Lightning Imaging Sensor

Software Design Specification

MSFC-SPEC-2058

Prepared By:

Maytha L. Frankford,
Flight Software

Kevin S. Wallace,
EGSE Software

Designers:

Lisa D. Coe/EB33
Maytha L. Frankford/EB33
Jon R. Rehage/EB33
David J. Trice/EB33
Kevin S. Wallace/EB33
DeLisa Wilkerson/EB33

Edited By:

Kevin S. Wallace

Table Of Contents

1.0 Introduction And Scope 1-1
1.1 Identification Of Document 1-1
1.2 Scope And Precedence 1-1
1.3 Purpose And Objectives 1-1
1.4 Document Status And Schedule 1-1
1.5 Document Organization 1-2
1.6 Definitions, Abbreviations, And Acronyms 1-2

2.0 Related Documentation 2-1
2.1 Reference Documents 2-1
2.1 Information Documents 2-1

3.0 LIS FLIGHT SOFTWARE DESIGN 3-
3.1 Flight Software Development 3-
3.1.1 Computer Software Configuration Items 3-
3.1.1.1 Initialization 3-
3.1.1.2 System Test 3-
3.1.1.3 Main Operating Loop 3-
3.1.1.4 Interrupt Handlers 3-
3.1.1.5 1773 Bus Communication 3-
3.1.1.6 Engineering and Analysis Software 3-
3.1.2 Development Tools 3-
3.1.3 Module Formulation and Structure 3-
3.1.4 Executable Code Creation 3-
3.1.4.1 Required Hardware 3-
3.1.4.2 Required Software 3-
3.1.4.3 Procedure 3-
3.2 Hardware Summary 3-
3.2.1 Microprocessor Architecture 3-
3.2.2 System Control 3-
3.2.3 Input/Output Capability 3-
3.2.4 Operating Characteristics 3-
3.2.5 Interrupts 3-
3.3 Software Functional Characteristics 3-
3.3.1 Program Control 3-
3.3.2 Program Input/Output Capability 3-
3.3.3 Operational Modes 3-
3.3.4 Database Definition 3-
3.3.5 Memory Allocation 3-
3.3.6 Storage Allocation 3-
3.3.7 Restrictions and Constraints 3-
3.4 Decomposition Description 3-
3.4.1 Initialization 3-
3.4.1.1 TMS320C25 Initialization 3-
3.4.1.2 BCRT Initialization 3-
3.4.1.3 RTEP Threshold Memory Initialization 3-

Table Of Contents

(continued)

3.4.1.4 RTEP Background Initialization 3-
3.4.1.5 Focal Plane Array Initialization 3-
3.4.1.6 Housekeeping Buffer #1 Initialization 3-
3.4.2 Packet Communication 3-
3.4.2.1 CCSDS Format 3-
3.4.2.2 FDS Communication Protocol 3-
3.4.2.3 Science Packets 3-
3.4.2.4 Housekeeping Packets 3-
3.4.2.5 Packet Formation Operation 3-
3.4.2.6 Time Data Format and Operation 3-
3.4.3 Interrupt and Exception Routines 3-
3.4.3.1 RTEP Interrupt 3-
3.4.3.2 Time-Mark Interrupt 3-
3.4.3.3 BCRT Interrupt 3-
3.4.3.4 Filter Temperature Exception Monitoring 3-
3.4.4 Background Operations 3-
3.4.4.1 Background Memory Read 3-
3.4.4.2 Packetize Background 3-
3.5 Interface Description 3-
3.5.1 LIS Hardware Interfaces 3-
3.5.2 Software Interfaces 3-

4.0 EGSE Software 4-
4.1 Development 4-
4.1.1 CSCI's 4-
4.1.1.1 Main Operation Loop 4-
4.1.1.2 EGSE TRMM Interface Simulator Drivers 4-
4.1.1.3 1773 Communication 4-
4.1.1.4 Discrete Telemetry Protocol 4-
4.1.1.5 Data Format, Storage, And Display 4-
4.1.1.6 Calibration/Test Analysis Software 4-
4.1.1.7 Engineering and Analysis Software 4-
4.1.3 Module Formation And Structure 4-
4.1.2 Development Tools 4-
4.1.4 Executable Code Creation 4-
4.1.4.1 Required Hardware 4-
4.1.4.2 Required Software 4-
4.1.4.3 Procedure 4-
4.2 Hardware Capabilities 4-
4.2.1 Microprocessor Architecture 4-
4.2.2 System Control 4-
4.2.3 Input/Output Capability 4-
4.2.4 Operating Characteristics 4-
4.2.5 Interrupts 4-
4.3 Software Functional Characteristics 4-
4.3.1 Program Control 4-
4.3.2 Program Input/Output Capability 4-

Table Of Contents

(continued)

4.3.3 Operational Modes 4-
4.3.4 Database Definition 4-
4.3.5 Memory Allocation And Variable Declaration 4-
4.3.6 Storage Allocation 4-
4.3.7 Restrictions And Constraints 4-
4.4 Decomposition Description 4-
4.4.1 Main Loop 4-
4.4.2 User Interface 4-
4.4.3 Test Routines 4-
4.4.4 Interface Simulator Drivers 4-
4.4.5 Bus Interface Protocols 4-
4.4.6 Command And LIS Data Formats 4-
4.4.7 Discrete Telemetry Formats 4-
4.4.8 Display/Storage Routines 4-
4.5 Interface Description 4-
4.5.1 Hardware Interfaces 4-
4.5.2 Software Interfaces 4-
4.5.2.1 Software Running Other Ground Systems 4-
4.5.2.2 COTS Software 4-
4.6 Detailed Design 4-

Appendix A: Derivation Of Translation Equations

Housekeeping DAS Equations A-
Thermistor Voltage-to-Temperature A-

Appendix B: Flight Software Resource Utilization

Estimated Lines Of Code B-
Estimated ROM Requirements B-

Estimated RAM Requirements B-

1.0 Introduction

1.1 Identification Of Document

This document, MSFC-SPEC-2058 constitutes the software design specification for the Lightning Imaging Sensor (LIS), a science experiment for the Tropical Rainfall Measuring Missing (TRMM).

1.2 Scope And Precedence

This document interprets the requirements for LIS software established in MSFC-SPEC-2026, the "Lightning Imaging Sensor Software Requirements Specification," (Reference 1) and establishes a design to be used for coding that will implement the requirements. This design is an implementation of the plans described in MSFC-PLAN-2025, the "Lightning Imaging Sensor Software Management, Development And Test Plan" (Reference 2), sections 3.0 and 4.0.

In the event of conflict between this document and Reference 1, Reference 1 shall take precedence.

1.3 Purpose And Objectives

This document provides the design to be used for coding of LIS software. This document is a tool used to produce code that will satisfy all requirements (Reference 1).

1.4 Document Status And Schedule

This document will be reviewed at the Software Preliminary Design Review (SWPDR) in September, 1992. Although the document is not expected to be completed, the scope and format of the document should be established, and the major topics addressed to a preliminary level, ready for detailed development. The purpose of reviewing the document at this time is to assure that all requirements are being addressed, and that the design effort is proceeding in a manner consistent with the program schedule. This version of the document is known as the "Preliminary Software Design Specification."

This document will be completed and reviewed at the Software Critical Design Review (SWCDR). The purpose of this document review is to establish the completeness and accuracy of this document, in preparation for software coding. This document will be baselined at SWCDR and maintained under configuration control. This version of the document is known as the "Software Design Specification: Code-To."

At the software Test Readiness Review, this document, and all Engineering Change Orders (ECOs) applied to the document, will be reviewed. This review will establish that the design document is consistent with the software code. This version of the document is known as the "Software Design Specification: As-Built."

At Configuration Inspection (CI), this document will be reviewed a final time. This final review will certify that all changes to the design, as required by the testing, have been incorporated into the design document. The Ci Review will again baseline this document as the final version. This version of the document in known simply as the "Software Design Specification."

1.5 Document Organization

This document is organized into four major sections:

Section 1.0 (this section) contains configuration information. Section 2.0 contains reference information. Section 3.0 contains the software design for the LIS flight instrument. Section 4.0 contains the software design for the electrical ground support equipment (EGSE) used as a portable checkout tool and TRMM interface simulator.

1.6 Definitions, Abbreviations, And Acronyms

AR	Acceptance Review
A/D	Analog to Digital
ADP	Automatic Data Processing (Equipment)
BCRT	Bus Controller/Remote Terminal
CCSDS	Consultative Committee For Space Data Systems
CDMS	(LIS) Command And Data Management System
CDR	Critical Design Review
C&DH	(TRMM) Command And Data Handling (System)
CI	Configuration Inspection
COTS	Commercial Off-The Shelf (Software)
CSCI	Computer Software Configuration Item
DAS	(Housekeeping) Data Acquisition System
DMS	(LIS) Data Management System
E&A	Engineering And Analysis (Software)
EGSE	Electrical Ground Support Equipment
FDS	(TRMM) Flight Data System
FPA	Focal Plane Array
GSFC	Goddard Space Flight Center
H-II	(Japanese Launch Vehicle for TRMM)
H/K	Housekeeping
Hz	Hertz
IRD	Interface Requirements Document
ISA	Industry Standard Architecture
IGSE	Instrument GSE (for LIS, the 'EGSE')
I/O	Input Output
LIS	Lightning Imaging Sensor
LSB	Least Significant Bit
MIL-STD	Military Standard
ms	Millisecond
MSFC	Marshall Space Flight Center
N/A	Not Applicable
NASA	National Aeronautics and Space Administration
OOP	Object Oriented Programming
PDR	Preliminary Design Review
PROM	Programmable Read Only Memory
RAM	Random Access Memory
ROM	Read Only Memory
RTEP	Real-Time Event Processor
SGSE	(TRMM) Spacecraft GSE
SWRD	(LIS) Software Requirements Document
S/W	Software
S/W CDR	Software Critical Design Review
S/W PDR	Software Preliminary Design Review
S/W RR	Software Requirements Review
TBD	To Be Determined
TBS	To Be Supplied
TRR	Test Readiness Review
TRMM	Tropical Rainfall Measurement Mission
XDS	Extended Development Support

2.0 Related Documentation

2.1 Reference Documents

Reference 1	MSFC-SPEC-2026, "Lightning Imaging Sensor Software Requirements Specification,"
Reference 2	MSFC-PLAN-2025, "Lightning Imaging Sensor Software Management, Development And Test Plan"
Reference 3	TMS320C2X Users Manual
Reference X	Turbo C++ Version 3.0 Users Guide
Reference X	MS-DOS Version 5.0 Users Guide
Reference X	TRMM IGSE/SGSE Interface Requirements Document

2.1 Information Documents

Information 1	CCSDS Packet Telemetry Recommendations (Green Book)
Information 2	CCSDS Time Code Recommendations (Blue Book)
Information 3	CCSDS Recommendations For Formatted Data Units (Red Book)
Information 4	Kerigan And Richie "C Programming"
Information 5	(TBD) C++ Programming Reference

CCSDS Telecommands

3.0 LIS Flight Software Design

LIS flight software will be developed to run the onboard operation of the Lightning Imaging Sensor. Flight software executes on one Texas Instruments TMS320C25 microprocessor which is part of the LIS Controller I/O board.

3.1 Flight Software Development

Flight software is developed in assembly language for the TMS320C25 microprocessor. Assembly language is considered adequate for microprocessor operation, and control of the LIS instrument.

Flight software will be developed by personnel in the Communication Systems Branch of the Computers and Communication Systems Division of the Information and Electronic Systems Laboratory.

The LIS data systems development laboratory, located in EB33, is the primary facility for flight software development. Code can also be developed using office IBM compatible computers. The LIS development facility will provide the location for software maintenance and some testing. It houses the LIS software library which contains hard copies of code, unit development folders, and pertinent LIS software documents.

3.1.1 Computer Software Configuration Items (CSCIs)

LIS flight software is developed, tested, and maintained under configuration controlled by using a CSCI design structure. Each CSCI is given a unique name, WBS number, and brief description. The flight software CSCI's are listed below, and can also be found in the LIS S/W Management, Development and Test Plan (Reference 2).

3.1.1.1 Initialization

This CSCI covers the initialization of the LIS system. The microprocessor, BCRT, Focal Plan Array, RTEP, Heater Controller, housekeeping buffer, and packet buffers are initialized upon system power up, or upon receipt of a RESET command.

The requirements governing this CSCI are given in SWRD 3.1

3.1.1.2 System Test

This CSCI refers to an operational test of the LIS system. The test determines if the value and location of a light source is correctly received and sent to ground.

The requirements governing this CSCI are given in SWRD 3.4

3.1.1.3 Main Operating Loop

This CSCI covers the microprocessor operation of packet building. The information in the data packets consists of lightning event values, line address, pixel address, background data, and time stamp.

The requirements governing this CSCI are given in SWRD 3.4

3.1.1.4 Interrupt Handlers

This CSCI provides routines to acknowledge and respond to the three external interrupts to the microprocessor. These interrupts are generated by the Real-Time Event Processor (RTEP), the BCRT interface to the 1773 bus, and time mark interface.

The requirements governing this CSCI are given in SWRD 3.5

3.1.1.5 1773 Bus Communication

This CSCI governs command reception, and data packet transmissions over the 1773 bus, including the packetization algorithms, and the FDS handshaking protocol.

The requirements governing this CSCI are given in SWRD 3.6

3.1.1.6 Engineering and Analysis Software

This CSCI refers to the software code written to support hardware development. There are no firm requirements, or test plans levied on this software. In general, E&A software will be code similar in function to the final EGSE code, but written informally on as-needed basis.

When formal coding begins, E&A software that has been utilized will prove a valuable reference, and will be a reference for a successful coding program.

The requirements governing this CSCI are given in SWRD 3.8

3.1.2 Development Tools

Flight software will use IBM compatible computers, and the Texas Instruments TMS320 development support tools. These include a TMS320 Macro Assembler/Linker, the TMS320C25 Simulator, the TMS320C25 Emulator (XDS/22). (See Figure TBD.)

3.1.3 Module Formulation and Structure

Flight software is structured in modular fashion. A module of flight code serves as a functional unit. The flight software modules follow, as closely as possible, the decomposition descriptions listed in section 3.4 of this document.

3.1.4 Executable Code Creation

3.1.4.1 Required Hardware

The hardware required to create executable flight software consists of:
1) IBM compatible computer,
2) floppy disk containing flight source code,
3) the TI XDS/22 emulator,
4) a TBD EPROM programmer.

3.1.4.2 Required Software

The software required to create executable flight software consists of:

1) the assembly language flight source code,
2) the TI TMS320 Assembler/Linker,
3) TBD EPROM programming software.

3.1.4.3 Procedure

TBS

3.2 Hardware Summary

3.2.1 Microprocessor Architecture

The TMS320C25 is the second generation of the TI TMS320 family of microprocessors. The C25 is a CMOS device, with an instruction cycle time of 100 ns. It has a total of 133 instructions, eight auxiliary registers, and an eight level hardware stack.

The C25 has a modified Harvard architecture. This allows transfers between program and data spaces, maximizing processor power and execution speed.

The C25 has 544 16 bit words of on chip data RAM. Of this, 288 words are data memory, and 256 words can be configured as either program or data memory. This size allows the C25 to handle a data array of 512 words, and still have 32 words available for intermediate storage. While using on chip program RAM or ROM, the C25 runs at full speed without wait states.

There are three separate address spaces for program, data, and I/O. The C25 has a 32 bit accumulator which is split into two 16-bit segments for storage in data memory.

Flight software design uses three of the six on- chip memory mapped registers, and 8 auxiliary registers for indirect addressing or temporary data storage. The

C25 instruction set provides three memory addressing modes: direct and indirect, for accessing data memory, and immediate addressing.

The C25 has a repeat feature that can be used in conjunction with other instructions, allowing a single instruction to be performed up to 256 times. Multicycle instructions are pipelined, and effectively become single cycle instructions.

3.2.2 System Control

System control can be accomplished by the program counter, hardware stack, external reset signal, interrupts, hold operation and wait states. A three level pipeline organization provides independent prefetch, decode, and execute instructions.

In addition, if power is cycled, the system will reinitialize.

3.2.3 Input/Output Capability

The TMS320C25 software instructions can access 16 input ports, and 16 output ports. The four LSB of the address bus specify the port being accessed. I/O devices are mapped into the I/O address space and into off chip data memory mapped registers.

3.2.4 Operating Characteristics

The operating characteristics for the TMS320C25 can be found in Appendix A in the TMS320 User's Guide (Reference #TBD).

3.2.5 Interrupts

The TMS320C25 has three maskable external interrupts, INT2 - INT0. RESET is the only non maskable external interrupt.

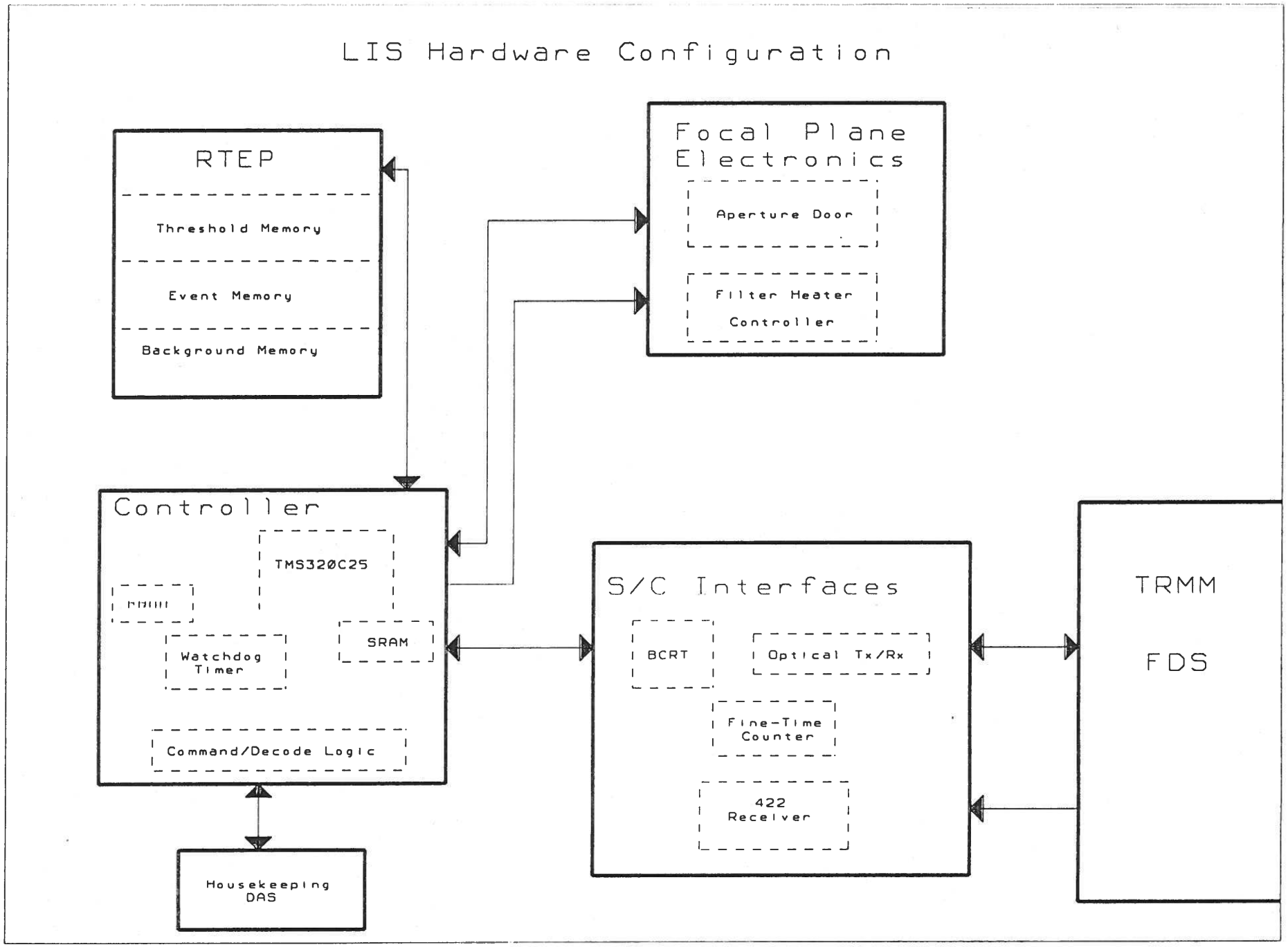
The flight software design uses two of the C25's internal interrupts: the timer interrupt (TINT), and the software interrupt (TRAP).

Details about the interrupt locations and priorities can be found in section 3.8 of the TMS320 User's Guide (Reference #TBD).

3.3 Software Functional Characteristics

3.3.1 Program Control

Flight software program control is accomplished by interrupts and commands. Three external interrupts are the RTEP interrupt, the BCRT interrupt, and the Time Mark interrupt. Commands are received by the 1773 interface and decoded and executed by software.


LIS Hardware Configuration
RTEP
Threshold Memory
Event Memory
Background Memory
Focal Plane Electronics
Aperture Door
Filter Heater Controller
Controller
TMS320C25
RAMM
Watchdog Timer
SRAM
Command/Decode Logic
Housekeeping DAS
S/C Interfaces
BCRT
Optical Tx/Rx
Fine-Time Counter
422 Receiver
TRMM
FDS

The interrupts are defined in section 3.4.3 of this document. A list of commands is found in section 3.3.4 of this document.

3.3.2 Program Input/Output Capability

The LIS flight software is required to accept the following inputs from the TRMM spacecraft:

- -Instrument commands
- -FDS acknowledges
- -Time broadcasts
- -Time mark counts

LIS flight software is required to provide the following outputs to the TRMM spacecraft in the proper packet format:

- -Real Time Event Processor (RTEP) data
- -Housekeeping information
- -Miscellaneous status information

3.3.3 Operational Modes

LIS flight software is designed to have three modes of operation. The commands for entering or exiting these modes are received over the TRMM 1773 fiber optic bus, via a DMA interface with a UTMC1553B BCRT.

1) Background Send Mode ON: This is considered the normal mode of operation for the LIS instrument. In this mode, the packets sent to ground contain both event data and background information. It is the default mode of operation upon completion of the RESET routine. The mode is terminated when commanded to another mode from the ground.

2). Background Send Mode OFF: In this mode, the instrument will send packets to the ground that contain event data only. This mode is initiated and stopped as commanded from the ground.

3). System Test Mode ON/OFF: This mode consists of an instrument self-test that runs when commanded from the ground to determine the operational health of the LIS.

~~4). Observatory Engineering Mode: TBD~~

3.3.4 Database Definition

The LIS RT address is 20. The LIS real-time serial commands are given in Figure TBD. ?

HOUSEKEEPING STATUS WORD															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bit Position	Meaning
15	Background Send Mode (1=ON; 2=OFF)
14	Self Test (1=ON; 2=OFF)
13	Aperture Door (1=OPEN; 0=CLOSED)
12	Watchdog (1=ENABLE; 0=DISABLE)
11	Primary Heaters (1=YES;0=NO)
10	Redundant Heaters (1=ON; 0=OFF)
9	Events In Packet (1=YES;0=NO)
8	Background In Packet (1=YES;0=NO)
7	Valid Command Received And Executed
6	Invalid Command Received
5	Threshold Update Error
4	Threshold Update Successful
3	Threshold Update Unsuccessful
2	Aperture Door Open Error
1	Heater Controller 28V
0	Spare

	COMMAND STATUS															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	<------- LAST COMMAND RECEIVED --------->															
1	3	2	1	0	3	2	1	0	3	2	1	0	3	2	1	0
	<------- LAST COMMAND EXECUTED -------->															
2	3	2	1	0	3	2	1	0	3	2	1	0	3	2	1	0

	8 PACKED THRESHOLD WORDS																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
1	0	0	0	0													<-- WORD NUMBER
					0	0											<-- FILLER
							4	3	2	1	0						<-- 5-BIT WORD (1)
												4	3	2	1	0	<-- 5-BIT WORD (0)

1	0	1	1	1													<-- WORD NUMBER
					0	0											<-- FILLER
							4	3	2	1	0						<-- 5-BIT WORD (15)
												4	3	2	1	0	<-- 5-BIT WORD (14)

INSTRUMENT PACKET WORD FORMATS

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

CPU WORD FORMAT

PRIMARY HEADER (3 WORDS)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	< ---------- PACKET IDENTIFICATION WORD ---------- >															
1	0	0	0	<-- VERSION NUMBER												
				0	<-- TYPE											
					0	<-- SECONDARY HEADER FLAG										
						10	9	8	7	6	5	4	3	2	1	0
	<------ PACKET SEQUENCE CONTROL WORD ------->															
2	1	1	<-- SEGMENT FLAGS													
			13	12	11	10	9	8	7	6	5	4	3	2	1	0
	<-------- PACKET LENGTH WORD -------->															
3	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

<-- APPLICATION PROCESS ID

<-- SOURCE SEQUENCE COUNT

MISSION ELAPSED TIME WORDS

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
1	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	<-- MS SECONDS
2	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	<-- LS SECONDS
3	ss	ss	ss	ss	ss	ss	ss	ss	ss	ss	ss	ss	ss	ss	ss	ss	<-- MS SUBSECONDS
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	<-- LS SUBSECONDS

FRAME SYNC WORD

	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	1	1	1	1

EVENT LOCATION WORD

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	<-- FILLER													
			6	5	4	3	2	1	0	<-- ROW ADDRESS						
										6	5	4	3	2	1	0

<-- COLUMN ADDRESS

EVENT VALUE WORD

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	0	1	1	0	0	<-- EVENT ID							
									7	6	5	4	3	2	1	0

<-- VALUE

BACKGROUND VALUE WORD

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	1	0	<-- BACKGROUND ID											
					11	10	9	8	7	6	5	4	3	2	1	0

<-- VALUE

HOUSEKEEPING ANALOG WORD

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	3	2	1	0	<-- CHANNEL ADDRESS											
					11	10	9	8	7	6	5	4	3	2	1	0

<-- VALUE

LIS REAL-TIME SERIAL COMMANDS

The LIS has the following 1773 command requirements:

1 -- Background Send Mode ON. (Normal mode)
2 -- Background Send Mode OFF.
3 -- Threshold Adjustment.
4 -- Filter Temperature Set Point.
5 -- Self Test.
6 -- Aperature Door Open.
~~7 -- Aperature Door Closed~~.
8 -- Safe Mode.
9 -- Watchdog Enable.
10 - Watchdog Disable.

