NASA Contractor Report 166071

DEVELOPMENT AND EVALUATION OF A FAULT-TOLERANT MULTIPROCESSOR (FTMP) COMPUTER
Volume I
FTMP Principles of Operation

T. Basil Smith, III and Jaynarayan H. Lala

THE CHARLES STARK DRAPER LABORATORY, INC.
555 Technology Square
Cambridge, Massachusetts 02139

CONTRACT NAS1-15336
MAY 1983

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665

FOREWORD

This report was authored by Dr. T. Basil Smith, III and Dr. Jaynarayan H. Lala. Dr. Smith was the project manager. The NASA technical monitor for the period from January 1982 to December 1982 was Mr. Charles Meissner and from August 1978 to December 1981 was Mr. Nicholas Murray. Some of the many people who contributed to the success of this project are as follows.

CSDL

Dr. Albert Hopkins
Mr. Jack Mckenna
Ms. Linda Alger
Mr. Kevin Koch
Mr. Robert Scott
Mr. Joseph Marino
Mr. David Hauger
Mr. Mario Santarelli

COLLINS AVIONICS

Mr. Ron Coffin
Mr. Charles Schulz

This Page Intentionally Left Blank

TABLE OF CONTENTS

	Page
1. Introduction	1
2. Overall FTMP Architecture	2
2.1 Processor Region	2
2.2 Slave Region	3
2.3 Clock Generation Region	5
2.4 Bus Guardian Units	6
2.5 System Bus Interface Circuits	7
2.6 Power System	7
2.7 Software Overview	8
3. System Bus	9
3.1 System Bus Design and Operation	10
3.2 LRU System Bus Interface Design	18
3.3 Bus Guardian Unit Design	21
4. Processor Region Design and Operation	23
4.1 The Processor Region Transfer Bus	25
4.2 Cache RAM	27
4.3 Cache PROM	27
4.4 Memory Management Unit (Mapper)	27
4.5 Interval Timer	29
4.6 Control and Communication Registers	30
4.7 System Bus Controller	33
4.8 The CAPS-6 Processor	41
5. Slave Region Design and Operation	98
5.1 Bus Coupler/ Transfer Bus	98
5.2 System RAM	102
5.3 Real Time Clock/Counter	102
5.4 Control, Communication and Status Registers	103
5.5 I/O Port	106
6. Clock Generation Region Design and Operation	110
7. Power System	116

CHAPTER 1. - Introduction

This report is Volume I of a multi-volume report on the Fault-Tolerant Multiprocessor (FTMP) project sponsored by the Langley Research Center of the National Aeronautics and Space Administration under contract NAS1-15336. The major topic covered by this volume is the FTMP architecture and principles of operation. Volume II describes the FTMP software, Volume III describes the FTMP test and evaluation results and Volume IV is an executive summary of the project.

This volume is intended to serve as a comprehensive guide to the hardware organization and operation of the Fault-Tolerant Multiprocessor engineering model. The FTMP engineering model was constructed by the Collins Avionics Division of Rockwell International Corporation to the architectural specifications provided by the Charles Stark Draper Laboratory. The architecture of this engineering model is similar to the machine architecture developed under a predecessor contract (NAS1-13782). Certain architectural modifications were made to this original baseline in order to facilitate the construction of this engineering model by Collins in a form which is compatible with current commercial avionics packaging and practice. It is believed that most of these modifications, in addition to the immediate goal of making the engineering model more practical, were desirable enhancements of the architecture in and of themselves. Chapter 2 summarizes the overall architecture of the FTMP and is intended to provide the necessary context for the hardware details of the later chapters. Chapter 3 discusses the System Bus design and operation. Chapter 4 discusses the Processor Region design and operation. Chapter 5 discusses the Slave Region design and operation. Chapter 6 discusses the Clock Generation Region design and operation. Chapter 7 summarizes the Power System design.

CHAPTER 2. - Overall FTMP Architecture

This chapter outlines the functional and design concepts of the Fault-Tolerant Multiprocessor engineering model, FTMP. The system is constructed of ten identical line replaceable units (LRU's). Each LRU contains a Processor region, a Slave region, a Clock Generation region, two Bus Guardian Units, System Bus Interface circuits, and a Power Subsystem.

2.1 Processor Region

The processor regions operate in groups of three called processor triads. Processor triads are formed by assigning the processor regions of any three LRU's to work together in tight synchronism. It is possible for up to three processor triads to be in operation simultaneously, utilizing nine of the ten available LRU's. The processor region of the tenth LRU serves as a spare. A processor triad functions as if it were a single processor executing a single instruction stream. With three triads operating simultaneously, three instruction streams are in parallel execution. The system is then functioning as a normal three-processor multiprocessor. The failure of a single processor region of a triad does not impact the correct execution of that instruction stream, because voting is used to mask the effects of the failure. Comparison techniques also enable the failed region to be detected and identified. A spare processor region can then be used to replace the failed element of the triad. If no spares are available, the damaged triad is retired from service with the surviving functional elements being used to replenish the spares pool.

The processor triads write data to, or read data from, locations within the system bus address space by means of the System Bus. This bus is a quintuply redundant fully duplex eight megabit per second serial bus. During a block transfer, data can be written from a processor triad at a peak rate of one word every 5 microseconds. Data can be read by a processor triad at a peak rate of one word every 3 microseconds. The error correction and error detection relies upon voting and comparison of data and addresses appearing on redundant elements of the System Bus.

At any one time three of the five redundant bus lines are active. These active lines are called a bus triad. Each element of a processor triad transmits data and addresses on a different

one of the bus triad's lines. Since the elements of the processor triad are all operating in tight synchronism, it is possible for any unit receiving a processor triad read or write request to compare the separate versions of that request by examining the separate copy of that request arriving on each bus of the bus triad. The receiving unit can correct any errors caused by a single processor region failure or single bus failure by using majority voting.

2.2 Slave Region

The Slave Region contains a number of subsystems all of which are addressed, read and written, as locations in the System Bus address space. These subsystems are the System Memory Module, the I/O Port, LRU Control/Status and Communications Registers, and a Real-Time Clock/Counter. Certain of these modules are normally operated within triads of three like modules assigned to the same function and location, others are operated as single units.

System Memory Modules are normally operated within a triad. The system memory modules of three LRU's are assigned to function together servicing a 16k word block of the system bus address space. Up to three memory triads can be formed from the system memory modules of nine of the ten LRU's. The tenth LRU's system memory module could then serve as a spare. Each memory triad is assigned to serve a different 16K block of the system memory address space. With three memory triads operating simultaneously, 48K words of the system memory address space can be served. The failure of a system memory module within a triad does not impact the integrity of data stored in that block, as voting is used to mask the effects of the failed module. Comparison techniques also enable the failed module to be detected and identified. A spare memory module can then be used to replace the failed element of the triad.

The real-time clock/counters of each LRU are also intended to operate together as a triad. All real-time clocks are addressed by the same system bus address. A processor triad write to that location sets all real-time clocks to the same value. The real-time clocks of three LRU's can be armed to respond to read requests. Only those three LRU's respond to any processor triad read requests and as such they function as the real-time clock triad. A failure of any element of that triad is masked by voting. Comparison techniques enable the faulty unit to be identified. Any one of the unarmed real-time clocks can be used to replace the failed element of the real-time clock triad. Note that even the unarmed real-time clocks respond to write commands from a processor triad, thus they will always agree with the elements of the real-time clock triad and can therefore be used to replace an element of the triad without reinitialization.

Unlike the processor regions, the memory modules, and the real-time clock/counters, the I/O ports operate independently of one another. Each I/O port responds to its own unique set of system bus addresses. Data and command words are transferred from a processor triad to an I/O port over the System Bus, appearing to the processor triad as routine system bus writes. As with any processor triad writes, voting at the receiving end serves to mask the failure of any one of the processor triad elements. The I/O port buffers any I/O transmissions, assembling an entire message before initiating an I/O bus transaction. The I/O port also buffers any incoming I/O transaction, assembling an entire remote terminal message. The entire transaction is then transferred as a block to a processor triad in response to a read request from that processor triad. The I/O port utilizes MIL-STD-1553A data bus protocols and signalling standards in its communications with the exterior. Two twisted shielded pairs are used, one for transmitting and one for reception, creating a fully duplex data link. If these two pairs are tied together they conform to all specifications of MIL-STD-1553A. A MIL-STD-1553A avionics data bus is a 1 MHZ serial data bus employing Manchester encoding to send both clocking and data information on a single shielded twisted pair bus line. Maximum I/O transaction length can be 32 data words, one command word and a status word requiring up to 700 microseconds for the transaction to be made. During this period the I/O port can act independently, and the processor triad may release the System Bus for regular bus traffic. Since each I/O port can operate independently, it is possible for the FTMP to be engaged in up to ten I/O bus transactions simultaneously, one on each of the I/O buses dedicated to each of the ten I/O ports.

The remaining elements of the Slave Region are System Control/Status and Communications Registers. These elements are used to control various parts of an LRU, to read the status of the error detection circuitry of an LRU, and to provide direct processor triad to processor triad communications.

The control registers are all write only. They are assigned fixed locations within the system bus address space depending upon LRU identification number. These LRU control registers control which bus lines the LRU uses for voting, triad assignment for the processor region, memory relocation factor for the system memory, whether the real-time clock is armed or not and other LRU assignments or functions.

The status registers, or error latches, can only be read by a processor triad. They report any bus errors observed by the error detection circuits of the LRU. Like the LRU control registers, the status registers are assigned fixed system bus addresses dependent upon their LRU identification number.

The communications registers are used to implement direct processor triad to processor triad communications. Each communications register can only be written using the system bus.

The communications register can be read by the processor region of the LRU directly, appearing as a local memory locations on its internal processor region data bus. The system bus address assignment, of each communication register within the LRU, is keyed to the processor region triad assignment of that LRU. This assignment is contained in one of the control registers of the LRU. The local processor region transfer bus address of each communications register is fixed and is the same for all LRU's.

Only one LRU responds to control register writes or status register reads, that LRU being determined by the system bus address of the register being accessed. All LRU's with the appropriate processor region triad assignment will respond to communication register writes. When multiple LRU's are responding to a communication register write they act in tight synchronism with one another.

Each LRU's Slave region is assigned to transmit on only one element of the redundant system bus. These assignments are made so that each element of a system memory triad or real-time clock/counter triad is assigned to a different bus. Each element of a processor triad therefore has simultaneous access to the redundant replies from each element of a responding triad. Each element of a processor triad can therefore mask a fault within a responding triad by appropriate majority voting circuitry. When reading from a simplex source, such as the I/O port or status register, the processor triad does not receive redundant information, but instead must accept the data from the single system bus line on which it appears and verify its accuracy by other means.

2.3 Clock Generation Region

All elements of the multiprocessor operate using a common time reference. This time base is provided by the Clock Generation Regions of four LRU's which are phase locked to one another. The Clock Generation Regions of the remaining LRU's are then phase locked to any three of elements of the clock quad. Each clock generator thus provides a timing source for its LRU which is in synchronism with all other correctly functioning generators. Such a system can tolerate the failure of any one of the clock generators. All correctly functioning clock generators remain synchronized despite such a failure. A failure within the quad is detected and identified and another clock generator can be assigned to replace the failed unit. Of the ten clock generators, four are assigned to the quad clock and the remaining six are either spare and in standby mode, or failed.

2.4 Bus Guardian Units

The overall integrity of the system relies upon the ability to reliably control the access that any element of an LRU has to the

system bus. Each LRU of the system has two Bus Guardian Units, BGU's, which function to protect the system bus from a failed or malperforming LRU or element within the LRU. Each BGU has bus enabling registers which control the LRU access to each individual line of the system bus. In order that the LRU be enabled to transmit on any line of the system bus it is necessary that the enabling bit from both BGU's within that LRU be set. Either BGU can block the LRU's ability to transmit on a line.

When the configuration control program creates a processor triad, for example, it must first assign the processor regions of three LRU's to the same processor triad. It does this by writing into the system control registers of the selected LRU's. It must then assign each processor region to transmit on separate lines of the system bus triad. It does this by writing to the enabling registers of both BGU's of each selected LRU, assigning each LRU to the appropriate processor transmit lines of the system bus.

The register loading mechanism of each BGU responds to a unique system bus address keyed to that BGU's LRU and location within the LRU. This register loading mechanism allows each BGU register to be written by a normal system bus write transaction. Because it is important that a BGU act independently of the LRU in which it is located, each BGU receives all the redundant copies of all processor triad transmissions appearing on the system bus. It decodes the system bus address of each write transaction, recognizing any writes addressed to itself. The data word of a write is then examined and enabling register contents are altered in response to a correctly formatted command contained in the data word. Since this requires that the BGU be aware of the current three of five bus lines which are the active triad, each BGU also has one internal register which is used to store the select code which determines which System Bus processor lines are to be used in the voting process. This processor line select register is also written by a processor triad using the same mechanism as is used to write to the enabling registers. All registers within the BGU are nonvolatile so that the bus assignments and line select codes are remembered during a power failure.

2.5 System Bus Interface Circuits

The actual connections to the system bus are made by the System Bus Interface Circuits within an LRU. These circuits perform several functions.

First, and most obviously, they provide the necessary drivers and receivers for each individual bus line. Selection of these drivers and receivers is dictated by electrical constraints such as adequate power and noise immunity.

Secondly, the interface circuits provide the means for controlling the fault environment of the system bus. The driver

circuit for each line performs the necessary gating of LRU signals onto system bus lines depending upon the values obtained from the BGU enabling registers. Thus it is this driver circuit which actually functions to cut the connection between an LRU and the system bus. The receiver circuits distribute independent copies of the system bus data, as required to each BGU within the LRU and to the other LRU circuitry. Each BGU and the input processing circuitry of the slave region must receive independent copies of the processor transmissions lines so that they each can each perform an independent vote and act independently of one another in response to processor triad commands. Partitioning within the interface circuitry region is designed so that a single fault can bring down no more than one element of the system bus or pollute no more than one copy of the received data from the system bus. As will be clearer later, the partitioning and fault containment aspects of these interface circuits are critical to the overall fault tolerance of the FTMP.

2.6 Power System

Each LRU has its own power subsystem. This subsystem consists of a power supply, which provides all the required voltages used within the LRU, and a battery backup circuit, which provides low power battery power for maintaining the CMOS system memory and the nonvolatile registers during primary power loss. This local power subsystem is overvoltage and overcurrent protected.

The local power subsystem of each LRU draws power from four 28 VDC power buses. Each LRU is fused at its connection point to this quad redundant power bus so that internal shorts within an LRU can at most only momentarily disrupt power on the primary power buses. The local power supplies have adequate energy storage to tolerate these interruptions while the fuses blow. The local power supplies draw power evenly from all of the power buses. Thus under most circumstances, each of the power buses is fairly equally loaded. Any one of the power buses is capable of fully supplying all power to operate the entire FTMP.

The four primary power buses are driven from four independent primary power supplies. In this particular implementation, each of these power supplies is identical and converts three phase 208 VAC, 400 Hz input power to 28 VDC.

2.7 Software Overview

Figure 2.1 illustrates the functioning of this system from a software or programmer's viewpoint. Three processor triads function as the logical equivalent of three simple processors with shared access to a single shared memory. Similarly, the processor triads share access to the ten I/O ports, a real-time clock/counter, and control/status registers. Each processor triad can directly write to the communications registers of the other processor triads using the system bus. A processor can read its own communications registers directly using its own internal transfer bus. The entire system is synchronized by the equivalent of a single system wide clock. The actual redundancy underlying the buses, system memory triads, processor triads, the clock quad and the I/O system is invisible to the programmer.

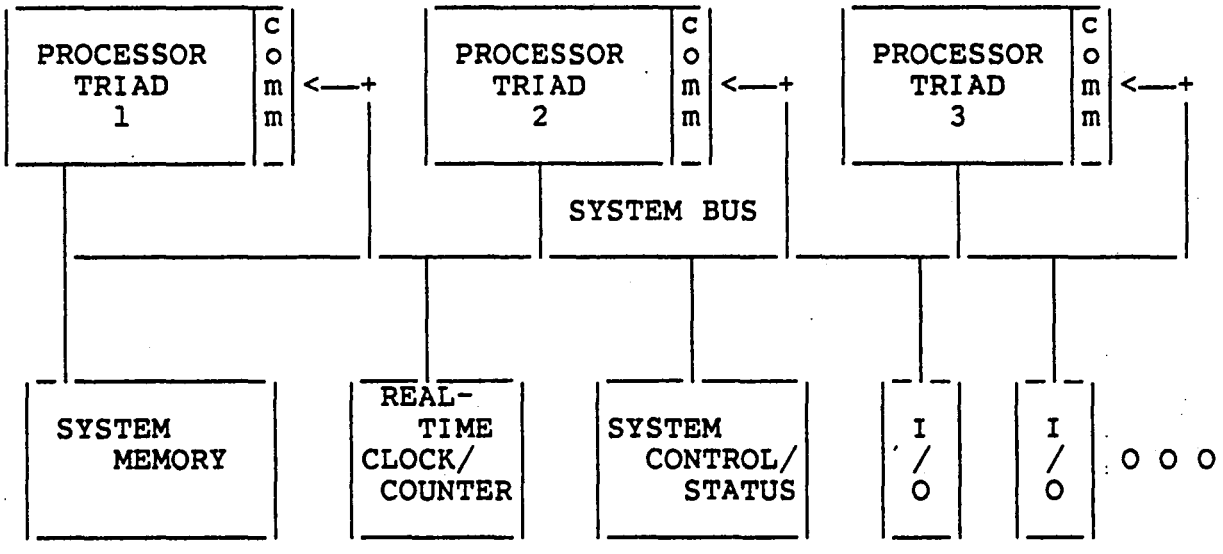


Figure 2.1 Software Appearance of FTMP

CHAPTER 3. - The System Bus

The System Bus interconnects all LRU's of the Fault-Tolerant Multiprocessor and is the transfer medium for exchanging all data, clocking, control and status signals. The bus system is five fold redundant, consisting of five identical bus sets. Each bus set is composed of four lines. These are a Poll line, P line, a processor Transmit line, T line, a processor Receive line, R line, and a Clock line, C line. Each LRU of the system is interfaced such that it always receives all bus lines. In addition it can be dynamically configured such that it may transmit on any bus line. The design of the interface circuitry of the LRU is such that any single point fault within an LRU can at most disable only one of the bus sets. Bus Guardian Units, BGU's, within each LRU are responsible for providing enabling signals to the interface circuitry which allow that LRU to transmit on particular bus lines. Each LRU has two BGU's. In order that the LRU be enabled to transmit on a bus line it is necessary that the interface circuitry receive enabling signals from both BGU's of that LRU. Thus the LRU can still be disconnected from the bus (blocked from transmitting) even if one of the BGU's should fail such that it is providing enabling signals to the interface circuitry.

The basic operating principles of the FTMP require that triads of processors and memories be formed and that each member of a triad be assigned to transmit on a different bus set. The triads operate in tight synchronism so that the transmissions from each element of a triad will be synchronized with one another. It is thus possible to listen to the three separate bus lines on which a triad is transmitting, and to perform a majority vote to synthesize a correct transmission even if one element of the triad should fail. It is also possible to note and record any disagreements between elements of a triad which might become apparent during this voting process. Processor triad, system memory triad and real-time clock/counter transmissions are always of this triplex form. The receiving party is responsible for performing the vote and error masking function, should one of these triads have a failed member.

Synchronous operation requires that all LRU's operate with a common time base. This common time base is provided by the clocking system. Clock generators within each LRU lock to a voted version of the redundant system clocks which appear on the system bus clock lines. These redundant clock signals are

provided by gating the clock generator outputs of selected LRU's onto the C lines of the system bus. The selected LRU's effectively lock to each other while all other LRU's lock to the selected LRU's.

This chapter discusses the System Bus design and operation, the LRU System Bus Interface design, and the Bus Guardian Unit design and operation.

3.1 System Bus Design and Operation

The System Bus interconnects all LRU's of the FTMP providing the timing, control and data paths for all intermodule communication and synchronization. The basic construction provides five fold redundancy. There are five complete and independent bus sets. Each bus set is made up of a poll line, P line, a processor Transmit line, T line, a processor Receive line, R line, and a Clock line, C line. Each of these logical bus lines is a single twisted pair of wires. One of these wires serves as the signal wire, the other is grounded. The bus wires interconnect the LRU's in a multidrop fashion with each LRU attaching to the bus by means of a short stub to the LRU interface circuitry. LRU slots 00 and 11 are at opposite ends of the bus with the intervening slots being fairly uniformly spread along the bus. Each end of the bus is terminated with the characteristic impedance of the transmission line so that signal reflections from the ends do not occur. Each bus operates as a wired 'OR'. If there are simultaneous transmissions from several LRU's onto one bus line, then the result is the logical 'OR' of the multiple transmission signals. The transmission line termination is such that the undriven state (when no transmissions are in progress) is a logical 'zero'. The overall length of the bus is roughly three meters, propagation time from one end of the bus to the other being about 20 nsec. The basic signal bandwidth of the bus is in excess of 20 MHz.

The signal formats and protocols for each of the bus lines is described in the following subsections.

3.1.1 The Clock Buses.

Each of the bus sets contains a clock line. These lines are used to distribute redundant copies of a common system wide time base, called the system clock. At any one time either three or four of the C lines are active, the other line(s) being either failed or spare. A copy of the system clock appears as a 1 MHz. square wave on one of the active C lines. One period of this square wave is called a system epoch. The system epoch begins at the rising edge of the system clock. All unfailed and active C lines carry independent copies of the system clock. All of these copies are identical excepting for some small time skews between them which result from circuit variations and propagation delay

variations due to LRU location on the bus. This time skew is a small fraction of the epoch time.

An LRU receives all C lines, selects 3 of the 5 lines, performs a majority circuit reduction of those 3 signals to 1 and phase locks its own crystal oscillator to that one signal. Presumming that the LRU is configured to select 3 active C lines, this effectively creates within the LRU its own private copy of the system clock signal. Since all LRU's can be configured to lock to the active C lines, each LRU's crystal oscillator can in effect be locked to the same common time base.

The LRU's themselves are used as the source of the individual system clock signals. The system is either configured such that four LRU's have their internal clock signal gated out onto separate C lines and each of these LRU's selects and locks to the clock signals of the three other LRU, or it is configured such that only three LRU's have their internal clock signal gated onto separate C lines and each of these LRU's selects and locks to the clock signals of the other two LRU's and itself. The first configuration allows all correctly functioning LRU's to remain synchronized with one another in the presence of any single fault in the clock system. LRU's which are not system clock sources can select and lock to any three of the four active C lines. The second configuration is nearly as good excepting for certain pathological single point failures which could induce lose of synchronization. In this case since there are only three active C lines, every LRU selects and locks to the same three signals. The likelihood that a single point failure is one of the pathological cases which could cause lose of synchronization is very remote.

Should an LRU system clock source fail it is possible to detect that failure, disconnect that LRU from its C line and to connect another LRU to the same line. Since there are ten LRU's within the system a large number of this type of failures could be tolerated without degradation of the clocking system. If one of the C lines itself should fail it can be replaced by an inactive but functional C line from another bus set. Initially, one such failure can be tolerated and still maintain the quad clock source configuration. A second such failure necessitates reversion to the triplex clock source configuration.

Note, that even though the C lines are wired 'OR' and no physical damage occurs if two LRU's are gated onto the same line, multiple transmissions onto the same line can considerably distort the square wave characteristic of the resultant system clock on that line. In the worst case the skews between the LRU's can produce a resultant clock signal which is always 'one'. This could frustrate the system's ability to obtain and maintain synchronization. If multiple clock sources are gated onto more than one of the active clock lines the operation of the clock system is indeterminate. Multiple clock sources may be gated onto one of the active clock lines and the clock system will

continue to function correctly as long as no clock faults or bus faults occur which disable any of the other active clock lines.

3.1.2 The Poll Bus

Each of the bus sets contains a poll line. These lines are used to arbitrate among the processor triads seeking control of the system bus. At any one time three of the five P lines are active the other two lines are failed or spare. The data rate on the P line is 1 Mbit/sec. Transmissions onto the P lines are synchronous with the system clock. Transmission format is NRZ (Non Return to Zero). A transmitting processor's system bus controller drives the P line to the correct bit value at the beginning of the epoch and holds that value until the next epoch. The P line bit value is read at the epoch midpoint. The time skews between LRU's is small enough to assure that the P line signal values will be correct and stable at all LRU's at the local timebase epoch midpoint.

The resultant value read from the three P lines is the majority function of the three values sampled at the epoch midpoint. Each element of a processor triad is assigned to a different one of the three active P lines. Processor triads use the P lines in allocating control of the system bus. A processor triad participates in a competitive poll when it seeks control of the system bus. The winner of the poll obtains control of the bus. During a poll simultaneous transmissions onto the P lines by multiple triads may occur. The signal value on any one line is the wired 'OR' of the multiple transmissions onto that line during that bit period. The resultant value read from the P lines during any bit period is then the majority function of these wired 'OR' signals.

