failure message is enqueued back to the requesting task. If in the correct mode, the device mode is marked for self test mode and the next pass of the RTD Scanning handler will issue the self test frame. The response enqueue of success or failure is enqueued by the Self Test Reception module during RTD scanning.

Broadcast self diagnostics is accomplished by enabling device self diagnostics on all online devices in the proper mode. This is done by looping through each device on the active scan list and changing the service mode to self test if the device is in the proper mode. The broadcast self test flag is set to inform the Polling Statistics module to check for completion and to signal the self test reception module to count the number of completed devices. If no devices are available for broadcast diagnostics, a message is enqueued immediately to inform the requesting task. Otherwise, the completion message is enqueued by the Polling Statistics module when all devices have been accounted for.

Finally, the hold mode request handler marks the service mode on the chosen device for hold mode only if the device is presently offline. The hold mode is used to allow command queue activity without RTD scanning. While in the hold mode, the ladder logic program in the PLC can be altered without affecting data acquisition. If the device is not offline, a failure message is enqueued back to the calling task. Otherwise, a completion response is sent. When a device enters the hold mode, no checks are performed on the operational condition of the device such as those performed in the initialization mode. It is the responsibility of the task driving the command queue to perform tests on the identity and health of the chosen controller.

CHAPTER V

TEST RESULTS

Real Time Data Scanning Performance

Testing of the Communications Control task is necessary to provide information on the performance of the network under various loading conditions. One of the most critical areas is the performance of the real time data scanning function. RTD scanning must be tested with a simulated network load equivalent to a normal load in actual system usage at a refinery. This performance testing will also validate the effectiveness of the scan tuning algorithm and communication flow control method.

To effectively test the RTD scanning performance, the actual environment of the CC task must be simulated. This is necessary because of the lack of a completely functional TFA system at the time of testing and also to reduce the level of complexity in a multi-tasking environment. A testbed program was written to simulate the TFA system environment and to provide tools for assessing network performance. The testbed creates the CC task and a simulated load task. A configurable amount of the CPU resource is absorbed by the load task to simulate the other TFA tasks operating. The testbed task allows the engineer to create command queue and service queue messages for the CC task and examine the response message from the CC task. Statistics can be viewed as the CC task operates to assess performance.

Network loading also presents a testing problem because it is impractical to test with over 200 tank transmitters and controllers.

This problem was solved with the development of a universal device simulator. This device will respond to 105 different network addresses and implements all the network primitives. The universal device simulator also contains five V memory locations for each secondary which can be altered using a local terminal. This makes the device ideal for simulating large numbers of tank transmitters on each channel. The controllers are simulated with actual PLCs running ladder logic programs. These ladder logic programs simulate live data changes in the RTD area of each PLC.

The first test conducted is to assess single device performance on the network. This test provides information on the maximum rate of data flow from a single device. For this test, the load task is set to absorb less than five percent of the system resources. A tank transmitter, a moderately loaded PLC, and a heavily loaded PLC were used to conduct the test. The tank transmitter requires only one gather frame of 8 bytes to retrieve the tank data. An average scan time of 0.15 seconds was obtained on the tank transmitter. To retrieve the complete data area of the medium load PLC required two gather frames. The first gather frame was 256 bytes in length and the second frame was 224 bytes long for a total of 480 bytes retrieved. A complete gather of the two frames by the RTD scanning handler required an average scan time of 1.2 seconds. The heavily loaded PLC required five gather frames to retrieve the complete RTD area defined in the device. Frame data length ranged between 128 bytes and 256 bytes of data with a total data retrieval of 1088 bytes. The RTD scanning handler retrieved the complete data area in an average of 4.7 seconds per scan. These rates

represent the maximum rates obtainable from the CC task under ideal conditions. In actual usage, the CC task will be allowed only a portion of the CPU resources and will be scanning many devices on the network simultaneously.