The above commands shall be sent as a 1773 message to subaddress #5 ,

The following defines the 1773 message required for each command.

1 -- Background Send Mode ON.
Send the following one (1) word (hex) message:
1111.
2 -- Background Send Mode OFF.
Send the following one (1) word (hex) message:
2222.
3 -- Threshold Adjustment.
Send the following nine (9) word (hex) message:
3333, 0xxx, 1xxx, 2xxx, 3xxx, 4xxx, 5xxx, 6xxx, 7xxx.
4 -- Filter Temperature Set Point.
Send the following two (2) word (hex) message:
4444, 0xxx.
5 -- Self Test.
Send the following one (1) word (hex) message:
5555.
6 -- Aperture Door Open.
Send the following one (1) word (hex) message:
6666.
7 -- Aperture Door Closed.
Send the following one (1) word (hex) message:
7777.
8 -- Safe Mode.
Send the following one (1) word (hex) message:
8888.
9 -- Watchdog Enable.
Send the following one (1) word (hex) message:
9999.
10 -- Watchdog Disable.
Send the following one (1) word (hex) message:
AAAA.

Jon Richard Rehage, EB33, 05/06/92 08:57 AM (c:\winword\lis\lis_cmd1.doc)

LIS Real-Time Serial Commands
(-continued-)

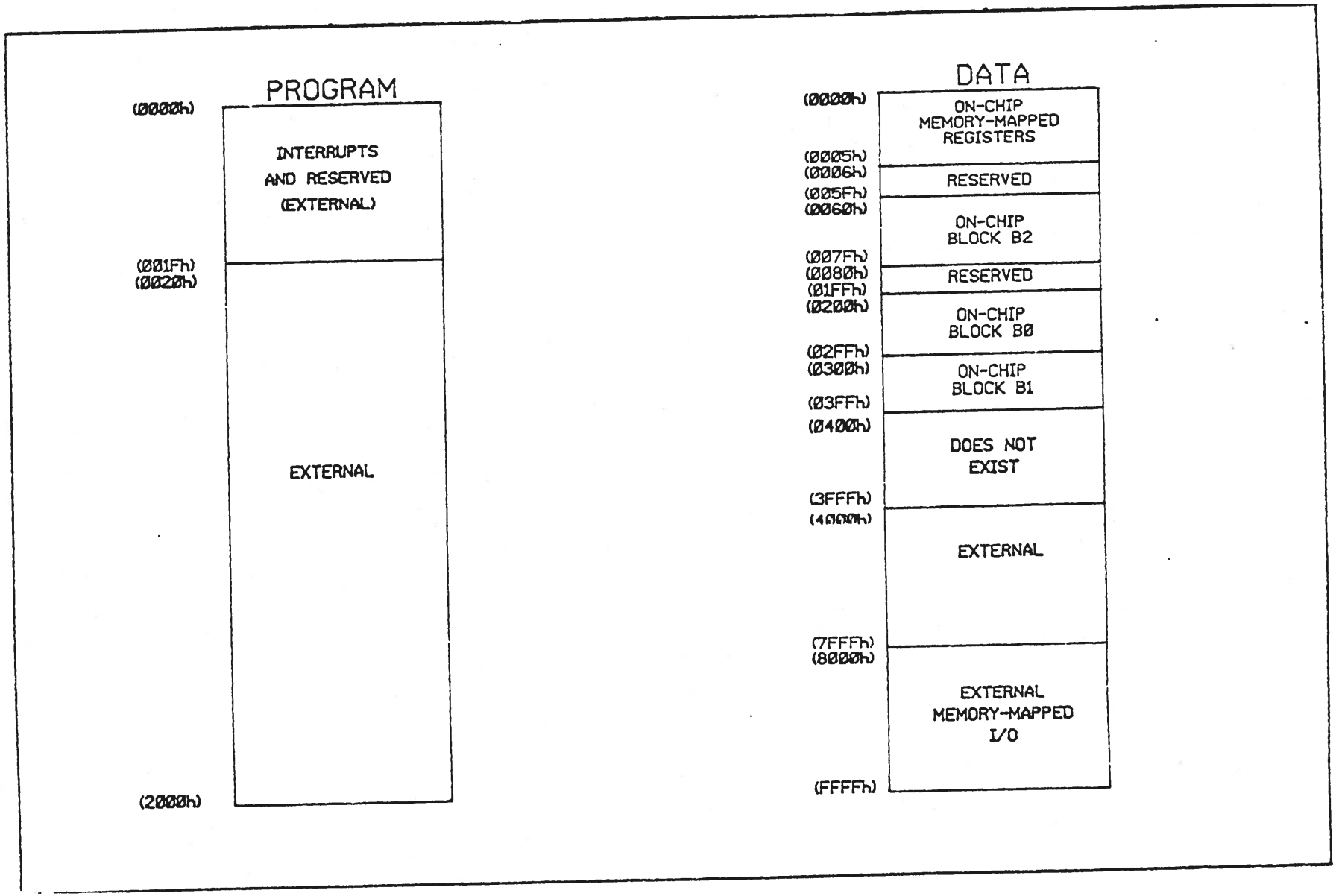
11 - Primary Heater On
Send the following one (1) word (hex) message: BBBB

12 - Redundant Heater On
Send the following one (1) word (hex) message: CCCC

13 - Enable Filter Monitoring
Send the following one (1) word (hex) message: DDDD

14 - Disable Filter Monitoring
Send the following one (1) word (hex) message: EEEE

TMS320C25 Memory Space



MEMMAP3.WK1

DETAILED MEMORY MAP/PAGE 4

FROM (HEX)	TO	# OF WORDS	# WORDS (HEX)	DESCRIPTION
4770	4777	8	8	RCV SUBADDRESS MESSAGE STATUS SPACE (8)
4778	478C	21	15	XMIT SUBADDRESS MESSAGE STATUS SPACE (21)
478D	478F	3	3	MODE CODE MESSAGE STATUS SPACE
4790	47AF	32	20	INTERRUPT LOG LIST/SCRATCH PAD
		1968		

EXTERNAL MEMORY

STARTING ADDRESS	ENDING ADDRESS	# WORDS(d)	DESCRIPTION
4000h	413F	320	REMOTE TERMINAL DESCRIPTOR SPACE
4140h	4151	18	RT INTERNAL REGISTERS
4152h	41EDh	156	DESCRIPTOR POINTERS
41EEh	42A7h	186	RCV SUBADDRESS SPACE
42A8h	44A9h	514	XMIT SUBADDRESS DATA SPACE/DATA PACKET 1
44AAh	46ABh	514	XMIT SUBADDRESS DATA SPACE/DATA PACKET 2
46ACh	46EDh	66	XMIT SUBADDRESS DATA SPACE/HOUSEKEEPING PACKET 1
46EEh	472Fh	66	XMIT SUBADDRESS DATA SPACE/HOUSEKEEPING PACKET 2
4730h	473Fh	16	MODE CODE DATA SPACE
4740h	478Dh	78	SUBADDRESS MESSAGE STATUS SPACE
478Eh	479Fh	18	INTERRUPT LOG LIST
47A0h	47AFh	16	SCRATCH PAD
47B0h	492Fh	384	BACKGROUND MEMORY BUFFER 1
4930h	4AAFh	384	BACKGROUND MEMORY BUFFER 2
4AB0h	7FFFh	13648	SCRATCH PAD

MEMORY MAPPED I/O

STARTING ADDRESS	ENDING ADDRESS	# WORDS(d)	DESCRIPTION
8000h	800Dh	14	BCRT INTERNAL REGISTERS READ/WRITE
A001h		1	FINE TIME COUNTER/READ TIME
A002h		1	FINE TIME COUNTER/LATCH TIME
D000h		1	WATCHDOG/RESET
D001h		1	WATCHDOG/DISABLE
D000h		1	WATCHDOG/ENABLE

DELISA WILKERSON
PAGE 1 OF

	DATA PAGE	START ADDRESS	STOP ADDRESS	DESCRIPTION	FUNCTION	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
RTDSSA	128	4000		RCV SUBADR #1 CMD WORD	TIME CODE																
		4001		#1 STATUS LIST POINTER																	
		4002		#1 DATA LIST POINTER																	
		4003		RESERVED																	
		4004		RCV SUBADR #2 CMD WORD	COMMANDS																
		4005		#2 STATUS LIST POINTER																	
		4006		#2 DATA LIST POINTER																	
		4007		RESERVED																	
		4008		RCV SUBADR #3 CMD WORD	THRESHOLD																
		4009		#3 STATUS LIST POINTER																	
		400A		#3 DATA LIST POINTER																	
		400B		RESERVED																	
		400C		RCV SUBADR #4-31 CMD WORD	ILLEGAL SUBADDRESS																
				#4-31 STATUS LIST POINTER																	
				#4-31 DATA LIST POINTER																	
			407B	RESERVED																	
RTDSSA+7C		407C	407F	UNUSED																	
RTDSSA+80	129	4080		XMIT SUBADR #1 CMD WORD																	
		4081		#1 STATUS LIST POINTER																	
		4082		#1 DATA LIST POINTER																	
		4083		RESERVED																	
		4084		XMIT SUBADR #2 CMD WORD	DATA PACKET																
		4085		#2 STATUS LIST POINTER																	
		4086		#2 DATA LIST POINTER																	
		4087		RESERVED																	
		4088		XMIT SUBADR #3 CMD WORD																	
		4089		#3 STATUS LIST POINTER																	
		408A		#3 DATA LIST POINTER																	
		408B		RESERVED																	
		408C		XMIT SUBADR #4 CMD WORD	HOUSEKEEPING PACKET																
		408D		#4 STATUS LIST POINTER																	
		408E		#4 DATA LIST POINTER																	
		408F		RESERVED																	
		4090		XMIT SUBADR #5-31 CMD WORD	ILLEGAL SUBADDRESS																
				#5-31 STATUS LIST POINTER																	
				#5-31 DATA LIST POINTER																	
			40FB	RESERVED																	
RTDSSA+FC		40FC	40FF	UNUSED																	
RTDSSA+100	130	4100		MODE CODE #1-16 CMD WORD																	
				#1-16 STATUS LIST POINTER																	
				#1-16 DATA LIST POINTER																	
			413F	RESERVED																	

BCRT MEMORY MAP 432
DELISA WILKERSON

MEMMAP3.WK1

DETAILED MEMORY MAP/PAGE 2

FROM (HEX)	TO	# OF WORDS	# WORDS (HEX)	DESCRIPTION
41B0	41CF	32	20	RCV SUBADDRESS #1 DATA SPACE COMMAND PACKET 1
41D0	41EF	32	20	RCV SUBADDRESS #2 DATA SPACE COMMAND PACKET 2
41F0	420F	32	20	RCV SUBADDRESS #3 DATA SPACE COMMAND PACKET 3
4210	422F	32	20	RCV SUBADDRESS #4 DATA SPACE COMMAND PACKET 4
4230	424F	32	20	RCV SUBADDRESS #5 DATA SPACE TIME CODE
4250	426F	32	20	RCV SUBADDRESS #6 DATA SPACE RESERVED
4270	428F	32	20	RCV SUBADDRESS #7 DATA SPACE RESERVED
4290	42AF	32	20	XMIT SUBADDRESS #1 DATA SPACE DATA PACKET 1/1
42B0	42CF	32	20	XMIT SUBADDRESS #2 DATA SPACE DATA PACKET 2/1
42D0	42EF	32	20	XMIT SUBADDRESS #3 DATA SPACE DATA PACKET 3/1
42F0	430F	32	20	XMIT SUBADDRESS #4 DATA SPACE DATA PACKET 4/1
4310	432F	32	20	XMIT SUBADDRESS #5 DATA SPACE DATA PACKET 5/1
4330	434F	32	20	XMIT SUBADDRESS #6 DATA SPACE DATA PACKET 6/1
4350	436F	32	20	XMIT SUBADDRESS #7 DATA SPACE DATA PACKET 7/1
4370	438F	32	20	XMIT SUBADDRESS #8 DATA SPACE DATA PACKET 8/1
4390	43AF	32	20	XMIT SUBADDRESS #9 DATA SPACE DATA PACKET 9/1
43B0	43CF	32	20	XMIT SUBADDRESS #10 DATA SPACE DATA PACKET 10/1
43D0	43EF	32	20	XMIT SUBADDRESS #11 DATA SPACE DATA PACKET 11/1
43F0	440F	32	20	XMIT SUBADDRESS #12 DATA SPACE DATA PACKET 12/1
4410	442F	32	20	XMIT SUBADDRESS #13 DATA SPACE DATA PACKET 13/1
4430	444F	32	20	XMIT SUBADDRESS #14 DATA SPACE DATA PACKET 14/1
4450	446F	32	20	XMIT SUBADDRESS #15 DATA SPACE DATA PACKET 15/1
4470	448F	32	20	XMIT SUBADDRESS #16 DATA SPACE DATA PACKET 16/1
[illegible]	[illegible]	[illegible]		[illegible]

MEMMAP3.WK1

DETAILED MEMORY MAP/PAGE 3

FROM (HEX)	TO	# OF WORDS	# WORDS (HEX)	DESCRIPTION
4480	44CF	32	20	XMIT SUBADDRESS #1 DATA SPACE DATA PACKET 1/2
44D0	44EF	32	20	XMIT SUBADDRESS #2 DATA SPACE DATA PACKET 2/2
44F0	450F	32	20	XMIT SUBADDRESS #3 DATA SPACE DATA PACKET 3/2
4510	452F	32	20	XMIT SUBADDRESS #4 DATA SPACE DATA PACKET 4/2
4530	454F	32	20	XMIT SUBADDRESS #5 DATA SPACE DATA PACKET 5/2
4550	456F	32	20	XMIT SUBADDRESS #6 DATA SPACE DATA PACKET 6/2
4570	458F	32	20	XMIT SUBADDRESS #7 DATA SPACE DATA PACKET 7/2
4590	45AF	32	20	XMIT SUBADDRESS #8 DATA SPACE DATA PACKET 8/2
45B0	45CF	32	20	XMIT SUBADDRESS #9 DATA SPACE DATA PACKET 9/2
45D0	45EF	32	20	XMIT SUBADDRESS #10 DATA SPACE DATA PACKET 10/2
45F0	460F	32	20	XMIT SUBADDRESS #11 DATA SPACE DATA PACKET 11/2
4610	462F	32	20	XMIT SUBADDRESS #12 DATA SPACE DATA PACKET 12/2
4630	464F	32	20	XMIT SUBADDRESS #13 DATA SPACE DATA PACKET 13/2
4650	466F	32	20	XMIT SUBADDRESS #14 DATA SPACE DATA PACKET 14/2
4670	468F	32	20	XMIT SUBADDRESS #15 DATA SPACE DATA PACKET 15/2
4690	46AF	32	20	XMIT SUBADDRESS #16 DATA SPACE DATA PACKET 16/2
4680	46CF	32	20	XMIT SUBADDRESS #17 DATA SPACE HOUSEKEEPING /2
46D0	46EF	32	20	XMIT SUBADDRESS #18 DATA SPACE RESERVED
46F0	470F	32	20	XMIT SUBADDRESS #19 DATA SPACE RESERVED
4710	472F	32	20	XMIT SUBADDRESS #20 DATA SPACE RESERVED
4730	474F	32	20	MODE CODE DATA SPACE
4750	476F	32	20	SCRATCH PAD

3.3.5 Memory Allocation

A memory map for the TMS320C25 is provided in Figure TBD.

3.3.6 Storage Allocation

All LIS storage for orbital operations is within volatile RAM. Therefore, there is no provision for storage, beyond the memory to contain the two packets of science data, the two packets of housekeeping data, the two coarse-time locations, and scratch-pad memory.

3.3.7 Restrictions and Constraints

LIS flight software is designed with the following restrictions and constraints:

1) Memory Allocation: Capability to store two 8 kilobit data packets, plus two 64 bit housekeeping packets.

correct to ~~8 kbit~~ 32 ~~8~~ words

2). Timing Requirements: up to 20 commands/sec could be recieved. This should go in SRD.

a) A packet is formed every 512 frames, or when there are a sufficient number of events to fill a packet.

b) Commands will be read within ~~31.25~~ 32 milliseconds.

c) No repeat commands which last more than 1.9 microseconds since this is a non-interruptible state which would prevent the BCRT from gaining DMA control.

3.4 Decomposition Description

The following description provides more information for the flight software modules required. This structure is designed to represent a logical continuity of function and requirements, irrespective of the CSCI structure. Top level flow diagrams show overall modular structure.

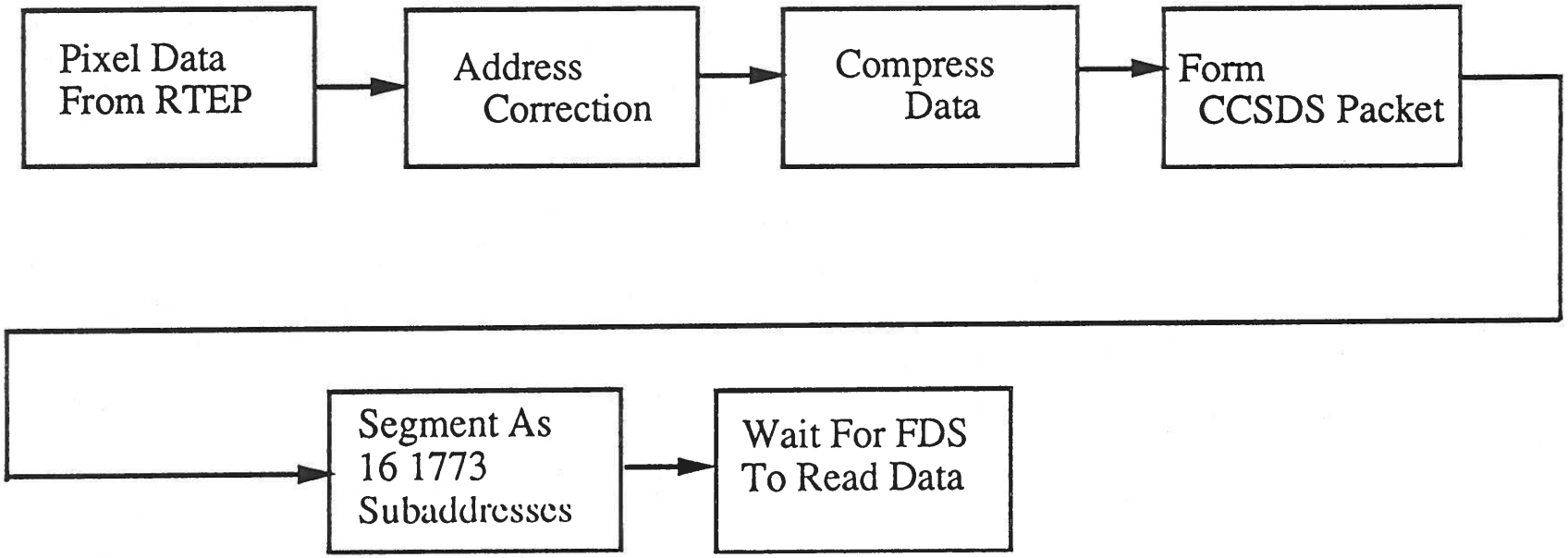
3.4.1 Initialization

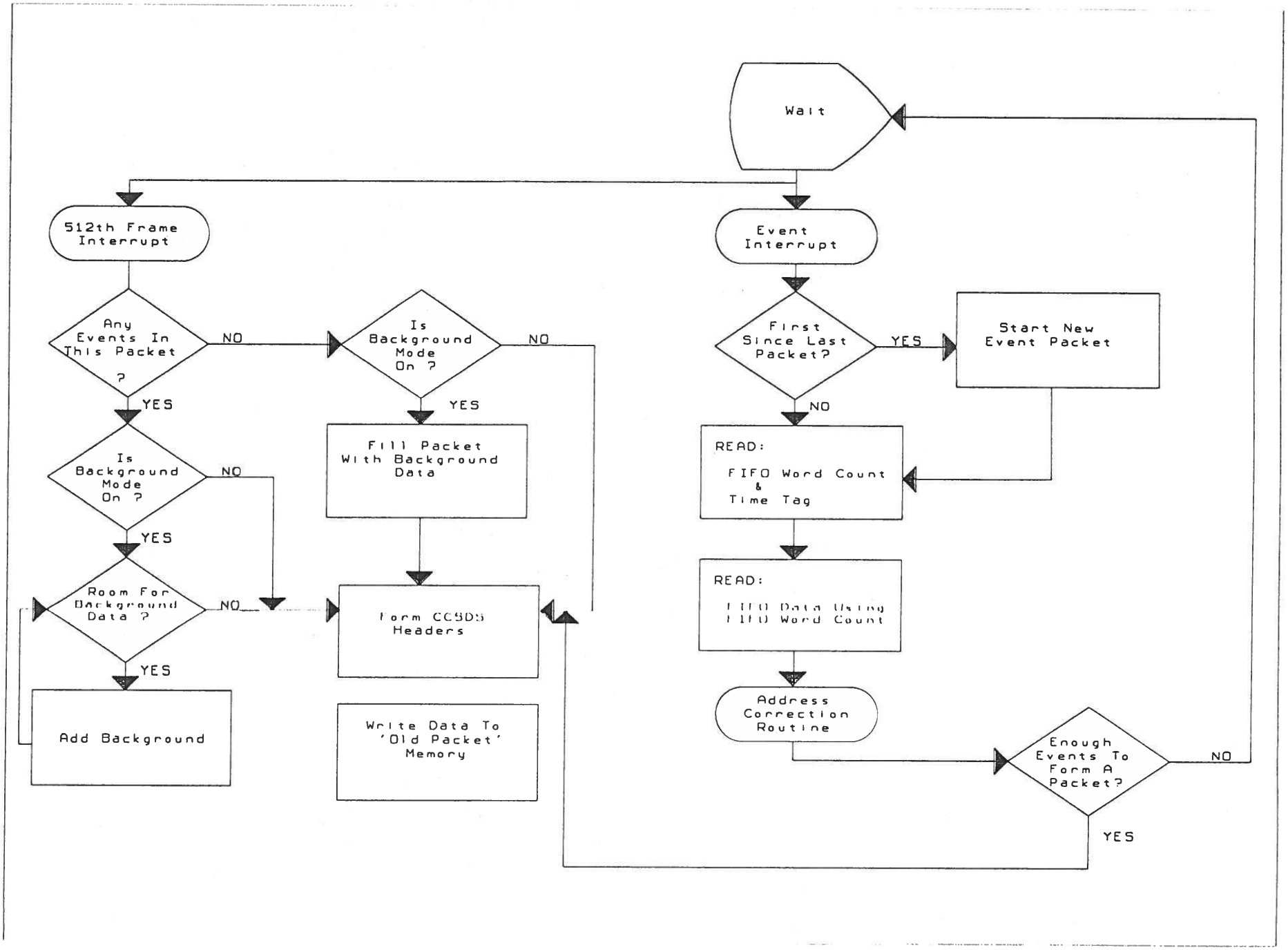
These software modules comprise the total initialization process, both of the microprocessor and its resources, and the other systems within the LIS instrument.

3.4.1.1 TMS320C25 Initialization

After system power up, the threshold memory is written, the BCRT is initialized, the background buffer is emptied, and the background is frozen. The background pointer is set to point to buffer #1, and read. Then background pointer is set to point to buffer #2, and read. See Figure TBD.