A triad recognizes that the system bus is free if the P bus is 'zero' for four consecutive bit periods. If a triad seeks control of the bus it then initiates a poll. The basic format of the bus poll is shown in Figure 3.1. The first bit of the poll is a 'one'. The next three bits are the most significant bits of the processor poll number. The fifth bit is a 'one' which prevents an inadvertent series of four 'zero's. The sixth thru eighth bits are the next three most significant bits of the processor poll number. Again this is followed by a 'one' to prevent an inadvertent series of four 'zero's. The tenth through twelfth bits of the poll are the least significant bits of the poll number. If at any time during this poll, the processor triad reads back a poll bit which differs from its transmitted bit, it immediately drops out of the poll. The triad with the highest poll number is the only triad that survives to the end of the poll sequence. This triad gains control of the system bus. It retains control by continually transmitting 'one's onto the P bus. When it wishes to release control, it simply stops transmitting. This will result in 'zero's on the P bus. After four bit periods the bus is recognized as free. The bus then

remains free until another triad(s) initiates the next poll. The functioning of the poll is relatively simple when all processor triads are synchronized. It is not as simple when processor synchronization has not been achieved. The reader is referred to Section 4.7 on the processor system bus controller for a more detailed description of the polling complexities and the dynamics of the interactions among the system bus controllers during polling sequences.

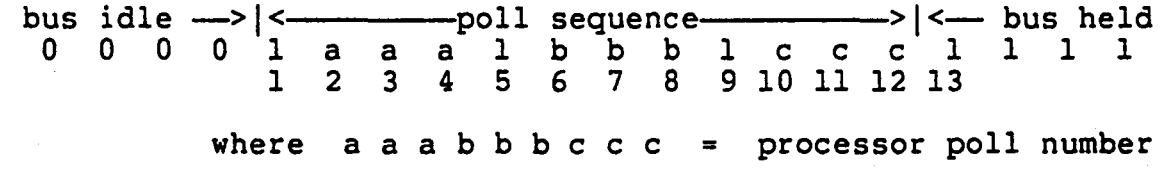


Figure 3.1 Basic Polling Sequence Format

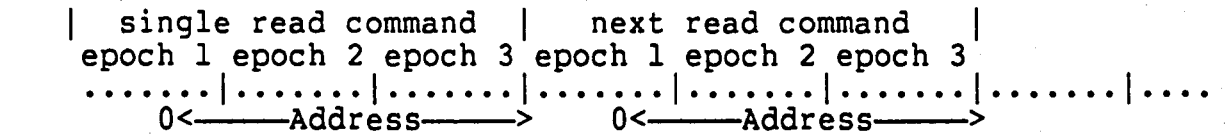
3.1.3 The Processor Transmit Bus

Each of the system bus sets contains one processor Transmit line, a T line. The T lines are used in transmitting processor triad read and write commands. During a processor triad write the T lines carry both the system bus address and the data word. During a processor triad read the T lines carry only the system bus address.

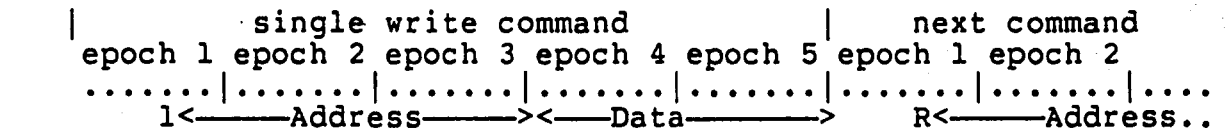
Each element of a processor triad is assigned to transmit on a different T line. All elements of a processor triad transmit their read or write commands in tight synchronism with one another. Processor commands are transmitted serially onto the T bus at an 8 Mbit/sec. rate. Figure 3.2 illustrates the format for both processor reads and writes. It is impossible for the separate elements of a processor triad to maintain synchronous operation to the extent that the skews between them would be less than one half of one of the T bus bit periods (62.5 nsec.) The time skew between processors of a single triad can in fact be several such bit periods. Because of this encoding of the transmitted bit streams at the source and resynchronization of the separate incoming bit streams at the receivers is required. The transmitter does not begin a transmission until the middle of an epoch. Since the synchronization of the processors is to better than a fraction of an epoch, any receiving element cannot mistake the epoch in which the transmission began. FIFO's within receiving circuitry provide adequate buffering so as to allow a

half epoch delay from the point when the second of the triplex transmissions begins to arrive at the receiver, to the point where the FIFO's begin to be unloaded. All of the FIFO's are then unloaded synchronously with the receiving LRU's internal system clock. In effect the receiving element merely decides in which epoch a triplex transmission began, delays a half epoch, and starts accepting the received data bit by bit synchronously with its own internal view of the epoch alignment. Skew is never large enough to cause the beginning of transmission from the separate elements of a triad to appear in different epochs. The FIFO's hold the variable number of bits required in adjusting for the time skew between the transmitting processor system bus controller's epoch and the receiving units epoch. Since the same bit of the processor command is unloaded simultaneously from all three incoming channels, a simple voting circuit is adequate to perform error correction should one of the processor elements of a triad fail.

This encoding and resynchronization requires that the serial bit streams from each source contain clock and transaction synchronization information. The beginning of a transaction synchronization is provided by a half epoch (the first half of the epoch) of null transmission. The address and data bits are then transmitted as a series of pulse width modulated pulses. The pulses are used to carry the clocking information, the width of the pulses carries the data. A 'one' is encoded as a long pulse on the T line, a 'zero' is encoded as a short pulse. Within the 125 nsec. bit period of the T bus a 'one' is encoded by transmitting an 83 nsec 'one' followed by a 42 nsec 'zero'. A 'zero' is encoded as a 42 nsec 'one' followed by a 83 nsec. 'zero'. Figure 3.3 illustrates this encoding.



System Read Command Format



System Write Command Format

R = 0 read command
= 1 write command
Address transmitted most significant bit first
Data transmitted most significant bit first

Commands may be tightly packed or idle epochs may exist between them. Reads and Writes may be intermixed.

Figure 3.2 System Bus Read/Write Command Formats

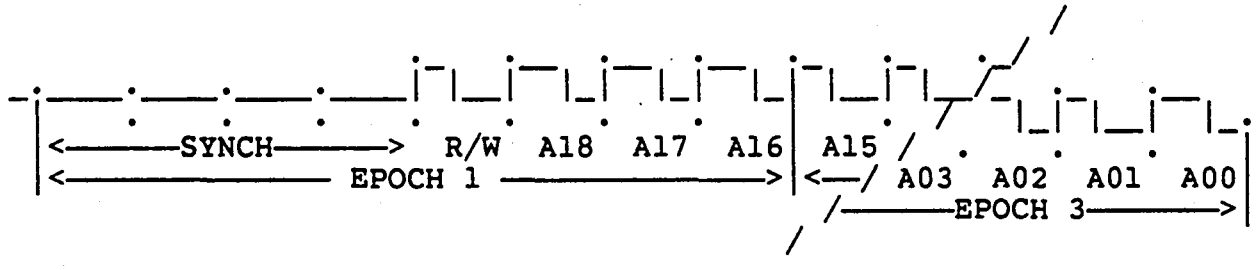


Figure 3.3 T Bus Signal Encoding Format

3.1.4 The Processor Receive Bus

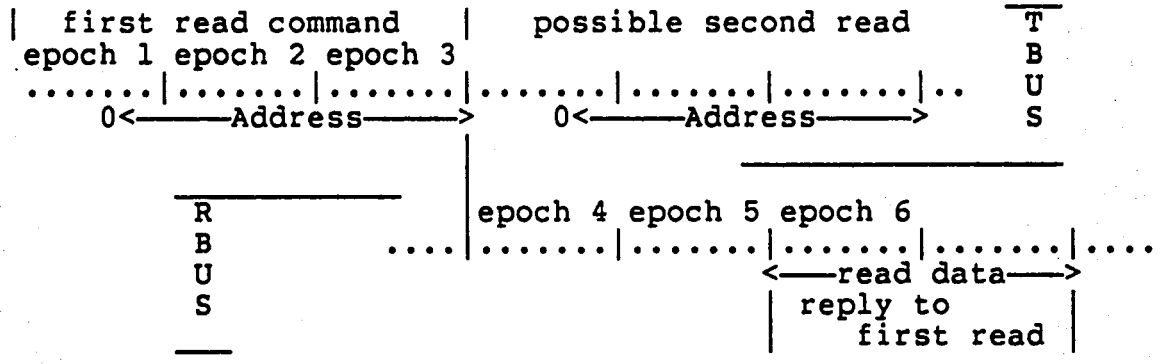
Each of the system bus sets contains one processor Receive line, an R line. The R lines are used in receiving slave region replies to processor triad system bus read commands. During a processor triad read the R lines carry the data word being transmitted by the responding slave region(s).

At any one time three of the five R lines are active; the other two are either failed or spare. This set of three lines is called the R line triad, or simply the R bus. Each slave region is assigned to transmit on one of the active R lines. These assignments are made such that each element of a system memory triad is assigned to transmit on a different R line. Processor commands are transmitted serially onto the T bus at an 8 Mbit/sec. rate. Slave region replies are locked to the processor read command, following in epochs six and seven of a processor read transaction. Note that in a series of tightly packed read commands the T bus transmissions of the next transaction overlap the R bus transmissions of the current transaction. Figure 3.4 illustrates the format for both processor read commands and the interlocking timing of the slave response.

A single slave region may reply to a read command, as is the case when reading I/O port registers or LRU status registers. In such cases the receiving processor triad must accept data from one of the lines of the active R lines. Voting among redundant copies of the same transmission is not done in this case, but instead this single copy is accepted as is. In this case, synchronization among the incoming bits of multiple copies is not necessary. This is not the only form of a read transaction, however. Three slave regions may reply to a read command, as is the case when reading system memory or the real-time clock/counter. In these cases the receiving processors must accept the redundant copies of the incoming data from the R bus, synchronize the multiple bit streams and vote so as to mask any single element errors. This requires that many of the same techniques used in the encoding and resynchronizing of T bus transmissions must also be used for the R bus. It is impossible for the separate elements of a slave triad to maintain synchronous operation to the extent that the skews between them would be less than one half of one of the R bus bit periods (62.5 nsec.) The time skew between slaves of a single triad can in fact be several such bit periods. Because of this encoding of the transmitted bit streams at the source resynchronization of the separate incoming bit streams at the receivers is required. The slave begins transmission of the requested data at the beginning of epoch 6 of the read transaction. Since the synchronization is to better than a fraction of an epoch, all slave regions begin transmission of the reply during the same epoch. FIFO's within receiving processors permit each processor to delay until the middle of epoch six before beginning to unload the incoming bits. All of the FIFO's are then unloaded synchronously with the receiving LRU's internal system clock. In effect the receiving processors delay until they are confident that their FIFOs contain data and then start accepting the received data bit by bit synchronously with their own internal view of the epoch alignment. The FIFO's hold the variable number of bits required in adjusting for the time skew between the transmitting slaves and the receiving processors. Since, the same bit of the slave reply is unloaded simultaneously from all three incoming channels, a simple voting circuit is adequate to

perform error correction should one of the elements of a triad fail.

This encoding and resynchronization requires that the serial bit streams from each source contain clock information. The data bits are transmitted as a series of pulse width modulated pulses. The pulses are used to carry the clocking information, the width of the pulses carries the data. A 'one' is encoded as a long pulse on the R line, a 'zero' is encoded as a short pulse. Within the 125 nsec bit period of the R bus a 'one' is encoded by transmitting an 83 nsec 'one' followed by a 42 nsec 'zero'. A 'zero' is encoded as a 42 nsec 'one' followed by a 83 nsec 'zero'. Figure 3.5 illustrates this encoding of the R bus reply. In this example, fragments of the wave form of two tightly packed read replies are shown. Note that each of the replies in a series of tightly packed responses is separated by an epoch of null activity on the R bus. This results from the difference in reply transmission time, two epochs, and read command transmission time, three epochs. Read commands can be packed no more tightly than one every three epochs, thus the replies can be no more tightly packed than one every three epochs.



Address transmitted most significant bit first
Data transmitted most significant bit first

Commands may be tightly packed or idle epochs may exist between them. Reads and Writes may be intermixed.

Figure 3.4 System Bus Read Format

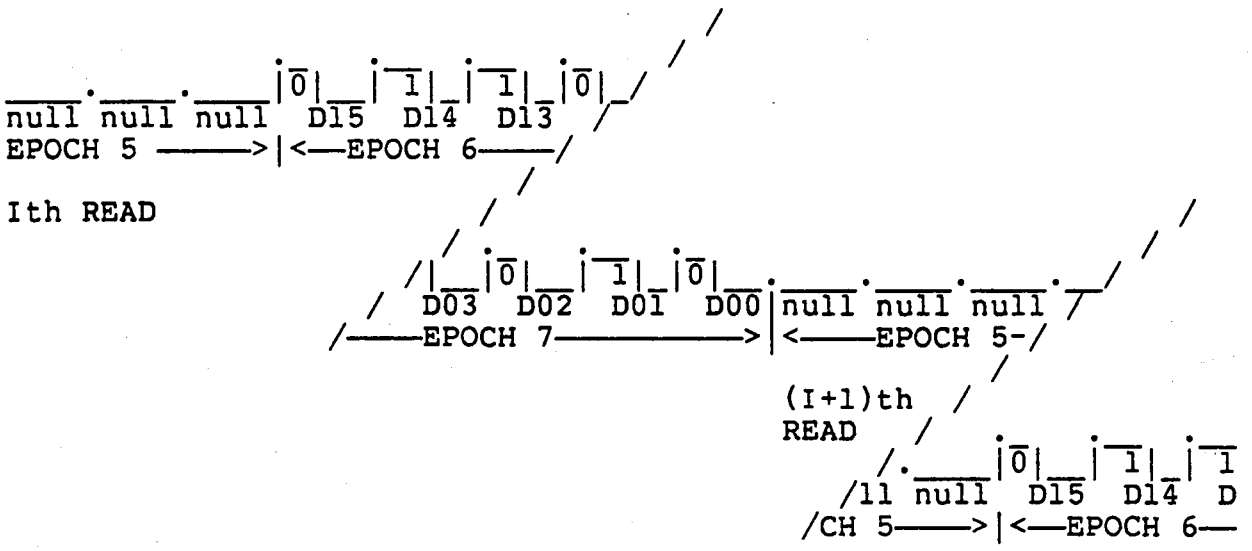


Figure 3.5 R Bus Signal Encoding Format (Signal Fragment)

3.2 LRU System Bus Interface Design

Each LRU is interfaced to the system bus by means of System Bus interface circuitry. The design of this circuitry is critical from both a reliability and performance viewpoint. The design must be such that any single failure within the interface circuitry does not disable the bus system. Additionally, the communication data rates and signal modulation bandwidths are high enough to place rather rigid constraints on the electrical performance of all interface circuitry.

Each bus set of the system bus is interfaced to the LRU by means of an interface circuit. Individual interface circuits are isolated from one another such that a single point failure can effect at most one of these circuits. A single point failure can therefore at most effect only one bus. These circuits listen to or receive the signal present on each line of the bus set. There are four receivers in each interface circuit, one for each line of the bus set, the P, T, R, and C lines.

The T line is then buffered so as to create three copies. One copy of the T line is then run to each BGU and one copy is run to the slave region bus coupler. The T line is buffered in this fashion so that if a short should occur on the T line input to either of the BGU's or on the T line input to the slave region bus coupler, then that element is the only T line destination affected. A short at the input to one BGU for example does not impact the integrity of the T line signals being received at the input to the other BGU or the slave region coupler.

The P and R signal values are run to the processor region system bus controller. The C signal is run to the clock generation region. Again as with the T line signals, design of the buffering is such that any shorts on the outputs of the line receivers will not propagate back onto the system bus itself, or to another output of the interface circuit.

Each interface circuit also contains a driver circuit for transmitting onto each line of the system bus set. A driver circuit must be enabled by a dedicated enabling line from each BGU before transmission onto a bus line is enabled. The driver circuit for the P and T lines of the bus set, when enabled, is driven by the P and T outputs of the processor region system bus controller. The driver circuit for the R line is driven by the R output of the slave system bus coupler. The driver circuit for the C line is driven by the C output of the clock generator.

The P, T, R, and C signals provided to a bus interface circuit are individually buffered copies of these outputs. This buffering is done at the signal source so that any shorts on these inputs to a bus interface circuit do not affect the integrity of the inputs to the other four interface circuits of the LRU. For example, the clock signal C is buffered so as to create five copies of this signal, one copy of which is distributed to each of the five bus interface circuits. Figure 3.6 illustrates the consumption of these signals by a single bus interface circuit. If a failure of one of these interfaces should short its inputs to ground, the other interfaces would still receive a valid C signal.

Each of the interface circuits is fault isolated from all others so that any failure within such a circuit cannot affect any of the other interface circuits or the buses to which the other interfaces are attached, and the buffering is such that any fault cannot physically propagate across these fault isolation boundaries. In the worst case, a single interface circuit can at most disable only one bus set of the five.

Figure 3.6 illustrates the logical organization of a bus interface circuit.

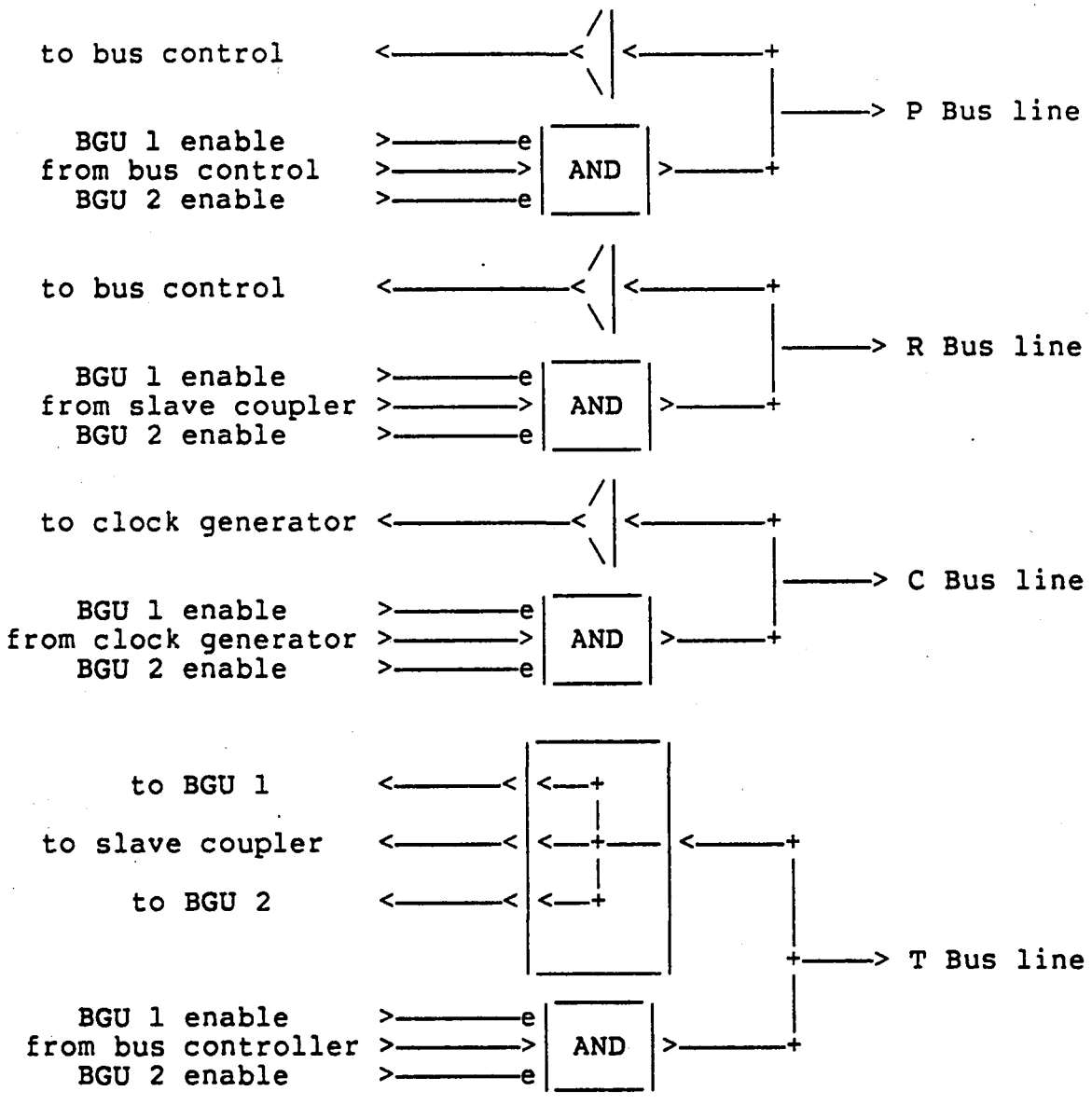


Figure 3.6 Single Bus Interface Circuit.

Electrically, each of the transceivers are of identical design. Signal formatting, encoding, decoding, etc. are not performed within this circuitry, but are done at the ultimate source or destination of the signal. As such they are just high speed buffering circuits. The 8 Mbit/sec data rates of the T and R bus lines place the most constraining specifications for buffer performance on the circuit design.

3.3 Bus Guardian Unit Design and Operation

Each LRU contains two Bus Guardian Units, which function together and with the system bus interface circuits, to protect the system bus from LRU faults and failures.

Each BGU provides, as its sole output, 20 enabling lines. Each enabling line runs to the driver of a bus interface circuit where it is used as an enabling line by that driver. The driver must have an enabling signal from both BGU's of the LRU before it will allow transmission by the LRU onto the associated bus line. Thus either BGU can, by asserting a disable on the appropriate enabling line, block LRU transmissions onto a bus line. Both BGU's operating together are required to enable transmissions onto a bus line.

The status of the enabling lines from a BGU is held by that BGU in enabling registers. There are four enabling registers, each of five bits. Enabling register 0 contains the enabling bits for each of the five P bus lines on which an LRU might transmit. Enabling register 1 contains the enabling bits for each of the five R bus lines. Enabling register 2 contains the enabling bits for each of the five T bus lines. Enabling register 3 constrains the enabling bits for each of the five C bus lines. Bit 0 of each of these four registers corresponds to the enabling bits for all lines, P, R, T, and C, of bus set 0. Bit 1 corresponds to the enabling bits for bus set 1, bit 2 for bus set 2, etc. A one stored in a particular bit position provides an enabling signal to the bus interface. A zero provides a disabling signal. The register contents are backed up by LRU battery power should there be a primary power failure. Thus their contents are nonvolatile during power interruptions. While primary power is down the BGU's provide disabling signals to the interface circuitry. Should battery power fail (or be turned off) while primary power is off, the register contents are reset to zero.

The register contents of a BGU may be altered by any processor triad by means of an appropriately constructed system bus write transaction. All five copies of the T bus are provided to each BGU by the bus interface circuitry. The BGU selects three of these five signals, processes them through the necessary deskewing circuits, voter, serial to parallel converter, and address recognition circuits. Each BGU of the system responds to one system bus address. Processor triad system bus write commands which are addressed to that BGU's system bus address are acted upon. All read commands and write commands to other than that BGU's unique address are ignored. The data word which is written to a BGU system bus address is interpreted as a command. Bits D10, D09, and D08 of the data word are used to select a BGU register. The least significant bits of the data word are taken as the new content to be stored into that register.

In addition to the first four registers, which contain the enabling bits for the bus lines, there is a fifth register, of four bits, which is used to hold the five to three select code to be used by the BGU's T bus input select circuitry. A individual select code must be provided wherever redundant bus data is reduced and voted. In this case each BGU must select the three of five R buses which are active and then vote their content. The select code is used by the BGU to select which of the 5 T bus lines are to be used. All other BGU registers are spare and writes to them are ignored. Figure 3.7 illustrates these register functions and the system bus address format for writing to a BGU.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
X	X	X	X	X	0	0	0	X	X	X		p4	p3	p2	p1	p0
X	X	X	X	X	0	0	1	X	X	X		r4	r3	r2	r1	r0
X	X	X	X	X	0	1	0	X	X	X		t4	t3	t2	t1	t0
X	X	X	X	X	0	1	1	X	X	X		c4	c3	c2	c1	c0
X	X	X	X	X	1	0	0	X	X	X	X	S3	S2	S1	S0	

enables

select

BGU Command Word Formats

18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	1	1	1	1	a	a	a	a	1	1	1	s

aaaa = LRU identification number
s = BGU number within LRU

BGU System Bus Address Format

Figure 3.7 BGU Command and Address Formats

CHAPTER 4. - Processor Region Design and Operation

This chapter discusses the processor region design and operation. It is intended to provide all the information required by the software designer on the operation of the processor region hardware. Hardware detail is provided to a level required for designing and implementing the software, and to a level adequate for the understanding of failure modes and their effects. This chapter should also serve as an outline or guide in understanding the detailed hardware documentation and logic diagrams. It is not however intended to be the detailed hardware documentation manual, such as might be used in making or effecting repairs to the system. Such documentation is provided by Collins Avionics in the form of a data package to be delivered with the FTMP engineering model itself.