The next RTD scanning performance test was conducted on a typical network load in a simulated TFA system environment. During this test, the load task was set to absorb 87.5 percent of the system resources. The load task does this by implementing a loop counter for 875 milliseconds and then suspending operation for 125 milliseconds. The testbed task also absorbs some system resource as it is used to view the scanning statistics. Figure 51 shows the system environment used for the full load scanning test. To simulate a tank farm load, the network was loaded with four universal simulators. Each simulator produced a network load of 50 tank transmitters. Four PLCs were also used to simulate the controller load. The first PLC contained 1088 bytes of RTD information which was retrieved in 5 gather frames. The other three controllers presented a combined load of 658 bytes of data. This data is retrieved with two gather frames for the medium load PLC and one gather frame for the other two devices. Each of the simulation devices was connected to a different network channel.

The scan tuning priority of the devices was set to provide equivalent scanning on each device. This required a priority of four for the first PLC, priority level three for the medium load PLC, priority one for the lightly loaded controllers, and priority zero for the tank transmitter devices. In this priority arrangement, the tank transmitters are scanned every fifth scanning cycle. This provides

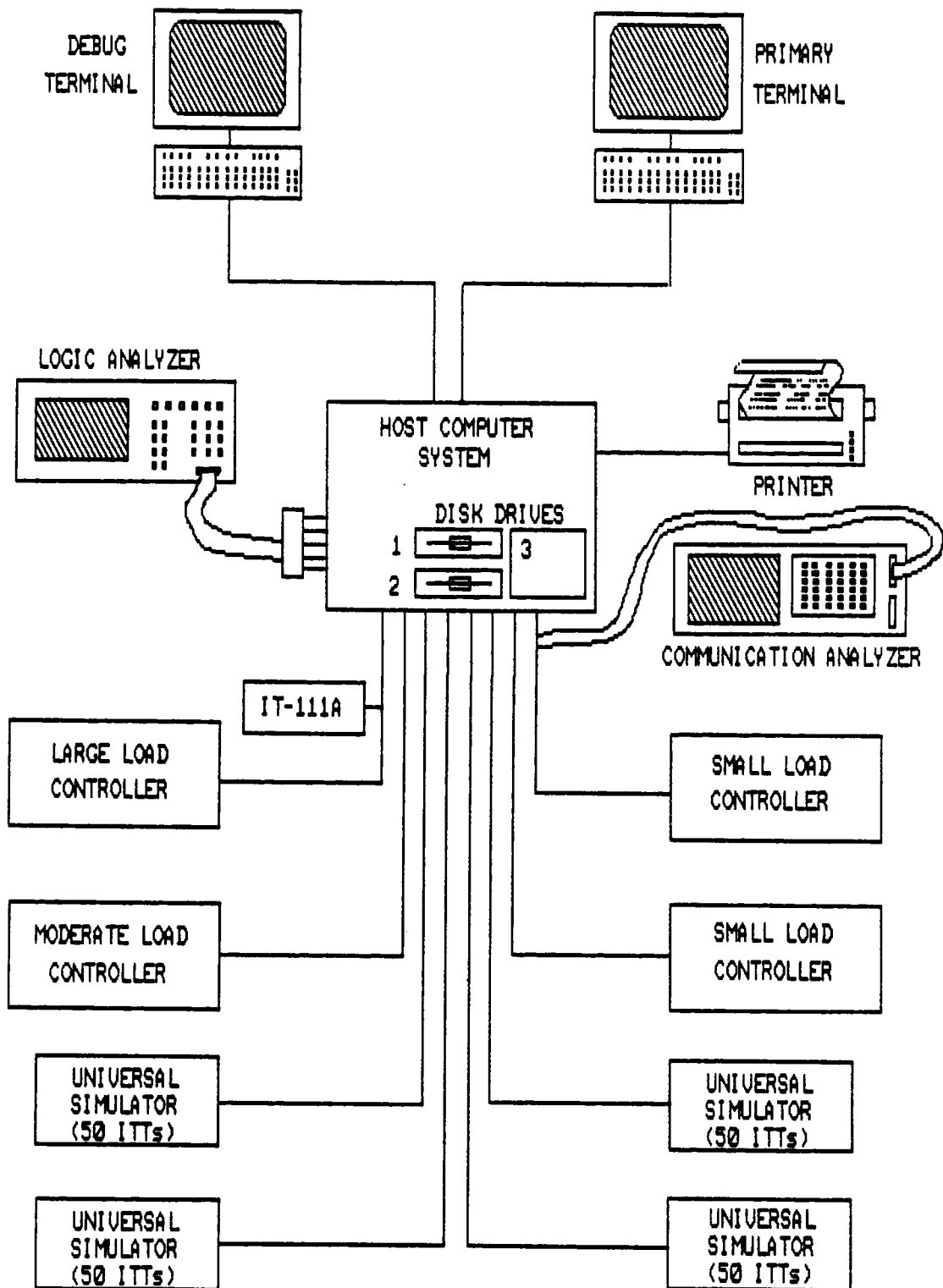


Figure 51. Scanning performance test environment.