Flight Software Science Data Processing

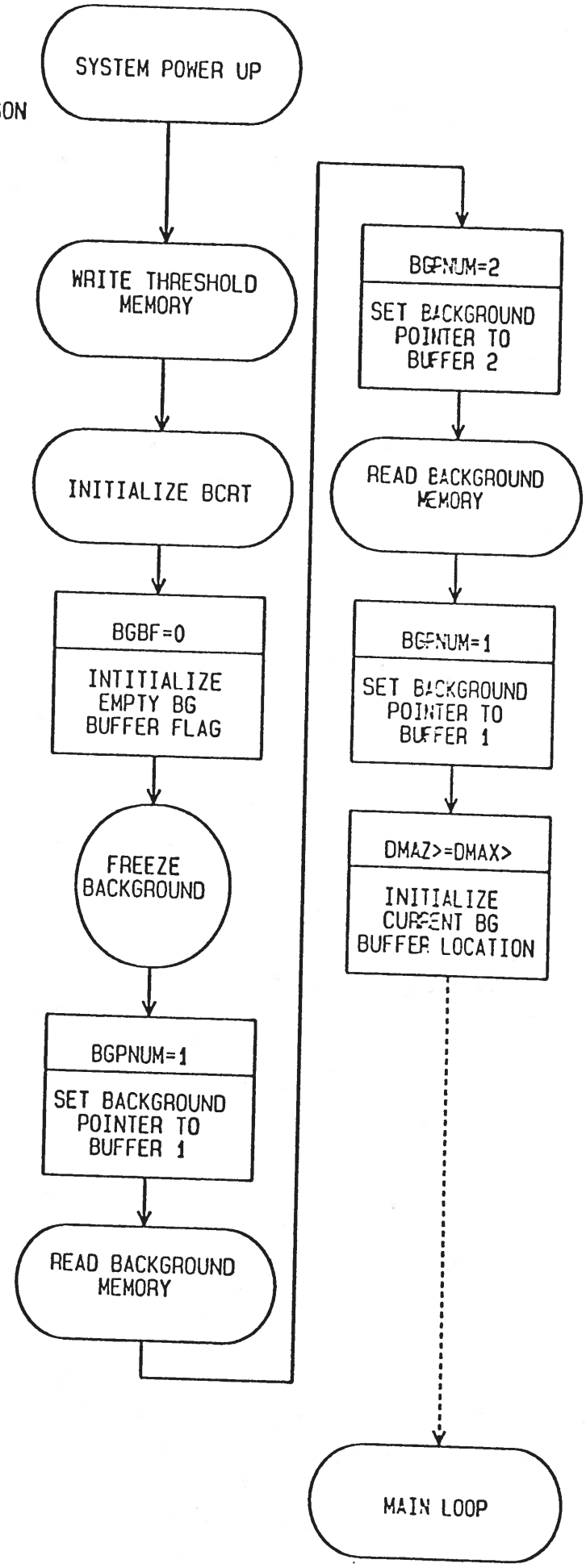



Wait
512th Frame Interrupt
Event Interrupt
Any Events In This Packet ?
NO
YES
Is Background Mode On ?
NO
YES
Is Background Mode On ?
NO
YES
Room For Background Data ?
NO
YES
Add Background
Fill Packet With Background Data
Form CCSDS Headers
Write Data To 'Old Packet' Memory
First Since Last Packet?
YES
NO
Start New Event Packet
READ:
FIFO Word Count
&
Time Tag
READ:
FIFO Data Using FIFO Word Count
Address Correction Routine
Enough Events To Form A Packet?
NO
YES

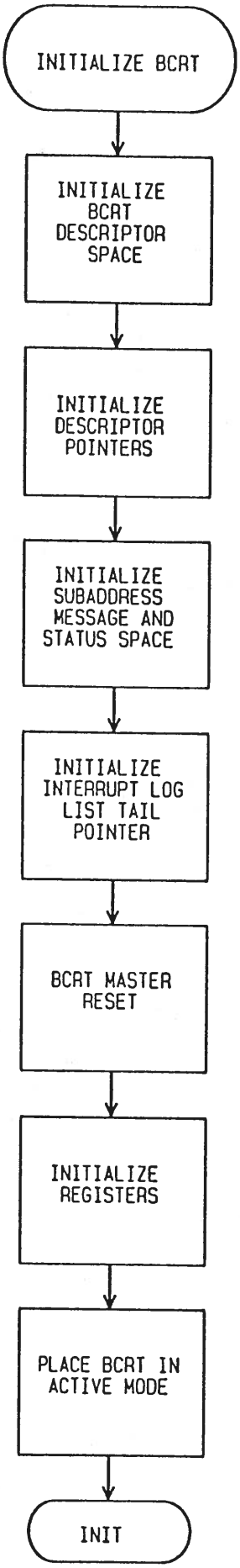
MICROPROCESSOR INITIALIZATION

INIT.EFS

DELISA WILKERSON



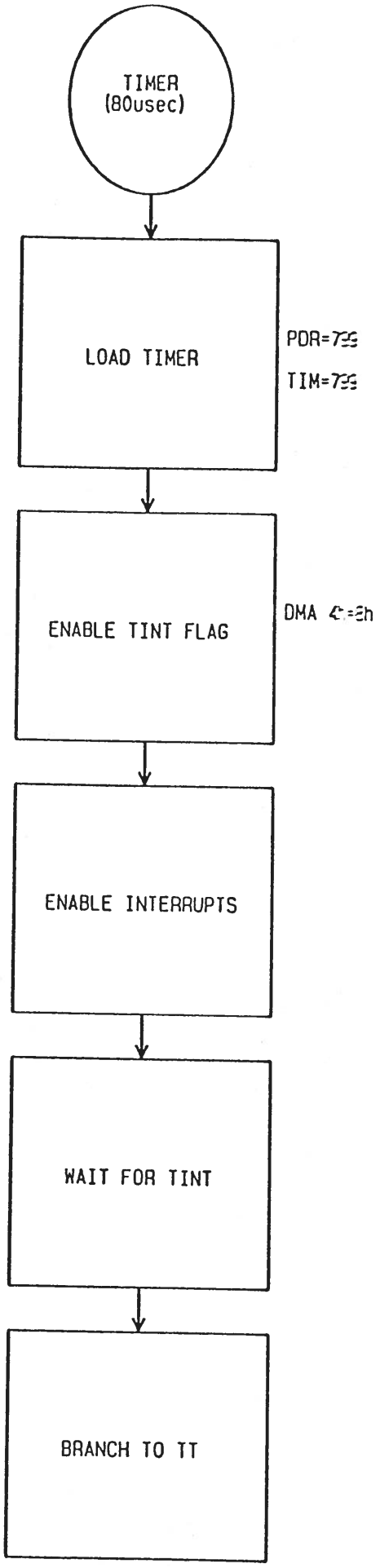
BCRT INITIALIZATION



BCRT00.EFS

DELISA WILKERSON

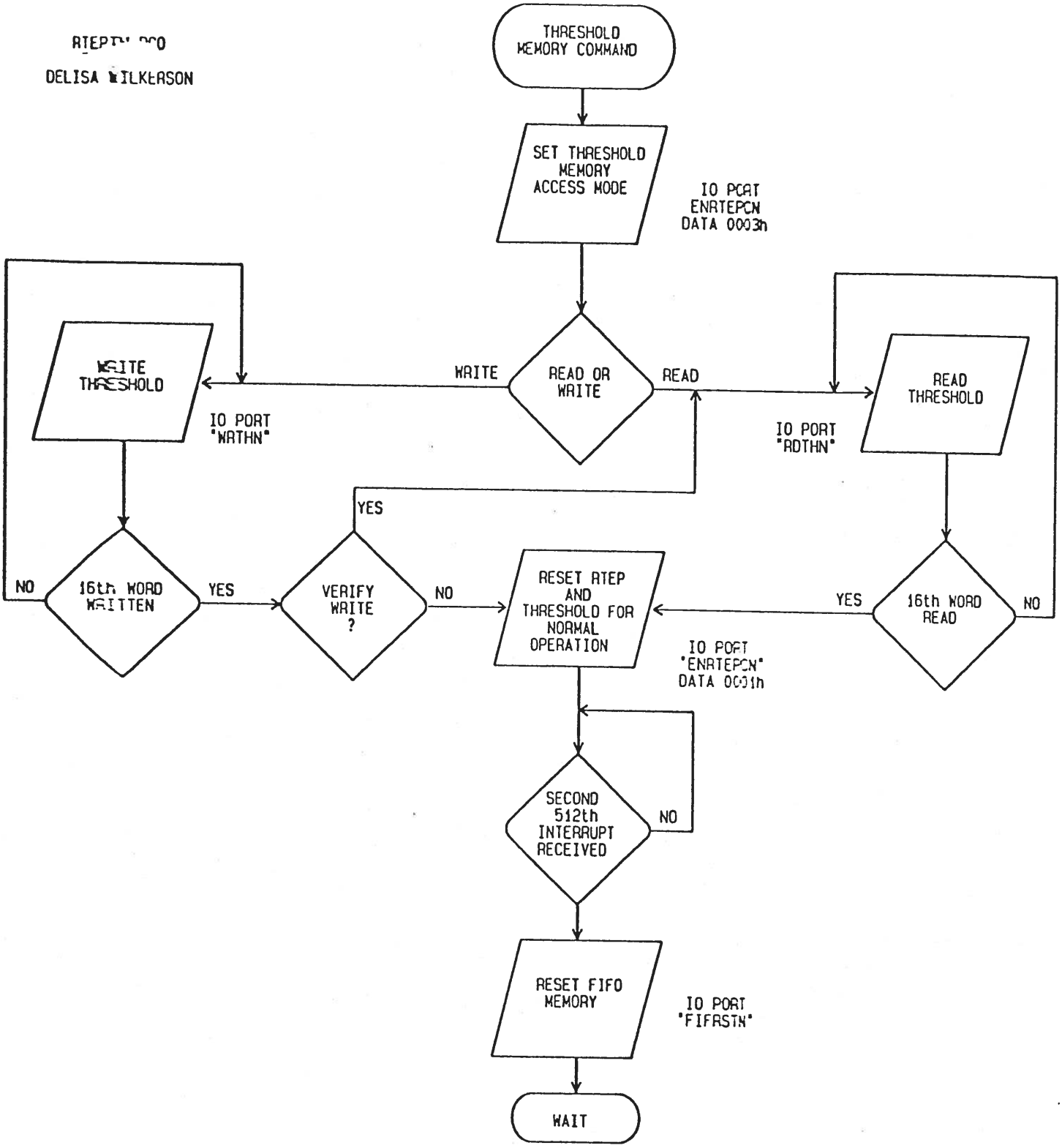
SUBROUTINE TIMER



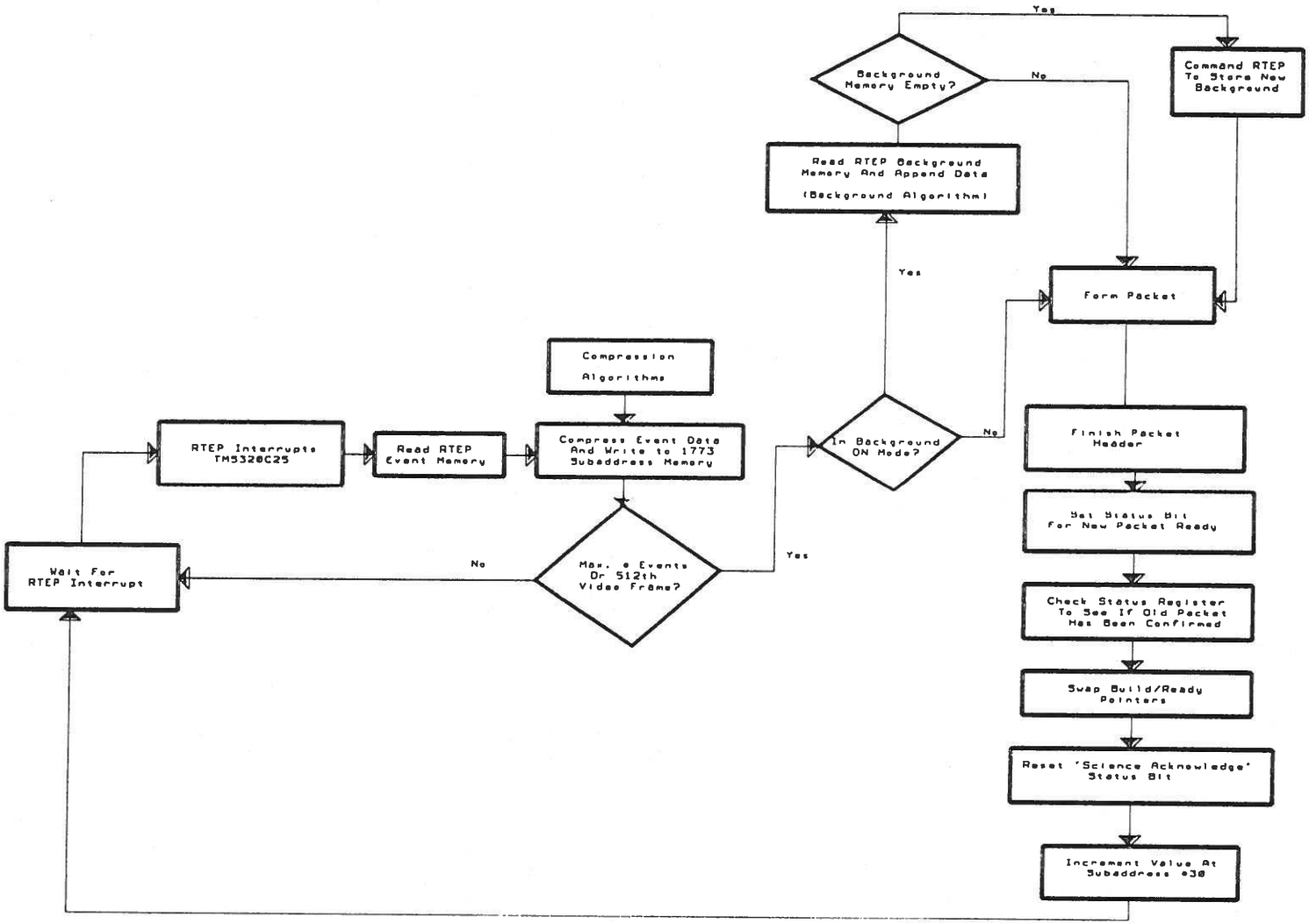
RTEP THRESHOLD MEMORY

RTEPTH PRO

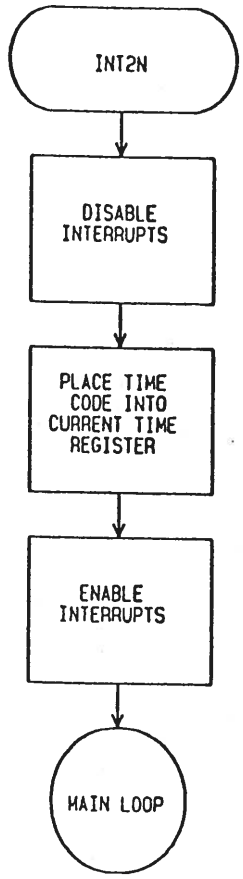
DELISA WILKERSON



Science Packet Formation



TIMEMARK INTERRUPT



3.4.1.2 BCRT Initialization

The BCRT descriptor space, and the descriptor pointers are initialized. The subaddress message and status space is initialized. The interrupt log list tail pointer is initialized. The BCRT master is reset. The registers are initialized, and the BCRT is put in active mode. See Figure TBD.

3.4.1.3 RTEP Threshold Memory Initialization

This module will load the RTEP threshold memory with the default values. After a two second wait, the RTEP is started, and the RTEP interrupt is enabled. See Figure TBD.

3.4.1.4 RTEP Background Initialization

The RTEP background memory if filled, and a timer interrupt enabled. The background send mode is set "ON" in the status word.
See Figure TBD.

3.4.1.5 Focal Plane Array Initialization

The default FPA temperature set point is written. The timer is set, and the timer interrupt is enabled. The aperture door is commanded "OPEN". The aperture door status is checked, and the appropriate bit (open, or error) is set in the housekeeping status word. See Figure TBD.

3.4.1.6 Housekeeping Buffer #1 Initialization

Housekeeping buffer #1 is pointed to, and the housekeeping DAS board is reset. The timer is set for 40 microseconds, and the timer interrupt enabled. The first time flag bit is checked. If true, a housekeeping word is read, and the word count is zero. If false, the word count is incremented. This is repeated until the word count equals 16. Then, the housekeeping DAS buffer #1 empty flag equals false, and a housekeeping packet is ready to be sent. See Figure TBD.

3.4.2 Packet Communication

3.4.2.1 CCSDS Format

Flight software follows the CCSDS Format specified in the LIS ICD (Reference TBD). The details for the LIS packets is given in Figure TBD.

3.4.2.2 FDS Communication Protocol

All data transferred shall be transmitted as CCSDS telemetry packets, as given in section 2.2.2, "Packet Communications" in the TRMM Flight Data System document. In addition, the CCSDS packet is distributed across the 32 word 1773 subaddresses. The FDS reads a LIS subaddress to determine if the data counter has been incremented. If so, the FDS transfers the 1773 subaddresses, then commands a subaddres read to signal confirmation of the transfer. This signals the microprocessor that the transfer has tacken place and that a new packet can be built in that memory space.

3.4.2.3 Science Packets

The science data packets consist of RTEP and/or background data collected. This data is compressed to remove superflous time, line, and pixel address information, and to correct for the addressing offsets introduced by the RTEP pipeline structure and the scan pattern of the CCD readout electronics.

3.4.2.4 Housekeeping Packets

A Housekeeping instrument packet consists of 16 housekeeping words collected from the housekeeping DAS, plus status word(s). Status words show error status, command verification, and other miscellaneous status. (See figure TBD).

3.4.2.5 Packet Formation Operation

Flight software maintains pointers to two locations for science packets, and two locations for housekeeping packets. A completed data packet is stored in one location, which is ready for transmission to the TRMM FDS. The second location contains a new packet in formation.

After the transmission of a data packet is confirmed, flight software swaps the pointer, and a new packet is formed in the location of the packet that has been confirmed.

3.4.2.6 Time Data Format and Operation

Coarse time data is transmitted by the FDS over the 1773 bus. The format is described in the FDS Appendix (Reference TBD), section 2.2.6. Software maintains pointers to two locations. One points to the current caorse time, the other to the location of new coarse time. The time-mark interrupt is the signal to swap these pointers, updating the coarse time.

3.4.3 Interrupt and Exception Routines

Exceptions and interrupts are used to govern the interaction of the microprocessor with the other LIS subsystems. When the microprocessor is interrupted,

further interrupts are disabled, and the FIFO count is read. When an exception is being processed, the microprocessor may still be interrupted, and will return to the exception following processing of the interrupt.

3.4.3.1 RTEP Interrupt

The RTEP status is read in. The main loop counter is initialized. The value of bit 14 is checked. If B14 = 1, the RTEP status bits are stored, the FIFO count is stored, and the upper 4 bits are zeroed. The main loop counters are stored with the FIFO count.

3.4.3.2 TimeMark Interrupt

The time mark is sent to the microprocessor via RS-422. The time mark pulse is sent once a second. Once received, the microprocessor's fine time is updated (automatically by the hardware) and the microprocessor swaps the coarse-time pointers.

3.4.3.3 BCRT Interrupt

When a valid instrument command is received, (as designated by the BCRT recognizing the appropriate RT and subaddress addressing of the 1773 command word), the microprocessor is interrupted by the BCRT to implement the command. The microprocessor reads the command from the memory space, where it has been stored by the BCRT.

3.4.3.4 Filter Temperature Exception Monitoring

The microprocessor software will monitor the temperature (via the analog housekeeping DAS) to determine wheter the filter temperature is being maintained within the nominal operating range. If not, the software will reset both heater controllers, and activate the one that was not functioning when the exception was generated.

3.4.4 Background Operations

3.4.4.1 Background Memory Read

The command is sent to capture the background (GETBMN). There is a 6 millisecond wait, and the the background memory is read on I/O port RDBMN. This is repeated until 127 lines of background memory have been read.

3.4.4.2 Packetize Background

The background is loaded into a packet. The background packet storage line

count is initialized (BGPCNT=0). The background data store is read, and the packet storage address, and the buffer address are incremented. These steps are repeated until the background packet = 128.

After 128 background packets are stored, the background packet lines stored is checked. If BGPLS=0, the one line is stored, and then return to the main loop.

3.5 Interface Description

3.5.1 Hardware Interfaces

The only interface external to the LIS instrument is the LIS to TRMM interface. This entails software interfaces that can receive commands across the MIL STD 1773 bus. LIS acts as a 1773 remote terminal in this system. There is also a Rs-422 reciever, used to receive the time-mark.

Within LIS, the controller interfaces to the RTEP, the FPA, and the DAS, as well as the memory space shared with the BCRT for 1773 interface.

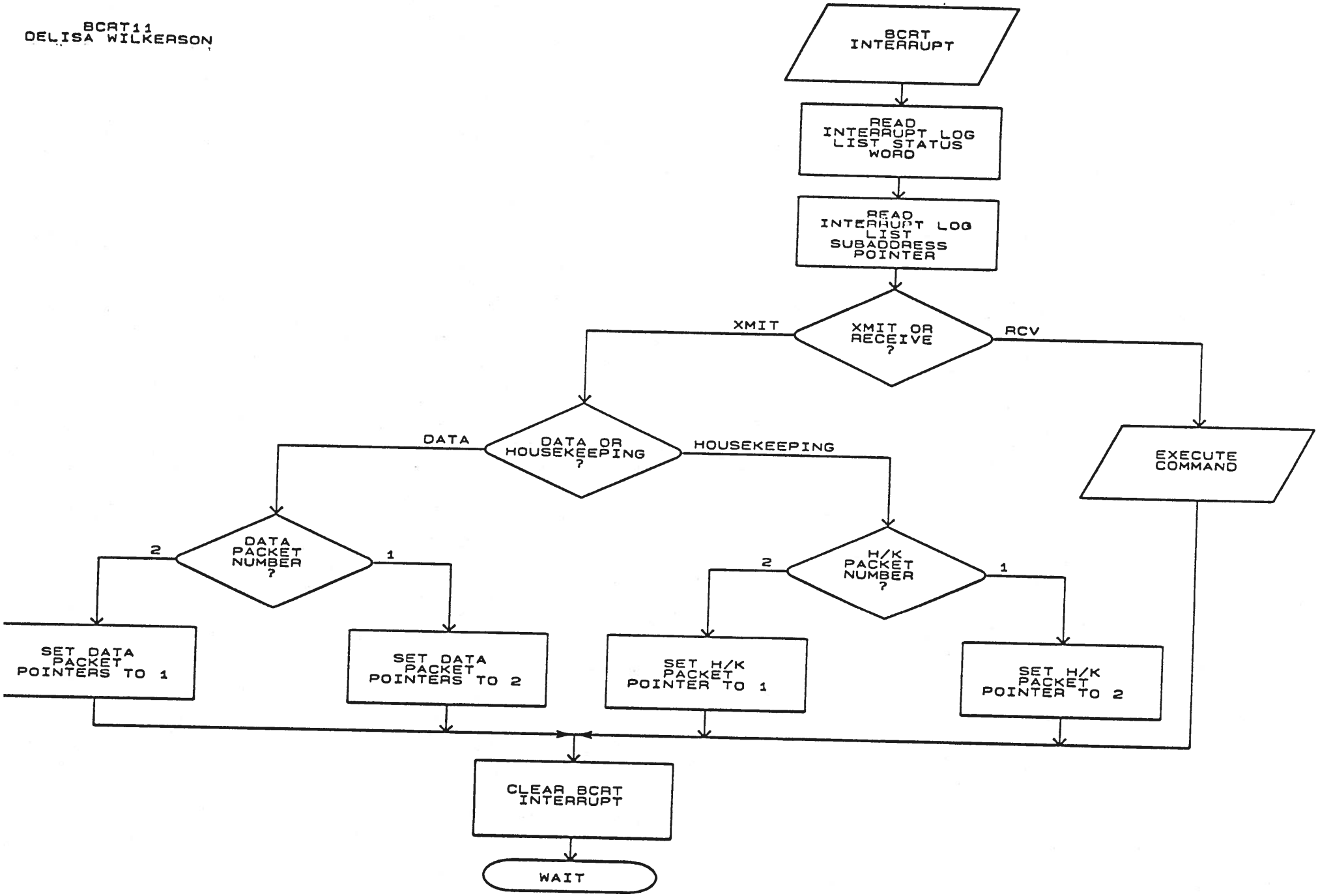
3.5.2 Software Interfaces

The BCRT and the microprocessor share memory space. This forms a type of software interface, since both can act on the data without knowledge of the he other. In addition, when the BCRT is addressing the memory, the microprocessor is placed in a hold state, and can not address external memory or peripherals. The microprocessor alters the pointers that the BCRT uses to locate the various 1773 subaddresses. This is explained more fully in figure TBD.