The processor region of the LRU consists of a CAPS-6 processor, a cache memory management unit, local cache memory in the form of both PROM and RAM modules, an interval timer, a communications and control register interface, and a system bus coupler. These elements are interconnected by the processor region transfer bus, a parallel 16 bit address and 16 bit data bus. Figure 4.1 illustrates the overall organization of this LRU region. The transfer bus of the processor region extends to a test connector at the front of the LRU. The CAPS-6 test adapter can be connected at this point and the activity of the transfer bus can then be remotely monitored or controlled. This chapter does not discuss the test adapter further and for purposes of this discussion the processor region is assumed to be operating without the test adapter attached.

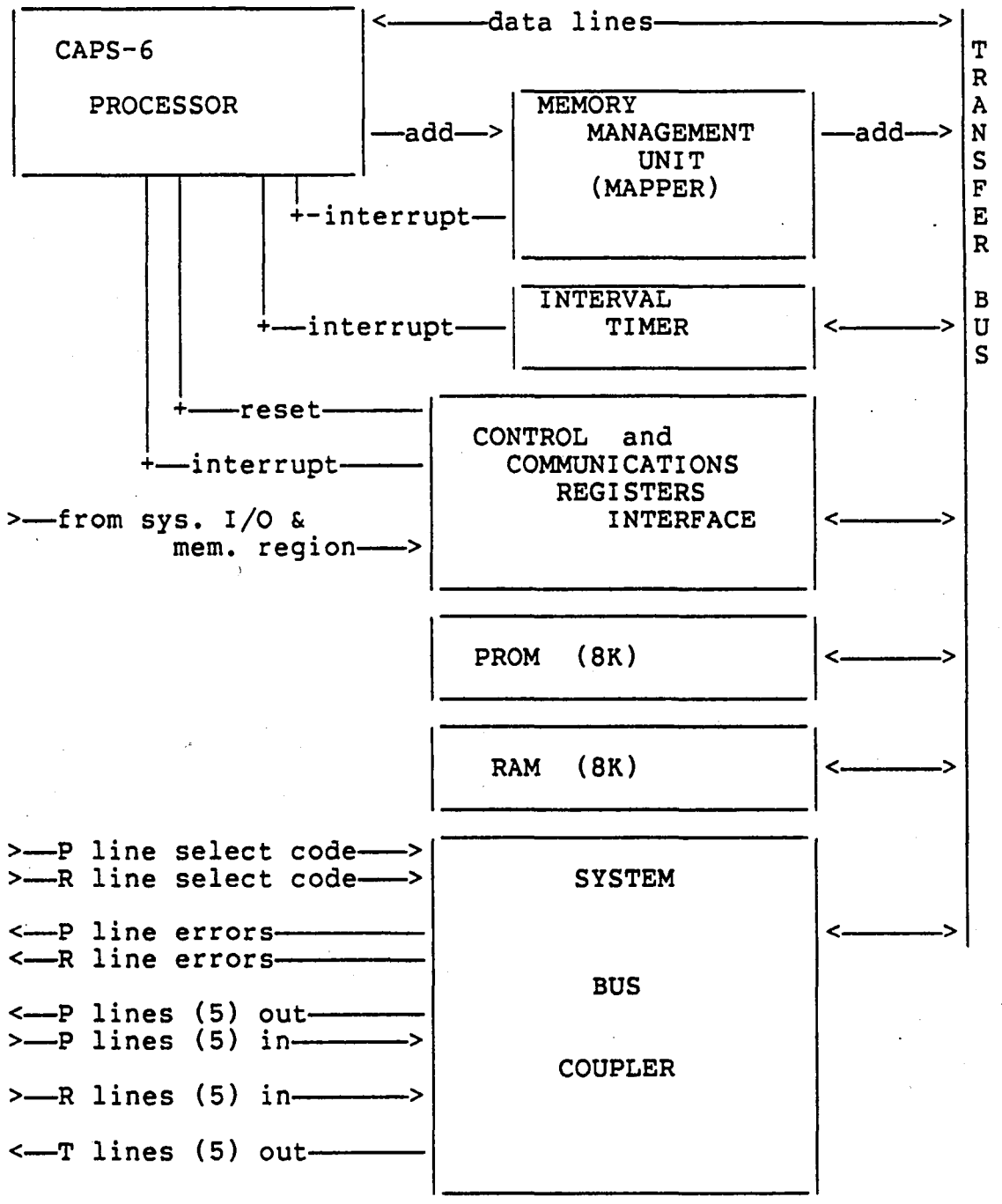


Figure 4.1 Processor Region Organization

4.1 The Processor Region Transfer Bus

Data can be transferred between two units of the processor region by means of the processor region transfer bus. During each transfer, the bus is controlled by the unit which initiated the data exchange. This unit is called the bus master. The responding unit is called the target. Only one unit at a time may be bus master. Control is allocated to units desiring control of the transfer bus by a hardwired arbitration circuit. The only units capable of initiating a transfer and of controlling the transfer bus are the processor and the system bus coupler. Once it gains control, the bus master selects a target device for a transfer by outputting a transfer bus address and asserting appropriate control signals so as to effect a read (transfer of data to the master) or write (transfer of data from the master). Each potential target device responds to fixed transfer bus addresses. The master unit then transfers data from the target device or to the target device. A single target device may respond to a number of transfer bus addresses with each address referring to a different information source or destination within the device. For example, the RAM memory unit responds to 8192 transfer bus addresses corresponding to 8192 memory locations. Each of the memory locations can be either individually read or written by any bus master. Certain target addresses are either read only or write only. Reads of write only or non-existent target addresses return "0000". Writes to read only or non-existent target addresses have no effect. As an example, the PROM memory is read only and responds to 8192 memory locations. These locations cannot be altered by transfer bus writes. It is also possible for otherwise unrelated functions to share the same address, as might be the case of a write only command register and a read only status register. In this case, a write to a particular location need not directly affect the information content which might subsequently be read from that location. Reading or writing to certain target addresses might also initiate peripheral actions beyond simple stores or fetches. A write to the command register of the system bus coupler can cause data blocks to be transferred from the processor region's local memory to system memory for example. Figure 4.2 summarizes the device location and function of all transfer bus addresses.

Transfer Bus Address	Device	Function
0000—>1FFF	PROM	Read only memory array
2000—>3FFF	RAM	CMOS read/write memory array
FE00—>FEFF	Mapper	Memory Management Page Table
FFE8—>FFEB	Control	Control registers (read only)
FFEC—>FFEF	Comm	IPC registers (read only)
FFF0	Timer	Interval timer register
FFF1—>FFF7	SBC	System bus controller command/status registers.
FFFC	Mapper	Map interrupt fault address
FFFD		Map clear command/status reg.
FFFE		Map enable/disable reg.
FFFF	Rupt	Interrupt register

Figure 4.2 Processor Transfer Bus Address Assignments

These transfer bus address assignments are permanent and cannot be altered (except by means of wiring changes).

The processor when accessing the transfer bus often configures the cache memory management unit, the mapper, so that it transforms the address provided by the processor into a new address, which is the actual address used on the transfer bus during that read or write. The input address to the mapper is called the virtual address and the output address is the real address. The operation of the mapper can cause the apparent addresses of devices, registers or memory locations to change as viewed from the processor. The operation of the mapper and the apparent address alterations should not be confused with the fixed real addresses of all devices, registers and memory locations. When the operation of the mapper is disabled, the virtual address provided by the processor is used as the real address for a bus transaction. The operation of the mapper is discussed in further detail in Section 4.4.

4.2 Cache RAM

The cache RAM is an 8192 word CMOS memory that can be read or written by using the processor region transfer bus. The 16-bit transfer bus real addresses to which cache RAM responds are 2000 to 3FFF. After a power interruption or cold start the cache RAM memory contents are indeterminate.

4.3 Cache PROM

The cache PROM is an 8192 word semiconductor memory that can be read by using the processor region transfer bus. The 16-bit transfer bus real addresses to which cache PROM responds are "0000" to "1FFF". The PROM is made up of 2K X 8 programmable read only memory arrays (2716). These PROM chips can be erased with ultra-violet light and reprogrammed. It is not necessary to remove the chips from the cards to reprogram them. The PROM memory card must be removed from the LRU and inserted into a special programming fixture in the ground support equipment to reprogram it. The PROM is non-volatile and holds its contents until reprogrammed.

4.4 Memory Management Unit (Mapper)

The function of the memory management unit or the mapper is to map processor generated virtual addresses into real addresses. These real addresses are then used in place of the virtual addresses in processor controlled transfer bus transactions. The mapper mechanism uses a 256 word page table of 12-bit words. When in operation the mapper uses the high order byte of the virtual address to index into the 256 word page table, and obtains a replacement high order byte. The least significant byte of the page table entry replaces the high order byte of the virtual address creating a real address. Each entry in the page table effectively relocates or maps a 256 word page of virtual memory onto a 256 word page in real memory. Since it is not always possible to assure that the virtual page is present in memory and since it will also be necessary to write protect pages of the real memory store, two of the remaining four bits of each page table entry are used to indicate absence of the associated real page, and to write protect the page. Bit 09 of the page table entry should be set to 'one' to indicate that the page is present in real memory and should be set to 'zero' to indicate its absence. A processor reference to an absent page will cause a page fault interrupt. The instruction in progress is interrupted, the processor state is backed up to the pre-instruction state, and processor accepts the page fault interrupt. Bit 08 of the page table entry is used to write protect the associated page of real memory. If set to 'one', the page is write protected. A processor store to a write protected page while the processor is in user mode interrupts the instruction in progress, backs up the processor to the the pre-instruction state, and causes the processor to immediately accept

the write protect interrupt. A write protect bit is ignored when the processor is operating in the privileged state.

The map fault register latches and holds the virtual address whenever an instruction causes a page fault or write protect interrupt. The map fault register can be read by the appropriate interrupt handling routine from real address "FFFC". It is read only.

The mapper page table may be read by the processor if the processor is in privileged mode by reading any address that maps into real address "FE00" through "FEFF". The mapper page table may be written by the processor only if the mapper is turned off and the processor is in privileged mode by writing real addresses "FE00" through "FEFF". Location "FE00" corresponds to page "00", "FEFF" corresponds to page "FF". The mapper may be turned on or off (enabled or disabled) by writing a 'one' or a 'zero' in location FFFE, provided the processor is in privileged mode. Location FFFE may also be read to determine mapper status.

The processor reset microcode disables the mapper and initializes the processor to interrupt mode.

If the processor is in privileged mode and the mapper is disabled the page table can be quickly initialized by writing a 'one' into the least significant bit of real address "FFFD". This results in all 256 entries of the mapper being set to "0FE". The address "FFFD" may be read to determine when the initialization procedure has been completed. The least significant bit of the word will read 'one' as long as the procedure is in progress and will read 'zero' when it has been completed. The initialization procedure takes about 32 microseconds.

Figure 4.3 illustrates the overall functional organization of the Memory Management Unit. Note that the mapper only maps virtual addresses of the processor. Any other bus master deals exclusively with the real address space of the transfer bus. In controlling any DMA devices, such as the system bus controller, the programmer must explicitly translate addresses from the virtual address space, used by the processor, to the real address space of the transfer bus before setting up the DMA control registers.

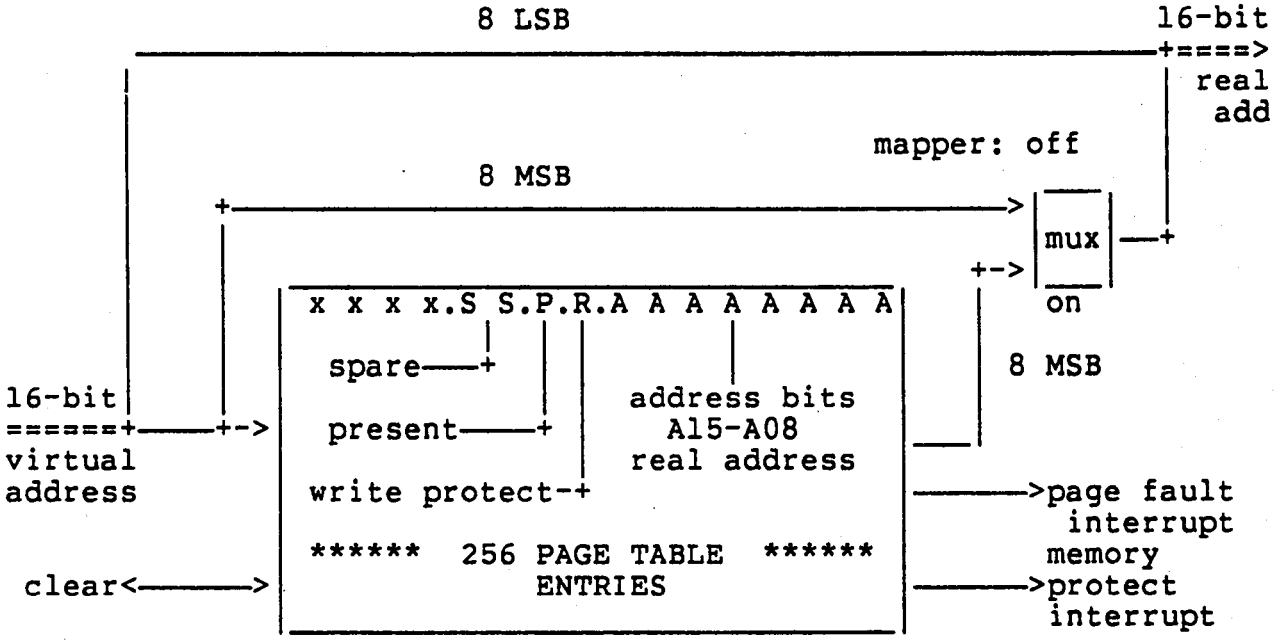


Figure 4.3 Memory Management Unit Organization

4.5 Interval Timer

The interval timer is a system clock driven 16 bit count down register and interrupt mechanism that can be used to accurately meter time intervals. The system clock is divided by 250, and the output of the divider network is used to decrement the timer count down register. The register may be loaded by writing to real address "FFF0". The intermediate states of the divider network are cleared whenever the register is written, thus the first decrementing occurs exactly 250 microseconds after the register has been loaded. The register may be read at any time without affecting its operation. A timer interrupt is requested whenever the register is decremented to zero. The timer interrupt is interrupt number C. The content of the timer register can be treated as a 16 bit positive number. It can therefore be set to request an interrupt at up to 16 seconds into the future. The timer is continually decremented and a timer interrupt is requested every 16 seconds. The timer interrupt may be disabled by masking interrupt 'C'. The timer interval can be reset at any time by a new store to the timer register. The timer register is volatile and is reset to zero after a power interruption.

4.6 Control and Communication Registers

There are sixteen control and communication registers within each LRU. Each of these registers can be written by means of a standard system bus write transaction to the appropriate system bus address. They are all write only from the system bus, they cannot be read by means of a system bus read transaction. However, eight of these registers can be directly read by the processor by means of the processor region transfer bus. These registers are read only from the transfer bus. The writing of these registers must therefore be done by a triad of processors (any triad can write any register), while the reading of the registers can only be done by the single processor of the LRU containing the register. Four of these eight registers are the CPU control registers. Four of them are inter-processor triad communication registers.

The four CPU control registers are each 4 bits wide, with the 4 bit wide bit field being right justified in the word. Certain of these registers in addition to being directly accessible to the processor, have hardwired control functions. Register 0 controls whether the processor is in the run or reset state and is called the RESET register. Bit 0 (B00) of that register is used to control the processor reset signal. When it is 'zero' the reset signal is asserted. When it is 'one' the reset is unasserted. The processor executes the standard microcode reset sequence when the reset signal is released. Control register 0, bits B03, B02, and B01, dictate the three bit triad identification code of the processor region of the LRU. This code is used by the poll sequence mechanism of the system bus controller and by the slave region's communication register address recognition mechanism. Registers 1, 2 and 3 can be read by the processor and have no hardware assigned function. Figure 4.4 illustrates the register system bus address (for writing), the processor transfer bus address (for reading) and the function of any hardware assigned bits of each register.

Reg	System Bus Address	Transfer Bus Add	Format
00 0	111,1111,1111,iiii,0000 7 F F I 0	1111,1111,1110,1000 F F E 8	t2 t1 t0 rr
01 1	111,1111,1111,iiii,0001 7 F F I 1	1111,1111,1110,1001 F F E 9	ua ua ua ua
10 2	111,1111,1111,iiii,0010 7 F F I 2	1111,1111,1110,1010 F F E A	ua ua ua ua
11 3	111,1111,1111,iiii,0011 7 F F I 3	1111,1111,1110,1011 F F E B	ua ua ua ua

iiii — LRU identification (binary)
I — LRU identification (hex)
rr — reset/run
t2,t1,t0 — triad identification
ua — unassigned

Figure 4.4 Processor Control Registers.

The four Inter-Processor triad Communication (IPC) registers provide the means for implementing direct processor triad to processor triad communications. Each of the IPC registers is four bits wide with the bit field being right justified in the word. An IPC register can be written by a system bus write transaction to the appropriate address just as the control registers can. Unlike the control registers, the system bus address of an IPC register is keyed to the processor triad identification code of the LRU (contained in processor control register 0) instead of by the LRU identification directly. It is therefore possible to simultaneously write to the communication registers of every processor with the same triad identification. In effect this provides the means of directly transmitting from one processor triad to another by means of system bus write transactions. Two of these IPC registers (2 and 3) generate an IPC interrupt ("B") when they are written into. Figure 4.5 summarizes the system bus addresses and transfer bus addresses for each of these registers.

Reg	System Bus Address	Transfer Bus Add	Interrupt
00 0	111,1111,1111,11tt,t000 7 F F - -	1111,1111,1110,1100 F F E C	no
01 1	111,1111,1111,11tt,t001 7 F F - -	1111,1111,1110,1101 F F E D	no
10 2	111,1111,1111,11tt,t010 7 F F - -	1111,1111,1110,1110 F F E E	yes "B"
11 3	111,1111,1111,11tt,t011 7 F F - -	1111,1111,1110,1111 F F E F	yes "B"

ttt — processor triad identification

Figure 4.5 Inter-Processor triad Communication (IPC) Registers

The CPU control registers as well as the IPC registers are provided with a battery back-up, and are therefore non-volatile. If the battery power is lost when the primary power is off the registers are reset to zero. A processor is initially held in the reset state when power is first applied after loss of both battery and primary power, as a consequence of control register 0 having been reset. This control register reset can be circumvented by means of a shorting plug which can be inserted into the front of an LRU. If a processor reads the reset/run bit and finds it set to zero then the shorting plug must be inplace on that LRU.

4.7 System Bus Controller

The System Bus Controller is designed to transfer blocks of from 1 to 256 words between the local processor region memory and system memory. It also serves as a synchronizing mechanism, whereby the operation of two or more processor regions can be brought into synchronism.

A single transfer of a block of data of from 1 to 256 words is called a system bus transaction. A transaction may either be a system memory write: data is moved from the processor region memory to system memory, or a transaction may be a system memory read: data is moved from system memory into the local processor region memory. It is necessary that the system bus controller perform a number of sequential operations in order to effect either a read or a write transaction. First the controller must gain control of the system bus. It then retains control of the bus while it effects the desired data transfer. Finally, it releases the system bus. Once a processor triad has gained control of the system bus it retains control until it voluntarily releases the bus. The controller may be instructed to hold the bus between transactions, allowing the processor to string together a number of transactions. In this mode of operation the controller need only obtain control at the beginning of the compound transaction and release it at the end. Alternatively, the controller may be instructed to obtain control of the system bus, perform a single transaction, and immediately release the bus. This is called a simple transaction.

Arbitration for control of the bus is provided by having each of the contending processor triads participate in a competitive cooperative poll each time they desire control. The processor triad with the highest competitive poll number is the processor which gains control of the bus. The poll number is a nine bit code consisting of three, three-bit subfields. The three most significant bits are the static priority code, SPC. The next three-bit subfield is the dynamic priority code, DPC. The least significant three bit field is the unique triad identification code, TID, of the requesting processor triad. The static priority code is set by the processor and is used by the processor to give its request for bus service priority over less urgent requests. If a conflict occurs between two processor triads, each seeking control of the bus at the same time, then control always passes to the triad with the highest static priority. If multiple triads are seeking control, all of which are using the same static priority code, then control will pass to the controller triad with the highest dynamic priority code. When a system bus controller initiates the effort to obtain control of the system bus it initially sets the dynamic priority code to zero. It then increments this code by one each time it loses a polling sequence to another controller of equal static priority. This effectively boosts the priority of the requesting controller by a factor related to that controller's waiting time for the bus. A triad is thus assured that it cannot be

repeatedly beaten by another triad of equal static priority. Finally, if both the static and dynamic priority of the competing triads are equal the poll sequence is decided by the triad identification. The bus is granted to the unit with highest triad identification. Since the triad identification is unique to a triad this serves to break any ties.

The operation of the separate processor regions of a processor triad is normally tightly synchronized. Under such circumstances this replicated operation can be discussed as if it were a single unit. The most noteworthy exception to this generalization is the operation of the system bus controllers during a polling sequence and before the separate elements of a processor triad have achieved synchronous operation. It is this behavior which provides the mechanism for synchronizing the separate elements of a processor triad.

A system bus controller participates in a poll sequence by transmitting one bit at a time on its assigned P line and listening to the voted input from all three active P lines. A poll may begin when the system bus is free. The system bus is free if during the preceding four bit periods the voted P bus value was 'zero'. 'Zero' is the undriven state of the P bus lines, therefore zero will be the voted result when no triad is controlling the bus and holding the P lines at 'one'. A bus controller, seeking control of the system bus, recognizes that the bus is free by detecting four sequential 'zero's on the P lines and attempts to initiate a poll by transmitting a 'one' on its assigned P line.

Under normal circumstances when the bus controller is synchronized with the other two members of its triad, the resultant voted P bus value is a 'one', as each element in synchronism transmits on its assigned P bus. This series of at least four 'zero's followed by a 'one' marks the beginning of the poll. The poll then proceeds and for the next three bit periods the controller transmits the static priority code, most significant bit first. If for any bit period the voted result of the P bus, differs from the transmitted bit, then the controller immediately drops out of the poll and waits for the system bus to become free again. Under normal circumstances when all triads are synchronized, the only time the resultant might differ from the transmitted value of a controller is when its 'zero' transmission is overwritten by a 'one' transmitted by a competing triad, with a higher poll number. After the static priority code is transmitted a 'one' is transmitted to prevent the possibility of four 'zero's in a row. This is followed by the dynamic priority code another 'one' and finally the triad identification. Again, if for any bit period the voted results differ from the transmitted bit then the triad immediately drops from the poll. Thus under normal circumstances, when all triads are synchronized, the triad with the highest nine-bit poll number will be the only triad surviving to the end of the poll. It then retains control of the system bus by transmitting 'one's on the

poll bus lines until it wishes to release it. Any triads which dropped out of the poll wait for the bus to be released and then reattempt the poll sequence.

This fairly tidy behavior is somewhat more complex if the individual bus controllers are not synchronized with each other at the beginning of the poll. Under this circumstance the voted result read from the P bus can differ from the bit transmitted because of the wired 'OR' characteristic of the bus, or it can differ because the single element transmitting on the bus was not joined by a partner. Thus, a controller may transmit a 'one' and read back a 'zero', it may transmit a 'zero' and read back a 'one', or it may actually read back what it transmitted. At the beginning of the poll a controller will detect that the bus is idle by a series of four or more 'zero's on the P bus and attempts to initiate a poll by transmitting a 'one' on its assigned P bus line. If it is the only controller transmitting a 'one' then it will read back a 'zero'. The controller in effect stalls at this point repeatedly transmitting a 'one' on its assigned bus as it attempts to initiate a poll sequence. It remains stalled at this point until it is joined by another controller, assigned to another bus, which is also attempting to initiate a poll sequence. Together, with each transmitting a 'one' on separate P lines they can initiate a poll sequence. The voted result will be 'one' and the poll begins. These two bus controllers are in fact now synchronized with one another. If these two bus controllers are members of the same triad they will continue through the poll sequence, gain control of the system bus, perform any commanded transaction and then release their processors to continue instruction execution. After obtaining synchronization the bus controllers remain synchronized, and the release of their processors is synchronous. Operation of the bus controllers and processors is such that this synchronous release in effect synchronizes the operation of the separate processor regions. If the individual processors are so configured as to have been in identical states (executing the same program, being at the same point in that program, etc.) then they will remain synchronized with one another. This synchronization process is likely to succeed in only synchronizing two of three elements of a triad on the first pass. When all the elements of a triad are completely out of synch then the leading element seeks control of the bus, but stalls and is forced to wait for the second element. When the second element catches up with the first they synchronize and together perform the poll sequence and gain control and proceed. The third element arriving late is left behind and stalls at the bus request stage. The two synchronized elements must then execute the proper procedures for looping back and picking up the tardy processor region. They can do this by repeating the bus request sequence. This causes them to sweep by and pick up the stalled processor region. Appropriate coding will assure that when this micro level of synchronization is obtained, that it can be preserved, and that all processors of the triad will execute the same code from that point onward.