equivalent scanning for the large PLC which completes a scan every five cycles. The medium load PLC is serviced every other cycle. Since this device requires two gather frames to retrieve all data, a scan is completed once every four scan cycles. This is approximately the same scan rate as all the other devices on the network.

With the described test environment, the RTD scanning function achieved an average scan rate on the large PLC of 30.6 seconds to retrieve all data. To complete a gather of all data on the moderately loaded controller required an average of 24.5 seconds for both data blocks. The tank transmitters were scanned at an average rate of 30.9 seconds for one gather frame on each of the 200 devices configured. This test can be considered a good approximation of actual expected performance at a refinery on a heavily loaded system and network.

The next RTD scanning test was conducted to determine the effect of system loading on the average scan times of the various devices. During the test, the device priorities were tuned for equivalent scanning. The test was conducted by varying the amount of system resources absorbed by the load task. This would simulate the load imposed on the system by the other tasks during various conditions. Figure 52 shows a graph of the results of this test. At 90 percent system loading, the large PLC achieved a scan rate of 35.3 seconds. At this loading level, the CC task is allowed only 10 percent of the available computing resources. In actual usage, the CC task should be using about 25 percent of the resources. This would provide scan rates on the order of 11 to 15 seconds for all devices. The loading levels at 10 and 35 percent represent the best and worst case conditions.