3.6 Detailed Design

E RT INTERRUPT

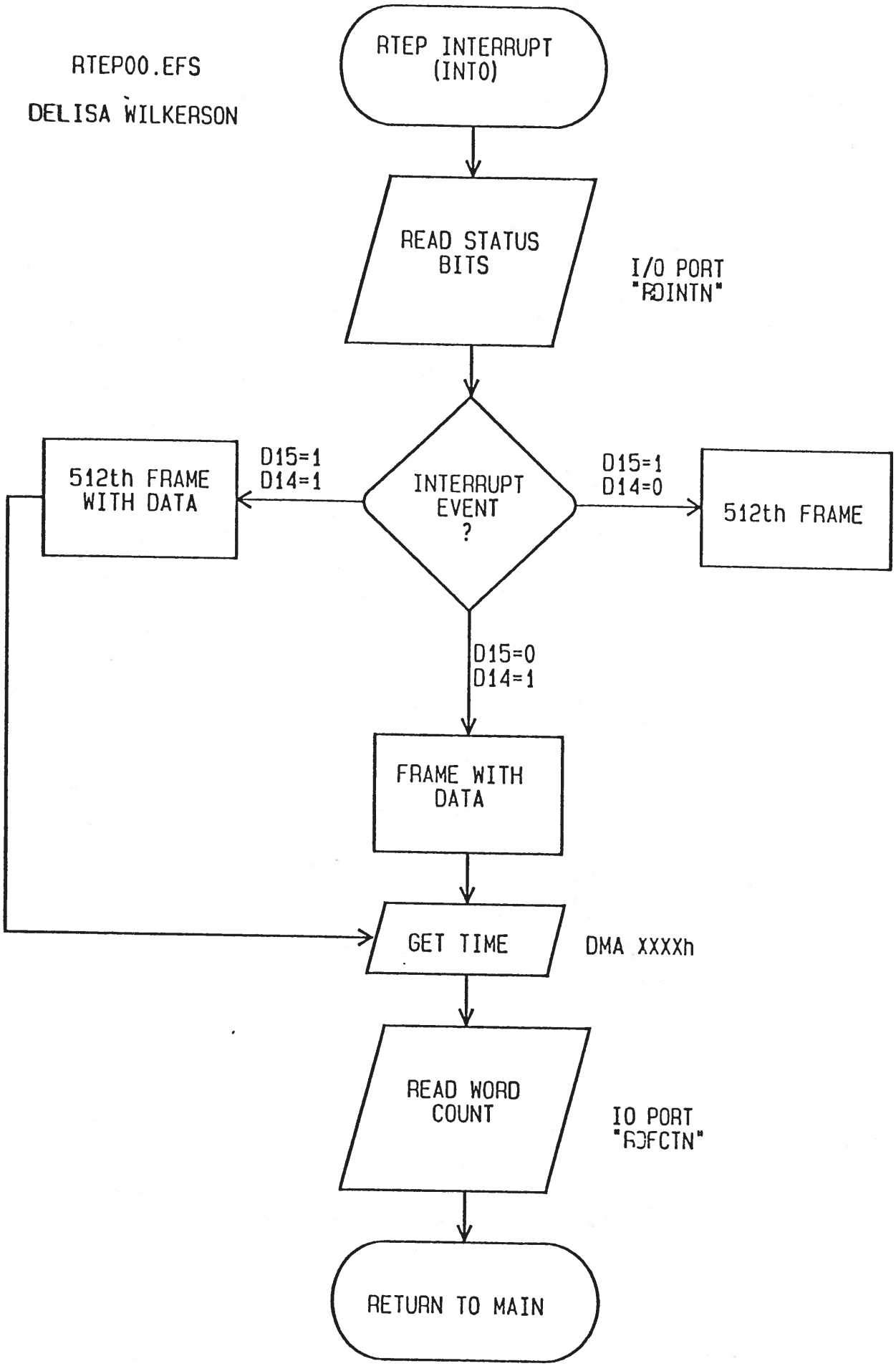
BCRT11
DELISA WILKERSON



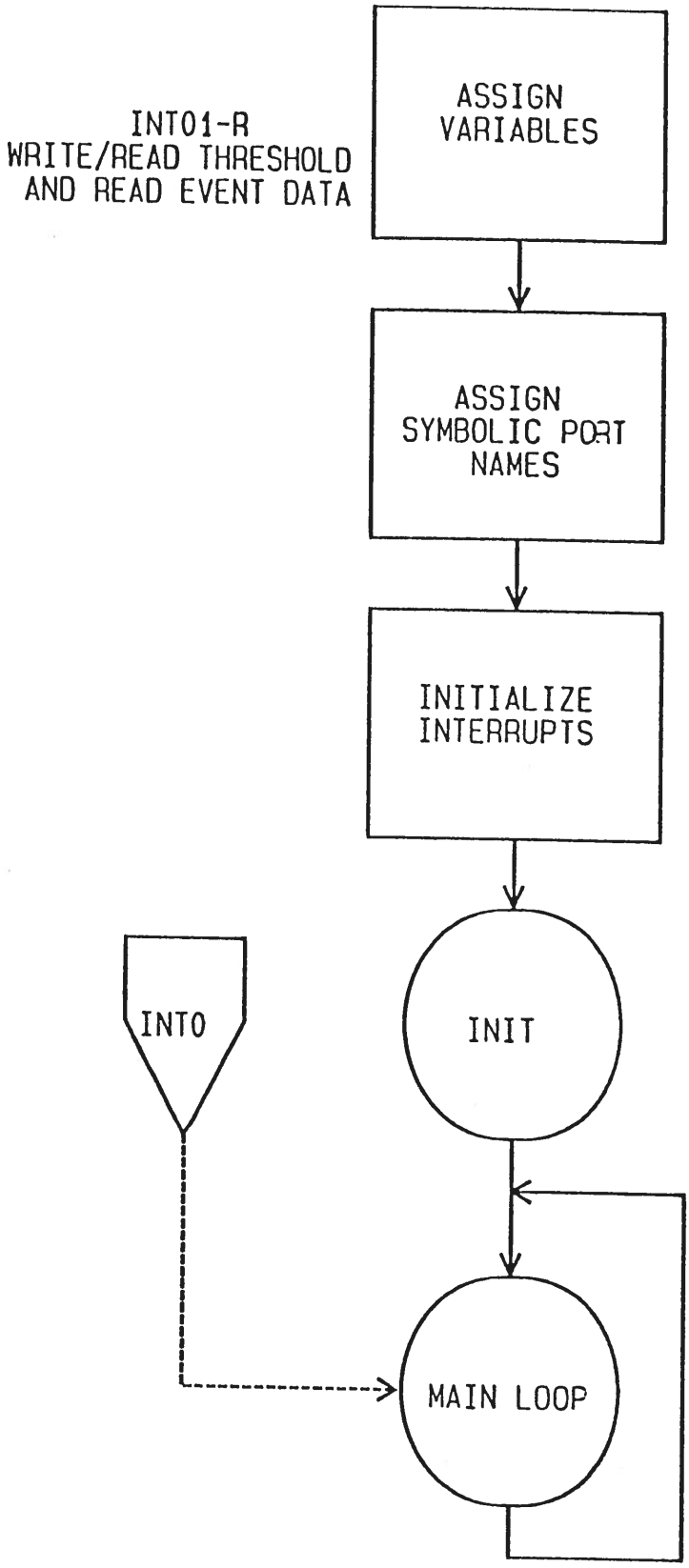
RTEP INTERRUPT

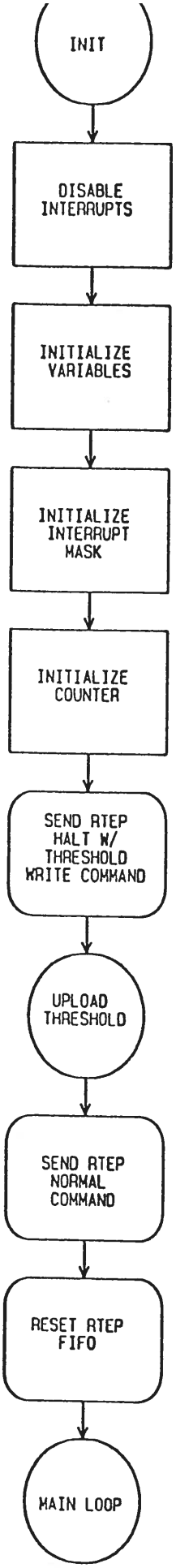
RTEP00.EFS

DELISA WILKERSON



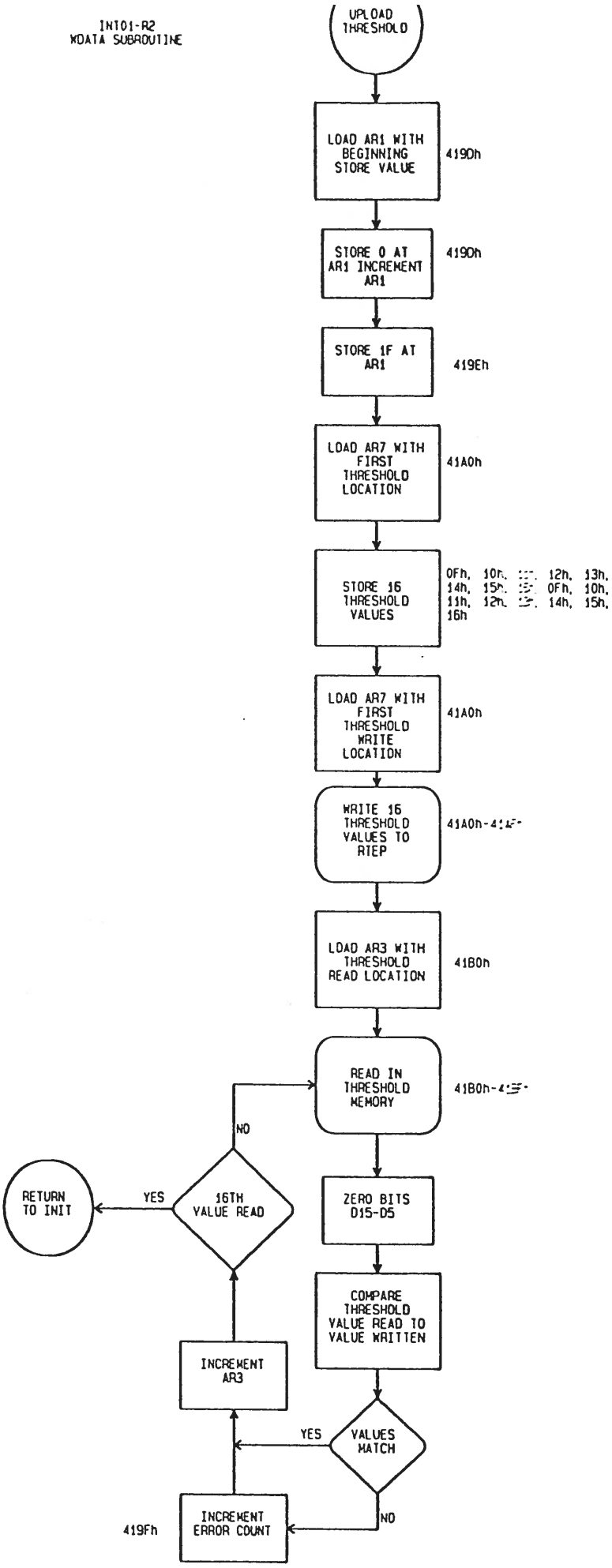
PROGRAM FOR READING HILF EVENT DATA



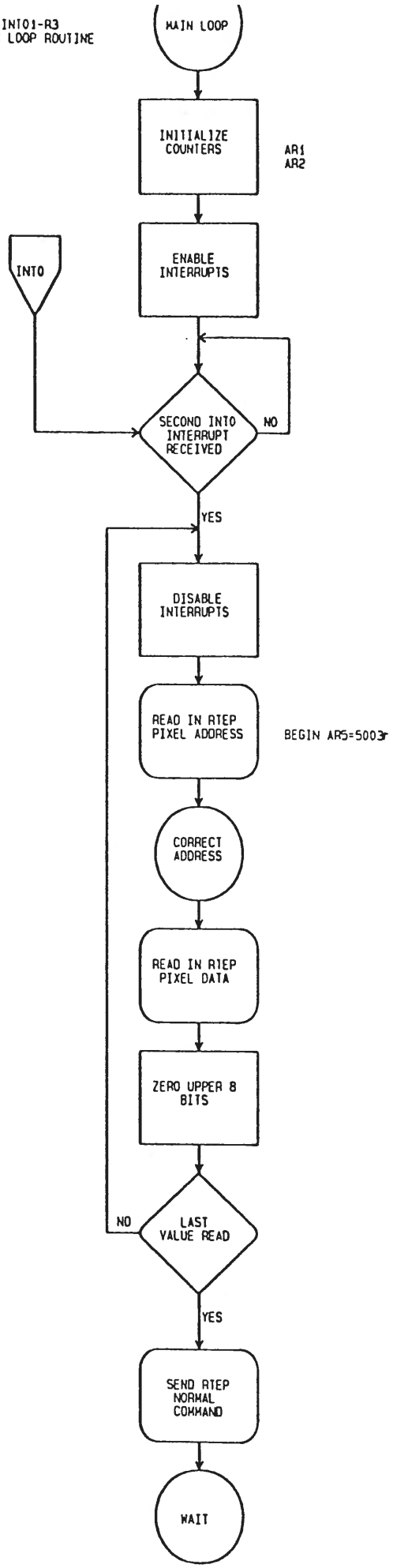


INT01-R1
INIT ROUTINE

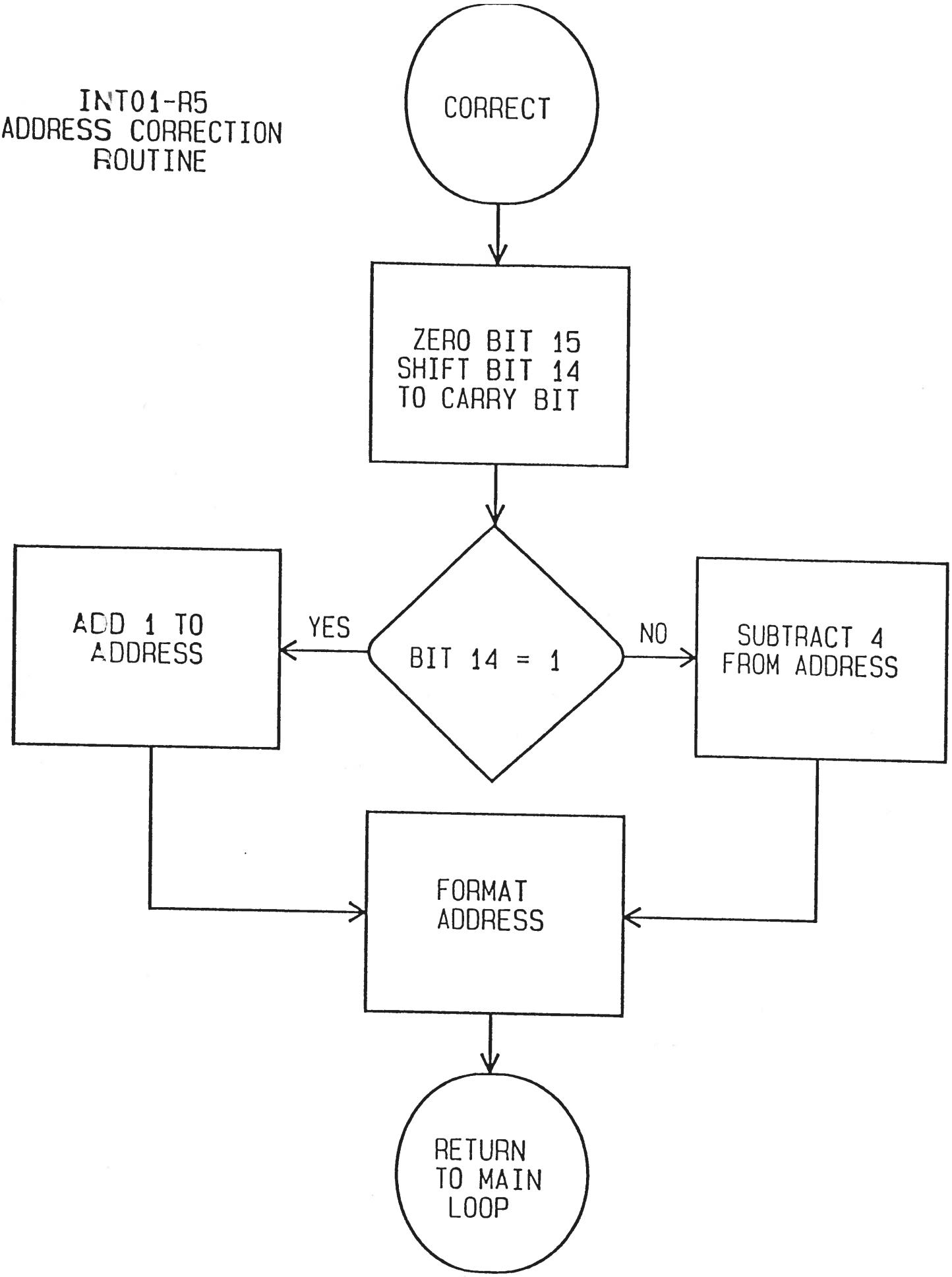
INT01-R2
WDATA SUBROUTINE



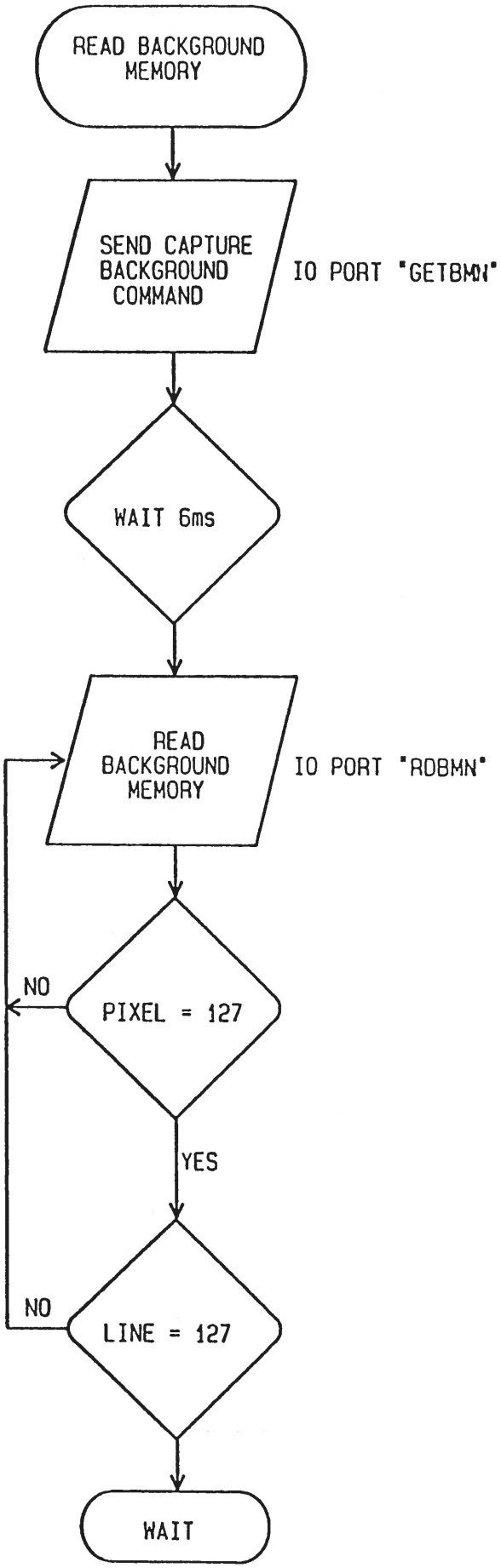
INT01-R3
MAIN LOOP ROUTINE



INT01-R5
ADDRESS CORRECTION
ROUTINE



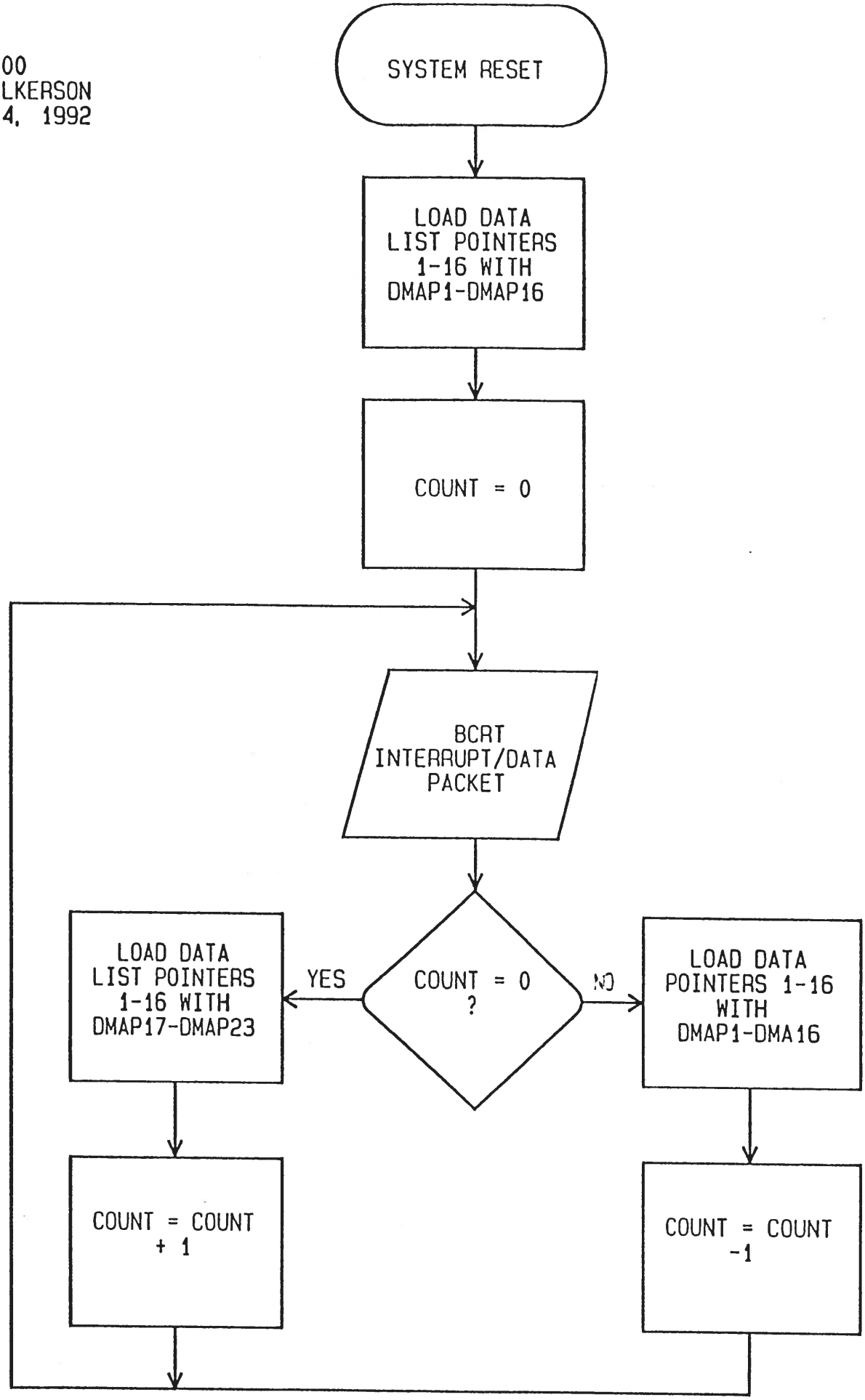
BACKGROUND MEMORY READ



RTEPBG00

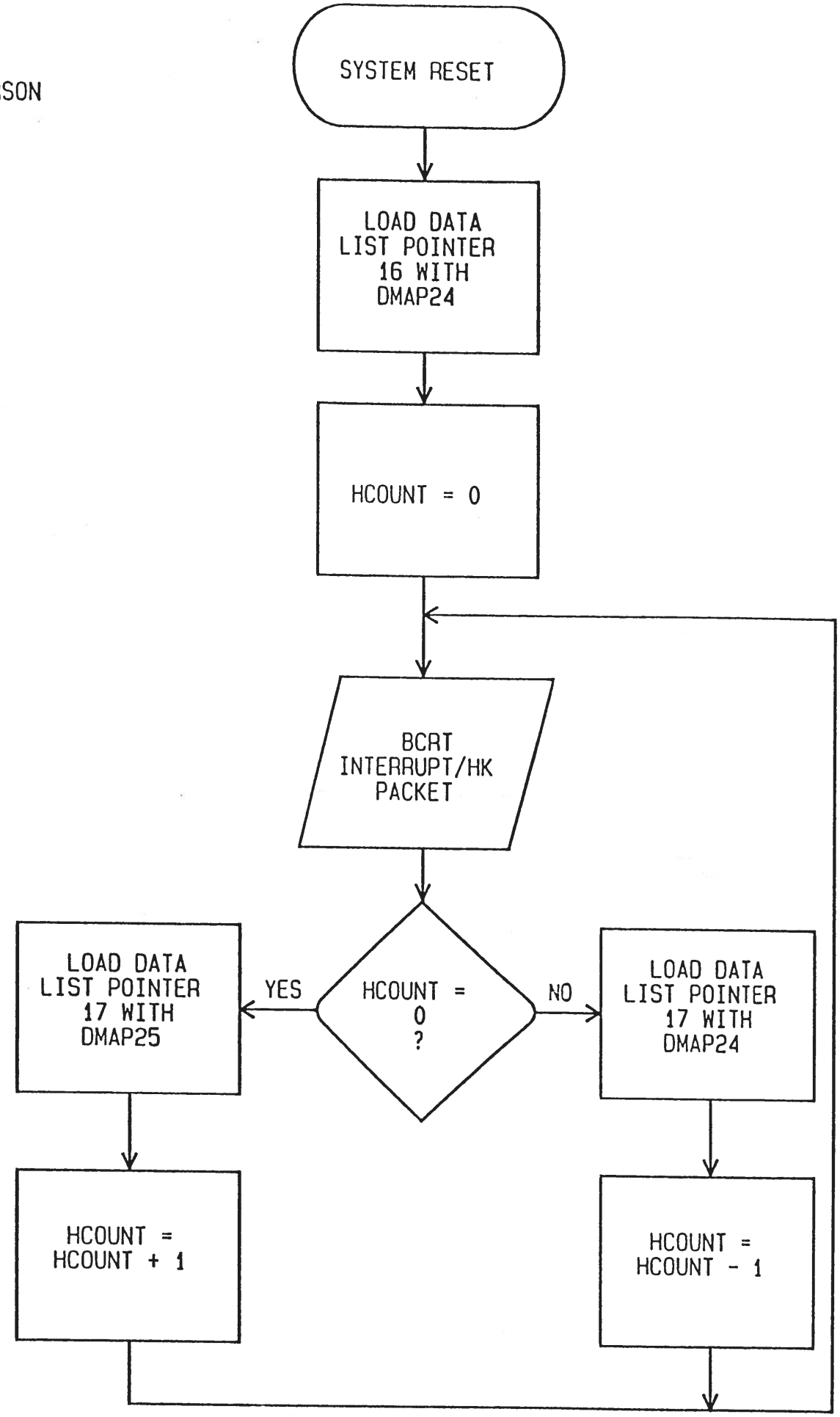
DELISA WILKERSON

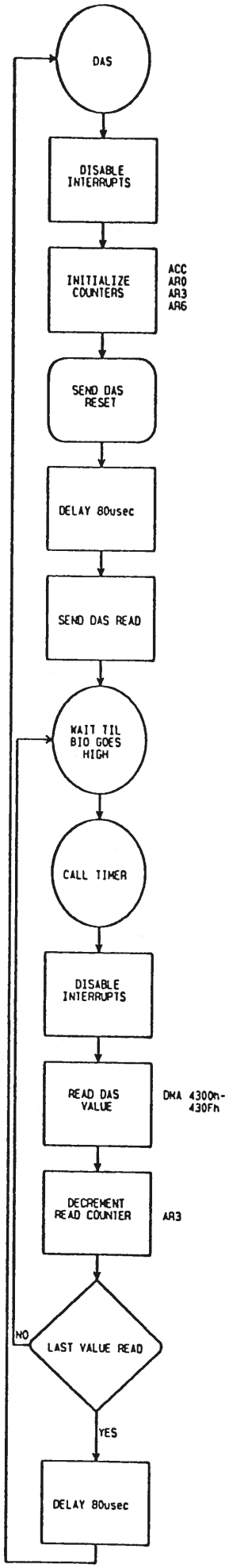
PACK00
DELISA WILKERSON
JANUARY 14, 1992