A different and more complex sequence of events occurs if a bus controller is joined not by a member of its own triad, but by a member of a second triad. Together they can and do initiate the poll sequence. However since they are using different poll numbers they are unable to complete the sequence and will both drop out before gaining control. This will cause them to recycle to the point where they are both waiting for the poll to begin again. Since they have both stopped transmitting, four 'zero's will occur and another poll sequence will begin. There may in fact be a number of these false starts before a successful poll sequence takes place. Greater complexity occurs when the competing and separate bus controllers overwrite each other when assigned to the same bus. In the end these repeated numbers of false starts succeed in stalling everyone until a high priority triad synchronizes and gains control of the bus. Note that the dynamic priority adjustment mechanism could frustrate this effort to synchronize. A means of disabling the dynamic priority mechanism is therefore provided.

The processor controls the operation of the system bus controller by writing into several control registers. Figure 4.6 summarizes the function of these registers and their real address assignments on the processor region transfer bus.

Processor Transfer Bus Address		Control Register Content
FFF1	write read	Command (write) and Status (read) Register | X| X| X| X| X| X| X| X|WR|RQ|P2|P1|P0|SX|HB|HA| | X| X| X| X| X| X| X| X|WR|RQ|P2|P1|P0|SX|BG|ER| 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 WR - Read(0)/Write(1) System Memory RQ - Transfer Request(1)/NOP(0) P2,P1,P0 - Static Priority Code SX - Simplex(1)/Voted(0) Read HB - Hold bus(1)/Release bus(0) after Transaction HA - Increment(0)/Do not Increment(1) System Memory Displacement Reg. during transaction BG - Bus Grant: Holding Bus(1)/Not Holding Bus(0) ER - System Page Boundary Overflow
FFF2	RO	Last Word Transferred, dddddddddddddddd | d| d| d| d| d| d| d| d| d| d| d| d| d| d| d| d| 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
FFF3	WO	System Memory Page, nnnnnnnnnnn | X| X| X| X| X| n| n| n| n| n| n| n| n| n| n| n| 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
FFF4	WO	Page Displacement, dddddddd | X| X| X| X| X| X| X| X| d| d| d| d| d| d| d| d| 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
FFF5	WO	Local Memory DMA Control Word | 0| 0| 0| 0| 0| 0| 0| 0| 0| 0| 0| 0| 0| 0| 0| 0| 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
FFF6	WO	Word Count, cccccccccccccccc | c| c| c| c| c| c| c| c| c| c| c| c| c| c| c| c| 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
FFF7	WO	Starting Address, aaaaaaaaaaaaaaaa, in Local Memory | a| a| a| a| a| a| a| a| a| a| a| a| a| a| a| a| 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Figure 4.6 System Bus Controller Register Assignments

Data is transferred between system memory and local memory by first writing into the control registers (FFF3-FFF7) and then by initiating a transfer by writing into the command register (FFF1) of the system bus controller. The order in which the control registers are written is not significant.

Location "FFF3", the System Memory Page Register, should be set to point to the page of system memory from/to which the transfer is to take place. The system memory page of a word is determined by the 11 most significant bits of the system bus address of the first word of the block.

Location "FFF4", the displacement register, points to the starting location within the system memory page of the data block to be transferred. The displacement within the page is determined by taking the 8 least significant bits of the system memory starting address of the block to be transferred. At the end of a transaction this register will point to the next word of the system memory page which would have been transferred if the transaction had continued one more word. Note that it is impossible to perform a block transfer across a system memory or cache memory page boundary. If such a transfer is attempted the transaction is terminated when the 8 bit displacement address is incremented past "FF" to "00", and the word count has not been exhausted. In such a case the displacement register is left equal to zero at the end of the transaction.

Location "FFF5", the local memory DMA control register, controls the processor region transfer bus DMA mechanism. It should be set to zero. It is unchanged at the end of a transaction. Note that since this register should always be zero, it will only be necessary to write it once during the initialization procedure following the power on reset.

Location "FFF6", the word count register, specifies the number of words to be transferred. If an attempt is made to transfer a block of data which spans two pages of system memory or cache memory then the transaction is terminated after the last word of the first page is transferred and the ER bit of the status register is set. The status register is a read only register residing at location "FFF1".It mirrors the contents of the command register in bits 2 through 7. At the end of a transaction, the word count register will contain the number of words of the original block remaining to be transferred. Assuming normal termination (no attempt to transfer across a page boundary), the register will be zero.

Location "FFF7", the local memory starting address, points to the starting address in the processor region transfer bus real address space of the block of addresses from/to which data is to be read/written. This register is incremented after each word transfer. At the end of a transaction it points to the next address in the processor region transfer bus real address space from/to which the next word would have been read/written if the

transfer had continued one more word.

A data transfer is initiated by writing to location "FFF1", the command register.

Bit 6, the RQ or Request bit, of that write must be a one to initiate the transfer.

Bit 7, the Write/Read bit, determines the direction of the transfer. A one commands a transfer to system memory from local memory, a zero commands a transfer from system memory to local memory.

Bit 0, the HA or Hold Address bit, inhibits the normal incrementing of the system memory page displacement register as each word is transferred. When HA equals one, each read or write of system memory is from or to the same address. Note that although the system memory address is frozen during the transaction, the normal incrementing of the local memory addresses still occurs. This mode of transfer is used in reading or writing the I/O port FIFO buffer or in writing multiple registers of a BGU in a single transaction.

Bit 1, the HB or Hold Bus bit, when set to one commands controller to hold the bus after the transaction is complete. If set to zero the system bus will be released at the end of the transaction.

Bit 2, the SX or Simplex read bit, when set to one configures the R bus input circuitry to 'OR' all incoming data from the three selected R buses, and to ignore disagreements. When set to zero, the incoming data from the three selected R buses are voted normally and the error detection/monitoring circuitry is armed. This bit has no effect when the direction of the data transfer is from local memory to system memory.

Bits 3, 4 and 5, the static priority bits, specify the static priority code to be used by the controller in its competition for the bus. If the bus controller already controls the system bus this field has no effect on the transaction. If the controller does not already have control of the bus it will acquire control in order to effect the commanded transfer.

The command register must be rewritten each time a transaction is initiated.

Bit 0, the ER bit, is set to one if the last transaction was terminated by an attempt to transfer data past a system page boundry.

Bit 1, the BG bit, is set to one if the controller is holding control of the system bus. If HB set was specified in the previous request this bit should read one. If HB reset was specified in the previous request this bit should read zero.

Location "FFF2" can be read and will contain a copy of the last data word transferred by the system bus controller.

Upon receiving a transfer request the bus controller seizes control of the processor region transfer bus and holds it until the transfer is complete. This effectively holds the processor in a suspended state awaiting completion of the transfer. If necessary the controller then seeks control of the system bus. If it already holds control of the system bus or after obtaining control it then affects the requested transfer. Once the transfer is complete it releases the processor region transfer bus and either releases or holds the system bus depending upon whether the HB was zero or one. The polling sequence, used in acquiring the system bus, will synchronize those controllers which participated in in and won the poll to the same state. If they then execute the same transfer they will remain in synchronism. They will release control of their processor region transfer buses at the same time and their processors will therefore resume processing at exactly the same time. This effectively synchronizes those processors. If they then continue to execute the same instruction stream they will remain synchronized.

The bus controller accepts and processes the five redundant copies of the P and R system bus lines. It selects three of the five P lines and votes among them reporting any errors to the LRU status registers where they are latched. It uses the four bit P line select code provided by a system command register (the P Select Register, described in greater detail in Section 5.4.1) in determining which three of five P lines are selected. The voting and error detection takes place during each P bus bit period, even when the controller is idle. Thus, P bus errors are reported by all LRU's, even though some of the reporting LRU may be unconnected to the erroneous activity.

The bus controller accepts and processes the five redundant copies of the R bus only when it is expecting a reply over the system bus. It selects three of the five R lines for processing based upon the R line select code provided by a system control register. It either votes and performs error reporting on these lines or 'OR's them together suppressing error reporting depending upon the content of the SX bit of the controller command register. Only those bus controllers directly involved in the transaction will report R bus errors.

4.8 The CAPS-6 Processor

The Collins Avionics CAPS-6 processor is used as the processor element of the processor region. Alterations were made to the standard CAPS-6 processor so as to allow the processor to be driven from an external clock source, in this case the clock is provided by the clock generation region. Since all operations within the processor region are synchronous with the clock, and since the clock regions of the several LRU's can be phase locked with one another, the operation of separate processor regions will stay tightly synchronized, once initially synchronized, as long as each processor executes exactly the same instructions, as long as the phase locked clocks remain synchronized, as long as the communications interrupts occur synchronously in each processor (as will be the case if the triad identification is identical), and as long as the timer interrupts occur synchronously (as will be the case if the timer is set after synchronization is achieved). The CAPS-6 processor was additionally modified to provide for instruction backup should a mapper interrupt occur during an instruction execution. This restores the processor state to the point that existed immediately before the execution of the instruction which caused the mapper interrupt. These changes while critical to the correct operation of the FTMP do not affect the apparent operation of the machine at the architectural and instruction description level.

4.8.1 CAPS-6 Instruction Set and Architecture

In order to provide some context for the detailed instruction and architecture description which follows, it is necessary to provide a brief foundation description of the CAPS-6 overall architecture. This preliminary description is not intended to be complete but is intended to serve as a starting point. The detailed instruction set description which follows builds on this starting point, adding elaboration and detail as required and appropriate.

The basic architecture of the CAPS-6 processor is that of a stack machine. Figure 4.7 illustrates the basic processor organization as it is apparent to the programmer.

The program syllable counter, SPCR, points to the byte address in memory from which the next instruction is to be fetched. Addresses in memory are word addresses. A byte address is constructed by shifting a word address left by one and using the least significant bit to address a byte within the word. The even byte is the least significant byte of a word, the odd byte

is the most significant byte. Program addresses are the only byte addresses used by this machine. All memory references made by the processor are references to memory words. Where necessary the processor performs the necessary word disassembly in order to access program bytes.

The register TOS points to the top of stack. The word pointed to by TOS contains the value of the top word of the stack. The stack is stored in RAM memory. When a word is pushed onto the stack TOS is decremented by one and the word is stored into the memory location pointed to by the new value of TOS. When a word is pulled from the stack the memory location pointed to by TOS is read and the TOS is then incremented by one. As the stack grows the TOS points to successively lower addresses in memory. The register STKLM, stack limit, contains the lowest address that can be written into by a stack operation. An attempt to decrement the TOS register such that it would be less than the STKLM register will cause a stack overflow interrupt. The local environment register, LENV, points to the stack frame mark of the current local environment. The local environment is an area of memory that contains local variables of the current procedure. It can be accessed by using LENV as a base register. A stack frame mark contains three quantities, the address of the entry point of the current procedure (called the PROC ID), the return SPCR address of procedure which called the current procedure, and a pointer to the stack frame marker of this calling procedure. When a procedure is called a new set of local variables are allocated on the stack, a new stack frame mark is written and linked back to the previous mark, and the LENV register is adjusted to point to the new mark. When a return is executed this process is reversed. The function of the stack is treated in greater detail by the instruction set description.

In addition to the stack, the programmer may also use a base register table in accessing global memory. A register BRPT points to the first word of a 16 word block in memory where the active base register table is located.

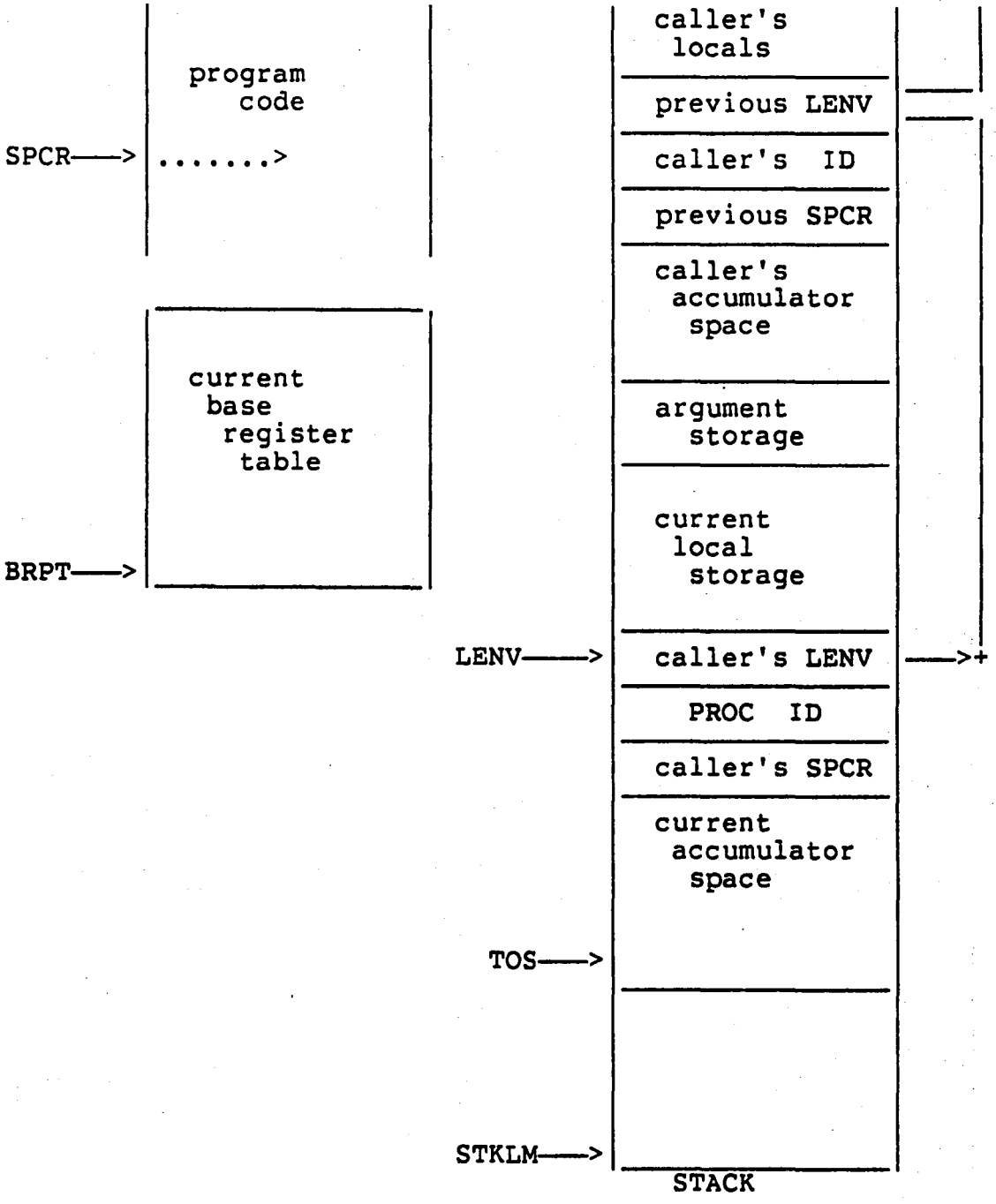


Figure 4.7 Basic Architecture

The following sections detail the CAPS-6 instruction set. All addresses refer to locations in the virtual or real address space depending upon whether the memory management unit operation is enabled or disabled. Except where noted, the instruction execution is unaffected by whether dynamic address translation is being performed. The following notational conventions are used:

```
        OPCODE <OPERAND>
        OPCODE OPERAND
 where: <OPERAND> implies the operand is contained in the
                  instruction stream.
    and OPERAND   implies the operand is contained in the
                  top of stack.

 additionally:
        (arg)     is the content of the memory location
                  whose address has a value of arg.
   for example:
        (TOS)     is the value stored at the top of stack
        (TOS+1)   is the value stored just under the top of
                  stack
        ((TOS))   is the value stored in the location pointed
                  to by the value stored at the top of stack
```

4.8.1.1 Data Transfer Instructions.

Data transfer instructions are used for either loading data onto the top of stack for a subsequent operation or storing data residing on the top of stack into memory.

LIT4 <K> : FOUR-BIT LITERAL.

A 4-bit literal is contained in the least significant bits of the LIT4 operation syllable. This literal field is extracted from the operation syllable and pushed, right-justified, into the top of the stack. The 12 most significant bits of the new top of stack are cleared to zero.

Stack Position	Stack Status Initial	Final
S0	A	K
S1	B	A
S2	C	B
S3	D	C

LIT8 <K> : EIGHT-BIT LITERAL.

An 8-bit literal is contained in the byte following the LIT8 operation syllable. This literal field is pushed, right-justified, into the top of the stack. The most significant byte of the top of stack is cleared to zero.

Stack Position	Stack Status Initial	Final
S0	A	K
S1	B	A
S2	C	B
S3	D	C

LIT16 <K> : SINGLE WORD LITERAL.

A 16-bit literal is obtained from the program syllable string and placed in the top of stack. The LIT16 operation syllable is interpreted to reference the following two 8-bit syllables of the program and place them into the stack. The two successive syllables appear in order of the least significant half of the literal and then the most significant half of the literal. The literal may be in the same 16-bit memory word or it may bridge two consecutive memory words.

Stack Position	Stack Status Initial	Final
S0	A	K
S1	B	A
S2	C	B
S3	D	C

LIT32 <K> : DOUBLE WORD LITERAL.

A 32-bit literal is obtained from the program syllable string and placed in the top of stack. The LIT32 operation syllable is interpreted to reference the following four 8-bit syllables of the program and place them into the stack. The 4 successive syllables appear in order of the least significant byte of the literal and then the next least significant byte, etc.

Stack Position	Stack Status Initial	Final
S0	A	K(LS HALF)
S1	B	K(MS HALF)
S2	C	A
S3	D	B

REFLS <K> : REFERENCE LOCAL SINGLE WORD.

References (reads) a 16-bit word from the Local Environment portion of memory and places it in the top of the stack. The base address is the contents of LENV register. K, the four least significant bits of the REFLS syllable, is the index on LENV (i.e., the sum of LENV and K points to the Kth word of the Local Environment).

Stack Position	Stack Status Initial	Final
S0	A	(LENV+K)
S1	B	A
S2	C	B
S3	D	C

REFLD <K> : REFERENCE LOCAL DOUBLE WORD.

References (reads) a double word (two 16-bit words) from the Local Environment and places it in the two top words of the stack. The base address for the double word is the contents of the LENV register. K, the four least significant bits of the REFLD operation syllable, is the index on LENV. The sum of LENV and K points to the least significant half of the double word.

Stack Position	Stack Status Initial	Final
S0	A	(LENV+K)
S1	B	(LENV+K+1)
S2	C	A
S3	D	B

ASNLS <K>,V : ASSIGN TO LOCAL SINGLE WORD.

The 16-bit word in the top of the stack is read destructively and stored (assigned) into memory in the Local Environment. The address in the memory where this assignment is made is computed by adding K to the contents of the LENV register. K is the four least significant bits of the ASNLS syllable.

Stack Position	Stack Status Initial	Final
S0	V	A
S1	A	B
S2	B	C
S3	C	D

ASNLD <K>,V' : ASSIGN TO LOCAL DOUBLE WORD.

The double word, V', (two 16-bit words) is initially in the two top words of the stack; the least significant half is S0, and the most significant half is S1. These two words are read destructively from the stack and stored (assigned) into two consecutive words of memory in the Local Environment. The assignment is made by computing addresses utilizing the LENV register and K, the four least significant bits of the ASNLD operation syllable.

Stack Position	Stack Status Initial	Final
S0	V'(LS half)	A
S1	V'(MS half)	B
S2	A	C
S3	B	D

REFLSE <K> : EXTENDED REFERENCE LOCAL SINGLE WORD

References (reads) a 16-bit word from the Local Environment portion of memory and places it in the top of the stack. The base address is the contents of LENV register (a pointer to the first word of the Local Environment). K, the 8-bit value of the byte following the REFLSE syllable, is the index on LENV (i.e., the sum of LENV and K points to the Kth word of the Local Environment). The Local Environment has zero origin (i.e., the first word is word zero).

Stack Position	Stack Status Initial	Final
S0	A	(LENV+K)
S1	B	A
S2	C	B
S3	D	C

REFLDE <K> : EXTENDED REFERENCE LOCAL DOUBLE WORD

References (reads) a double word (two 16-bit words) from the local Environment portion of memory and places it in the two top words of the stack. The base address for the double word is the contents of the LENV register (a pointer to the first word of the Local Environment). K, the value of the byte following the REFLDE operation syllable, is the index on LENV. The sum of LENV and K points to the least significant half of the double word.

Stack Position	Stack Status Initial	Final
S0	A	(LENV+K)
S1	B	(LENV+K+1)
S2	C	A
S3	D	B

ASNLSE <K>,V : EXTENDED ASSIGN TO LOCAL SINGLE WORD.

The 16-bit word in the top of the stack is read destructively and stored (assigned) into memory in the Local Environment. The address in the memory where this assignment is made is computed by adding K to the contents of the LENV register (a pointer to the first word of the local environment) K is the value of the byte following the ASNLSE syllable and is the index on LENV (i.e., the sum of LENV and K is the address of the Kth word of the Local Environment). The Local Environment has zero origin (i.e., the first word is word zero).

Stack Position	Stack Status Initial	Final
S0	V	A
S1	A	B
S2	B	C
S3	C	D

ASNLDE <K>,V' : EXTENDED ASSIGN TO LOCAL DOUBLE WORD.

The double word, V', (two 16-bit words) is initially in the top words of the stack; the least significant half is S0, and the most significant half is S1. These two words are read destructively from the stack and stored (assigned) into two consecutive words of memory in the Local Environment. The assignment is made by computing addresses utilizing the LENV register (a pointer to the first word of the Local Environment) and K, the value of the byte that follows the ASNLDE operation syllable.

Stack Position	Stack Status Initial	Final
S0	V'(LS half)	A
S1	V'(MS half)	B
S2	A	C
S3	B	D

LOCL I : GLOBAL RELATIVE ADDRESS OF LOCAL WORD.

Generates the absolute address of the Ith word of the Local Environment. This computed address is placed in the top of the stack.

Stack Position	Stack Status Initial	Final
S0	I	LENV+I
S1	A	A
S2	B	B
S3	C	C

REFGS I : REFERENCE GLOBAL SINGLE WORD.

References (reads) a 16-bit word from memory and places it in the top of the stack. I, a 16-bit value initially in the top word of the stack, points to the word to be referenced. when the 16-bit word is referenced (fetched) from memory, it replaces I in the top word of the stack (i.e., I is destroyed).

Stack Position	Stack Status Initial	Final
S0	I	(I)
S1	A	A
S2	B	B
S3	C	C

REFGD I : REFERENCE GLOBAL DOUBLE WORD.

References (reads) a double word (two 16-bit words) from memory and places it in the two top words of the stack. The address for the double word is I, a 16 bit value initially in S0, the top word of the stack. The stack is first pushed, moving I to S1, to make room for both halves of the double word. The least significant half goes to S0; the most significant half to S1. I is destroyed when S1 is loaded with the referenced most significant half of the double word. I is the address of the least significant half of the double word, while I + 1 points to the most significant half of the double word.

Stack Position	Stack Status Initial	Final
S0	I	(I)
S1	A	(I+1)
S2	B	A
S3	C	B

ASNGS V,I : ASSIGN GLOBAL SINGLE WORD.

The 16-bit word in the second word of the stack is read destructively and stored (assigned) into memory. The address in memory where this assignment is made is I, a 16-bit value initially in the top word of the stack. Upon completion of the assignment, both I and the value to be assigned are removed from the stack by incrementing TOS by two (i.e., popping the stack twice).

Stack Position	Stack Status Initial	Final
S0	I	A
S1	V	B
S2	A	C
S3	B	D

ASNGD V',I : ASSIGN GLOBAL DOUBLE WORD.

The address, I, is initially in the top of the stack (S0). The second and third words of the stack contain the double word, V', (two 16-bit words) with the least significant half in S1 and the most significant half in S2. The double word, V', is read destructively from the stack and stored (assigned) into two consecutive words in memory. The least significant half goes into the address I; the most significant half to I + 1. Upon completion of the double word assignment, the stack is popped three times to remove the double word and the address I.

Stack Position	Stack Status Initial	Final
S0	I	A
S1	V'(LS HALF)	B
S2	V'(MS HALF)	C
S3	A	D

REFSA <K> : REFERENCE SINGLE ABSOLUTE. References (reads) a 16-bit word from memory and places it in the top of the stack. The address K, is the 16-bit value in the two bytes following the REFSA syllable.

Stack Position	Stack Status Initial	Final
S0	A	(K)
S1	B	A
S2	C	B
S3	D	C

REFDA <K> : REFERENCE DOUBLE ABSOLUTE.