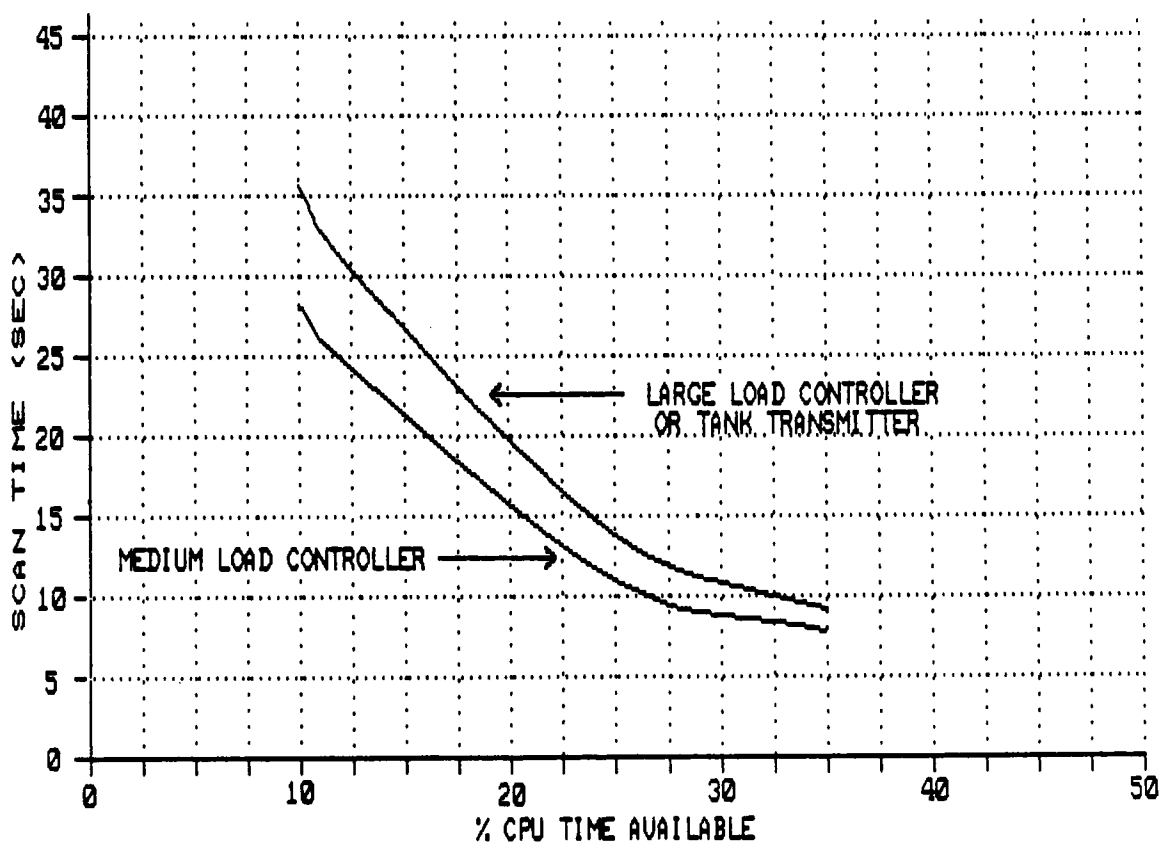


Figure 52. System loading effects on scan rate performance.

The final RTD scanning test was conducted to determine the performance of the scan tuning algorithm. This test was performed using the same setup as described above. During this test, all devices were started at priority zero. The priority of the heavily loaded PLC was raised one priority level and the scanning statistics were gathered again. Scan priority was increased until the PLC was at the highest priority level (20). Figure 53 shows the average scan time of the heavily loaded PLC, the moderate load PLC, and the tank transmitters as a function of the priority of the heavily loaded PLC. As can be observed, the scan rate of the loaded PLC decreased as the priority was raised. The scan rate of the tank transmitters and the other two controllers increased as more communication resources were devoted to the maximum priority device. However, the rate of increase of the scan time on the low priority devices was less than the corresponding decrease in the scan rate of the maximum priority PLC. The point at which the tank transmitter line crosses the large PLC line is the equivalent scanning point for these devices. If the priority of the moderate load PLC were raised, the scan rate for this device would also decrease and cross near the equivalent scanning point.