HOUSEKEEPING PACKET SELECTION

PACK01
DELISA WILKERSON

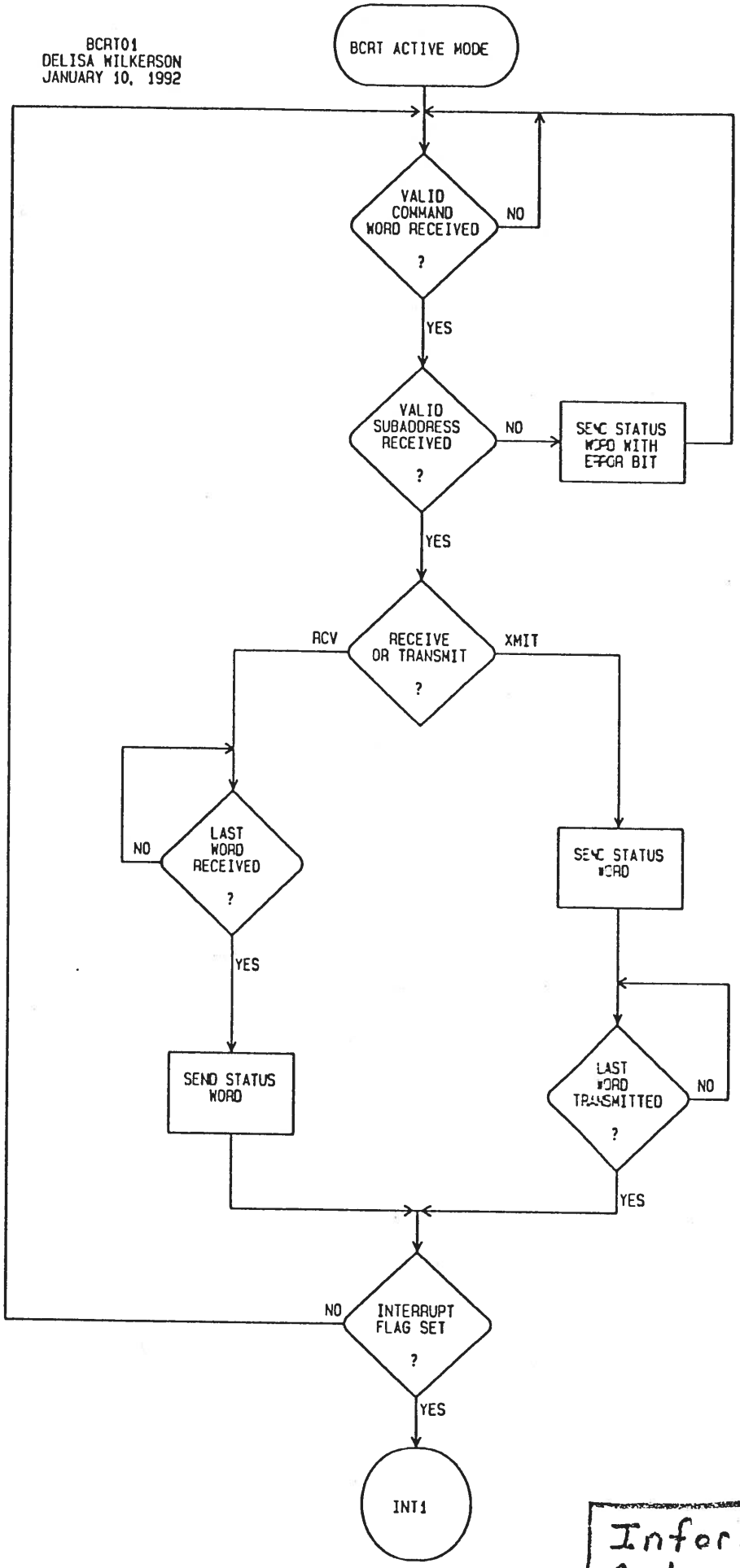




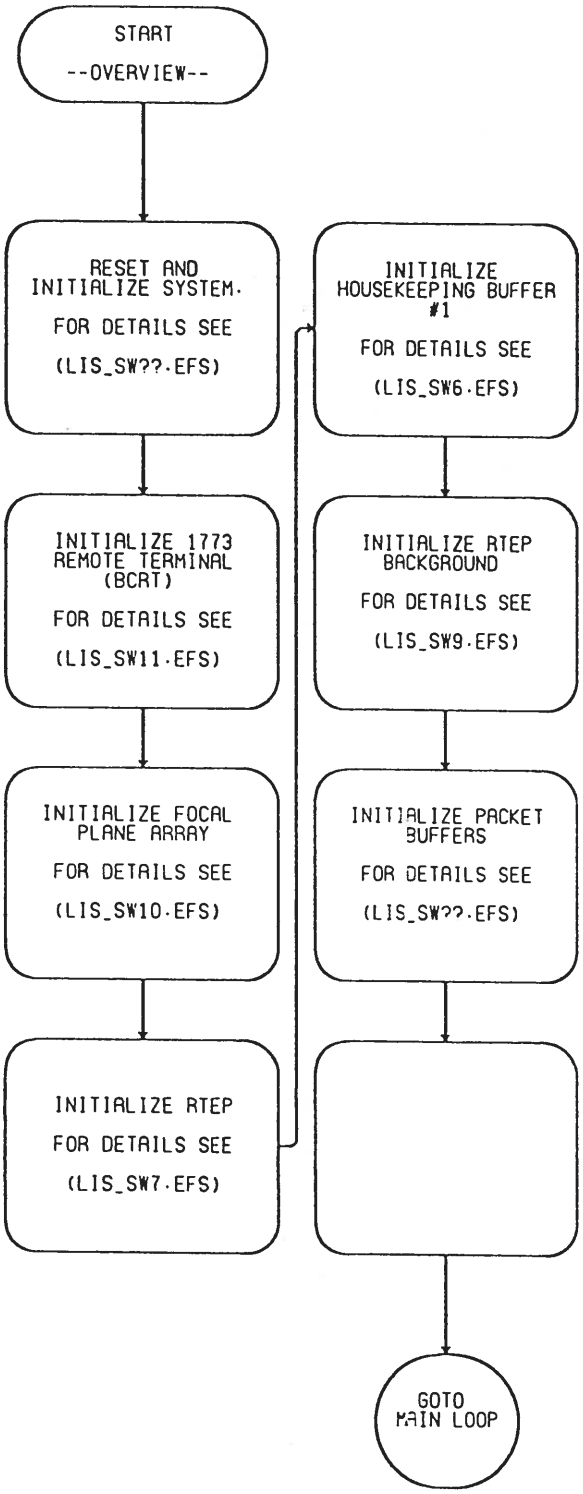
Preliminary

BCRT OPERATION
(All Operations done by BCRT)

BCRT01
DELISA WILKERSON
JANUARY 10, 1992

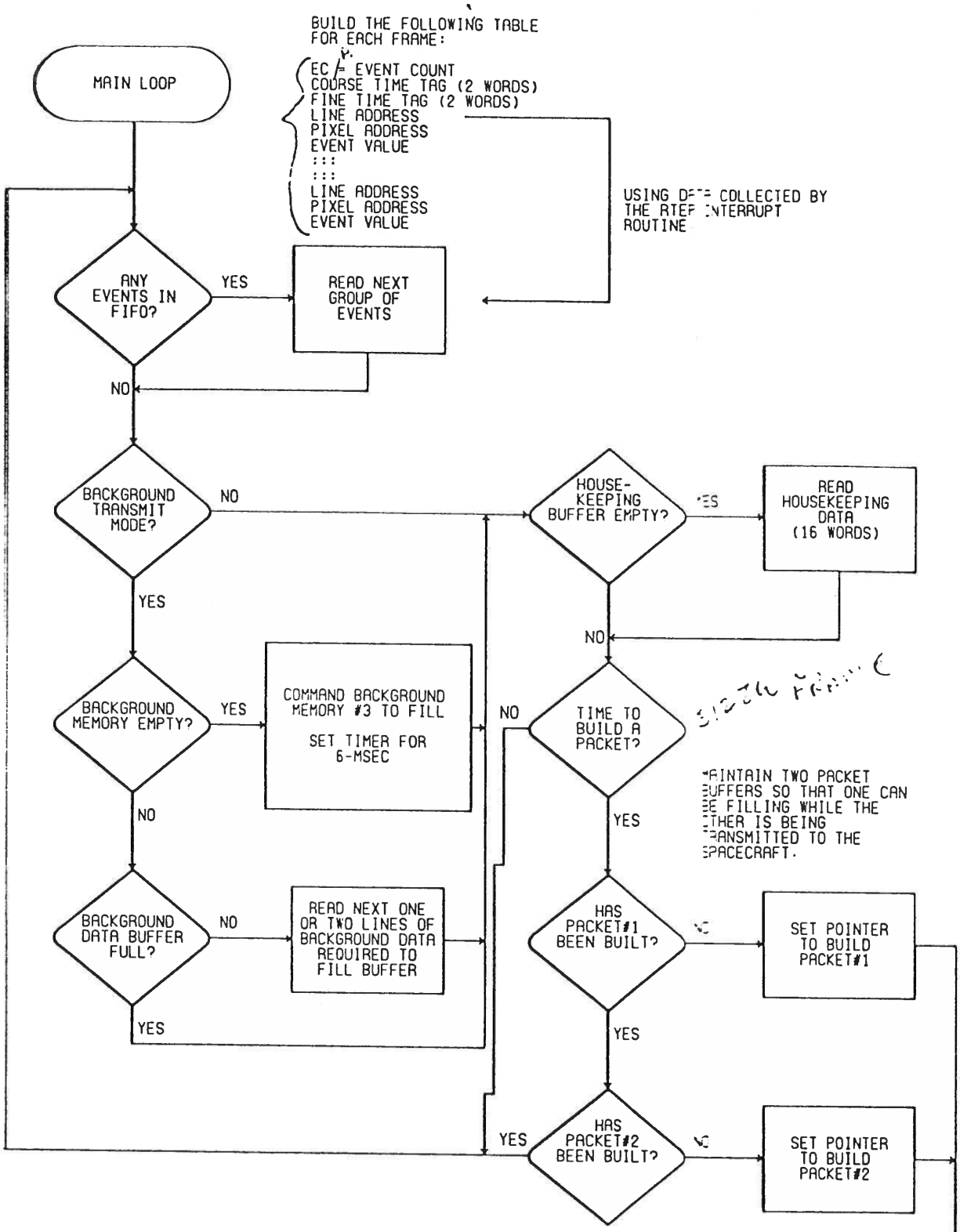


Information Only



LIS_SW1.EFS

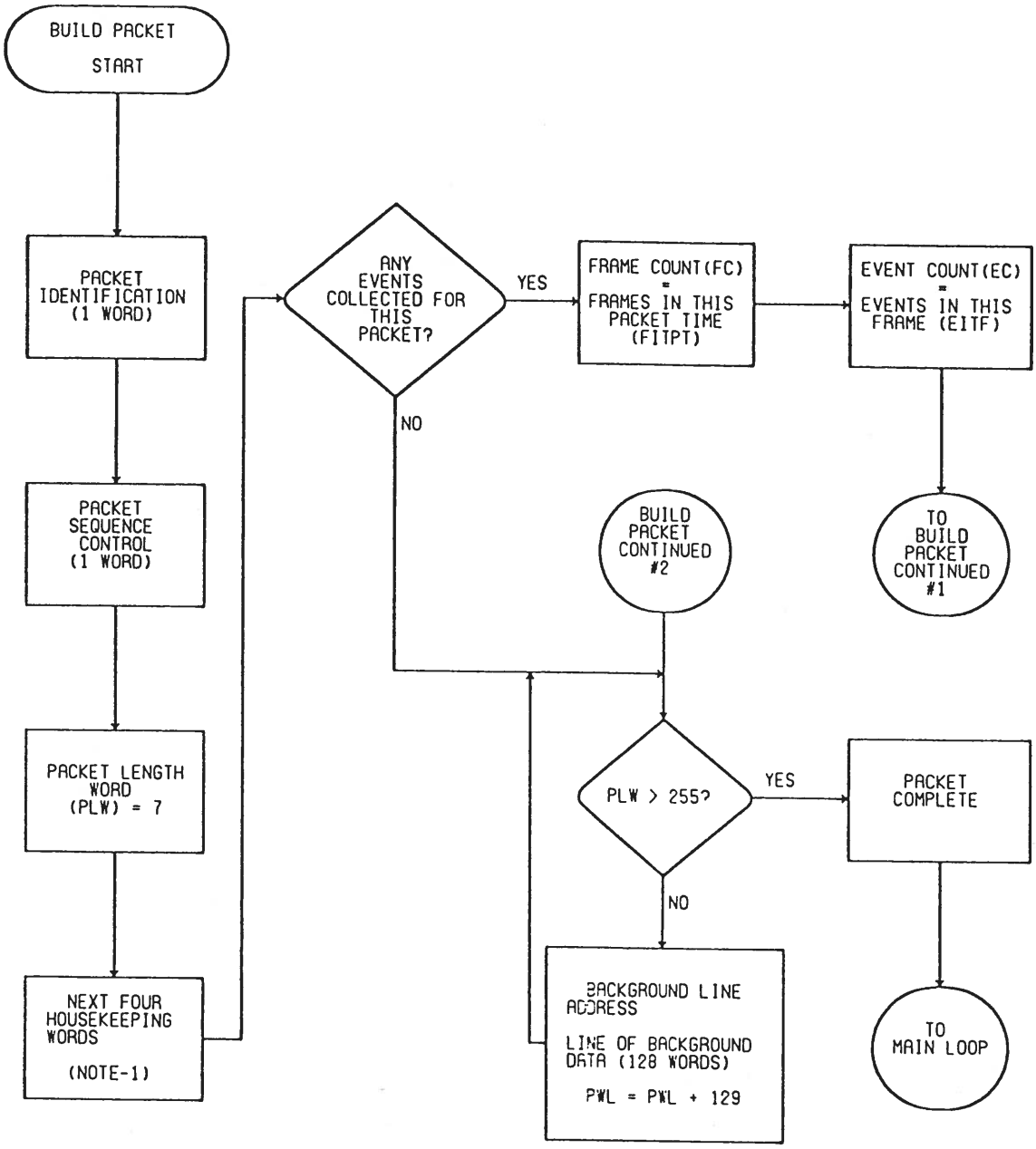
JON R. REHAGE



LIS_SW2.EFS

JON R. REHAGE

PAGE 2 of



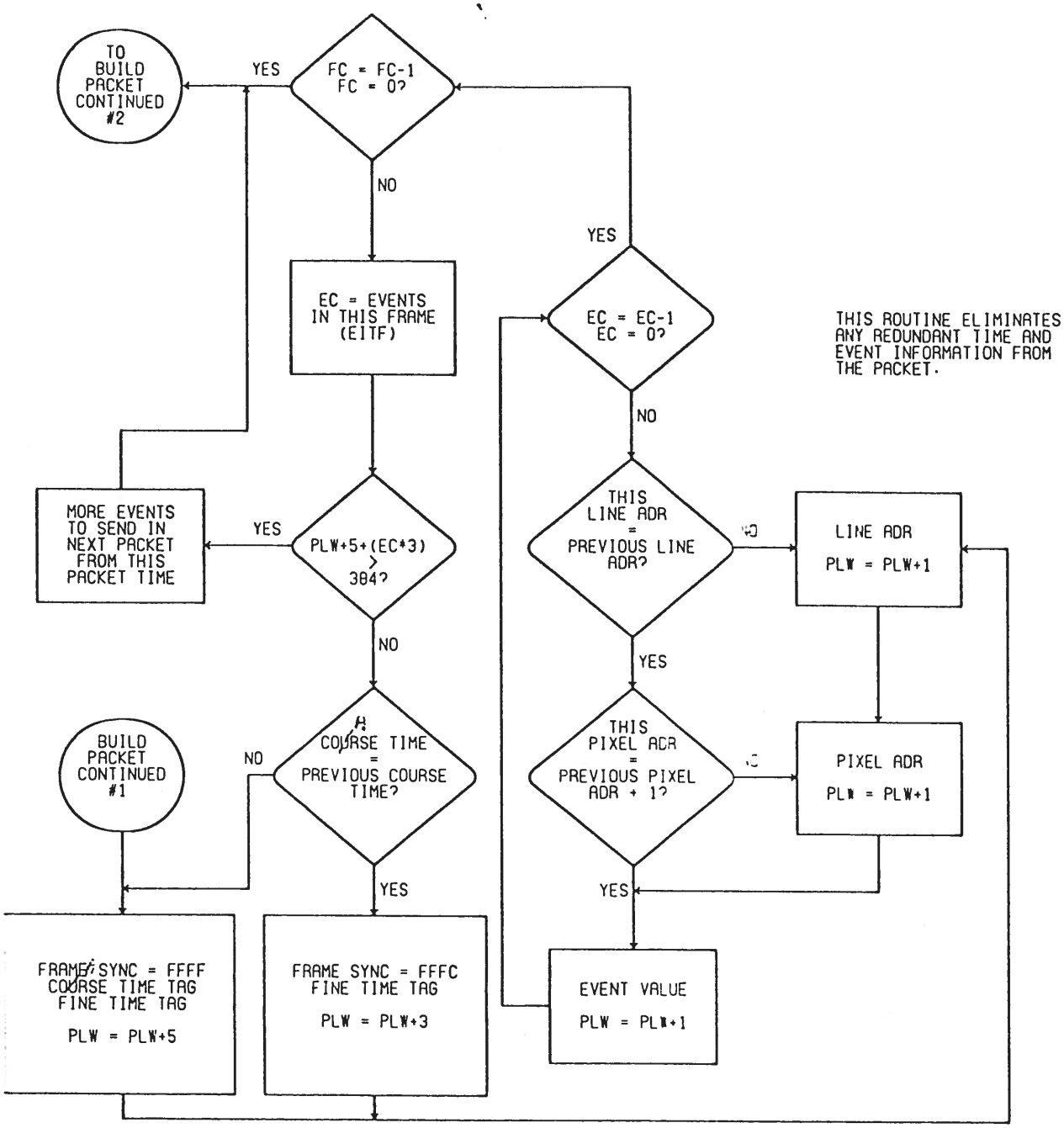
LIS_SW3.EFS

JON R. REHAGE

PAGE 3 of

NOTE-1:

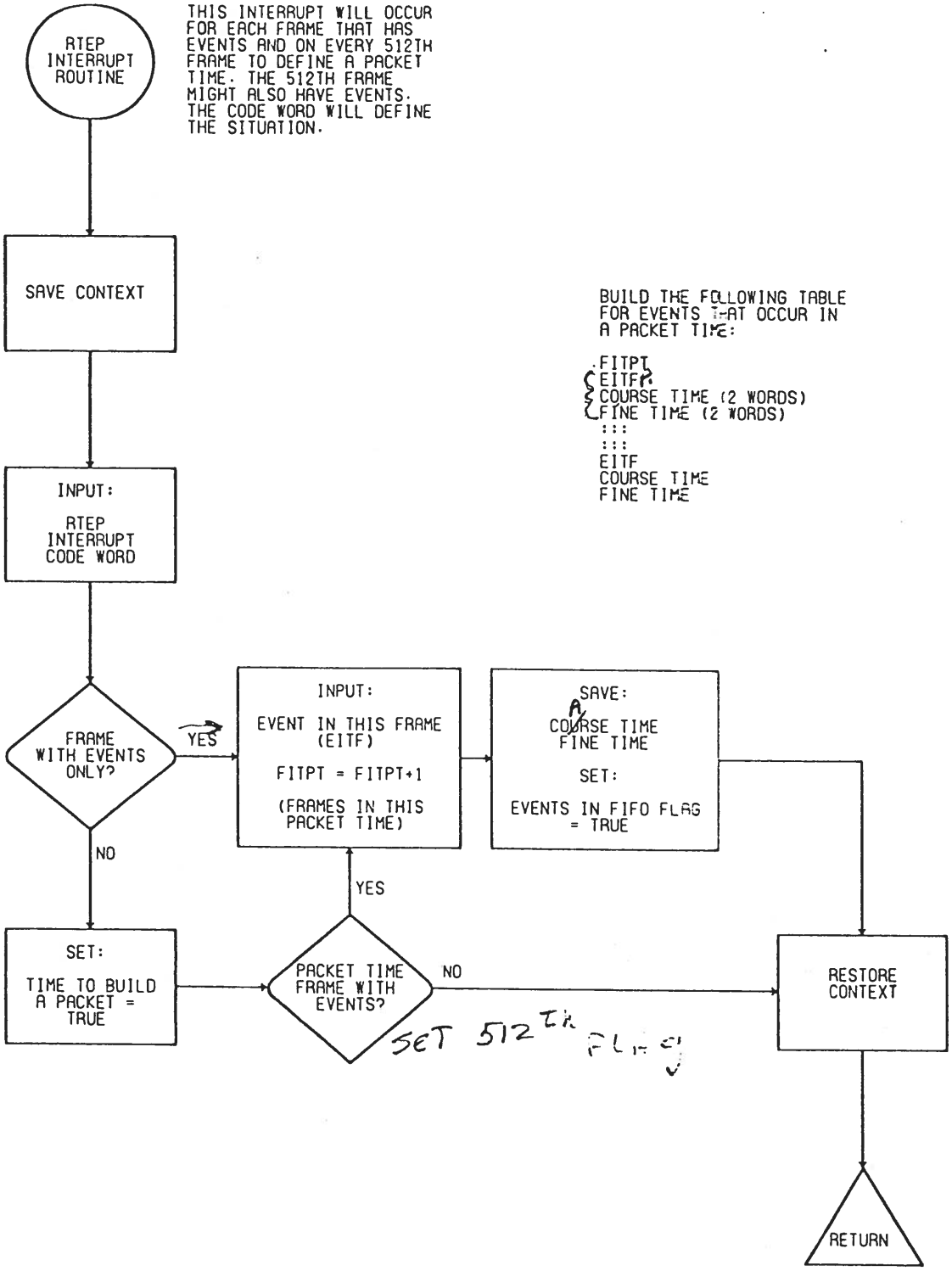
THE TWO LS-BITS OF THE PACKET SEQUENCE CONTROL--DEFINES THE FOUR HOUSEKEEPING WORDS IN THE PACKET--
00=1st FOUR,
01=2nd FOUR, ETC.