References (reads) two 16-bit words from memory and places it in the top of the,stack. The address K, is the 16-bit value in the two bytes following the REFDA syllable.

Stack Position	Stack Status Initial	Final
S0	A	(K)
S1	B	(K+1)
S2	C	A
S3	D	B

ASNSA <K>,V : ASSIGN SINGLE ABSOLUTE.

The 16-bit word, V, in the top of the stack is read destructively and assigned (stored) into the memory. The address K, is the 16-bit value in the two bytes following the ASNSA syllable.

Stack Position	Stack Status Initial	Final
S0	V	A
S1	A	B
S2	B	C
S3	C	D

ASNDA <K>,V' : ASSIGN DOUBLE ABSOLUTE.

The double word, V', is initially in the two top words of the stack; the least significant half is S0, and the most significant is S1. These two words are read destructively from the stack and assigned (stored) to the memory. The address K, is the 16-bit value in the two bytes following the ASNDA syllable.

Stack Position	Stack Status Initial	Final
S0	V'(LS half)	A
S1	V'(MS half)	B
S2	A	C
S3	B	D

REFSC <O,I> : REFERENCE SINGLE COMPONENT.

References (reads) a 16-bit word from memory and places it in the top of the stack. O is a positive offset <16. I is an index from the LENV register and points to the I'th word in the local environment. The address from which the word is obtained is computed by adding O to the contents of the word pointed to by LENV + I. If, however, the index I is 0 then the address is obtained by adding O to the contents of the top of the stack.

Stack Position	Stack Status Initial	Final
S0	A	((LENV+I)+O)
S1	B	A
S2	C	B
S3	D	C

REFDC <O,I> : REFERENCE DOUBLE COMPONENT.

References (reads) a double word from memory and places it in the top two words of the stack. O is a positive offset < 16. I is an index from the LENV register and points to the I'th word in the local environment. The addresses from which the double word is obtained are computed by adding O and O + 1 to the contents of the word pointed to LENV + I. If, however, the index i is 0 then the addresses are obtained by adding O and O + 1 to the contents of the top of the stack.

Stack Position	Stack Status Initial	Final
S0	A	((LENV+I)+O)
S1	B	((LENV+I)+O+1)
S2	C	A
S3	D	B

ASNSC <O,I>,V : ASSIGN SINGLE COMPONENT.

The 16-bit word, V, in the top of the stack is read destructively and stored (assigned) into memory. O is a positive offset < 16. I is an index from the LENV register and points to the I'th word in the local environment. The address in which the word is to be stored is obtained by adding O to the contents of the word pointed to by LENV + I. If, however, the index I is zero then the address is obtained by adding O to the contents of S1.

Stack Position	Stack Status Initial	Final
S0	V	A
S1	A	B
S2	B	C
S3	C	D

ASNDC <O,I>,V' : ASSIGN DOUBLE COMPONENT.

The double word, V', in the top two words of the stack are read destructively and stored (assigned) into memory. O is a positive offset < 16. I is an index from the LENV register and points to the I'th word of the local environment. The addresses in which the double word is to be stored are obtained by adding O and O + 1 to the contents of the word pointed to by LENV + I. If, however, the index I is 0 then the addresses are obtained by adding O and O + 1 to the contents of S2.

Stack Position	Stack Status Initial	Final
S0	V'(LS half)	A
S1	V'(MS half)	B
S2	A	C
S3	B	D

REFSP <P,O> : REFERENCE SINGLE USING POINTER TABLE

References (reads) a 16-bit word from the Global Environment portion of memory and places it in the top of the stack. P is an offset into the currently active base register pointer table (pointed to by the BRPT register) and O is either a positive offset or zero. The address from which the word is obtained is computed by adding O to the contents of the word pointed to by BRPT + P.

Stack Position	Stack Status Initial	Final
S0	A	((BRPT+P)+O)
S1	B	A
S2	C	B
S3	D	C

REFDP <P,O> : REFERENCE DOUBLE USING POINTER.

References (reads) a double word from the Global Environment portion of memory and places it into the top two words of the stack. P is an offset into the current base register pointer table and O is either a positive offset or zero. The addresses from which the double word is obtained are computed by adding O and O + 1 to the contents of the word pointed to by BRPT + P.

Stack Position	Stack Status Initial	Final
S0	A	((BRPT+P)+O)
S1	B	((BRPT+P)+O+1)
S2	C	A
S3	D	B

ASNSP <P,O>,V : ASSIGN SINGLE USING POINTER.

The 16-bit word, V, in the top of the stack is read destructively and stored (assigned) into memory. P is an offset into the current base register pointer table (pointed to by the BRPT register) and O is either a positive offset or zero. The address in which the word is to be stored is obtained by adding O to the contents of the word pointed to by PDTR + P.

Stack Position	Stack Status Initial	Final
S0	V	A
S1	A	B
S2	B	C
S3	C	D

ASNDP <P,O>,V' : ASSIGN DOUBLE USING POINTER.

The double word, V', in the top two words of the stack are read destructively and stored (assigned) into memory. P is an offset into the current base register pointer table (pointed to by the BRPT register) and O is either a positive offset or zero. The addresses in which the double word is to be stored are obtained by adding O and O + 1 to the contents of the word pointed to by BRPT + P.

Stack Position	Stack Status Initial	Final
S0	V'(LS half)	A
S1	V'(MS half)	B
S2	A	C
S3	B	D

REFSPI <P,O,I> : REFERENCE SINGLE USING POINTER AND INDEX.

References (reads) a 16-bit word from memory and places it in the top of the stack. P is an offset into the current base register pointer table (pointed to by the BRPT register). O is either a positive offset or zero. I is an index from the LENV register and points to the I'th word of the local environment. The address from which the word is obtained is computed by adding the contents of the word pointed to by LENV + I to the result of adding O to the contents of the word pointed to by BRPT + P.

Stack Position	Stack Status Initial	Final
S0	A	((BRPT+P)+O+(LENV+I))
S1	B	A
S2	C	B
S3	D	C

REFDPI <P,O,I> : REFERENCE DOUBLE USING POINTER AND INDEX.

References (reads) a double word from the Global Environment portion of memory and places it into the top two words of the stack. P is an offset into the current base register pointer table (pointed to by the BRPT register). O is either a positive offset or zero. I is an index from the LENV register and points to the I'th word of the local environment. The addresses from which the double word is obtained are computed by adding twice the contents of the word pointed to by LENV + I to the results of adding O and O + 1 to the contents of the word pointed to by PDTR + P.

Stack Position	Stack Status Initial	Final
S0	A	((BRPT+P)+O+2*(LENV+I))
S1	B	((BRPT+P)+O+1+2*(LENV+I))
S2	C	A
S3	D	B

ASNSPI <P,O,I>,V : ASSIGN SINGLE USING POINTER AND INDEX.

The 16-bit word, V, in the top of the stack is read destructively and stored (assigned) into memory. P is an offset into the currently active page definition table (pointed to by the BRPT register) and O is either a positive offset or zero. I is an index from the LENV register and points to the I'th word of the local environment. The address in which the word is to be stored is obtained by adding the contents of the word pointed to by LENV + I to the result of adding O to the contents of the word pointed to by BRPT + P.

Stack Position	Stack Status Initial	Final
S0	V	A
S1	A	B
S2	B	C
S3	C	D

ASNDPI <P,O,I>,V' : ASSIGN DOUBLE USING POINTER AND INDEX.

The double word, V', in the top two words of the stack are read destructively and stored (assigned) into memory. P is an offset into the current base register pointer table (pointed to by the BRPT register) and O is either a positive offset or zero. I is an index from the LENV register and points to the I'th word of the Local environment. The addresses in which the double word is to be stored are obtained by adding twice the content of the word pointed to by LENV + I to the results of adding O and O + 1 to the contents of the word pointed to by BRPT + P.

Stack Position	Stack Status Initial	Final
S0	V'(LS half)	A
S1	V'(MS half)	B
S2	A	C
S3	B	D

4.8.1.2 Arithmetic Instructions.

ADD X,Y : TWO's COMPLEMENT INTEGER OR FRACTIONAL ADD.

The ADD operator performs binary addition on two 16-bit operands initially in the top two positions of the stack (S0 and S1) and pops the stack to replace the original operands, leaving the result in the top of the stack (S0). The result is the binary sum of the two 16-bit operands. Overflow occurs if the result X+Y can not be represented in 16 bits.

Stack Position	Stack Status Initial	Final
S0	Y	X+Y
S1	X	A
S2	A	B
S3	B	C

SUB X,Y : TWO's COMPLEMENT INTEGER OR FRACTIONAL SUBTRACT.

The subtract operator performs binary two's complement subtraction of Y, a 16-bit operand initially in the top of the stack (S0) from X, a 16-bit operand initially in the second word of the stack (S1). The stack is popped to remove the original operands, leaving the result in the new top of the stack (S0). The result is the two's complement difference X-Y. Overflow occurs if the result X-Y can not be represented in 16 bits.

Stack Position	Stack Status Initial	Final
S0	Y	X-Y
S1	X	A
S2	A	B
S3	B	C

MPYI X,Y : TWO's COMPLEMENT INTEGER MULTIPLY.

The MPYI operator performs binary multiplication of two 16-bit integer operands in two's complement form and pops the stack to replace the original operands, leaving the 16-bit result in the new top of stack. Initially, the multiplier Y is in the top of the stack (S0) and the multiplicand X is the next position (S1). The result is the binary product Y times X, modulo 2expl6; i.e., the least significant 16 bits of the double-length product. Overflow occurs if the result X*Y can not be represented in 16 bits. In case of overflow the result in S0 is 7FFF or 8000 depending on the sign of the result.

Stack Position	Stack Status Initial	Final
S0	Y	X*Y
S1	X	A
S2	A	B
S3	B	C

DIVI X,Y : TWOs COMPLEMENT INTEGER DIVIDE.

The DIVI operator divides X, a fixed point two's complement integer in stack position S1, by Y, a fixed point two's complement integer in stack position S0. The original operands are eliminated by a stack adjustment and the result, the quotient represented as a two's complement integer, is placed in the top of the stack, S0. If the divisor is zero, the divide operation is not performed, and the result equal to zero is left in the top of the stack. For non-zero operands, the result of the DIV operator is the 16-bit signed two's complement representation of the integer binary divide operation on the operands (originally in two complement representation) converted to positive integer representation. The two's complement of the integer divide quotient forms the result if the original operands had opposite signs. Since an integer result is produced, the result quotient is zero if the absolute value of the dividend is less than that of the divisor. No remainder representation is preserved.

Stack Position	Stack Status Initial	Final
S0	Y	X/Y
S1	X	A
S2	A	B
S3	B	C

ABSV X : ABSOLUTE VALUE, SINGLE WORD.

This operator replaces the 16-bit word, X, in S0 by its absolute value. That is, if the leftmost bit (bit 15) of X is a 1, the two's complement of X is placed in S0; otherwise, the stack remains as it was prior to the operation. Overflow occurs if X="8000".

Stack Position	Stack Status Initial	Final
S0	X	|X|
S1	A	A
S2	B	B
S3	C	C

MPY X,Y : TWO's COMPLEMENT FRACTIONAL MULTIPLY.

The MPY operator performs binary multiplication of two 16-bit fraction operands in two's complement form and pops the stack to replace the original operands, leaving the 16-bit result in the new top of stack. Initially the multiplier Y is in the top of the stack (S0) and the multiplicand X is the next position (S1). The result is the binary product Y times X; i.e., the most significant 16 bits of the double-length product. Overflow occurs if X=Y="8000".

Stack Position	Stack Status Initial	Final
S0	Y	X*Y
S1	X	A
S2	A	B
S3	B	C

MPYE X,Y : TWO's COMPLEMENT FRACTIONAL MULTIPLY.

The MPYE operator performs binary multiplication of two 16-bit fraction operands in two's complement form and pops the stack to replace the original operands, leaving the 32-bit result in the top 2 words of stack. Initially the multiplier Y is in the top of the stack (S0) and the multiplicand X is the next position (S1). The result is the binary product Y times X; i.e., the the 32-bit double-length product. Overflow occurs if X=Y="8000".

Stack Position	Stack Status Initial	Final
S0	Y	X*Y (LS HALF)
S1	X	X*Y (MS HALF)
S2	A	A
S3	B	B

DIV X,Y : TWO's COMPLEMENT FRACTIONAL DIVIDE.

The DIV operator divides X, a fixed point two's complement fraction in stack position S1, by Y, a fixed point two's complement fraction in stack position S0. The original operands are eliminated by a stack adjustment and the result, the quotient represented as a two's complement fraction, is placed in the top of the stack, S0. If the divisor is zero, the divide operation is not performed, and the result equal to zero is left in the top of the stack. Overflow occurs if |Y|<|X| or X=Y="8000".

Stack Position	Stack Status Initial	Final
S0	Y	X/Y
S1	X	A
S2	A	B
S3	B	C

ADDD X',Y' : TWO'S COMPLEMENT DOUBLE WORD ADD.

The ADDD operator performs binary addition on two 32-bit (double) operands initially in the top four positions of the stack (S0 through S3) and pops the stack to replace the original operands, leaving the result in the top of the stack (S0, S1). The result is the binary sum of the two 32-bit operands. Overflow occurs if X'+Y' can not be represented in 32 bits.

Stack Position	Stack Status Initial	Final
S0	Y'(LS half)	X'+Y'(LS half)
S1	Y'(MS half)	X'+Y'(MS half)
S2	X'(LS half)	A
S3	X'(MS half)	B

SUBD X',Y' : TWO'S COMPLEMENT, DOUBLE WORD SUBTRACT.

The subtract operator performs binary two's complement subtraction of Y', a 32-bit operand initially in the S0 and S1 registers from X', a 32-bit operand initially in the S2 and S3 registers. The stack is popped to remove the original operands, leaving the result in the new top of the stack (S0, S1). The result is the two's complement difference X' - Y'. Overflow occurs if X' - Y' can not be represented in 32 bits.

Stack Position	Stack Status Initial	Final
S0	Y'(LS halt)	X'-Y'(LS half)
S1	Y'(MS half)	X'-Y'(MS half)
S2	X'(LS HALF)	A
S3	X'(MS HALF)	B

MPYD X',Y' : TWO's COMPLEMENT FRACTIONAL MULTIPLY, DOUBLE WORD

The MPYD operator performs binary multiplication of two 32-bit fraction operands in two's complement form and POPS the stack to replace the original operands, leaving the 32-bit result in the new top of stack Initially the multiplier Y' is in the S0 and S1 registers and the multiplicand X' is in S2 and S3. The result is the binary product Y' times X'; i.e., the most significant 32 bits of the four word product.

Stack Position	Stack Status Initial	Final
S0	Y'(LS HALF)	X'*Y'(LS half)
S1	Y'(MS HALF)	X'*Y'(MS half)
S2	X'(LS HALF)	A
S3	X'(MS HALF)	B

DIVD X',Y' : TWO's COMPLEMENT FRACTIONAL DIVIDE, DOUBLE WORD.

The DIVD operator divides X', a fixed point two's complement 32 bit fraction in (S2, S3), by Y', a fixed point two's complement 32-bit fraction in (S0, S1). The original operands are eliminated by a stack adjustment and the result, the quotient represented as a two's complement fraction, is placed into S0 and S1. If the divisor is zero, the divide operation is not performed, and the result equal to zero is left in the top of the stack.

Stack Position	Stack Status Initial	Final
S0	Y'(LS half)	X'/Y'(LS half)
S1	Y'(MS half)	X'/Y'(MS half)
S2	X'(LS half)	A
S3	X'(MS half)	B

ABSD X' : ABSOLUTE VALUE, DOUBLE WORD.

This operator replaces the 32-bit (double-length) word X' in (S1, S0) with its absolute value. That is, if the sign bit of X' (bit 15 of S1) is a 1, then the two's complement of X' is placed in (S1, S0); otherwise, the stack remains as it was prior to the operation. Overflow occurs if X'="80000000".

Stack Position	Stack Status Initial	Final
S0	X'(LS half)	|X'|(LS half)
S1	X'(MS half)	|X'|(MS half)
S2	A	A
S3	B	B

EXTS X : EXTEND SIGN.

This operator extends the sign bit of the operand in S0. The operand stays in the top of the stack (S0) and the extended sign (0 or -1) is stored in S1. Space for SIGN (X) is allocated on the stack by pushing the stack once and moving X to the top of the stack.

Stack Position	Stack Status Initial	Final
S0	X	X
S1	A	SIGN(X)
S2	B	A
S3	C	B

MPYID X',Y' : TWO's COMPLEMENT INTEGER MULTIPLY, DOUBLEWORD

The MPYID operator performs binary multiplication of two 32-bit integer operands in two's complement form and pops the stack to replace the original operands, leaving the 32-bit result in the S0 and S1 registers. Initially the multiplier Y' is in S0 and S1, and the multiplicand X' is in S2 and S3. The result is the binary product Y' times X', modulo 2 exp 32; i.e., the least significant 32 bits of the four word product.

Stack Position	Stack Status Initial	Final
S0	Y'(LS half)	X'*Y'(LS half)
S1	Y'(MS half)	X'*Y'(MS half)
S2	X'(LS half)	A
S3	X'(MS half)	B

DIVID X',Y' : TWO's COMPLEMENT INTEGER DIVIDE, DOUBLEWORD.

The DIVID operator divides X', a fixed point two's complement integer in S2 and S3 by Y', a fixed point two's complement integer in S0 and S1. The original operands are eliminated by a stack adjustment and the result, the quotient represented as a two's complement integer, is placed in the top of the stack, in S0 and S1. If the divisor is zero, the divide operation is not performed, and the result equal to zero is left in the top of the stack.

Stack Position	Stack Status Initial	Final
S0	Y'(LS half)	X'/Y'(LS half)
S1	Y'(MS half)	X'/Y'(MS half)
S2	X'(LS half)	A
S3	X'(MS half)	B

4.8.1.3 Bit Manipulation Instructions.

SRS X,L : RIGHT SHIFT.

This operator right shifts the operand X, in stack position S1, by the amount L in stack position S0. The shift is performed with a zero fill. The stack is popped once after the shift is performed.

Stack Position	Stack Status Initial	Final
S0	L	X
S1	X	A
S2	A	B
S3	B	C

SLS X,L : LEFT SHIFT.

This operator left shifts the operand X, in stack position S1, by the amount L in stack position S0. The shift is performed with a zero fill. The stack is popped once after the shift is performed.

Stack Position	Stack Status Initial	Final
S0	L	X
S1	X	A
S2	A	B
S3	B	C

INSERT Y,X,S,L : INSERT FIELD IN WORD.

The INSERT operator extracts a right-justified field of length L from Y, a single-word stack operand, and inserts it into a cleared field (starting at bit number S and length L) in another single-word operand, X. Initially the source operand X is in stack position S2, the operand Y in S3. The starting bit number S (0<S<15) is encoded in the least significant four bits of (S1), and the field length, L (0<L<16) is encoded in the least significant five bits of (S0). The extracted field begins at bit position S and ends at bit position S+L-1. For the purpose of counting bit positions, the least significant bit is '0' and the most significant bit is '15'. The 16-bit result with inserted field is placed at the top of the stack and the original stack contents (S0, S1, S2, S3) discarded.

Stack Position	Stack Status Initial	Final
S0	16-L	X
S1	S	A
S2	X	B
S3	Y	C

XTRACT X,S,L : EXTRACT FIELD FROM WORD.

The XTRACT operator extracts a field defined by a starting bit number and a length (number of bits) from a 16-bit word (X) in the third word of the stack. The result, the right-justified extracted field, is placed in the top word of the stack, and the field description parameters and the original word are discarded. Initially the operand X is in stack location (S2), the starting bit number, S (0<S<15) is in the least significant 4 bits of stack location S1, and the field length in bits, L (0<L<16) is in the least significant 5 bits of the top of the stack (S0). The extracted field begins at bit position S and ends at bit position S+L-1. For the purpose of counting bit positions, the least significant bit is '0' and the most significant bit is '15'.

Stack Position	Stack Status Initial	Final
S0	16-L	X
S1	S	A
S2	X	B
S3	A	C

REFBIT N,X : REFERENCE BIT.

This operator reads the Nth bit of memory pointed to by X. Initially, address X is in the top of the stack (S0) and N is in S1. Finally, X and N are popped from the stack and the bit value (0 or 1) is returned in S0.

Stack Position	Stack Status Initial	Final
S0	X	BIT VALUE
S1	N	A
S2	A	B
S3	B	C

ASNBIT V,X,N : ASSIGN BIT.

This operator assigns (writes) the value V (0 or 1) in the Nth bit of memory location X. Initially X, N and V are in S0, S1 and S2, respectively. Finally, all three operands are popped from the stack.

Stack Position	Stack Status Initial	Final
S0	X	A
S1	N	B
S2	V	C
S3	A	D

4.8.1.4 Logical Instructions.

AND X,Y : LOGICAL AND, SINGLE WORD.

This operator produces the logical bit-by-bit product of the two 16-bit words initially on top of the stack. That is, the word X in S1 is 'AND-ed' with the word Y in S0. The result is placed in S1 and then the TOS pointer is incremented by 1, leaving the result on top of the stack.

Stack Position	Stack Status Initial	Final
S0	Y	X AND Y
S1	X	A
S2	A	B
S3	B	C

OR X,Y : LOGICAL OR, SINGLE WORD.

This operator produces the logical bit-by-bit inclusive OR of the two 16-bit words initially on top of the stack. That is, the word X in S1 is 'OR-ed' with the word Y in S0. The result is placed in S1 and then the TOS pointer is incremented by 1, leaving the result on top of the stack.

Stack Position	Stack Status Initial	Final
S0	Y	X OR Y
S1	X	A
S2	A	B
S3	B	C

XOR X,Y : LOGICAL EXCLUSIVE OR, SINGLE WORD.

This operator produces the logical bit-by-bit exclusive-or of the two 16-bit words initially on top of the stack. That is, the exclusive sum of the word X in S1 and the word Y in S0 is formed. The result is placed in S1 and then the TOS pointer is incremented by 1, leaving the result on top of the stack.

Stack Position	Stack Status Initial	Final
S0	Y	X XOR Y
S1	X	A
S2	A	B
S3	B	C

NOT X : LOGICAL COMPLEMENT, SINGLE WORD.

This operator replaces the 16-bit word X in S0 by its one's complement.

Stack Position	Stack Status Initial	Final
S0	X	NOT X
S1	A	A
S2	B	B
S3	C	C

4.8.1.5 Relational Instructions.

GR X,Y : TWO's COMPLEMENT INTEGER GREATER THAN, SINGLE WORD.

This operator algebraically compares two 16-bit integers (fixed point representation) initially in the top of the stack. 'Is X greater than Y?' (X is in S1 and Y in S0.) If X is algebraically greater than Y, then the logical true value (all ONEs) is placed in the top of the stack (S0). If X is equal to, or algebraically less than Y, then the logical false value (all ZEROs) is placed in the top word of the stack. In either case, a stack 'pop' adjustment replaces the original operands with the logical results in the new top of the stack.

Stack Position	Stack Status Initial	Final
S0	Y	LOGICAL RESULT
S1	X	A
S2	A	B
S3	B	C

GRD x',y' : TWO's COMPLEMENT INTEGER GREATER THAN, DOUBLE WORD.

This operator algebraically compares two 32-bit integers (fixed point representation) initially in the top of the stack. 'Is X' greater than Y'?'. (X' is in (S2, S3); Y' is in (S0, S1)). If X' is algebraically greater than Y', then the logical true value (all ONEs) is placed in the top of the stack (S0). If X' is equal to, or algebraically less than Y', then the logical false value (all zeros) is placed in the top word of the stack. either case, a stack 'pop' adjustment replaces the original operands with the logical results in the new top of stack.

Stack Position	Stack Status Initial	Final
S0	Y'(LS HALF)	RESULT(LS HALF)
S1	Y'(MS HALF)	RESULT(MS HALF)
S2	X'(LS HALF)	A
S3	X'(MS HALF)	B

EQ X,Y : EQUAL, SINGLE WORD.

This operator compares two 16-bit operands in the top two stack positions for equivalence. If the contents of the top of the stack (S0) is equal to the contents of the next stack position (S1), then the result is the logical truth value (all ONEs); otherwise, the result is the logical false value (all ZEROs). In either case, a stack pop replaces the original operands with the logical result in the top of the stack (S0).

Stack Position	Stack Status Initial	Final
S0	Y	LOGICAL RESULT
S1	X	A
S2	A	B
S3	B	C

EQD X',Y' : EQUAL, DOUBLE WORD.

This operator compares two 32-bit operands in the top four stack positions for equivalence. If the contents of S2 and S3 (X') is equal to the contents of the S0 and S1 (Y') then the result is the logical truth value (all ones); otherwise, the result is the logical false value (all ZEROs). In either case, a stack pop replaces the original operands with the logical result in the top of stack (S0).