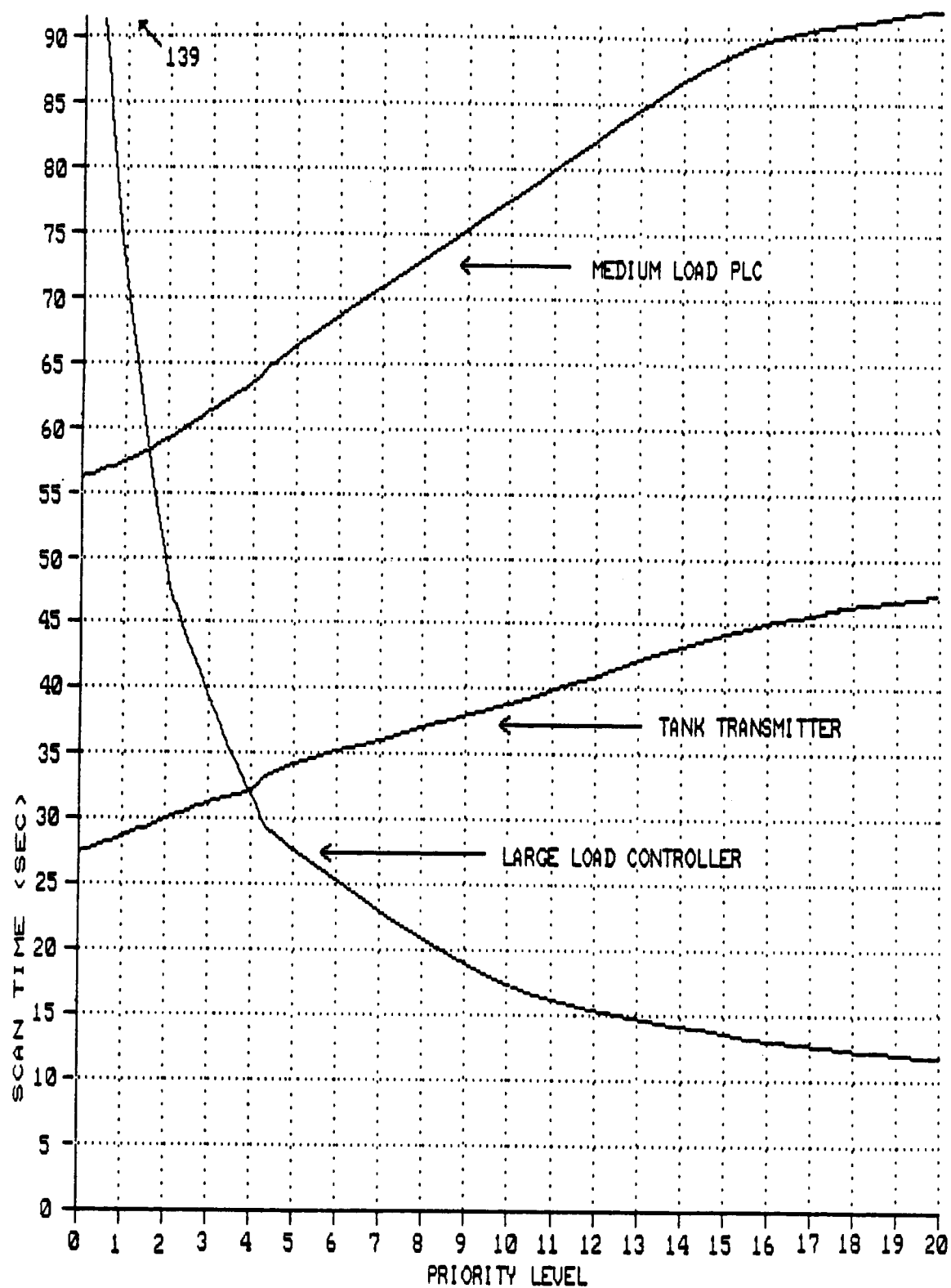


Figure 53. Scan tuning priority test results.

Command Queue Message Response

The command queue message function was tested to obtain performance data on the response rate for an application task sending a command frame to a PLC. Testing was conducted using the equipment configuration shown in Figure 51 and described in the previous section. The command queue messages were formatted in the testbed task and enqueued to the CC task. Time was measured between the enqueued transmission message and the response frame return in the reply queue.

Command queue response testing was performed on three different devices: the heavily loaded PLC, the moderate load controller, and a tank transmitter. First, a frame was transmitted to perform a write to one word of V memory. This would simulate a normal command being issued to turn on an output point. With the large controller, this frame transmission completed in an average of 8.5 seconds. This includes the time to enqueue the message to the CC task, internal processing time, network communication time, controller response time, time to deformat the response by the command queue reception handler, and time to enqueue the response back to the testbed. The response rate also includes the time to retrieve any previous outstanding frame from normal RTD scanning. A command queue frame to the medium load controller required 9 seconds and the tank transmitter responded in 8 seconds. Response rate to a large frame was also obtained by requesting a read of 100 V memory locations. This results in a 200 byte data retrieval. Average response rate for this transaction to the large PLC was 11.0 seconds while the medium controller responded in 6.0 seconds.