LIS_SW4.EFS

JON R. REHAGE

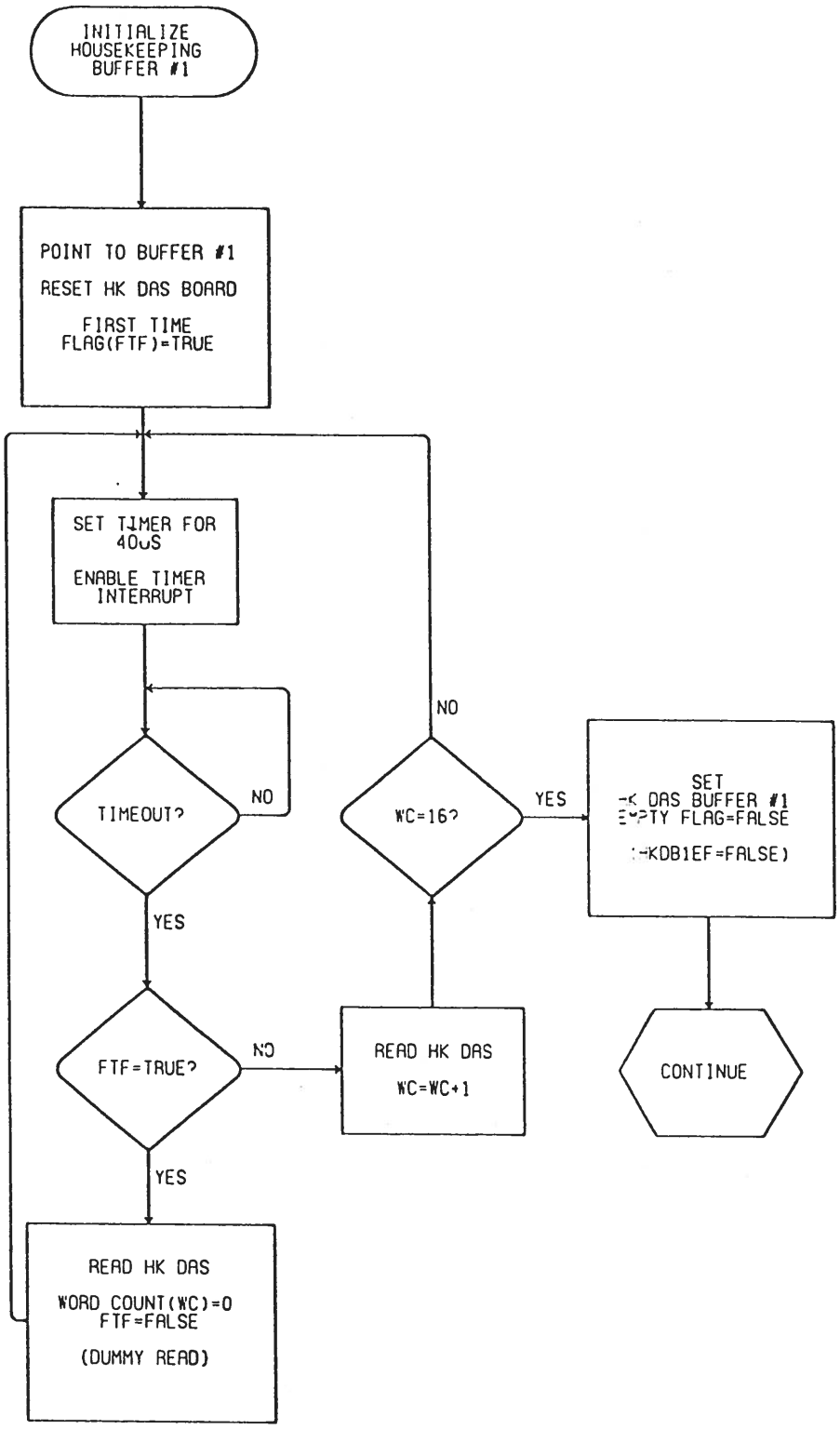
6-14-91



LIS_SW5.EFS

JON R. REHP

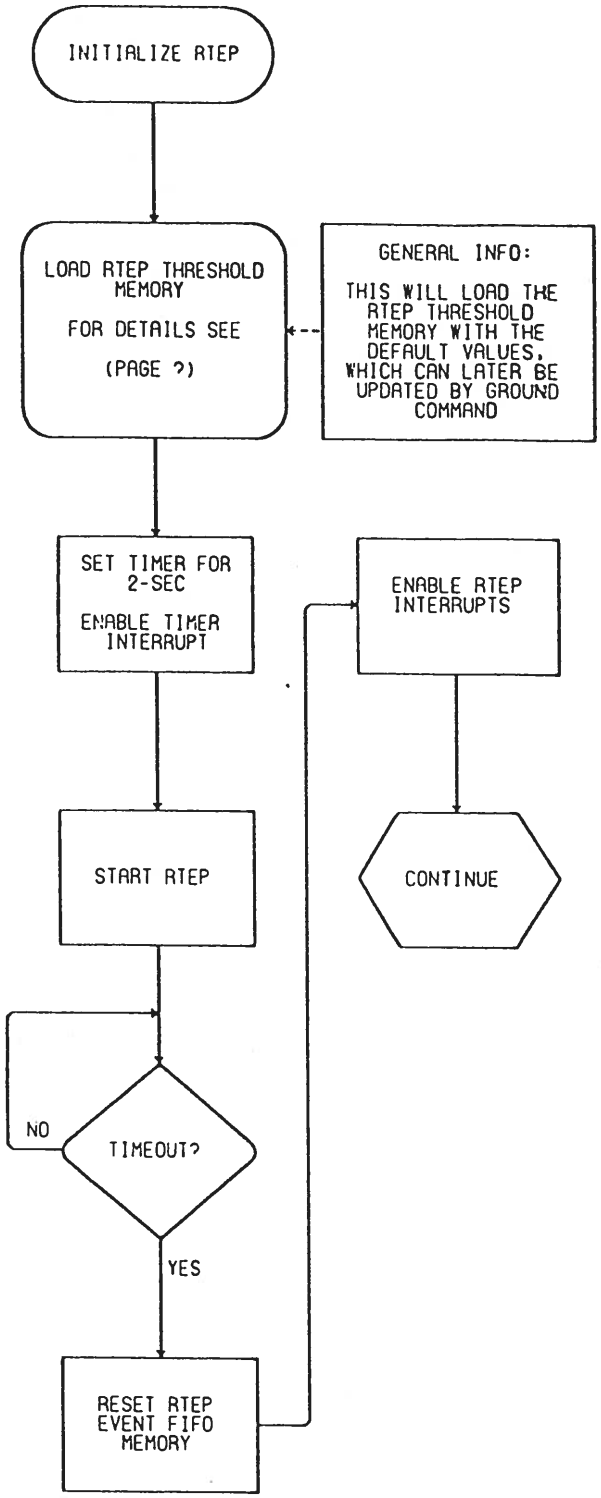
PAGE 5 of

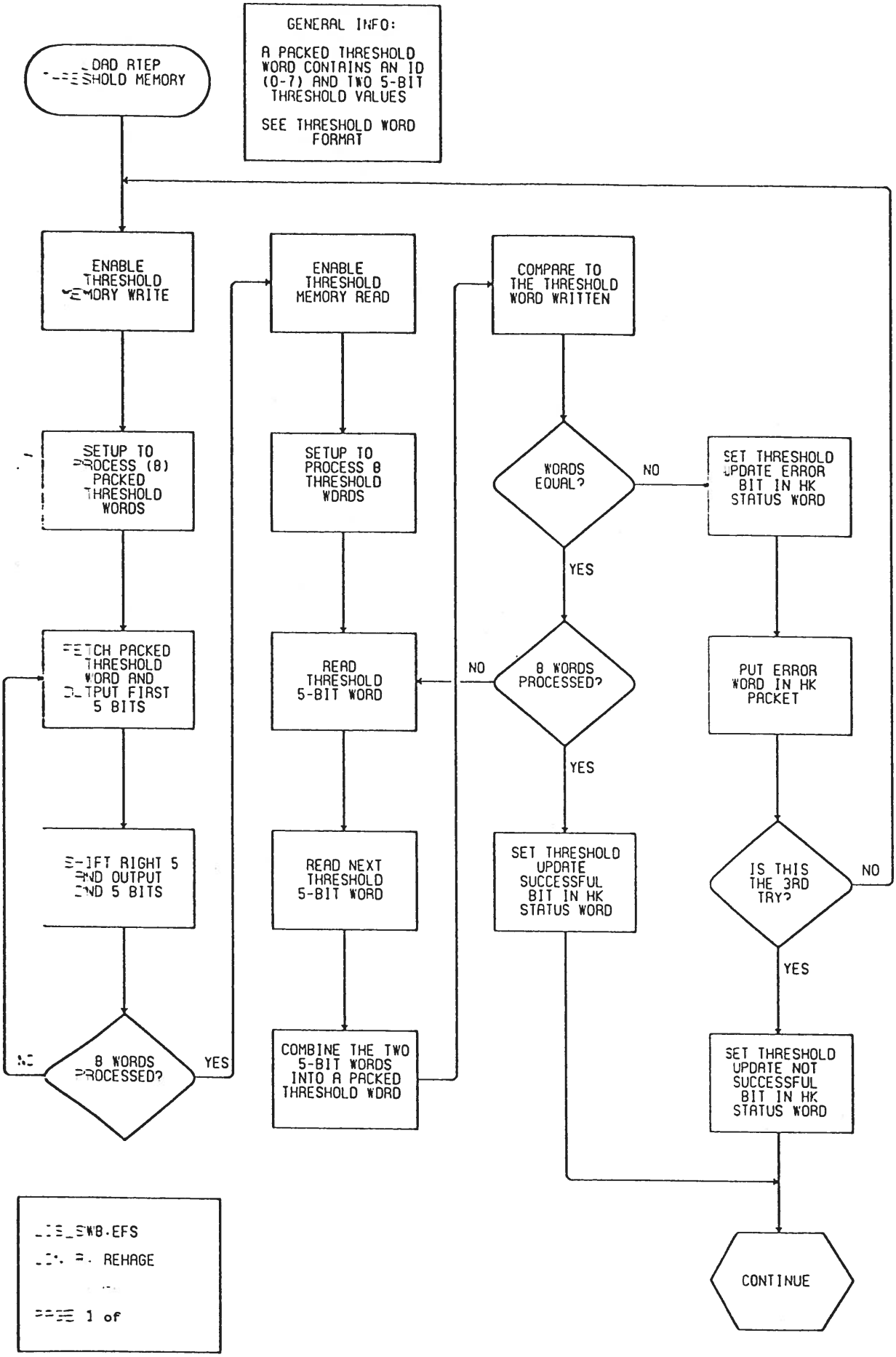


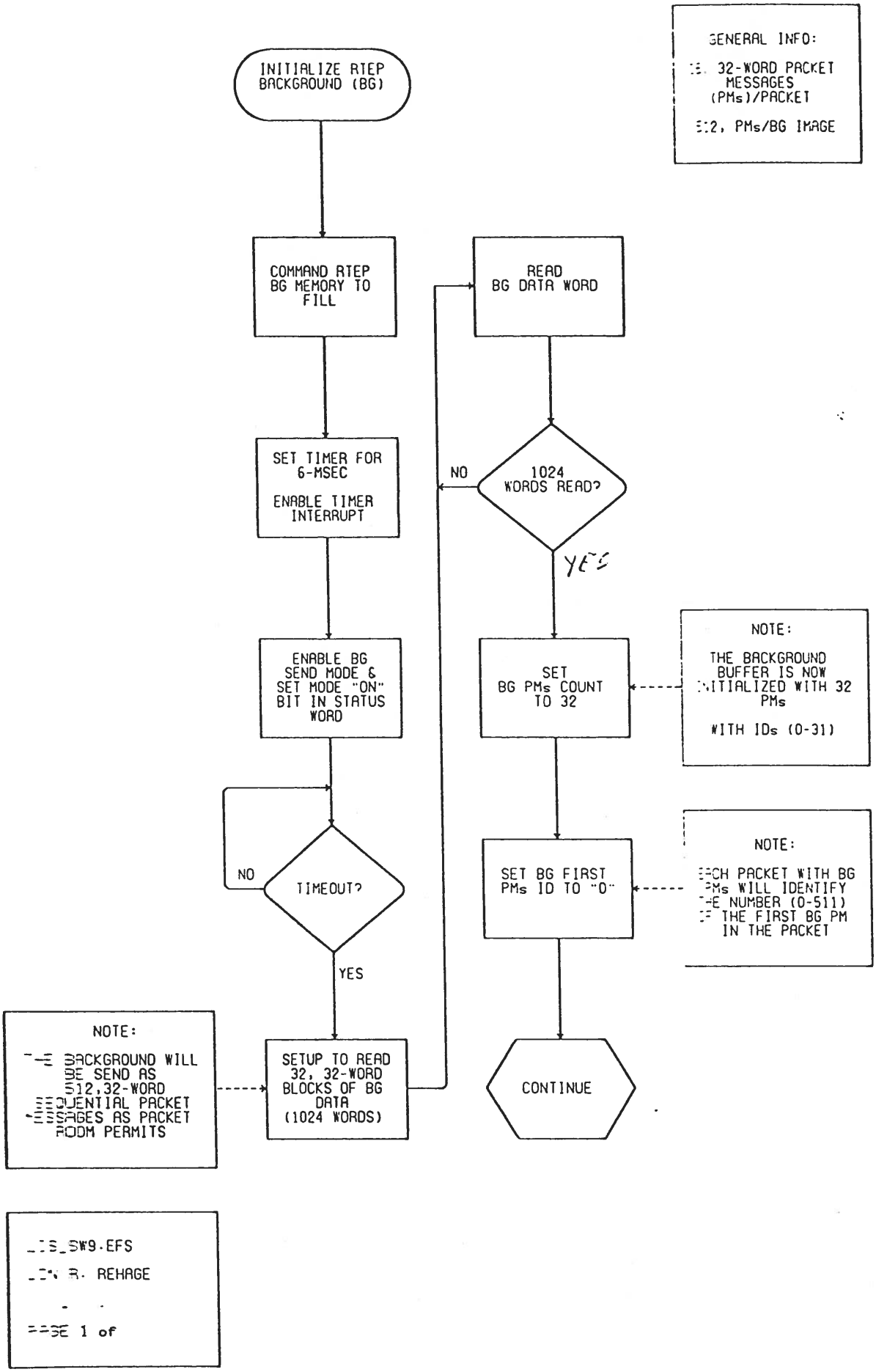
SW6.EFS

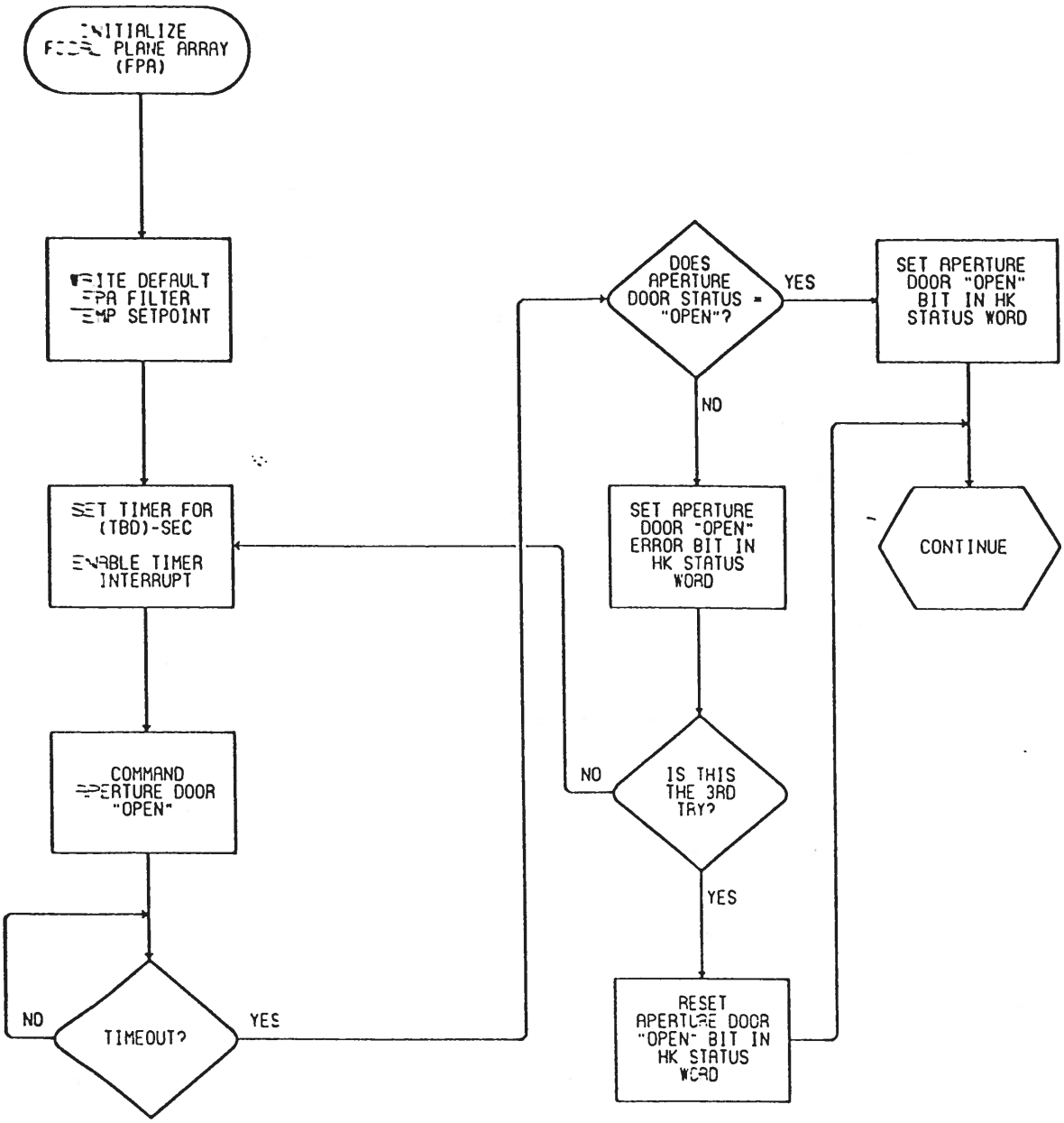
R. REHAGE

PAGE 1 of


INITIALIZE RTEP
LOAD RTEP THRESHOLD MEMORY
FOR DETAILS SEE
(PAGE 2)
GENERAL INFO:
THIS WILL LOAD THE RTEP THRESHOLD MEMORY WITH THE DEFAULT VALUES, WHICH CAN LATER BE UPDATED BY GROUND COMMAND
SET TIMER FOR 2-SEC
ENABLE TIMER INTERRUPT
ENABLE RTEP INTERRUPTS
START RTEP
CONTINUE
NO
TIMEOUT?
YES
RESET RTEP EVENT FIFO MEMORY


GENERAL INFO:
A PACKED THRESHOLD WORD CONTAINS AN ID (0-7) AND TWO 5-BIT THRESHOLD VALUES
SEE THRESHOLD WORD FORMAT
LOAD RTEP THRESHOLD MEMORY
ENABLE THRESHOLD MEMORY WRITE
SETUP TO PROCESS (8) PACKED THRESHOLD WORDS
FETCH PACKED THRESHOLD WORD AND OUTPUT FIRST 5 BITS
SHIFT RIGHT 5 AND OUTPUT 2ND 5 BITS
8 WORDS PROCESSED?
NO
YES
ENABLE THRESHOLD MEMORY READ
SETUP TO PROCESS 8 THRESHOLD WORDS
READ THRESHOLD 5-BIT WORD
READ NEXT THRESHOLD 5-BIT WORD
COMBINE THE TWO 5-BIT WORDS INTO A PACKED THRESHOLD WORD
COMPARE TO THE THRESHOLD WORD WRITTEN
WORDS EQUAL?
NO
YES
SET THRESHOLD UPDATE ERROR BIT IN HK STATUS WORD
PUT ERROR WORD IN HK PACKET
IS THIS THE 3RD TRY?
NO
YES
SET THRESHOLD UPDATE NOT SUCCESSFUL BIT IN HK STATUS WORD
8 WORDS PROCESSED?
NO
YES
SET THRESHOLD UPDATE SUCCESSFUL BIT IN HK STATUS WORD
CONTINUE
REHAGE
1 of

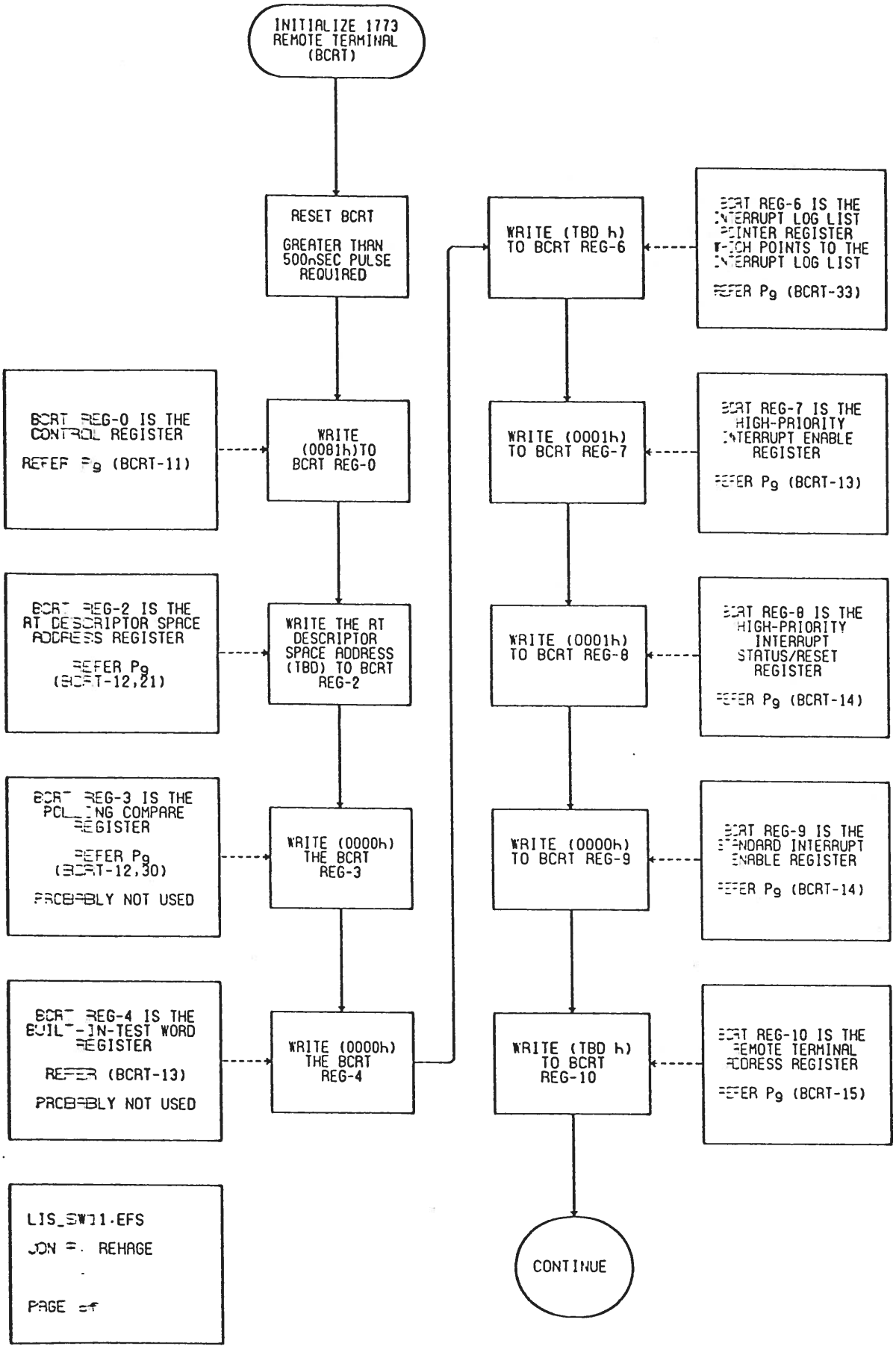

GENERAL INFO:
32-WORD PACKET MESSAGES (PMs)/PACKET
PMs/BG IMAGE
INITIALIZE RTEP BACKGROUND (BG)
COMMAND RTEP BG MEMORY TO FILL
SET TIMER FOR 6-MSEC
ENABLE TIMER INTERRUPT
ENABLE BG SEND MODE & SET MODE "ON" BIT IN STATUS WORD
TIMEOUT?
NO
YES
SETUP TO READ 32, 32-WORD BLOCKS OF BG DATA (1024 WORDS)
NOTE:
BACKGROUND WILL BE SEND AS 512,32-WORD SEQUENTIAL PACKET MESSAGES AS PACKET ROOM PERMITS
READ BG DATA WORD
1024 WORDS READ?
NO
YES
SET BG PMs COUNT TO 32
NOTE:
THE BACKGROUND BUFFER IS NOW INITIALIZED WITH 32 PMs
WITH IDs (0-31)
SET BG FIRST PMs ID TO "0"
NOTE:
EACH PACKET WITH BG PMs WILL IDENTIFY THE NUMBER (0-511) OF THE FIRST BG PM IN THE PACKET
CONTINUE
SW9.EFS
R. REHAGE
1 of