Stack Position	Stack Status Initial	Final
S0	Y'(LS HALF)	RESULT(LS HALF)
S1	Y'(MS HALF)	RESULT(MS HALF)
S2	X'(LS HALF)	A
S3	X'(MS HALF)	B

4.8.1.6 Control Transfer Instructions.

IFT C,N : IF CONDITION TRUE, CONTINUE, ELSE SKIP.

The IFT operator provides conditional self-relative program control transfer. If the condition, C, initially in stack location (S1) is true (not all ZEROS), then the program syllable execution continues in sequence. If the condition, C, is false (all ZEROS), the N program syllables are skipped. If the skip is to be performed, the current contents of the program syllable counter are incremented by N, a TWOs complement number in the top of the stack (S0) to form the address of the next program syllable to be executed. A stack pop of 2 eliminates both the condition and the skip count from the top of the stack.

Stack Position	Stack Status Initial	Final
S0	N	A
S1	C	B
S2	A	C
S3	B	D

IFF C,N : IF CONDITION FALSE, CONTINUE, ELSE SKIP.

The IFF operator provides conditional self-relative program control transfer. If the condition C, initially in the stack location (S1) is false (all ZEROS) then program syllable execution continues in sequence. If the condition C is true (not all ZEROS) then N program syllables are skipped. If the SKIP is to be performed the current contents of the program syllable counter are incremented by N, a twos complement number in the top of the stack (S0) to form the address of the next program syllable to be executed. A stack pop of 2 eliminates both the condition and skip count from the top of the stack.

Stack Position	Stack Status Initial	Final
S0	N	A
S1	C	B
S2	A	C
S3	B	D

BRANCH<L> : BRANCH TO LOCATION L.

This operator transfers control to address L by loading SPCR with L. Address L is contained in the 2 bytes following the BRANCH opcode byte.

SKIP N : UNCONDITIONAL SKIP.

The skip operator provides unconditional self-relative program control transfer, an unconditional skip by the number of program syllables specified by N, a 16-bit two's complement number in the top of the stack. After incrementing the current value of the program syllable count register by the contents of the top of the stack, N, a stack adjustment pops N from the stack.

Stack Position	Stack Status Initial	Final
S0	N	A
S1	A	B
S2	B	C
S3	C	D

SVSKIP N : SAVE CURRENT SPCR & SKIP N SYLLABLES.

This operator saves the current contents of the program syllable counter, SPCR, and skips N program syllables. The address of the next program syllable to be executed is formed by incrementing the contents of the top of the stack. The initial contents of the SPCR replace the skip number, N, in the top of the stack.

Stack Position	Stack Status Initial	Final
S0	N	SPCR
S1	A	A
S2	B	B
S3	C	C

POC : POWER ON CLEAR.

Upon application of power to the CPU, the control is transferred to the microcode for this instruction. This instruction expects cache locations 0 to 3 in ROM to contain the following initialization information:

0 - IPROC
1 - ITOS
2 - ISTKLIM
3 - IPSD

The POC instruction initializes the TOS and STKLIM registers to values contained in locations 1 and 2. The LENV and SPCR are initialized to 0. The privileged mode is turned on by setting PMR to 1 and the mapper is turned off by writing 0 to "FFFE". Control is then transferred to CALL microsequence which in turn passes control to the procedure pointed to by location 0. Location 3 of ROM should point to the PSD of this initial procedure.

4.8.1.7 Interrupt Related Instructions.

INTERRUPT PROCESSING.

The mechanism for handling interrupts includes procedures for interrupting a user program when interrupts are enabled, and an instruction for returning to user mode from interrupt mode, INTRTN. Since the procedure for interrupting a user program is not an instruction it is described here. For an interrupt to occur the processor must be in user mode (interrupts enabled). An interrupt can then occur at the end of the current instruction before the next instruction has begun. When this occurs the user stack becomes inactive and the interrupt stack is activated. The machine state is saved in the current Processor State Descriptor (PSD). The location of this PSD is contained on the top of the Interrupt Stack. The PSD contains the following information:

	PSD
0.	TOS
1.	STKLIM
2.	SPCR
3.	LENV
4.	PMR
5.	MAP SELECT
6.	OTHER PSD INFORMATION

The machine state is restored to the interrupt mode which is defined by the interrupt PSD. The interrupt PSD is pointed to by the internal register ISR. Next, the highest priority interrupt requesting service is pushed into the interrupt stack and that interrupt is cleared. Finally, all interrupts are disabled and interrupt program is activated. The instruction interrupt return, INTRTN, will reverse this process and restart the interrupted procedure. The instruction HALT will cause an interrupt (number 10'Hex). The HALT interrupt can not be disabled. The instructions assign mask, ASNMSK and reference mask, REFMSK, allow access to the interrupt mask. The table below summarizes the interrupt assignments for the implementation of the CAPS-6 processor.

Interrupt Number	Maskable	Assignment/function
0	yes	unassigned*
1	yes	unassigned*
2	yes	unassigned*
3	yes	unassigned*
4	yes	unassigned*
5	yes	unassigned*
6	yes	unassigned*
7	yes	unassigned*
8	yes	unassigned
9	yes	unassigned
A	yes	arithmetic overflow
B	yes	IPC register 2 or 3 written
C	yes	interval timer
D	yes	write protect violate
E	yes	page fault
F	yes	reserved (for test adapter)
10	no	halt instruction execution
11	no	illegal opcode
12	no	stack overflow
13	no	non-local search fault
14	no	privileged instruction fault
15	no	pmcall fault
16	no	unassigned
17	no	unassigned

* unavailable in this processor implementation

INTRTN PSD.PTR : INTERRUPT RETURN

Normally, this operator is executed in interrupt mode to return to a previously interrupted user mode program, the currently active stack is the interrupt stack (current contents of TOS point to the top of the interrupt stack). The current machine state is saved in the interrupt PSD. The user mode machine state is restored from the PSD pointed to by PSD.PTR (S0 of the interrupt stack). The interrupts are enabled and the user task resumes in the state when the task was interrupted.

HALT :

This operator causes an interrupt 10'Hex. Interrupt 10'Hex is nonmaskable.

REFMSK : REFERENCE MASK.

This operator references (reads) the mask register (M) from the interrupt controller and pushes the resulting word onto the top of the stack. Bit 0 of the mask register corresponds to interrupt 0, bit 1 to interrupt 1, etc. through bit 15 of the mask register which corresponds to interrupt 15. A bit set to 'zero' implies that the corresponding interrupt is disabled, a 'one' means it is enabled.

Stack Position	Stack Status Initial	Final
S0	A	M
S1	B	A
S2	C	B
S3	D	C

ASNMSK M : ASSIGN MASK.

This operator assigns (writes) M into the mask register of the interrupt controller. Initially M is in the top of the stack. After writing M into the mask register, the stack is popped once. The bit position n of the mask register corresponds to interrupt number n. A bit set to 'zero' implies that the corresponding interrupt is disabled, a 'one' means it is enabled.

Stack Position	Stack Status Initial	Final
S0	M	A
S1	A	B
S2	B	C
S3	C	D

SWPMSK M : SWAP MASK

This operator assigns (writes) M into the mask register of the interrupt controller. The original contents of the mask register are saved on top of the stack. Initially, M is in the top of the stack. That is, the new interrupt mask is swapped with the old mask.

Stack Position	Stack Status Initial	Final
S0	M	OLD.MASK
S1	A	A
S2	B	B
S3	C	C

CLRINT N : CLEAR INTERRUPT.

This operator clears the interrupt specified in the 4 least significant bits of the top of the stack. After the operation the stack is popped once.

Stack Position	Stack Status Initial	Final
S0	N	A
S1	A	B
S2	B	C
S3	C	D

4.8.1.8 Subroutine Linkage Instructions.

CALL X1,...,XN,F : CALL SUBROUTINE F.

This operator transfers program control to a "called" routine and provides for passage of parameters, allocation of dynamic storage on the stack for local variables, and saving of the code descriptor of the "Caller" routine for subsequent return.

a. Initially the top of the stack contains F, the 16-bit program syllable address of the called function or subroutine, and succeeding stack positions contain the respective parameters.

b. A stack adjustment moves TOS so as to provide space for local variables (owns and temporaries). The number, J, of such variables is obtained from the 16-bit header of the subroutine body. (The leftmost byte of the header is masked prior to use.)

c. The subroutine address ("procedure i.d."), LENV and SPCR are stored in the top three stack positions.

d. Initial contents of the SPCR are replaced with the procedure address +4 , effecting a transfer of control to the subroutine.

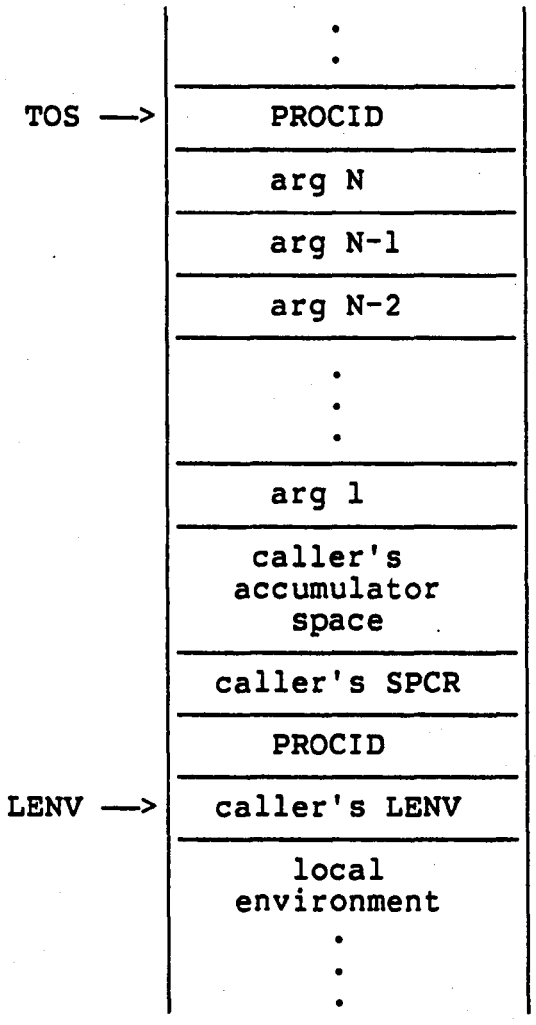
e. The contents of the LENV register are set equal to the stack pointer plus two, TOS+2.

The initial and final stack states are indicated in the Figure on the following page.

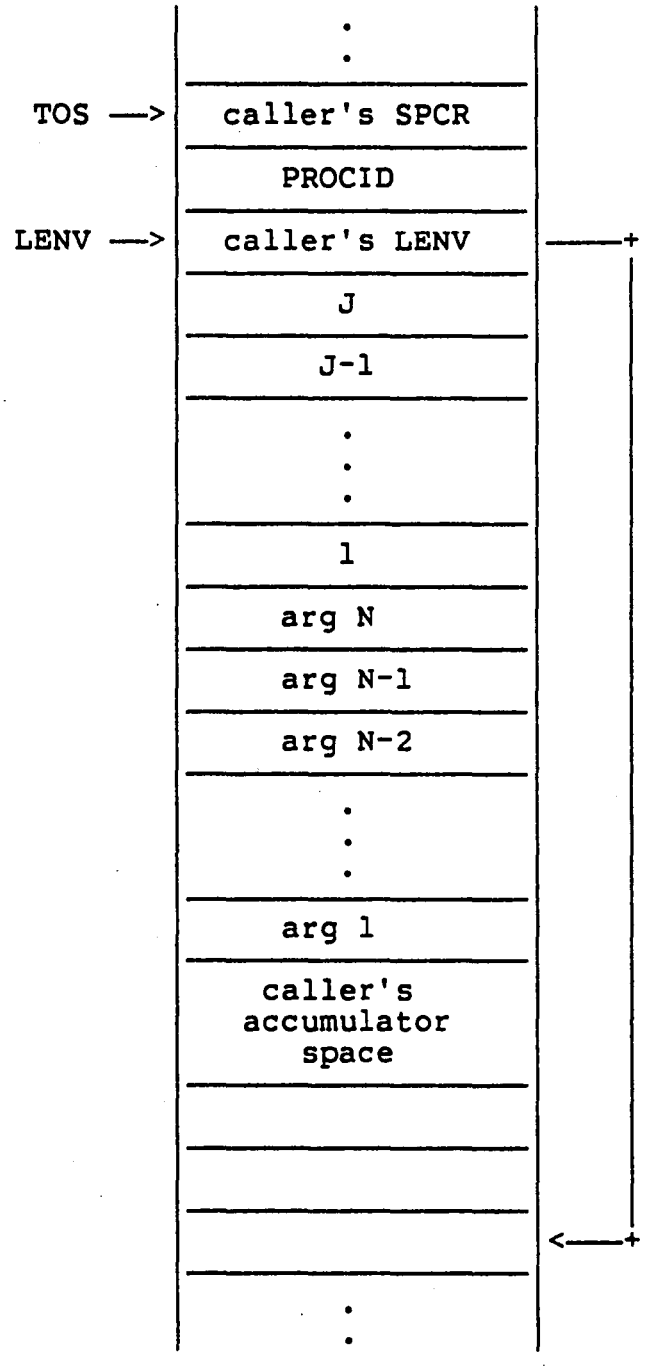
1 F actually points to the least significant byte of the header word of the procedure body.

2 The compiler places a 1 in the leftmost bit position of every procedure body header word, in order to distinguish proc body code from alias beads. This is useful in implementation of DOIT-type mechanisms.

STACK BEFORE EXECUTION OF CALL INSTRUCTION



STACK AFTER EXECUTION OF CALL INSTRUCTION



PMCALL X1,...,XN,N : CALL PRIVILEGED SUBROUTINE N.

This operator transfers program control to a call routine via an index into a table of privileged subroutine entry points. The table is pointed to by the real address 4. The index, N, is verified to be within the table limits. Upon CPU initialization the largest index is stored in real address 5. If the index is valid, the CPU enters the privileged mode state. S0 is replaced with the indexed entry of the privileged subroutines table and the CPU transfers control to the CALL instruction. (See CALL instruction)

Stack Position	Stack Status Initial	Final
S0	N	PROC.ID
S1	arg N	arg N
S2	arg N-1	arg N-1
.	.	.
.	.	.
.	.	.
Sn	arg 1	arg 1

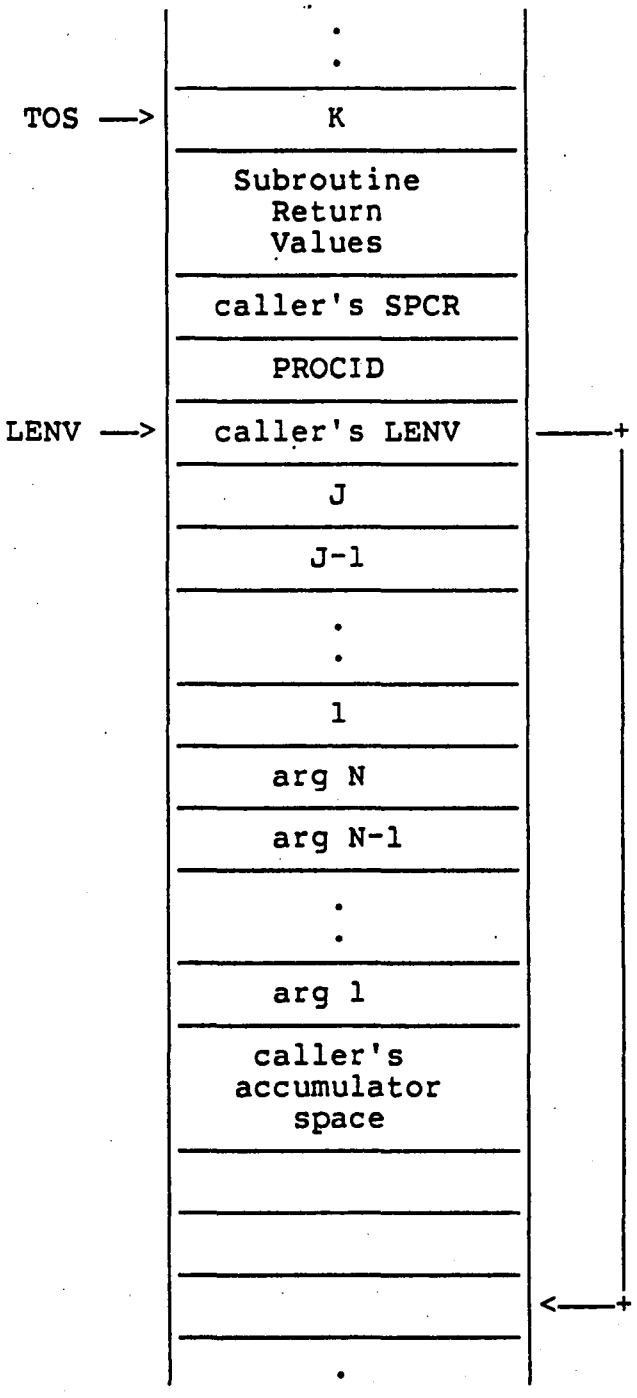
PMOFF : PRIVILEGED MODE OFF.

This instruction compares the privileged mode register, PMR, with the contents of the location pointed to by LENV. If a match occurs, PMR is replaced with zeros and the CPU disables privileged mode.

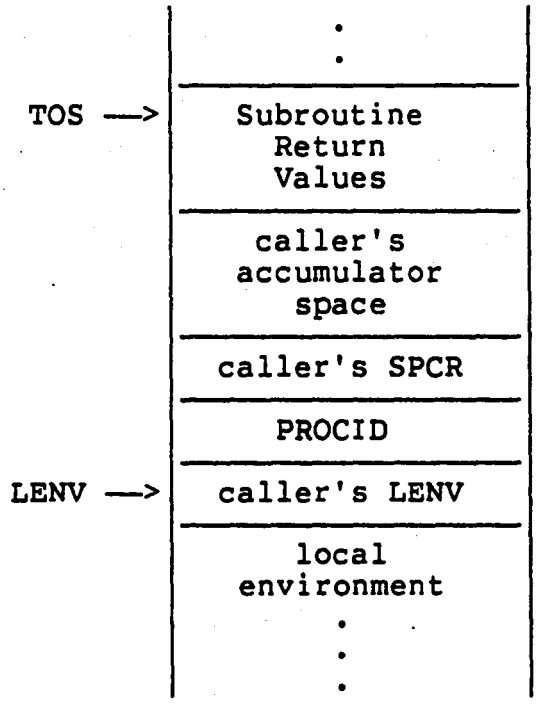
RETURN K : RETURN FROM SUBROUTINE.

This operator provides for re-establishing the necessary state to return to a "caller" routine and to return (in the stack) an arbitrary number of values that may have been generated by the called subroutine. The number of words K (owns plus temporaries plus argument words) to be de-allocated is initially on top of the stack. The SPCR and LENV values of the caller program saved in the stack at the current LENV-2 and LENV locations replace the current SPCR and LENV values. The subroutine's returned values are copied down in the stack as K stack locations are de-allocated (the called programs owns and temporaries plus any arguments). The initial and final stack states are shown in the figures on the following pages.

STACK BEFORE EXECUTION OF RETURN INSTRUCTION



STACK AFTER EXECUTION OF RETURN INSTRUCTION



GOTO A : TRANSFER PROGRAM CONTROL VIA ALIAS BEAD.

The GOTO instruction is used to jump to a program point which is outside the scope of the currently active subroutine. Control is passed by means of an alias bead. The formats for the four types of alias bead are shown below. Initially, S0 contains the byte address of the least significant byte of the first word of the alias bead. If the transfer is through a switch, S1 must initially contain the switch index value. The situation is as follows: The new SPCR value is obtained from the alias bead, as is the LENV if the destination is a recursive procedure. If the destination procedure is nonrecursive, the stack is "unwound" until a "proc i.d." is found in the stack frame which matches that of the alias bead. Then the associated local environment is activated. If the control transfer is to be through a switch, then the index value is moved from S1 of the initial stack to S0 of the resulting stack. The microsequence for the GOTO operator is shown on the following page.

LABEL AND SWITCH ALIASES.

Label Alias, Non-recursive:		
Label alias	Value	0
	VALUE	SPCR of label
	VALUE	procid
Label Alias, Recursive:		
Label alias	VALUE	1
	VALUE	SPCR of label
	VALUE	LENV
Switch Alias, Non-recursive:		
Switch alias	VALUE	2
	VALUE	SPCR of switch
	VALUE	procid
Switch Alias, Recursive:		
Switch alias	VALUE	3
	VALUE	SPCR of switch
	VALUE	LENV

Note: The code in the first word of an alias bead identifies its type. Any code value greater than 3 denotes a procedure; in fact, the leftmost bit of procedure body header word is always set to 1 by the assembler. Note: For recursive alias beads, the LENV (as well as the rest of the bead) is generated dynamically by run-time support software or microcode.

```
GOTO:
   S0 shifted right one ->S0
   M(S0) -> CODE; hardware temporary
   S0 + 1 -> t; hardware temporary
   'POP';
   CODE = 1 or 3 =>
        begin
             M(t+1) -> LENV;
             M(t) -> SPCR;
             go to alias.go;
        end
   M(t+1) -> p; hardware temporary
   LENV -> t1; hardware temporary Loop:  p = M<t1 - 1> =>
        begin
             t1 -> LENV;
             M( t ) -> SPCR;
             go to alias.go;
        end
   M(t1) -> t1;
   t1 = 0 =>
             begin
                   Set program interrupt bit;
                  DONE;
             end
   go to loop; Alias.go: CODE = 2 or 3 =>
             begin
             S0 -> M(LENV-3);
                  LENV-3 -> TOS;
             end
             else LENV-2 -> TOS;
        DONE;
```

4.8.1.9 Loop Control Instructions.

The instructions FOR, STEP, UNTIL, and WHILE described in this section are used to efficiently implement FOR loops in the procedural language. These instructions are used for loops of the form:

```
For I = I1 STEP I2 UNTIL I3 DO . . . .;
                    or
For I = I1 STEP I2 WHILE B3 DO . . . .; The code will be of
the form:
```

```
        LOC I             ;    Address of I
        REF I1            ;    initial value of I
        FOR L,B           ;    Save SPCR and branch to L
          .               ;
          .               ;
          .               ;    DO body code
          .               ;
        REF I2            ;    step I2
        STEP              ;    Increment I by I2
L       EQU  *            ;
        REF I3 or B3      ;    I3 or B3
        UNTIL or WHILE    ;    Test I against I3 or test B3
```

FOR I,I1,<L> : SAVE CURRENT SPCR AND BRANCH TO ADDRESS L.

This operator saves the current contents of the program syllable counter, SPCR, and transfers control to address L by loading SPCR with L. Address L is contained in the 2 bytes following the FOR opcode byte. S0 contains the initial value I1 of the loop variable I , S1 contains the address of the loop variable. S0 is assigned to the location pointed to by S1 and the saved SPCR replaces S0.

Stack Position	Stack Status Initial	Final
S0	I1 (value)	SPCR
S1	I (adr)	I
S2	A	A
S3	B	B

STEP I2 : STEP LOOP VARIABLE I BY AMOUNT I2.

This operator adds the loop increment I2 to the loop variable I. Initially the value of I2 is in S0 and the address of I is in S2. The stack is then popped once.

Stack Position	Stack Status Initial	Final
S0	I2 (value)	SPCR
S1	SPCR	I
S2	I (adr)	A
S3	A	B

WHILE B3 : TEST VARIABLE B3 AGAINST 0 AND LOOP OR EXIT.

This operator tests the boolean variable B3 to determine if the loop should continue or exit. Initially the value of B3 is in S0. If S0 is non-zero the loop continues by popping the stack once and storing S0 (the saved SPCR) in SPCR. Otherwise the loop exits by popping the stack three times.

Stack Position	Stack Status Initial	Loop	Exit
S0	B3	SPCR	A
S1	SPCR	I	B
S2	I	A	C
S3	A	B	D

UNTIL I3 : TEST LOOP VARIABLE I AGAINST I3 AND LOOP OR EXIT.

This operator tests the loop variable I against the final loop value I3 to determine if the loop should continue or exit. Initially the value I3 is in S0 and the address of I is in S2. If the value of I is less than or equal to I3 the loop continues by popping the stack once and storing S0 (the saved SPCR) into SPCR. Otherwise the loop exits by popping the stack three times.

Stack Position	Stack Status Initial	Loop	Exit
S0	I3	SPCR	A
S1	SPCR	I	B
S2	I	A	C
S3	A	B	D

LOOPT L,C : LOOP ON CONDITION TRUE.

This operator provides conditional loop return control (loop on condition true). Initially, the top of the stack (S0) contains a condition, C, and the next stack position (S1) contains a 16-bit absolute program syllable address corresponding to a loop head label, L. If the condition in S0 is true (not all ZEROS), then the address in S1 replaces the current contents of the program syllable count register SPCR and a stack pop adjustment eliminates the condition, C, from the stack. If the condition is false (all ZEROS), then the SPCR is not changed (next program syllable in sequence to be executed; loop exit) and a stack adjustment pops both L and C from the stack.