Service Queue Performance

The service queue performance was tested to find the time required to bring a device online, take a device offline, and execute self diagnostics on a device. Statistics and logging functions through the service queue present no communication load on the network and are almost instantaneous since they require only data structure manipulation. Testing was again performed with the equipment setup shown in Figure 51 on page 154.

The online and offline commands through the service queue received a reply in only four seconds since they only mark the control data structures. When the maximum priority device was commanded online or offline, the response time increased to 4.5 seconds due to the time required to resequence the service counts for all devices on the network. The time required to actually achieve RTD scanning varies according to the amount of initialization data to retrieve. When configured for four gather frames of initialization data, the large PLC required 70.0 seconds before the first scan cycle for data was complete. The tank transmitters have no initial gather frames and require only 42.0 seconds before the first scan is complete. This time can vary greatly due to the initial service count assigned to a device and the priority of the device.

The single device self test was conducted on a tank transmitter. A tank transmitter responds in an average of seven seconds to a diagnostic test request. Again, this can vary greatly due to the priority of each device and the timing of the service command relative to the network activity. The amount of delay configured for the self

test delay also affects the response speed. The time required to begin scanning again is similar to the time required to bring a device online because the device goes through initialization mode after a self test. Average time to resume scanning on a tank transmitter was 56 seconds.

Broadcast diagnostic testing is not possible because the universal simulator for the tank transmitters does not support this function. This prevents testing this function on a fully loaded network. However, the structure of the broadcast command indicates the response time should be a function of the slowest responding device on the network and the configured self test delay.

Error Recovery Response

The error recovery function response is critical for determining the time required to bring a device back online after a communication error has occurred. Testing of this function is primarily concerned with the reset recovery function. The abort function stops further communication and the retry function operates at the same rate as the normal communication transactions. Two parameters are to be tested: (1) the time to detect a failed device and (2) the time required to achieve normal RTD scanning after the device is repaired. This testing is done by disconnecting the PLC or by changing the mode switch on the PLC.

The first test involves disconnecting the network connector on the device and measuring the time for the CC task to send a message to the Alarm Handler task for the communication alarm. Similar to the other response tests, the time varies according to the timing relative to the scanning activity and the priority of the device. The large PLC

required an average of 8.0 seconds to notify the Alarm Handler task of the failure. A tank transmitter requires an average of 27 seconds due to the lower priority of the device and the time of disconnection relative to the service count remaining on the device.

When the device is reconnected, the reset recovery action will be in progress. This function has a built in counter which causes the SNRM frame to be sent at a slower periodic rate than the service rate of the device. This counter is configured for five cycles during this testing. With this reset counter value, a SNRM frame will be issued only once every five scans. Once the device is detected to be back online, the initialization frames are sent before the Alarm Handler is issued a clear command. Again, the amount of time required to complete initialization is a function of the number of frames necessary to retrieve the initialization data. The large PLC in the previous test arrangements required an average of 67.5 seconds between the time of reconnection and the queue message to clear the alarm. A tank transmitter averaged only 45.0 seconds due to the lower number of initialization steps.

The status fault test was conducted only on the large PLC because the tank transmitters do not have an inoperative state. The PLC status can be changed by operating a switch on the top of the device. Again, amount of time required to post and clear a status alarm is a function of the device priority. The testing conducted involved the large PLC operating at highest network priority during equivalent scanning. The average time required to post the status fail alarm is 10 seconds. When the status in any response frame indicates that the device is no

longer operational, the initialization mode is selected. Thereafter, a status read frame is repeatedly transmitted until the device returns to normal operation. When the mode switch is returned to the operational state, the time to clear the alarm averaged seven seconds. The heavily loaded PLC achieved normal RTD scanning after 47 seconds.