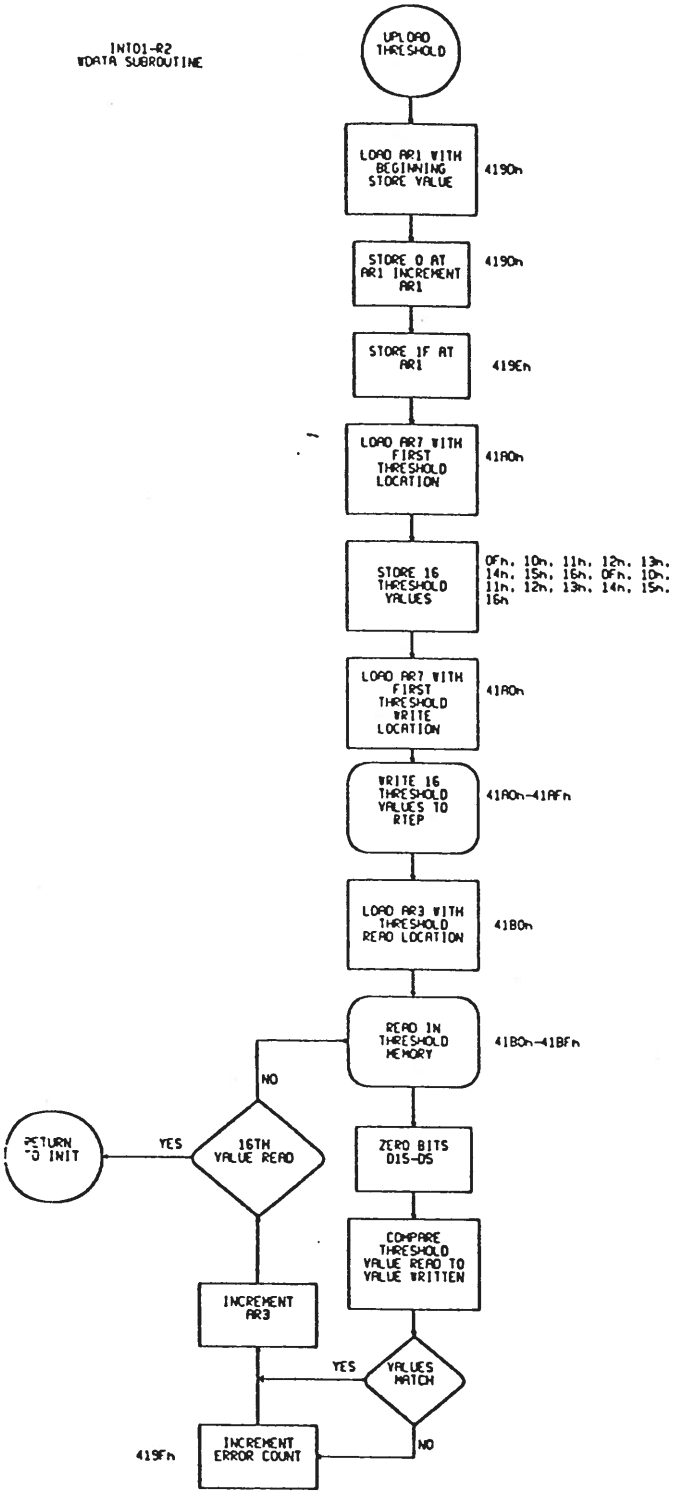
LIE_SW10.EFS

JOHN R. REHAGE

PAGE 1 of


INITIALIZE 1773 REMOTE TERMINAL (BCRT)
RESET BCRT
GREATER THAN 500nSEC PULSE REQUIRED
BCRT REG-0 IS THE CONTROL REGISTER
REFER Pg (BCRT-11)
WRITE (0081h)TO BCRT REG-0
BCRT REG-2 IS THE RT DESCRIPTOR SPACE ADDRESS REGISTER
REFER Pg (BCRT-12,21)
WRITE THE RT DESCRIPTOR SPACE ADDRESS (TBD) TO BCRT REG-2
BCRT REG-3 IS THE POLLING COMPARE REGISTER
REFER Pg (BCRT-12,30)
PROBABLY NOT USED
WRITE (0000h) THE BCRT REG-3
BCRT REG-4 IS THE BUILT-IN-TEST WORD REGISTER
REFER (BCRT-13)
PROBABLY NOT USED
WRITE (0000h) THE BCRT REG-4
WRITE (TBD h) TO BCRT REG-6
BCRT REG-6 IS THE INTERRUPT LOG LIST POINTER REGISTER WHICH POINTS TO THE INTERRUPT LOG LIST
REFER Pg (BCRT-33)
WRITE (0001h) TO BCRT REG-7
BCRT REG-7 IS THE HIGH-PRIORITY INTERRUPT ENABLE REGISTER
REFER Pg (BCRT-13)
WRITE (0001h) TO BCRT REG-8
BCRT REG-8 IS THE HIGH-PRIORITY INTERRUPT STATUS/RESET REGISTER
REFER Pg (BCRT-14)
WRITE (0000h) TO BCRT REG-9
BCRT REG-9 IS THE STANDARD INTERRUPT ENABLE REGISTER
REFER Pg (BCRT-14)
WRITE (TBD h) TO BCRT REG-10
BCRT REG-10 IS THE REMOTE TERMINAL ADDRESS REGISTER
REFER Pg (BCRT-15)
CONTINUE
LIS_SW11.EFS
JON F. REHAGE
PAGE of

INIT01-R2
WDATA SUBROUTINE



4.0 EGSE Software

Electrical Ground Support Equipment (EGSE) software will be developed to provide test tools, and portable checkout equipment for the Lightning Imaging Sensor (LIS).

EGSE software will execute on both the primary and secondary EGSE target systems. These platforms will provide user interface to the EGSE equipment which is provided to command the LIS instrument, and capture and store LIS data.

4.1 Development

EGSE software will be developed in the C++ programming language, with limited use of assembly code if required either for speed of execution, or access to a function not available to the higher order language. Presently, no requirement for the use assembly language has been identified.

Commercial off the shelf (COTS) or reusable software will not be constrained to development in C++ or assembly language. Source code will not be provided for COTS or reusable software, provided the executable code can be executed and controlled from within the main EGSE program.

The primary facility for development of EGSE software will be the LIS data systems development laboratory, located in EB33. This laboratory will contain the target EGSE computer systems, as well as the principle development tools, and the LIS breadboards.

In addition, the office computers of the developers can be utilized for code development and unit testing, provided they are IBM compatibles, operating a version of Turbo C that supports the language and complexity required for the code. These systems will be upwardly compatible with the target environment.

EGSE software will be designed and (with the exception of any COTS or reusable software) coded by the data system engineers designing and developing the EGSE hardware. Organizationally, the developers are located within the Communications Systems Branch, in the Computers and Communication Division, of the Information and Electronic Systems Laboratory.

4.1.1 Computer Software Configuration Items (CSCIs)

Computer Software Configurations Items (CSCIs) are units of management used to facilitate scheduling and configuration control of software under development. Each CSCI will be documented in a Unit Development Folder (UDF), as detailed in Reference 2. Each CSCI will be assigned a separate, equivalent, number in the WBS.

Broadly defined, each EGSE CSCI performs a collection of related tasks. Each CSCI is composed of one or more modules.

Each CSCI will satisfy one or more requirements. The collection of all EGSE CSCIs will satisfy all EGSE requirements. The section(s) of the requirements document (Reference 1) are given below for cross reference.

4.1.1.1 Main Operation Loop

This CSCI will provide the following routines:

initialization
user input
self-test

The requirements governing this CSCI are:

SWRD-4.1
SWRD-4.3.3

4.1.1.2 EGSE Spacecraft Simulator Interface Drivers

This CSCI will provide hardware drivers to operate the following boards interfacing, either directly to the ISA expansion bus, or via an intermediate communication board:

1) Mil-Std-1553 communication card
2) time-mark interface
3) passive analog interface
4) power distributor
5) ethernet interface to SGSE/TRMM simulator
6) #TBD serial interface to calibration facility

Note that items 1 - 4 will never be used in conjunction with item 5.

The requirements governing this CSCI are:

SWRD-4.2.1.1
SWRD-4.2.1.2
SWRD-4.2.1.3
SWRD-4.2.1.4
SWRD-4.2.2/4.2.4
SWRD-4.2.3

4.1.1.3 1773 Communication

This CSCI will provide the ability to implement:

1) the protocol of MIL-STD-1773 (the portions not implemented directly by the hardware) including transmission of mode commands;

2) the handshaking implemented by the TRMM FDS for command and

data transfers over the 1773 fiber optic bus;

3) time code updates provided by the TRMM FDS, formatted as CCSDS packets and transmitted over the 1773 fiber optic bus;

4) formation of commands, and validity checks of LIS data, in compliance with the CCSDS packet telemetry standards;

5) transmission of all valid LIS instrument commands, as well as non-LIS and invalid commands; and

Note that items 1-3 are requirements levied against EGSE software only when utilizing the MSFC TRMM interface simulator. Items 4 and 5 are required for all hardware configurations.

The requirements governing this CSCI are:

SWRD-4.4.3.1
SWRD-4.5.1
SWRD-4.4.3

4.1.1.4 Discrete Telemetry Protocol

This CSCI will provide the ability to simulate those portons of the FDS software that control hardware interfaces to the LIS instrument. This consists of the passive analog interface to the LIS thermistors, the redundant time-mark interface, and sequencing of the power distribution relays.

The protocol of the discrete telemetry will simulate the activity of the FDS and provide the ability to test the LIS response to various configurations of the power busses and time-marks, as well as the ability to diagnose the temperature at the locations of the thermistors within the LIS instrument. Thermistor data will be converted to temperature data for display.

The requirements governing this CSCI are:

SWRD-4.3.2

4.1.1.5 Data Format, Storage, And Display

This CSCI will be developed to manipulate LIS and EGSE data for communication,storage and display purposes. Compressed LIS data will be decompressed to reconstruct science and engineering data for storage and display. Engineering data will be converted to physical units (temperature, etc.)

Data received from or sent to the GSFC TRMM simulator, or the Calibration facility will be appropriately formatted.

The requirements governing this CSCI are:

SWRD-4.6
SWRD-4.7

4.1.1.6 Calibration/Test Analysis Software

This CSCI provides the ability to exchange data with other ground systems (the TRMM simulator, the calibration facility, and the SGSE). In addition, this CSCI provides the ability to manipulate the data that is exchanged, check for validity of format, and perform various other analysis functions upon the data.

The requirements governing this CSCI are:

SWRD-4.4.3

4.1.1.7 Engineering and Analysis Software

Engineering and analysis software is written primarily as a method of supporting the hardware development. In general, E&A software will be code similar in function to the final EGSE code, but written informally on as-needed basis.

When formal coding begins, E&A software that has been utilized will prove a valuable reference, and will be a reference for a successful coding program.

The requirements governing this CSCI are:

SWRD-4.1.4.7

4.1.3 Development Tools

The principle development tool for EGSE software will be the Borland "Turbo C" and "Turbo C++" software development packages. The Turbo C++ package supports both C and C++ programming. C++ is the object oriented version of the C language, and all code developed with the Turbo C package will be compatible with the Turbo C++ compiler/assemble.

Final executable code will be created using the package Turbo C++ Version 3.0, operating on the target EGSE computer.

The Borland packages provide a text editor for code creation, on-line compiler and linker, and trace and breakpoint tools for code debugging.

4.1.4 Executable Code Creation

4.1.4.1 Required Hardware

The hardware required to create the executable EGSE software consists of the following:

1) target EGSE computer system
2) floppy disk(s) containing EGSE source code

4.1.4.2 Required Software

The following software is required to create the executable EGSE software:

1) Source Code
2) Turbo C++ Version 3.0
3) Project File 'GSE.PRJ'
4) MS-DOS Version 5.0
5) The utility program 'CPYGSE.BAT'

The project file is a Turbo C++ convention that links together different files of code, for creation of a single executable file. The file gse.prj is created in Turbo C++, (see Reference #TBD for the procedure) and lists all '*.c' modules required for the EGSE software.

The utility program 'CPYGSE.BAT' will copy the EGSE source files from the floppy disk to the directories required by Turbo C++ to compile and link the EGSE code.

4.1.4.3 Procedure

Executable EGSE Code is created by implementing the following steps. Please note that the MS-DOS operations system is not case sensitive. All commands are terminated by pressing the 'Enter' key.

1) Power on the target computer hardware.

2) If the system does not power up in the DOS directory

```
E:\LIS\EGSESW
```

execute the commands:

```
e:
cd \
cd \lis\egsesw
```

3) Place the 5.25" floppy disk containing source code in drive A and close the drive door. Copy the files from drive a to the appropriate subdirectories by executing the command:

```
cpgse
```

4) Remove the disk from drive A.

5) Repeat steps 3 and 4 until all files containing source code are copied from floppy disk to the target system.

6) Enter the Turbo C compiler/linker environment by executing the command:

tc

7) Load the EGSE project file by executing the following steps:

<ALT>P

Backspace over the text "*.prj" and type "GSE.prj' followed by <Enter>.

<ALT>C
B

8) The Turbo C compiler/linker provides various options for executable code creation. The following options should be selected:

80486 target:	See Table 4-#TBD
80386 target:	See Table 4-#TBD2

9) If step 7 produces any error messages, creation of the executable file has failed. If the errors are listed as code errors, refer to the code debugging section. If the error represents errors in the Turbo C++ environment, refer to the users manual (Reference ##).

10) When the code has been successfully created, as indicated by the status messages returned by the Turbo C package, press the following sequence of keys to exit:

<ESC>
<ALT>X

11) The executable EGSE software now exists in the directory

E:\LIS\EGSESW

as the file "GSE.EXE". The software may be executed by entering the following command from the DOS prompt:

gse

4.1.5 Module Formation And Structure

A specific hierarchy is required for the formation of and integration of the EGSE software modules. This hierarchy is shown if figure 4-#TBD.

The largest unit of EGSE structure is defined as the 'program.' Each target system will have a program that has been compiled especially for that system. The EGSE program is composed of all EGSE modules, linked together by the 'project file.' The project file is a Turbo C designation for a file used to show the structure of a program. All modules required to compile and link the program are listed in the project file.

A 'module' of EGSE code is defined as all code resident within one MS-DOS file. Modules will be composed of one or more functions, and a collection of definitions of constants and global variable declarations. Modules in the Turbo C structure fall into two categories "*.C" and "*.H" where the '*' indicates any valid MS-DOS filename. "*.C" modules are composed of functions, and global variable allocations. "*.H" modules are composed of symbol definitions. Although other filenames can be used for modules, and incorporated into the overall design, only "*.C" and "*.H" delimiters will be utilized for EGSE implementation.

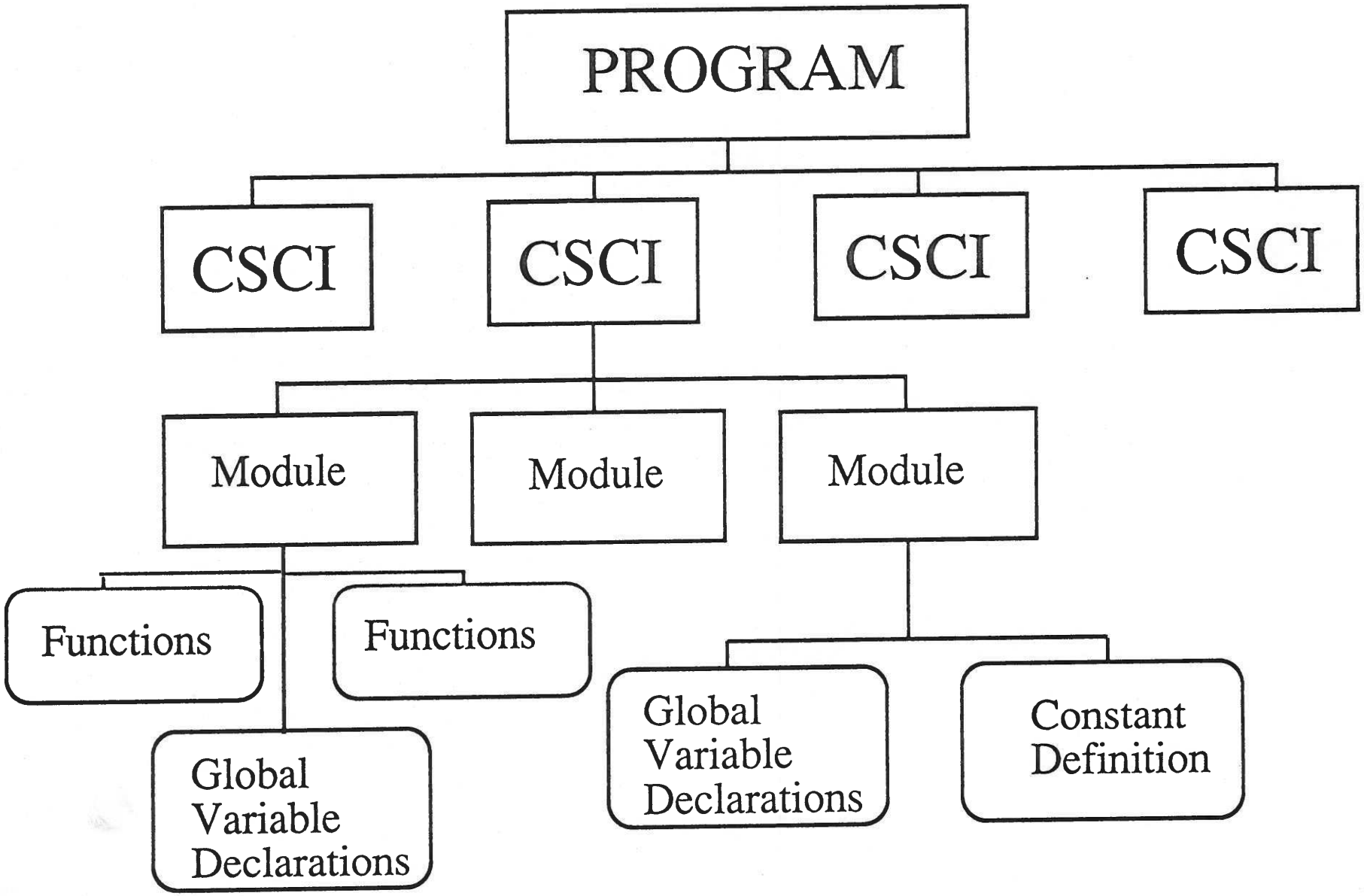
A 'function' of EGSE code is defined as any validly formed C++ function, as defined by the standards governing the C++ programming language, and the implementations supported by the Turbo C++ development package.

Each module of code will be labeled with the following identification header:

```
/*

Module Name:          'name of module'

Module Path:          'destination for compilation'

CSCI:                 'CSCI(s) addressed by module'

Version Number:       'version of module, referenced to UDF'

Revision Date:        'date of current revision, as recorded in UDF'

Programmer(s):        'name of principle programmer,
                       name(s) of all programmers involved'

Module Contents:      'description of module contents'

*/
```

The '/*' and '*/' symbols delineate comments.

EGSE Software Structure



4.2 Hardware Capabilities

Two target systems are a part of the electrical ground support equipment: the portable, deliverable EGSE used for qualification, post-ship, and post-integration test; and the development test-set used within the Communications Systems Branch.

Target System #1:

Target #1, the deliverable EGSE, is an Intel 80486-33MHz computer with an ISA bus structure, an IBM-AT compatible BIOS, and the MS-DOS 5.0 operating system. The interface characteristics of this hardware are given in Reference #TBD.

The Intel 80486 is a 32-bit microprocessor which includes an internal floating-point mathematics co-processor, memory caching, and an extended instruction set that is downwardly compatible with the 80386 instruction set.

This system is provided with 8 Megabytes of RAM memory. Total hard disk storage capacity is TBD(230) Megabytes.

Target System #2:

Target #1, the EGSE test-set, is an Intel 80386-25MHz computer with an external 80387 floating-point mathematics co-processor on the system board, an ISA bus structure, an IBM-AT compatible BIOS, and the MS-DOS 5.0 operating system. The interface characteristics of this hardware are given in Reference #TBD.

The Intel 80386 is a 32-bit microprocessor. The 80386 instruction set that is upwardly compatible with the 80486 instruction set.

This system is provided with 4 Megabytes of RAM memory. Total hard disk storage capacity is (110) Megabytes.

4.2.1 Microprocessor Architecture

Because of the use of a both an operating system and a higher-order language, the architecture of the microprocessor does not directly influence code creation. However, the compiler and linker will be set to optimize for the processor and resources used by the target system.

4.2.2 System Control

Control of the hardware is accomplished in four ways: system reset, power interruption, DMA, and interrupts.

The EGSE system can be reset two ways: software reset or hardware reset. The software reset is generated by MS-DOS by pressing the <CTRL><ALT><DELETE> keystrokes. This pattern generates an interrupt that MS-DOS used to perform a soft reset (See Reference #TBD). Although it is possible to disable, or alter this action, EGSE will not do so. A hardware reset is generated by pressing the reset button, located on the from of the EGSE computer case. This is a hard reset, and performs the same operations as cycling power.

Power interruption of very short duration may interrupt normal program execution, without performing a power-on reset of the microprocessor and operating system. Power interruptions should be of at least TBD seconds duration in order to properly reset the system. Interruption of power can be used as a last resort to stop program execution. All data stored to disk will be saved, whereas all data not stored to disk will be lost.

DMA (direct memory access) is a method of reading/writing data directly to memory, without microprocessor intervention. During a peripheral DMA transfer, the microprocessor is placed in a hold state, and is unable to access the memory space. However, the process can continue to perform actions on internal memory space. Interfaces to the ISA bus can perform DMA using the systems DMA controller. The commercial 1553 card will utilize DMA to transfer data in and out of memory.

4.2.3 Input/Output Capability

Input/output is provided via two means: the human operator of the EGSE, or the interfaces EGSE hardware.

Human input is provided via the keyboard. This takes the form of selecting options from the menus provided by the software, or typing in the value of data, options, or the names of files containing data and/or options.

Output for the human operator is accomplished using the video display capability and the printer.

Hardware-oriented input obtains data from the interface hardware to the EGSE, or reads the data from hard disk or floppy disk storage. All data is accessed by the EGSE software polling the data from the device. The only EGSE interrupt, the time update interrupt (section #TBD) does not produce data.

Hardware-oriented output consists primarily of storing data to hard or floppy disk. It is also possible to back-up disk data to the tape drive.

4.2.4 Operating Characteristics

The EGSE systems conform to the characteristics of IBM-AT compatible computers. The backplane structure is the Industry Standard Architecture (ISA), a 16-bit data bus.

4.2.5 Interrupts

The ISA bus structure provides 13 separate interrupts for peripherals to use when in need of attention. The EGSE hardware will be configured to produce only one interrupt in need of service from the EGSE software. (Any other interrupts will either be handled by the operating system, or by the COTS drivers supplied with the commercial hardware generating the interrupt.) The configuration of the EGSE hardware will be such that the interrupt produced will be unique to the generating hardware, i.e. only the time-update hardware can produce the interrupt on that bus channel.

4.3 Software Functional Characteristics

4.3.1 Program Control

Program control is accomplished by four methods: user inputs, timed loops, interrupts, and polling for data. User inputs will be guided by the menus displayed on the screen. An interrupt will be generated by the discrete telemetry board to initiate a broadcast of time data. Timed loops will perform operations in predefined sequences.

The availability of data will be used primarily when interfacing to the GSCF ground systems. In these configurations, the GSFC ground equipment is responsible for receiving data from the LIS and providing that data, asynchronously, to the EGSE. For this operation, the EGSE will poll for the availability of data, and then process the data when it is available.

At any step where the EGSE software accepts input, the keystroke <CTRL>C will exit the program, and return to the operating system.

4.3.2 Program Input/Output Capability

The EGSE software will operate from within an MS-DOS environment. The MS-DOS operating system will provide the majority of the low-level functions required for interfacing to the ADP computer hardware. For example, routines to access the BIOS for keyboard inputs, test output, interrupts, etc., are handled by the operating system, and can be accessed using the high-order functions of Turbo C++. (Reference #TBD)

4.3.3 Operational Modes

There are four operational modes for the EGSE software. Three of these modes are driven by the physical configuration of the EGSE.

One mode of operation uses the MSFC TRMM simulator. In this configuration, the EGSE software is responsible for implementing the FDS protocols, and the other interface controls.