Stack Position	Stack Status Initial	Loop	Exit
S0	C	L	A
S1	L	A	B
S2	A	B	C
S3	B	C	D

LOOPF L,C : LOOP ON CONDITION FALSE.

The operator provides conditional loop return control (loop on condition false). Initially, the top of the stack (S0) contains a condition, C, and the next stack position (S1) contains a 16-bit absolute program syllable address corresponding to a loop head label, L. If the condition in S0 is false (all ZEROS), then the address in S1 replaces the current contents of the program syllable count register SPCR and a stack "pop" adjustment eliminates the condition, C from the stack. If the condition is true (not all ZEROS), then the SPCR is not changed (next program syllable in sequence to be executed- loop exit) and a stack adjustment pops both L and C from the stack.

Stack Position	Stack Status Initial	Loop	Exit
S0	C	L	A
S1	L	A	B
S2	A	B	C
S3	B	C	D

4.8.1.10 Outer Block Data Transfer Instructions.

REFNSE<K,P> : EXTENDED REFERENCE NONLOCAL ENVIRONMENT - SINGLE WORD.

References (reads) a 16-bit word from the Nonlocal Environment portion of memory identified by the second and third syllables following the REFNSE syllable. The base address used is the address of the nonlocal stack frame located by matching the procedure I.D., P, to the proc. I.D. stored in the stack frame. If no match is found, a program interrupt will be generated. K, the value of the byte that follows the REFNSE syllable, is added to the base.

Stack Position	Stack Status Initial	Final
S0	A	(LENV@P+K)
S1	B	A
S2	C	B
S3	D	C

REFNDE<K,P> : EXTENDED REFERENCE TO NONLOCAL ENVIRONMENT - DOUBLE WORD.

References (reads) a double word (two 16-bit words) from the Nonlocal environment identified by P, the second and third syllables following the REFNDE syllable. The base address used is that of the nonlocal stack frame whose procedure I.D. matches P. K, the value of the byte following the REFNDE operation syllable, is the index used. The sum of base and K points to the least significant half of the double word, while the sum of base and K + 1 points to the most significant half of the double word. An interrupt occurs if no match on P is found.

Stack Position	Stack Status Initial	Final
S0	A	(LENV@P+K)
S1	B	(LENV@P+K+1)
S2	C	A
S3	D	B

ASNNSE <K,P>,V : EXTENDED ASSIGN TO NONLOCAL ENVIRONMENT - SINGLE WORD.

The 16-bit word in the top of the stack is read destructively and stored (assigned) into memory in the Nonlocal environment identified by P, the second and third syllables following the ASNNSE syllable. The base address used is that of the nonlocal stack frame whose proc I.D. matches P. K is the value of the byte following the ASNNSE syllable and is added to the base to form the address of the resulting store. A program interrupt is generated if no match on P can be found.

Stack Position	Stack Status Initial	Final
S0	V	A
S1	A	B
S2	B	C
S3	C	D

ASNNDE <K,P>,V' : EXTENDED ASSIGN TO NONLOCAL ENVIRONMENT - DOUBLE WORD.

The double word, V', (two 16-bit words) is initially in the two top words of the stack; the least significant half is S0, and the most significant half is S1. These two words are read destructively from the stack and stored (assigned) into two consecutive words of memory in the nonlocal environment identified by P, the second and third syllables following the ASNNDE syllable. The base address used is that of the nonlocal stack frame whose procedure I.D. matches P. K, the value of the byte following the ASNNDE byte, is added to the base to form the address of the least significant word of the resulting store. An interrupt occurs if no match on P can be found.

Stack Position	Stack Status Initial	Final
S0	V'(LS half)	A
S1	V'(MS half)	B
S2	A	C
S3	B	D

NLOCL I,P : GLOBAL RELATIVE ADDRESS OF NONLOCAL WORD.

Generates an address (relative to the Global Environment) that is the Ith word of the nonlocal environment identified by procedure I.D. P. This 1-byte instruction is analogous to the LOCL instruction, but refers to nonlocal environment rather than to local environment. The two top words of the stack must indicate (a) in S1, the environment being referenced, and (b) in S0, the relative displacement of the required variable within its environment. The stack is popped two places and then the global relative address of the requested variable is pushed onto the stack.

Stack Position	Stack Status Initial	Final
S0	I	LENV@P+I
S1	P	A
S2	A	B
S3	B	C

4.8.1.11 Stack Management Instructions.

POP N : POP STACK

This operator pops the stack by one word. The value of TOS is incremented by one.

DUPS V : DUPLICATE TOP OF STACK, SINGLE WORD.

The single word duplicate top of the stack operator executes a stack push operation and copies the initial contents V (16-bit word) of the top of stack into the location which is the new top of stack. The value V and its duplicate are left in the top two stack positions (S1 and S0).

Stack Position	Stack Status Initial	Final
S0	V	V
S1	A	V
S2	B	A
S3	C	B

DUPD V' : DUPLICATE TOP OF STACK, DOUBLE WORD.

The double word duplicate top of stack operator executes a push 2 operation and copies the double word V' initially in the top two positions of the stack into the corresponding two new positions at the top of the stack. The double word V' and its duplicate are left in the top four words of the stack.

Stack Position	Stack Status Initial	Final
S0	V'(LS HALF)	V'(LS HALF)
S1	V'(MS HALF)	V'(MS HALF)
S2	A	V'(LS HALF)
S3	B	V'(MS HALF)

EXCHS U,V : EXCHANGE, SINGLE WORD.

This operator exchanges the two words currently on top of the stack.

Stack Position	Stack Status Initial	Final
S0	U	V
S1	V	U
S2	A	A
S3	B	B

EXCHD U',V' : EXCHANGE, DOUBLE WORD.

This operator exchanges the two double-length words currently on top of the stack. That is, S0 and S2 are swapped, and S1 and S3 are swapped.

Stack Position	Stack Status Initial	Final
S0	U'(LS HALF)	V'(LS HALF)
S1	U'(MS HALF)	V'(MS HALF)
S2	V'(LS HALF)	U'(LS HALF)
S3	V'(MS HALF)	U'(MS HALF)

NOP : NULL OPCODE.

This operator has no effect on the stack contents. Control is simply passed to the next instruction in the program sequence.

CHAPTER 5. - Slave Region Design and Operation

This chapter discusses the Slave Region design and operation. Excepting for the BGU's, all devices which are addressed and accessed by means of the system bus are within the Slave Region. These devices or modules are system memory modules, system I/O ports, real-time clock/counters, and LRU control, status and communications registers. Each of these units is interfaced to a transfer bus which is internal to the Slave Region. A slave coupler couples this slave transfer bus to the system bus.

5.1 Slave System Bus Coupler

The Slave System Bus Coupler functions to accept the serial system bus commands, either reads or writes, convert them to the parallel format of the slave region transfer bus, and to control that transfer bus executing the indicated read or write to devices attached to the slave region transfer bus.

The slave bus coupler first converts the serial address of a system bus read or write into a parallel format and then maps this 19 bit address into a 16 bit address. In the case of a system bus write, it then converts the serial data word to a parallel format and executes a transfer bus write of that word to the mapped 16 bit address. Devices attached to the slave transfer bus are responsible for recognizing their own addresses (in the mapped address space) and storing the data addressed to them. In the case of a system bus read, the slave coupler initiates a transfer bus read using the mapped address. Devices attached to the slave transfer bus are responsible for recognizing their own addresses (within the mapped address space). A device which recognizes its own address responds with the requested data word and a positive acknowledgment. If the slave coupler receives a positive acknowledgment it accepts the data word, converting it to the serial format used on the system bus, and transmits it on the R bus outputs of the slave coupler. These R bus outputs are then gated directly to the system bus R line(s) where appropriately enabled by that LRU's BGU R line enabling registers.

Operation of the slave coupler, the slave transfer bus, and of the attached devices is synchronous with the LRU system clock. The arrival of a system bus command serves to synchronize the operation of the slave coupler to a particular frame of the LRU

system clock. In those situations where several slave couplers must respond to a system read or write simultaneously, as in the case of a system memory triad read or write, this synchronization to a particular frame of the LRU system clocks of each of the responding LRU's serves to assure simultaneous and tightly synchronized operation of each element of the responding triad, since the LRU system clocks of all LRU are synchronized by the system clocking mechanism. Thus the slave region is automatically synchronizing. No special action is required to achieve tightly synchronized operation of slave region triads, as is required to acheive processor triad synchronization.

The mapping of the 19 bit system bus address into a 16 bit mapped address, which is used internally on the slave transfer bus, is relatively simple. In the case where the five most significant bits of the system bus address equal the content of the five least significant bits of the system memory relocation register (an LRU control register) the mapped slave transfer bus address is created by directly using the 14 least significant bits of the system bus address as the 14 least significant bits of the slave transfer bus address and using 00 as the two most significant bits of the 16 bit slave transfer bus address. The 16K system memory module will respond to transfer bus addresses "0000" through "3FFF". The writing of the LRU system memory relocation register thus determines which 16K block of the system memory address space the system memory module of an LRU will respond to.

In the case where the 5 most significant bits of the system bus address do not equal the content of the relocation register the slave transfer address is constructed as follows. System bus address bits 18 through 10 are 'AND'ed together. If this result is 'zero' then 01 is used as the two most significant bits of the slave bus address and the 14 least significant bits of the system bus address are used as the 14 least significant bits of the slave bus address. There are no devices which respond to these addresses in the current implementation. If this result is 'one' then slave address bits 15, 13, 12, 11, and 10 are set to 'one', slave address bit 14 is set equal to system bus address bit 8, and slave address bits 9 and 8 are set equal to system bus address bits 9 and 8 except when the system bus address bits 9, 8, 3, 2, 1 and 0 equal 100011 in which case slave address bits 9 and 8 are set equal to 01. Figure 5.1 summarizes this mapping. Note that this entire mapping procedure is transparent to the processor triad and that for all practical purposes slave transfer bus devices can be treated as if they responded to system bus addresses directly. For this reason, all other sections of this volume discuss device function and system bus address assignments as if this were the case. Figure 5.2 summarizes the system bus address assignments for all slave units.

if rrrrr = reloc reg.

system bus address		slave bus address
rrr,rraa,aaaa,aaaa,aaaa	—>	00aa,aaaa,aaaa,aaaa

if rrrrrssss =|= 111111111

system bus address		slave bus address
rrr,rrss,ssaa,aaaa,aaaa	—>	01ss,ssaa,aaaa,aaaa

if rrrrrrrrr = 111111111 and abdddd =|= 100011

system bus address		slave bus address
rrr,rrrr,rrab,cccc,dddd	—>	1b11,11ab,cccc,dddd

if rrrrrrrrr = 111111111 and abdddd = 100011

system bus address		slave bus address
rrr,rrrr,rrab,cccc,dddd	—>	1011,1101,cccc,0011

Figure 5.1 System Bus Address to Slave Transfer Bus Address Mapping

18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	1	1	1	0	0	0	reg sel		LRU			

ERROR LATCH STATUS REGISTER ADDRESS

18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	1	1	1	0	$\overline{\text{LRU}}$				reg sel			

I/O PORT ADDRESSES

18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	1	1	1	1	LRU				reg sel			

LRU CONTROL REGISTER ADDRESSES

18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	1	1	1	1	1	1	triad			0	reg sel	

PROCESSOR TRIAD COMMUNICATION REGISTER ADDRESSES

Figure 5.2 System Bus Address Assignment Summary

5.2 System RAM

Each LRU in the FTMP contains 16K words of system memory. The system memory is made up of 1K x 4 bit CMOS RAM chips. It is backed by battery power to provide non-volatile storage. The memory can be read or written over the system bus. It responds to a 19-bit address when the 5 most significant bits of the 19 bit address match the 5 low order bits of the MRR control register. The 14 low order bits of the address, which can range from 0000 to 3FFF, select a word from the 16K memory array. Figure 5.3 illustrates the address format for accessing system memory.

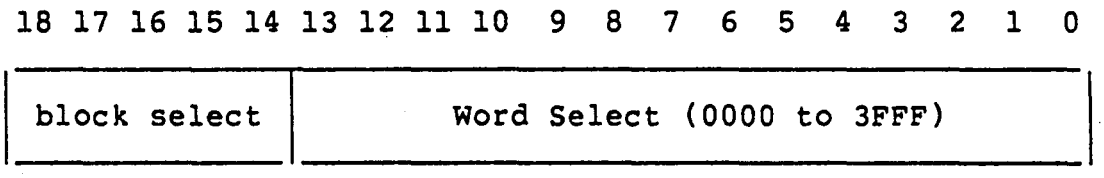


Figure 5.3 System Bus Address Format of System Memory

5.3 Real Time Clock

An accurate time reference is provided in the FTMP by a real time clock with a resolution of 250 microseconds. Logically, the real time clock is a 32-bit wide register that is incremented every 250 microseconds. Thus if reset, the clock can be incremented continually for approximately 12-1/2 days before overflowing.

The clock can be read on the system bus by addressing locations "7FFFF" (high order clock word) and "7FFFE" (low order word). A read from 7FFFE latches the state of the clock (all 32 bits) and the low order word is gated to the bus. A subsequent read from "7FFFF" gates the high order word that is already in the latch onto the bus. Each LRU has a 32-bit clock register but only one triad of LRU's is assigned to respond to reads of the real-time clock. All LRU's respond to writes to the real-time clock. An LRU may be armed to respond to clock read requests by setting bit 5 of its Memory Relocation Register to 1. When the least significant word of the real-time clock ("7FFFE") is written, the divider network, which is used to derive the 250 microsecond clock from the system clock, is cleared and held at zero. When the most significant word ("7FFFF") is then written the divider network is released. Since all real-time clocks and their associated divider networks respond to writes in synchronism with one another, a write serves to synchronize all clocks to one another. Furthermore, since they are all incremented by identical derivatives of the system clock time base, they will then stay synchronized. If an element of the clock triad fails it is not necessary to reinitialize the real-

time clock system in order to effect repair. The failed unit can be replaced by removing it from the system and assigning another unit in its place. That unit is already operating in synchronism with the elements of the real-time clock/counter triad.

The real-time clock is volatile and the loss of primary power causes its contents to be lost.

5.4 Control, Communication and Status Registers

There are 17 control, communication and status registers in each LRU. These are described in the following three sub-sections.

5.4.1 Control Registers

There are nine control registers within each LRU. Each register is used to perform control specific LRU functions. These registers are shown in Figure 5.4.

Registers 0 to 3 are 4-bit wide CPU control registers. The least significant bit of Register 0 controls the reset/run state of the CPU. The next 3 bits of Register 0 contain the processor triad identification assignment for the processor region. CPU control registers 1, 2 and 3 are presently unassigned.

Control registers 4 to 7 are 4-bit wide line select registers. Registers 4 and 5 are used by the processor region bus controller. Register 6 is used by the I/O region bus coupler and register 7 is used by the clock region bus coupler. Each four bit code designates which three of five lines are to be selected by the input voting circuitry. Figure 5.5 summarizes the code to line select relationship. The codes to line select mapping is identical from all lines. This code to line select mapping is also identical to the code to line select mapping used by the BGU's, however the select register and input circuitry of the BGU's are separate from this control and input circuitry.

Register 8 is a 6-bit system memory relocation register. The 5 low order bits of this register form the 5 high order bits (bits 14 to 18) of the 19 bit address space to which system memory in this LRU responds on the system bus. The most significant bit of the relocation register is used to arm/disarm the real time clock of the LRU. If the MSB is a 'one', the LRU responds to 'read clock' requests. Otherwise, read clock requests are ignored.

The system address format of control registers is shown in Figure 5.6. As can be seen from this figure, the address of the registers is LRU specific. Therefore, the contents of the control registers in a given LRU can be altered independent of other LRU's. The control registers can not be read over the system bus.

Reg. #	Register Function
0	CPU Control 0
1	CPU Control 1
2	CPU Control 2
3	CPU Control 3
4	P Select
5	R Select
6	T Select
7	C Select
8	Memory Relocation

Figure 5.4 Control Registers

Select Code	Selected Bus Set
0	1, 2, 4 <
1	1, 2, 5 <
2	1, 3, 4 <
3	1, 3, 5 <
4	2, 3, 4 <
5	2, 3, 5 <
6	2, 4, 5 <
7	3, 4, 5 <
8	1, 2, 3 <
9	1, 2, 3
A	1, 3, 3
B	1, 3, 3
C	2, 3, 3
D	1, 2, 3
E	3, 4, 5
F	1, 4, 5 <

< : legal select codes

Figure 5.5 Select Code to Selected Bus Set Mapping Table.

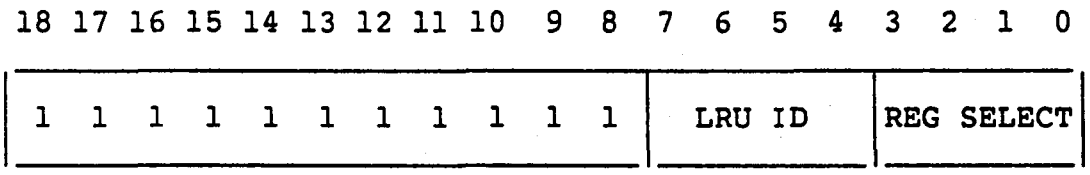


Figure 5.6 System Bus Address Format of Control Registers

5.4.2 Communication Registers

There are four Inter-Processor triad Communication (IPC) registers in each LRU which provide a direct communication link between processor triads. The registers are 4 bits wide. Their system bus address format is shown in Figure 5.7. IPC register addresses are keyed to specific processor triad identification. Therefore the IPC registers of any LRU with the appropriate processor triad identification will respond to a system bus write to an IPC register. Since all members of a processor triad have the same processor triad assignment, all members of a processor triad will receive the IPC write simultaneously. Writes to IPC register 2 or 3 of the LRU will pend an interrupt for the processor. The IPC registers can not be read over the system bus.

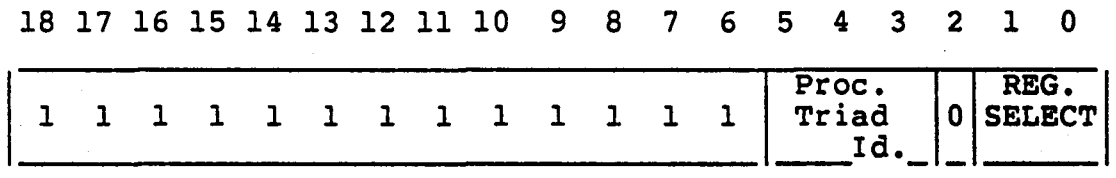


Figure 5.7 System Bus Address Format of IPC Registers

5.4.3 Status Registers

There are four status registers in each LRU and are known as the P, R, T and C bus error latches. The error latches are 5 bits wide. They can be read over the system bus. The address format is shown in Figure 5.8. Error latches have LRU specific addresses and only one LRU responds to a given error latch read request. Reading an error latch clears that latch to zero. The latches can not be written into via the system bus. The bits in the error latches are set in response to pulses from the associated input circuitry; the system bus controller input circuitry in the case of the P and R error latches, the slave system bus coupler input circuitry in the case of the T error latch, and the clock generator input circuitry in the case of the C error latch. Bit 0 set corresponds to an error on bus set 1, bit 1 to bus set 2, bit 2 to bus set 3, bit 3 to bus set 4, and bit 4 corresponds to an error on bus set 5. An error latch is reset when it is read.

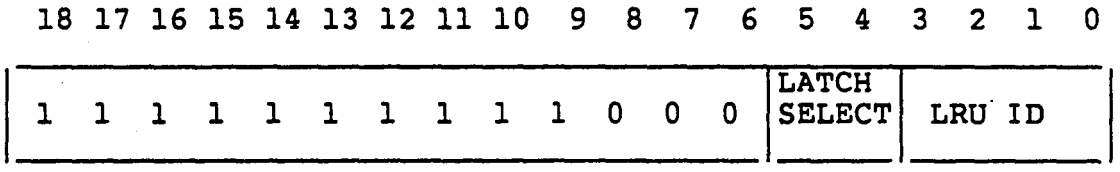


Figure 5.8 System Bus Address Format of Status Registers

5.5 I/O Port

Each LRU in the FTMP contains an Input/Output port. The I/O port links the LRU to remote terminals (RT) via an external 1553A bus. The port contains three 16-bit wide registers and a 32-word long first-in first-out (FIFO) type data buffer. The functions of the three registers are as follows. Register 0 (R0) is the 1553 command register. Register 1 (R1) is the port status register and register 2 (R2) is the port control register. These registers can be accessed on the system bus using the address format shown in Figure 5.9. The command and control registers (R0 and R2) are write only while the status register (R1) is a read only register. The FIFO buffer can be read and written over the system bus using the address shown in Figure 5.10. As can be seen from the figure the FIFO buffer in a given LRU responds to a single system bus address. The buffer as well as the registers have LRU specific system bus addresses and only one LRU responds to a given I/O port command.

Section 5.5.1 describes various fields in the command, control and status registers. Section 5.5.2 describes the operation of the I/O port to communicate with the remote terminals.

5.5.1 I/O Port Registers

The various fields of the 1553 command register are shown in Figure 5.11. Bits 11 to 15 (5 high order bits) of R0 contain the address of the remote terminal to which the transmission is directed. Bit 10 is the 'Receive/Transmit' bit. It determines the direction of the transmission. If bit 10 is zero the I/O port will transmit; if one the I/O port will receive. Bits 5 to 9 are the RT subaddress. If all zero then it is a mode command. Bits 0 to 4 (5 low order bits) contain the number of words to be transmitted. If all 5 bits are zero, the number is 32.

Figure 5.12 shows the detailed fields of the port status register. Bit 15 (the most significant bit) is the 'Ready' bit. It is set when the port is ready for another 1553 transaction sequence. Bit 14 is the 'Bus Busy' bit. It is set if a command word transmission is attempted while the 1553 bus is busy. Bit 13 is the 'Buffer Full' bit. It is set when the FIFO buffer is full, that is, contains 32 words. Bit 12 is the 'Buffer Empty' bit and is set if the FIFO is empty. Bit 11 is the 'Word Count Error' bit. It is set if a word count error is detected. The count error occurs if 1) the port transmitted a 'Receive' command and too few words were contained in the FIFO to transmit the required word count or 2) the port transmitted a 'Transmit' command and the RT transmitted too few or too many words. If the FIFO contains more words than required before a transmission no count error will occur. Bit 10 is the 'Message Error' bit. It is set when a message error during a 1553 transaction is detected. A number of different errors can cause this bit to be set. This bit will be set if the RT fails to send a status reply within 15 microseconds after the completion of the command word transmission. The message error bit is also set if a command word transmission is attempted when the 1553 bus is busy. This also sets the 'Bus Busy' bit, explained earlier. A word count error also sets the message error bit. Bits of 0 to 9 (the 10 low order bits) of R1 contain the 1553 RT status reply. Upon completion of a 1553 transaction the remote terminal sends its status into these bits.

A write into the port command register clears the status register to zero.

Figure 5.13 shows the format of the port control register. Bit 15 of the control register is the 'Reset' bit. Setting this bit clears the command and status registers and the FIFO data buffer. The reset bit itself is cleared by the port after the port reset sequence has been completed. The reset sequence may take up to 55 microseconds if the buffer is full. After a reset, the 'Empty' bit will be set in the status register indicating that the data buffer is empty and the 'Ready' bit will be set indicating that a transaction sequence may begin. Bit 0 of the control register is the 'Flush Buffer' bit. Setting this bit empties the FIFO buffer. The bit is reset by the port when it has completed emptying the buffer.

5.5.2 I/O Port Operation

All 1553 data words are written to or read from the 32-word FIFO buffer. To send a message to a remote terminal, it is necessary first to load the buffer with the data to be sent, assuming that the I/O port has already been reset by setting the 'Port reset' bit in the control register. The next step then is to check the 'Ready' bit in the status register to make sure that the port is ready for a 1553 transaction. If so, the command register is loaded with the RT address, the word count and Receive command since the RT is going to receive the message. Writing into the command register clears the status register to zero, causes the contents of the command register and 'word count' number of words from the FIFO buffer to be transmitted over the 1553 bus. At the completion of the I/O transaction the status register contains the status reply from the remote terminal. The status register also contains the status of the transaction. That is, 'Message Error', 'Word Count Error' and other bits in the status register are set to indicate whether the transmission was successful.

To receive a message from an RT, a 'Transmit' command is loaded into the command register after the data buffer has been cleared and the 'Ready' bit of the status register has been checked. Writing into the command register causes the contents of the command register to be sent on the 1553 bus. After the data from the RT has been received in the port buffer, the status register would contain the RT status reply and the outcome of the transaction.