CHAPTER VI

CONCLUSIONS

The tank farm automation system operates as a host computer connected to a variety of distributed controllers and data acquisition equipment across a communication network. In the host computer, the data from the distributed devices is collected, organized, and presented to the operator in an easily understood format. The communication control task provides a method to organize and control the data flow from the various devices. This task controls both the continuous retrieval of process data from the network as well as operator generated command messages to control the remote field devices. Error conditions are identified, classified according to appropriate recovery action, and posted by the alarming function to notify the operator of the problem. Recovery action is periodically attempted to restore normal operation to a failed device. A priority tuning structure gives the engineer a means to control the rate of information retrieval from individual devices commensurate with the amount and critical nature of the process data in each device. To prevent network congestion, a communication flow control algorithm controls the flow of network transactions. The statistics package gathers performance data needed for proper tuning of the network.

The modular construction of the communications control task allows easy modification to meet future demands of the host computer functionality. Additional data types and arrangements can be incorporated into RTD scanning by defining the format of the data

structure in the controller and creating a definition of the data type for the RTD Initialization and Scanning module. The command queue message facility allows creation and transmission of virtually any primitive frame structure. If a device is added to the network which supports a new communication primitive, a custom application task can utilize the command queue to retrieve information and send commands to the device. The priority tuning algorithm can be made adaptive to compensate for various conditions in the process controllers. For example, while a process is operating, the priority of the associated controller could be automatically increased to provide faster updates. After the process completes, the priority could be lowered to devote more communication resource to other devices. Similarly, if a tank is nearing capacity during a filling operation, raising the priority would automatically provide faster updates for more accurate loading operations. While a tank is closed, the scan priority would assume a low value for occasional checks on leakage conditions.

Testing of the communications control package indicates acceptable performance for both the RTD scanning activity and the command queue message transmissions. The slow nature of tank fluid movement in a refinery permits data gathering rates between 30 and 90 seconds. Autonomous control features in the distributed controllers automatically regulate the attached processes. This allows data retrieval rates in the order of 15 to 30 seconds for operator updates of the remote condition of these processes. The tuning priority method allows enough dynamic range for the communication scan rate to meet the projected needs for data retrieval.

BIBLIOGRAPHY

BIBLIOGRAPHY

Davies, D. W., et al. Computer Networks and Their Protocols.
Chichester: John Wiley & Sons, 1981.

Spenser, Kenneth. "Standards, organizations, and the seven-layer OSI model." reprint from I&CS-Industrial and Process Control Magazine, April 1985.

TIWAY I Network Protocol Specification. Texas Instruments Document No. 2457778, 1983.

TIWAY Primitive Compatibility Specification. Texas Instruments Document No. 2457787, 1984.

Voelcker, John. "Helping Computers Communicate." IEEE Spectrum, 23, No. 3 (March 1986), 61-70.

VITA

Gary E. Underwood was born in Knoxville, Tennessee on November 16, 1956. He attended Bearden Elementary school and Bearden Junior High School in that city and was graduated from Bearden High School in June 1975. The following September he entered the University of Tennessee. In June 1979 he received a Bachelor of Science degree in Electrical Engineering.

In June 1979, he accepted a position in the college graduate training program with Caterpillar Tractor Company in Peoria, Illinois. He was promoted to research design engineer in the Research Controls division in May 1980. While at Caterpillar, he attended Bradley University and participated in graduate level electrical engineering courses.

In August 1983, he accepted a position at the Industrial Systems Division of Texas Instruments Incorporated in Johnson City, Tennessee. His position is a software systems engineer in the Systems Development branch.

In the fall of 1983 he enrolled in the University of Tennessee and began study toward a Master's degree. He received a Master of Science degree in Electrical Engineering in August 1986.

The author is a member of Phi Eta Sigma, Eta Kappa Nu, the Institute of Electrical and Electronic Engineers, and the IEEE Computer Society.