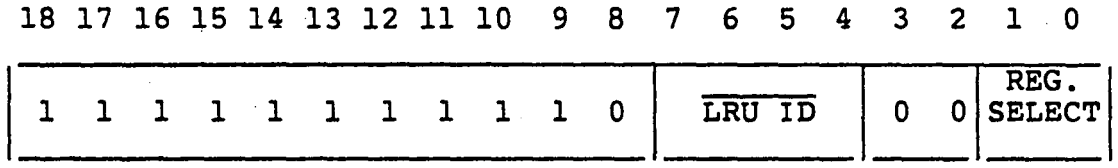


Figure 5.9 System Bus Address Format of I/O Port Registers

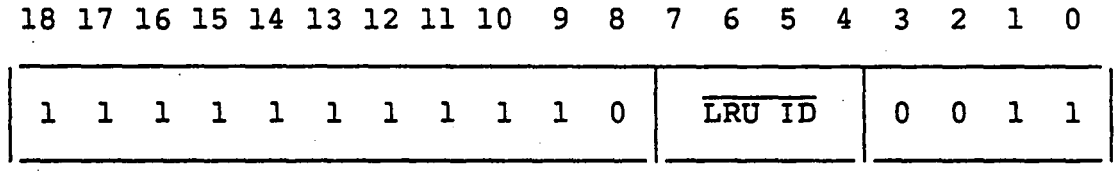


Figure 5.10 System Bus Address Format of FIFO Buffer

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RT ADDRESS					XMIT/ RECV	RT SUBADDRESS					WORD COUNT				

Figure 5.11 I/O Port Command Register(Reg. 0)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRT RDY	BUS BSY	BUF FUL	BUF ZRO	WRD CNT ERR	MSG ERR	REMOTE TERMINAL STATUS REPLY									

Figure 5.12 I/O Port Status Register (Reg. 1)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESET BIT	UNUSED														FLUSH BUFFER

Figure 5.13 I/O Port Control Register (Reg. 2)

CHAPTER 6. - Clock Generation Region Design and Operation

All circuitry of the LRU (except for the BGU's) use or rely upon the timing base provided by the clock region. This region provides the local LRU version of the 1 MHz. system clock, as well as clock multiples such as 16 MHz., 8 MHz., 4 MHz., 2 MHz. and the submultiples such as 500 KHz. All of these auxiliary frequencies are phase related to the local system clock by the divider and generation mechanism which are used to create them.

The heart of the clock region is a voltage controlled crystal oscillator, with a nominal center frequency of 16 MHz. All of the subfrequencies including the basic 1 MHz. system clock are created by dividing this source signal. The crystal oscillators are accurate to .001 % and may be pulled via the control voltage input by .01 %. Such variations in the clock frequencies are insignificant relative to the correct operation of the FTMP circuitry. Use of the system clock as the ultimate time base for the real-time clock/counter limits its accuracy to that of the system clock signal. In the worst case, this implies an error of about 1 minute per day. Such an error could manifest itself into navigation errors as large as four nautical miles over a ten hour mission. Typical performance should be about four or five times better than this worst case number.

Each clock generation region synchronizes with the clock generation regions of the other LRU's by phase locking its internal 1 MHz. system clock to the system bus timing signals. Certain of these LRU's serve as the sources of system bus timing signals when their internal 1 MHz. system clocks are gated onto system bus C lines. Others simply listen to the C bus and have no effect on its content. In either case an LRU generates an internal reference signal by receiving all five C lines, selecting three of these five and passing them through simple majority logic. This reference signal is then phase compared to the internal system clock. If the system clock leads the reference signal, an error signal proportional to this lead is generated and applied to the voltage control input of the crystal oscillator. This signal tends to depress the operating frequency of the oscillator. If the internal system clock lags the reference signal the error signal, proportional to the lag, is in the direction such that it elevates the operating frequency of the oscillator. The code used by the input select circuitry in determining which three of the five C lines are to be gated to the voter is provided by an LRU control register of the slave

region. Figure 6.1 illustrates the basic organization of the clock generation region. Figure 6.2 depicts this block diagram using classical control block representations for its operation.

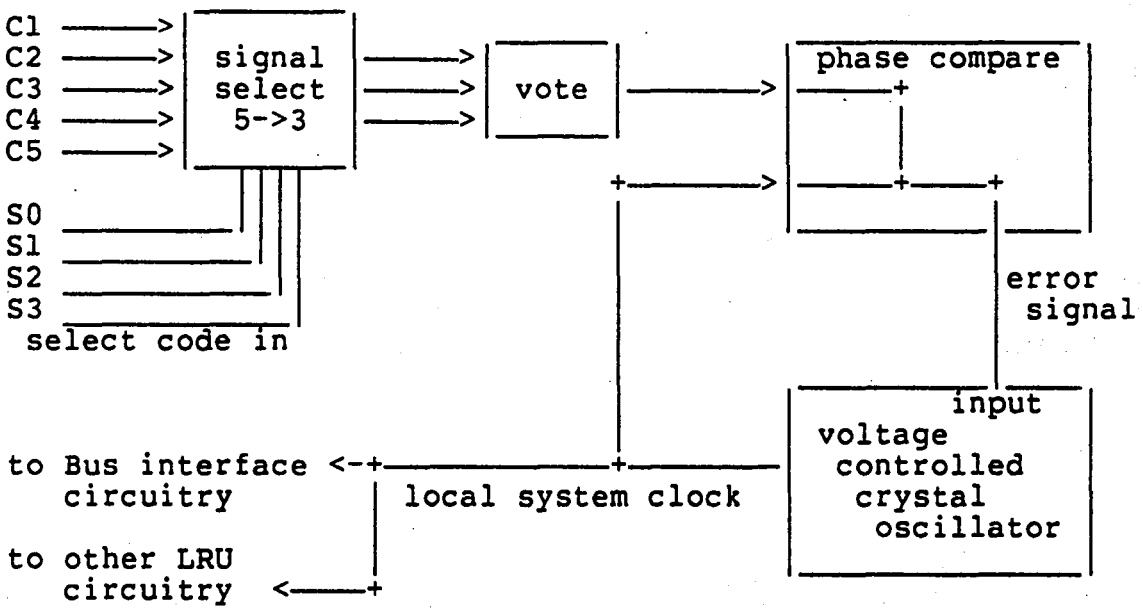


Figure 6.1 Clock Generation Region.

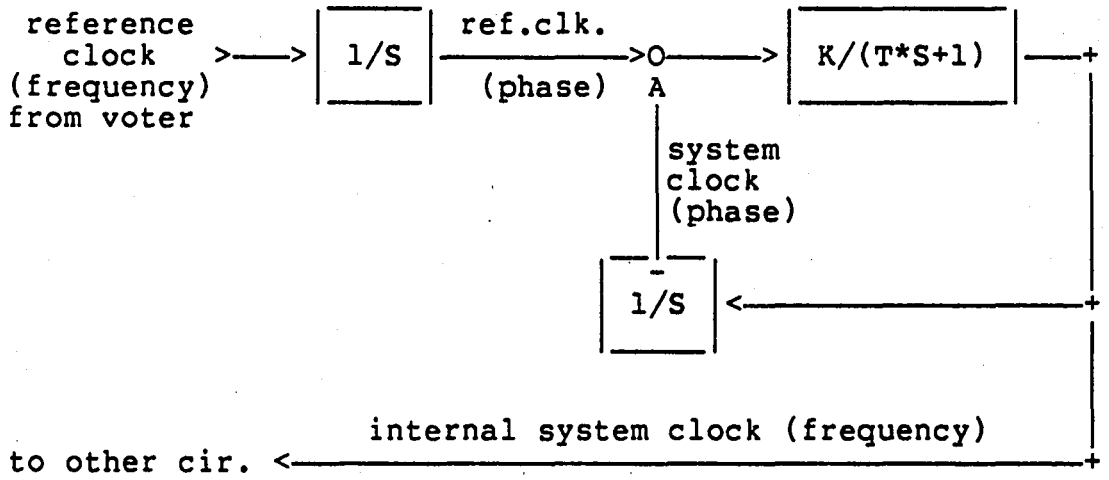


Figure 6.2 Clock Generator (Frequency/Phase Control Diagram)

The transfer function of such a system is:

$$(\text{freq. in})/(\text{freq. out}) = K / (T*S**2 + S + 1)$$

As a minimum for clock system stability this loop should be overdamped. Damping is:

$$D = .5 / \text{SQRT}(T*K)$$

The K chosen for this implementation is:

$$K = .1 \text{ Hz./deg. phase error}$$

or dimensionally more correct

$$K = 37 / \text{sec.}$$

which implies:

$$T < .006 \text{ sec.}$$

The T's for the voltage controlled crystals are all comfortably below this limit.

The phase locked loop also serves to filter the reference signal, eliminating the effect of high frequency glitches and asymmetric duty cycles. The output of the voter circuit may be asymmetric or have high frequency spikes due to legitimate skews between valid clock signals and the effect of a failed clock line on the operation of the voter circuit. The outputs of the crystal oscillator and its various divider circuits are always symmetric and the phase lock loop rejects frequency components in the reference signal which are beyond the capture range of the crystal. It is this basic unsuitability of the raw reference signal which compels the use of the crystal output for internal timing instead of the direct use of the reference for LRU clocking functions.

The clock generation regions of multiple LRU's may be used in either of two ways to create the common C line timing signals. Four LRU's may be selected as the clock quad, with the output of each LRU's clock generator being gated onto a different C line. Each LRU then selects and votes on the C lines being used by the other three LRU's. Configured in this fashion, the clock generators of the clock quad effectively lock to each other so that their transmissions onto the C lines are of the same frequency and are in phase with one another. Any single failure of the clocking system, either in a clock generator or in the bussing distributing the clocks can at most disable one of the C line bus signals. Any LRU which is not a member of the clock quad can synchronize its internal system clock to that of the clock quad by selecting and voting on any three of the four active bus lines to create its reference signal. All functioning LRU's can therefore maintain clock synchronism with one another

despite any single fault in either the clock quad or the bus system. When such a fault occurs the system can be repaired by assigning one of the functioning but passive LRU generators to take over transmissions onto a C line in place of a failed generator, or by using the spare C line in place of a failed C line.

An alternate configuration is to select three clock generators to function as a clock triad. Each element of the clock triad transmits onto a different C line. Each element of the clock triad selects and votes on the same C lines. Thus the reference signal used by a clock triad member is a combination of, the signals from the other two elements of the triad and one's own signal. The reference signal used by the LRU's which are not members of the clock triad is simply derived by voting on the three signals from the clock triad. The clock generators of all LRU's will remain synchronized with one another despite any single failure of a clock generator or any single failure of the bussing system, excepting for certain pathological and remote failure events. It is because of these pathological failure events that the clock quad configuration is preferred and the clock triad is only used when inadequate buses or generators remain to construct a fault free clock quad.

Figure 6.3 illustrates the interconnections to create a clock quad. Figure 6.4 illustrates the interconnections to create a clock triad.

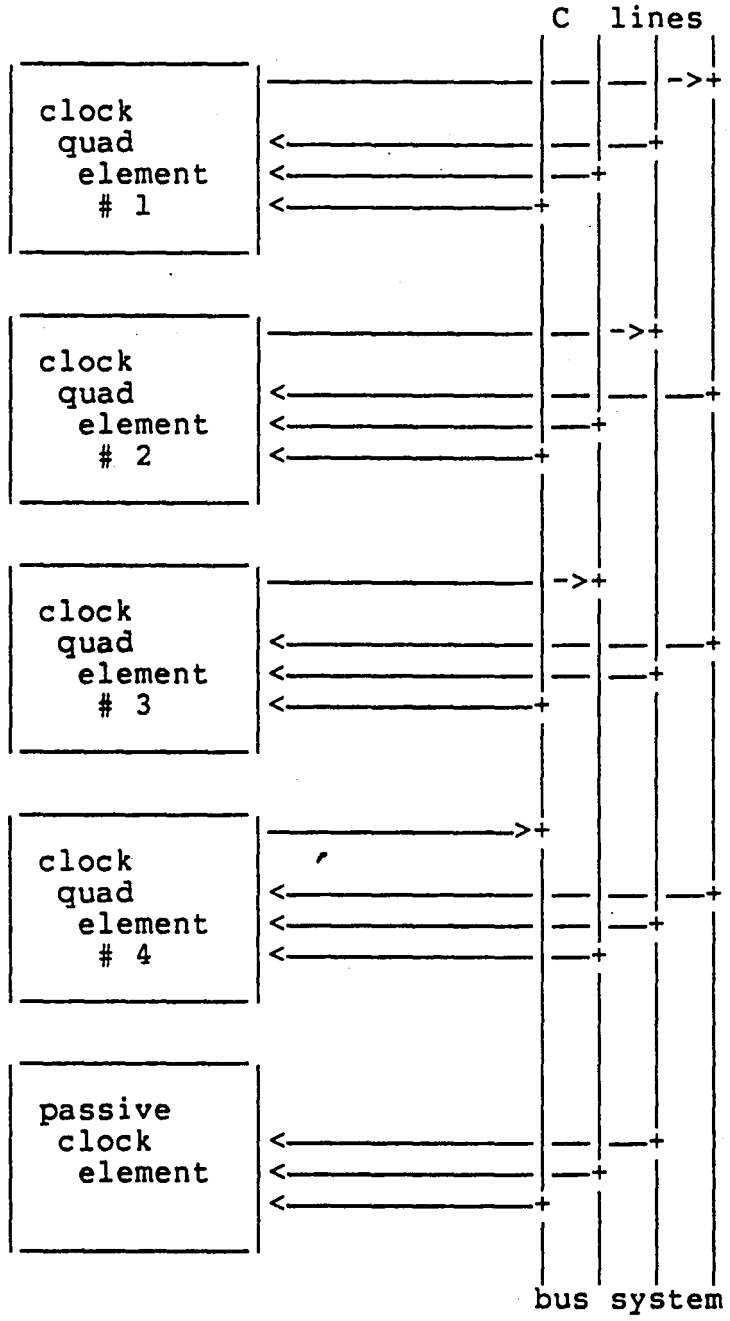


Figure 6.3 Clock Quad Interconnections

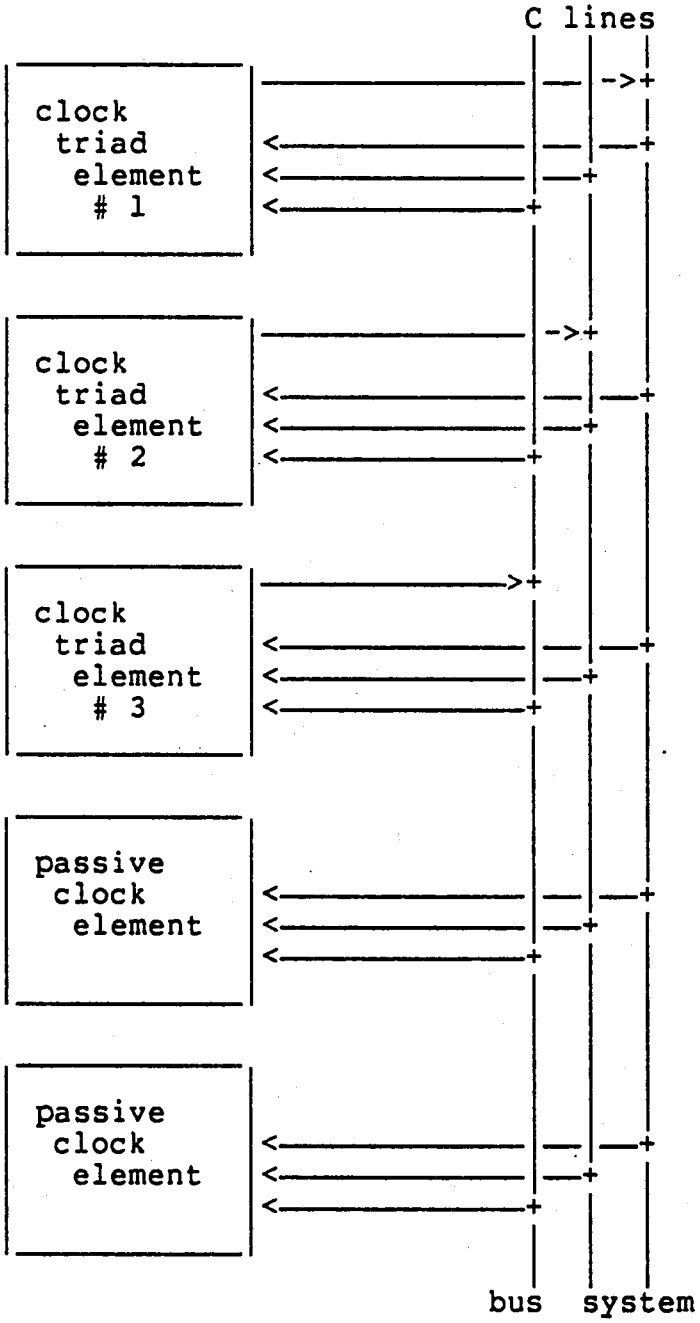


Figure 6.4 Clock Triad Interconnections

CHAPTER 7. - Power System

The power system consists of four primary power supplies, which function to provide quad redundant power to each LRU of the system, local power regulators in each LRU, and local battery backup within each LRU for maintaining system memory and system configuration registers during primary power interruptions.

Each of the four primary power supplies accepts 208 VAC, 400 Hz three phase power and outputs 28 VDC onto a different one of the four primary power buses. Each power supply can sustain 150 amperes output current indefinitely and in excess of 1500 amperes peak output current for short periods. Total power consumption of all LRU's can be sustained by any one of the four primary power supplies. Four primary power buses distribute power to all LRU's of the system. Each primary power supply drives a different power bus and each LRU has access to all power buses.

Each LRU has a power regulator which accepts power from the quad redundant primary power buses, and converts it to the appropriate LRU internal voltages. Each power regulator can draw power from any one of the four primary power buses, and will maintain internal LRU voltages within regulation as long as at least one of the power buses is within its input operating range of 18 to 40 VDC. Within each LRU, the combining of primary power is provided in a two stage process. First, two pairs are combined by a diode junction. Secondly, a dual switching regulator draws power from this intermediate pair, combining them to produce the regulated internal power. Each primary power bus is fused as it enters the LRU so that any internal short circuits within an LRU could at most only provide a momentary disturbance on the primary power buses before the fuses blew. Energy storage within the regulator is such that the LRU internal voltages will remain within regulation in the event that another LRU briefly shorts all primary power buses. Due to the nature of the diode junction which is used to combine the power from two buses, it is impossible to assure that most of the power being drawn from the diode junction is not being sourced from only one bus of the pair. If diode junctions were used exclusively to combine power it would in fact be difficult to assure that all power for all LRU's was not being drawn from one supply. The second stage dual switch regulator provides the means for balancing the power drain from each of the primary power supplies. Within the LRU, the dual regulator draws nearly equal power from each of the two diode junctions. Since it is possible that all of the power

drawn from a diode junction may be sourced from only one bus, a single LRU may under worst case draw about half its power from one power bus. If all LRU's combined the same pairs of primary power buses, then it would be possible that only two power supplies were driving the entire system. Each LRU is configured to combine differing pairs of the primary power buses so as to avoid this possibility. Thus while any single LRU may be drawing power from only two buses, it is impossible for all LRU's to be drawing power from only these two buses. This tends to distribute the load among the power supplies such that in the worst case power is drawn from at least three of the power supplies, and in most cases power is drawn from all power supplies. Figure 7.1 illustrates examples of interconnecting two LRU's to the primary power buses.

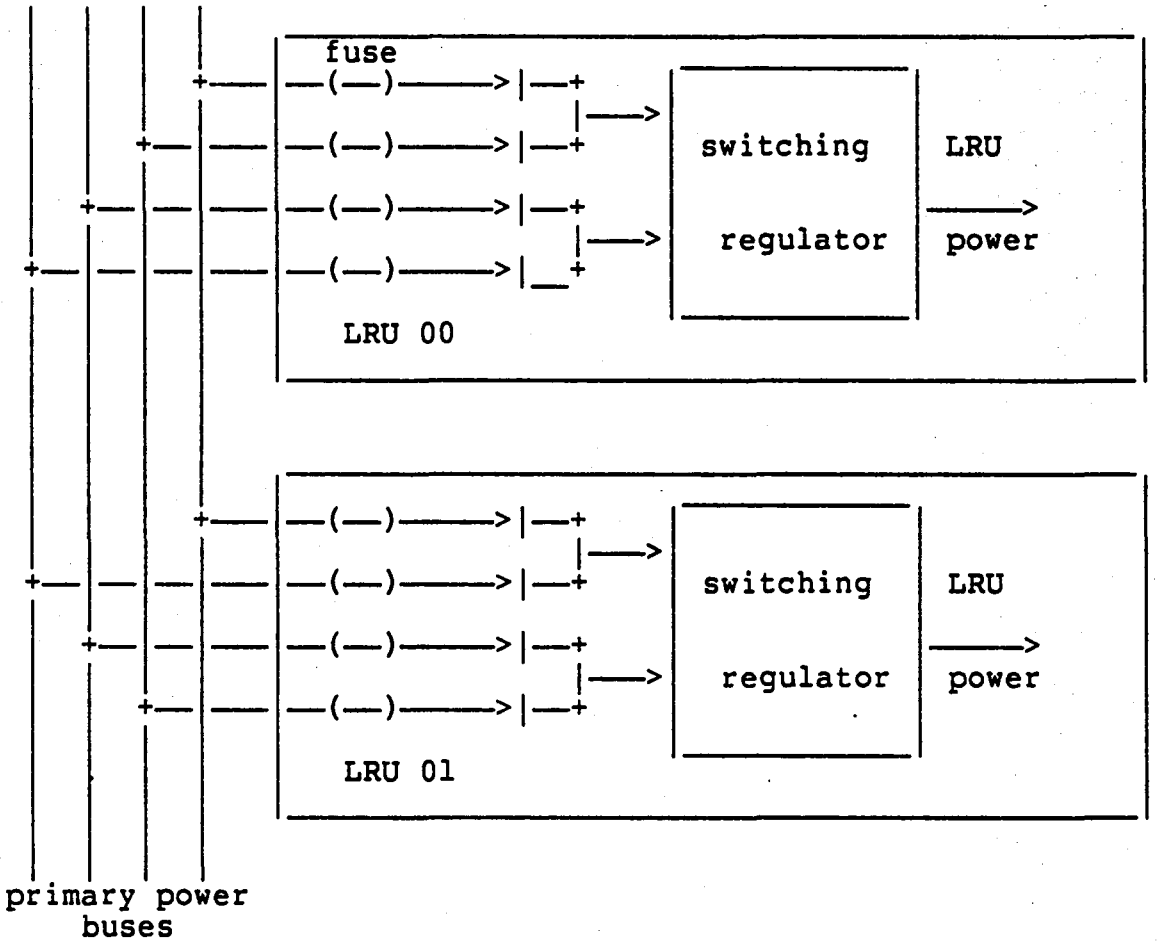


Figure 7.1 LRU Power Interconnection Examples.

Each LRU also contains battery backup circuitry to power the CMOS memory and configuration registers should primary power be lost. These batteries can provide some 100 hours of protection against power loss. Integral charging circuitry assures that the batteries are recharged when primary power is available. A switch on the front of each LRU allows the batteries to be disconnected and the LRU completely shut down for long term storage or shipment. This avoids discharging the batteries needlessly, and provides the means of assuring that all power has been removed from the backplane and circuit cards when repair is undertaken. Total charging time from complete discharge is eight hours.

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA CR-166071		

4. Title and Subtitle	5. Report Date
DEVELOPMENT AND EVALUATION OF A FAULT-TOLERANT MULTIPROCESSOR (FTMP) COMPUTER Volume I - FTMP Principles of Operation	May 1983
	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
T. B. Smith, III and J. H. Lala	CSDL-R-1600
	10. Work Unit No.

9. Performing Organization Name and Address	11. Contract or Grant No.
The Charles Stark Draper Laboratory, Inc. 555 Technology Square Cambridge, Massachusetts 02139	NAS1-15336
	13. Type of Report and Period Covered

12. Sponsoring Agency Name and Address	13. Type of Report and Period Covered
National Aeronautics and Space Administration Washington, DC 20546	Contractor Report
	14. Sponsoring Agency Code

15. Supplementary Notes

Langley Technical Monitor: Charles W. Meissner, Jr.
Final Report

16. Abstract

This report is Volume I of a four-volume report on the Fault-Tolerant Multiprocessor (FTMP) project. It covers in detail the FTMP architecture and principles of operation, and is intended to serve as a comprehensive guide to the hardware organization and operation.

The FTMP engineering model was constructed by the Collins Avionics Group of the Rockwell International Corporation to the architectural and functional specifications provided by the C. S. Draper Laboratory.

The basic organization of the FTMP is that of a general purpose homogeneous multiprocessor. Three processors operate on a shared system (memory and I/O) bus. Replication and tight synchronization of all elements and hardware voting is employed to detect and correct any single fault. Reconfiguration is then employed to "repair" a fault. Multiple faults may be tolerated as a sequence of single faults with repair between fault occurrences.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Fault-Tolerance Multiprocessor Synchronous Reconfigurable	Distribution Subject Category 62

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	122	

Available: NASA's Industrial Applications Centers

End of Document