The GSFC simulator mode and the TRMM SGSE mode both replace the hardware of the MSFC TRMM interface simulator. These ground systems are external to the EGSE, and provide the LIS data to the EGSE, and the EGSE commands to the LIS.

The analysis mode runs independently by processing previously collected LIS data, or in a limited form, concurrently with either of the previous two configurations.

The diagnostic mode is a mode for running either low-level checks and test of the EGSE itself, or performing very simplistic functions on the LIS instrument.

4.3.4 Database Definition

Definitions of all constants, for example, the LIS 1773 remote terminal (RT) address, will be coded as C constants in upper case. For example, the LIS RT address is 20. Rather than using 20 in the code, which would not make it obvious that the number was intended to represent the LIS RT address, EGSE code will use the expression.

LIS_RT_ADDRESS

Definition of this constant consists of the following statement, within a '*.h' file:

```
#define      LIS_RT_ADDRESS      20
```

In addition, other values, such as indications of 'end of file' or 'error' can be used as alphanumeric constants, and defined in the appropriately included '*.h' file.

The complete definition is given is section 4.6.TBD.

4.3.5 Memory Allocation And Variable Declaration

The Turbo C++ package automatically assigns memory locations for variable and data. In addition, memory can be reserved by using commands, allowing faster data handling. Where speed of execution is important, the EGSE software will manipulate memory allocation. As a default, the compiler will be allowed to handle this function automatically.

One memory allocation is presently planned. Memory space will be assigned for the 1553 card to use for performing DMA transfers.

There are two type of variable declarations: local and global. The value of one local variable can be passed to another local variable in another function. Global variable are common to all functions.

A local declaration is only valid within a function. Declaration of the same variable in two or more different functions is legal, and each variable can hold different information and/or different types of information. For example, the variable 'i' can be declared an integer in one function, and a character in another.

A global variable is declared outside of all functions. The type and value of the variable is common to all functions within the program. It is not legal to declare the same variable as both local and global. Global variables that are used within only one module will be declared within the '*.c' module using that variable. Global variables that are used by more than one module will be declared within '*.h' modules that are included by the calling modules. The global declaration will be included by the following statement:

extern <variable type> <variable name> /* location */

where 'location' is the name of the module where the global variable is declared.

The complete list of global variable is given in section 4.6.TBD.

4.3.6 Storage Allocation

The full available capacity of the hard disk and the tape drive can be used by EGSE software for storage of data collected. The following file names and descriptions will be used:

'LIS1773.DAT'	will contain unprocessed data from the LIS 1773 interface;
'PASSIVE.DAT'	will contain unprocessed data from the LIS thermistors and the status of the power distribution relays;
'EVENTS.DAT'	will contain processed LIS lightning and background data, where <RUN> is an identifier to identify which experimental run produced the data;

Data generated by differing runs will be labeled as 'LI-<NUMBER>.DAT', 'PA-<NUMBER>.DAT', and 'EV-<NUMBER>.DAT' where <NUMBER> is a unique number/label that will be cross referenced for cataloging.

4.3.7 Restrictions And Constraints

In order to provide simulation of the TRMM FDS while operation in the MSFC TRMM interface simulator configuration, the EGSE software will operate subject to the FDS timing constraints listed in the FDS Appendix (Reference #TBD), sections 2.2.2, 2.3.2, and 2.4.

In addition to constructing all valid LIS commands, the EGSE software will provision, while operating in the MSFC TRMM interface simulator configuration for generating non-LIS commands and invalid commands, to test the LIS response.

4.4 Decomposition Description

The EGSE software can be decomposed into eight major sections (see figure 3-#TBD): Main Loop, User Interface, Test Routines, Interface Simulator Drivers, Bus Transfer And Formats, Command And LIS Data Formats, Discrete Telemetry Formats, and Display/Storage Routines.

4.4.1 Main Loop

The main loop of the EGSE software is used to provide program control. The main loop itself operates in on of two modes: timer driven, or user input driven.

In the user input driven mode, individual actions will be selected by the user, with the software providing the correct timing and sequencing to implement the activity. For example, the user selected command "Get LIS science data" would implement the multi-step process that transfers 16 1773 subaddresses of data from the LIS instrument.

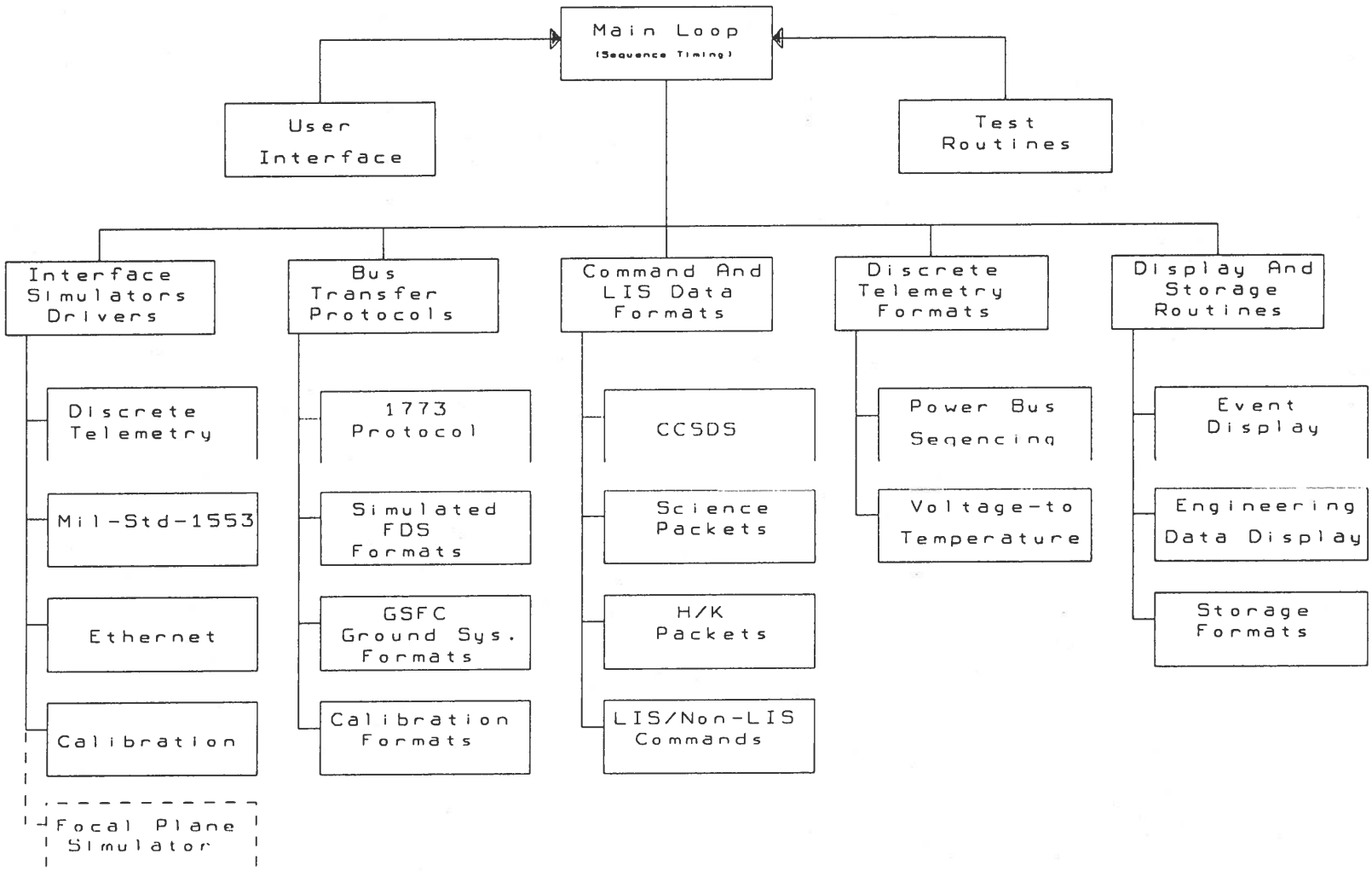
In the timer driven mode, the EGSE software will implement LIS commands and other interface activities in a predetermined sequence, based on elapsed time.

4.4.2 User Interface

The user interface is not tied to any specific CSCI or requirement. However, the use of an easy, effective interface is necessary to accomplish the overall goal of the EGSE: implementation of an automate test tool for LIS development.

Much of the user interface will be distributed across the various modules developed. This will include menu displays, input requests, and outputs. To the extent possible, all generic input routines will be collected into a module titled "INPUT.C" and all generic output modules will be collected into a module titled "OUTPUT.C".

EGSE Decomposition Description



4.4.3 Test Routines

Test routines are predetermined sequences of routines that can be executed automatically upon either a specific user input, or an elapsed timer. The routines required to support hardware development of both the EGSE or the LIS data system (i.e. the Engineering and Analysis CSCI) will be collected into two modules "DEVTEST.C" for the 1773 and microprocessor development routines, and "FPSIMTST.C" for the RTEP development routines.

Other routines of specific sequences will be developed as defined.

4.4.4 Interface Simulator Drivers

To provide adequate simulation for testing the LIS, the EGSE consists of a variety of simulators, in several configurations, that are controlled by the EGSE software.

Discrete Telemetry Simulator: This interface consists of two cards, with separate board addresses, plugged into the ISA expansion bus of the EGSE computer. As presently planned, these cards will only be used with the MSFC TRMM simulator configuration.

This simulator provides the readout of the 8 (redundant) LIS thermistors, the (redundant) time mark, and the command and status interface to the power distribution relays.

In addition, this board provides the interrupt that the EGSE will use as the signal to send a time packet update.

Mil-Std-1553: This board is a commercial procurement from SCI. When combined with the 1553/1773 converter, this will implement the 1773 interface. The board itself is plugged directly into the ISA expansion bus. The board will perform DMA transfers within the computer, for data transmitted and received of the 1773 bus.

It is planned to use COTS software, purchased with this board, to implement both the protocol of 1773 and the DMA transfers. This procurement is scheduled to be completed this fiscal year.

Until this card is available, the EGSE will utilize a 1553 card designed in-house, with in-house E&A software to support the protocol.

Ethernet: The interface to the GSFC TRMM Interface Simulator and the TRMM SGSE consists of a single thin-wire (10BaseT) ethernet. A commercial card, with COTS software drivers will be purchased. This procurement will be initiated in the near future in order to assure compatibility, and define any additional requirements required to interface the COTS software to the developed EGSE software.

Calibration Interface: The calibration interface has not yet been defined.

Focal Plane Simulator: The focal plane simulator is used to simulate controlled electrical outputs duplicating the LIS detector outputs. This interface consists of two sets of memories for holding 128x128 12-big values. The software loads both sets of memories with values representing CCD pixel values. One set of memories contains purely background values. The second set of memories contains the same background values, with the addition to selected pixels, of values representing lightning. To simulate the output, the EGSE outputs the background-only memories until selected to produce a lightning event. At this time, the output is switched by the software to the second memories that contain background-with-event values. This set is output for one video frame, and then the output is switched back to the background-only set.

4.4.5 Bus Transfer Protocols And Formats

There are 4 different protocols of communication for the EGSE.
The first is 1773 communication. Much/most of this protocol is provided by the hardware of the EGSE 1553 boards. Additional protocol will be implemented in software.

The TRMM flight data system (FDS) defines a specific protocol for the exchange of commands and data with the LIS instrument. Separate protocols govern: transfer of LIS science data from the LIS to the FDS; transfer of LIS housekeeping data from the LIS to the FDS; transfer of commands from the FDS to the LIS; and broadcast by the FDS of coarse time to the LIS.

The GSFC ground computers also define a specific protocol for the exchange of data with the LIS EGSE (See Reference #TBD). This format consists primarily of encapsulating LIS data from the GSFC equipment as CCSDS formatted data units, and passing that data to the LIS EGSE. Passing of LIS instrument commands, and control of the interfaces, is also defined by this protocol.

4.4.6 Command And LIS Data Formats

In addition to he protocol for bus transfers, the FDS and the LIS define additional formats for how that data will be formatted. The FDS mandates CCSDS packets. This structure is applied to both the science and housekeeping data. In addition, the LIS defines a compression format to the science data. The EGSE software will check these packet formats for validity of formation, and reverse the process to construct the original data.

The instrument commands are defined by LIS, within the FDS constraints. The EGSE will produce this structure, in order to properly command the LIS.

4.4.7 Discrete Telemetry Formats

The discrete telemetry format software executes the necessary timing and data sequences to: read the LIS thermistor temperature data as a digital voltage, and convert that data back to temperature; control the power distribution of the 4 power busses to the LIS instrument.

In the MSFC TRMM simulation mode, the EGSE software will have complete control of these functions. With other configurations, this control will be limited, but the status will still be available to the EGSE.

4.4.8 Display/Storage Routines

There are three ways to handle data collected by the EGSE: display the data as engineering data; display the data graphically, or store the data. Engineering display will include both the science data, and the housekeeping and passive data. This display will consist of numerical display, and can be directed either to the video screen or the printer. Graphically display will consist of both the housekeeping and passive telemetry (temperature, voltage, etc.) and the lightning image display. Data can be stored (to disk and/or tape) either in the form received by the EGSE, or in processed form that has the actual data reconstructed.

Display of data will be limited while in active simulation mode, but will also support more extensive display of data that has previously been stored to tape.

4.5 Interface Description

4.5.1 Hardware Interfaces

The particular interfaces to the EGSE computer are governed by the mode of operation in which the EGSE is functioning. These interfaces, for each configuration, are show in figures #TBD, #TBD, and #TBD.

4.5.2 Software Interfaces

The developed EGSE software will interface to two types of software: the software running in other ground computer systems; and the COTS software running on the EGSE computer.

4.5.2.1 Software Running Other Ground Systems

The EGSE software will not interface directly to the software running on other ground systems. Instead, the EGSE and other systems will exchange data packets, transmitted by either ethernet to the GSFC systems or by TBD interface to the calibration facility.

Data provided to the calibration facility will not influence the operation of the calibration facility. The EGSE has no command ability of the calibration facility, nor does the calibration facility have command capability over the EGSE.

The TRMM ground computers have no command control over the EGSE systems. The EGSE will poll the input buffer from the ethernet interface. When available, data received by the EGSE from the TRMM ground computers will be

processed and/or stored.

Data transmitted by the EGSE, to the TRMM ground computers will control the execution of some of the hardware within the other system. It is, however, the responsibility of the TRMM computer to check the data for validity, and screen out any commands that would have a deleterious effect on itself. Data that the TRMM ground computer forms into a command for the LIS instrument will be under the same formation restrictions as used in the MSFC simulator configuration. The SGSE will screen the commands for those that would have a deleterious effect on non-LIS instruments. These commands will be rejected.

4.5.2.2 COTS Software

The EGSE software will interface with COTS software. The EGSE will utilize commercially available software drivers for commercially purchased hardware interfaces or peripherals. This software falls into three four categories:

1) Drivers for commercially available portions of the EGSE spacecraft simulator:

the 1553 cards

2) Drivers to input/output or mass storage hardware associated with the EGSE:

the Exabyte tape drive

3) Data interfaces between the EGSE ADP equipment and other ground systems

the ethernet card
TBD calibration facility interface

4) The operating system, and other base level software, upon which the developed EGSE software operates.

For items 1-3, the developed EGSE program will interface to this COTS software by spawning a process to run the existing COTS routine, and then return to the calling program. In item 4, the interface consists of the operating characteristics and restraints imposed upon the creating of the developed software (Reference #TBD).

4.6 Detailed Design

Database Definition:

NOTE: '0x' denotes a hexadecimal number.

General Defines:

```
NULLSTRING          -13
TRUE                  1
NIL                    0
ESC                 0x1B
FIFO_EMPTY            0
FIFO_NOT_EMPTY        1
BAD_STATUS          -1
WRITE                 0
READ                   1
UNDEFINED            -1
RELAY_LATCH_TIME      0          /*          TBD              */
DISPLAY_TIME          1000       /*  Delay in milliseconds */
```

LIS Command Values:

```
LIS_ADDRESS                          20
BACKGROUND_SEND_MODE_ON              0x1111
BACKGROUND_SEND_MODE_OFF             0x2222
THRESHOLD_ADJUSTMENT                 0x3333
FILTER_TEMPERATURE_SET_POINT         0x4444
SELF_TEST                            0x5555
APERTURE_DOOR_OPEN                   0x6666
APERTURE_DOOR_CLOSED                 0x7777
SAFE_MODE                            0x8888
WATCHDOG_ENABLE                      0x9999
WATCHDOG_DISABLE                     0xAAAA
HEATER_A_ON                          0xBBBB
HEATER_B_ON                          0xCCCC
ENABLE_FILTER_MONITOR                0xDDDD
DISABLE_FILTER_MONITOR               0xEEEE
```

Data Acquisition Board Defines:

Defines for counter 1 addresses:

```
cntrl1_add          0x1703  /* byte */
IO_1_A              0x1702  /* lines 1-8, byte */
IO_1_B              0x1701  /* lines 17-24, byte */
IO_1_C              0x1700  /* lines 33-36, nibble * /
```

Defines for counter 2 addresses:

```
cntrl2_add              0x1707  /* byte */
IO_2_A                  0x1706  /* lines 9-16, byte */
IO_2_B                  0x1705  /* lines 25-32, byte */
IO_2_C                  0x1704  /* lines 37-40, nibble */
```

Status register address defines:

```
find_board              0x1310
term_panel_status       0x1314
int_status              0x1318
```

I/O setup values for counters:

```
io_setup_A              0x23
io_setup_B              0x2B
io_setup_C              0x06
```

Special I/O controls for open drain/normal i/o:

```
special_io_setup_A      0x24
special_io_setup_B      0x2c
special_io_setup_C      0x07
normal_output           0
open_drain_output        0xff
```

Data path polarity defines:

```
data_path_A_add         0x22
data_path_B_add         0x2A
data_path_C_add         0x5
non_inverting           0x00
inverting               0xff
```

Port mode specification register defines:

```
port_mode_spec_A_add    0x20
port_mode_spec_B_add    0x28
bidirectional           0xC0
```

Z8536 register defines:

```
master_interrupt_contrl_reg    0x0
```

Z8536 control codes

```
reset                     1
clear                     0
master_config_cntrl_reg 0x01
```

Port enable defines:

```
enable_all_chips        0x3
enable_all_ports        0x94
enable_port_A           0x4
enable_port_B           0x80
enable_port_C           0x10
```

General defines for boards:

```
all_output      0x0
all_input       0xFF
```

Board Select defines:

```
FPSIM_BOARD                         0x04    /* FPSIM Board */
```

defines for counter resets:

```
timer_one_stat_reg                  10
timer_one_stat_val                  0x6
timer_two_stat_reg                  11
timer_two_stat_val                  0x6
timer_one_mode_spec_reg             28
timer_two_mode_spec_reg             29
timer_one_time_constant_msb_reg     0x22
timer_one_time_constant_lsb_reg     0x23
timer_one_mode_value                0x26
timer_two_mode_value                0xe4
timer_msb_value                     0xf
timer_lsb_value                     0xff
```

Defines For The Focal Plane Simulator:

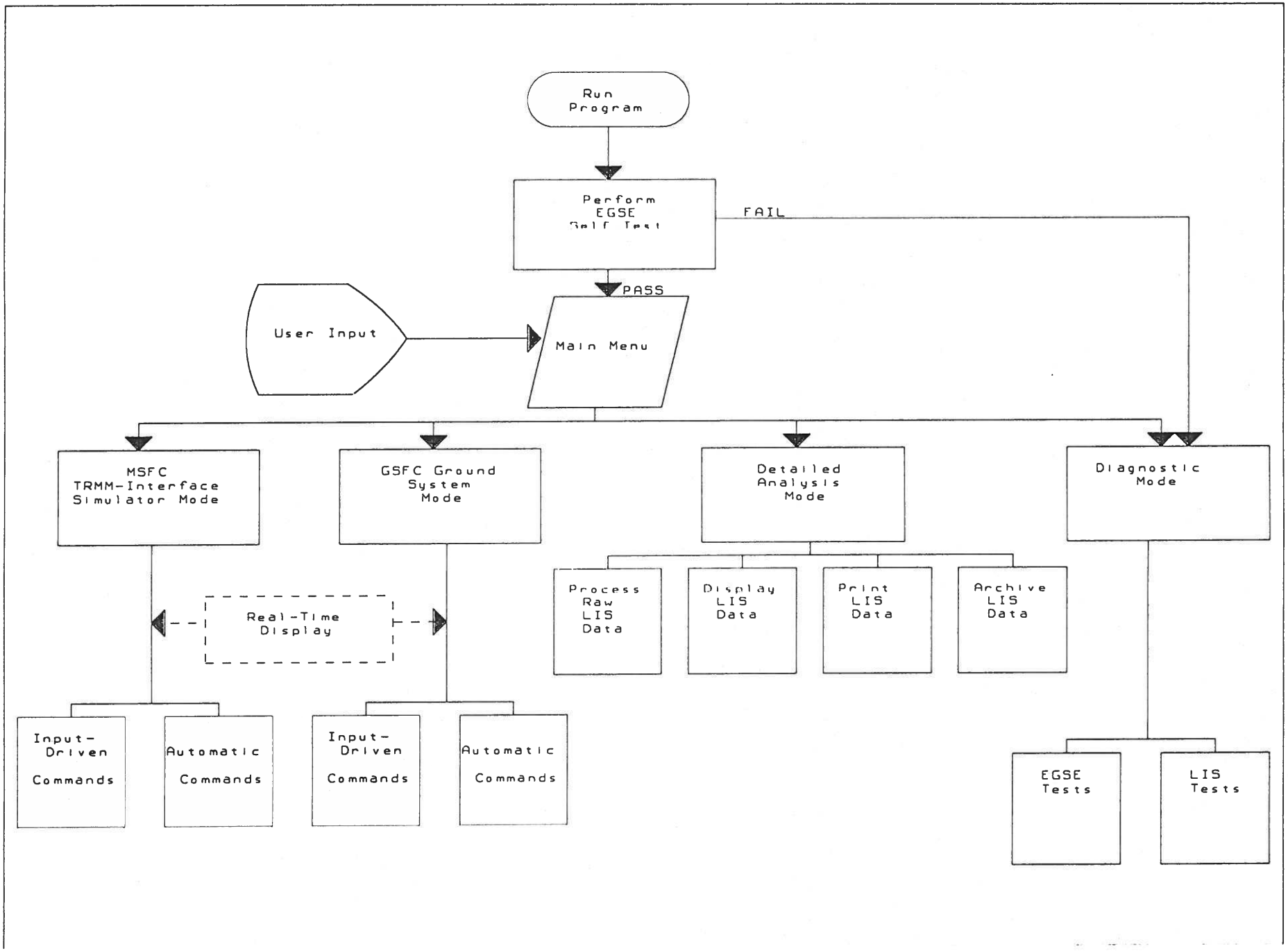
Address Defines

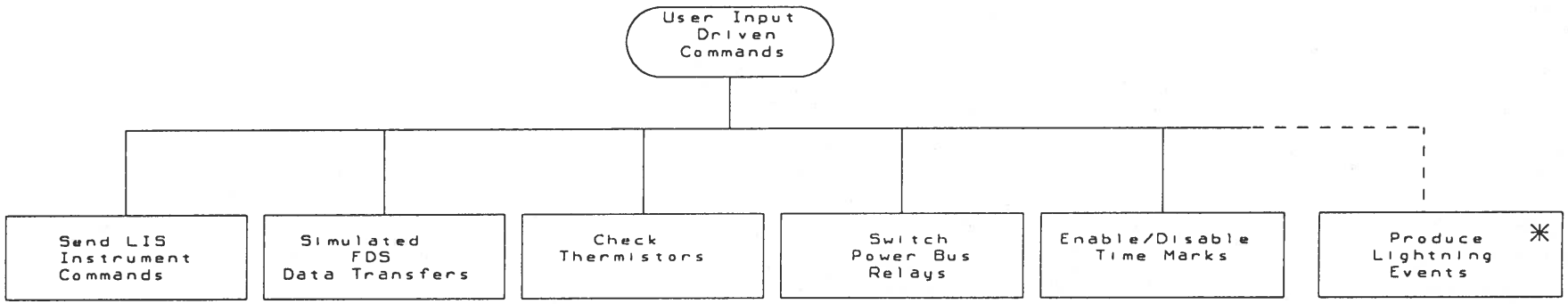
```
FPS_MEMORY_ADD              0x0
FPS_MEMORY_EN_ADD           0x1
FPS_DAC_ADD                 0x5
FPS_CONTROL_REG_ADD         0x6
FPS_COUNTER_ADD             0x7
MAX_LINE_NUM                128
MAX_PIXEL_NUM               128
```

Control Register Defines

```
EN_INT_CLK          0x01    /* C1 Active High */
EN_EXT_CLK          0x02    /* C2 Active High */
EN_COUNTER_PRESET   0x04    /* C3 Active High */
EN_COUNTER_CLR      0x08    /* C4 Active Low  */
EN_COUNTER_RCO      0x10    /* C5 Active High */
EN_FRAME_2          0x20    /* C6 Active High */
```

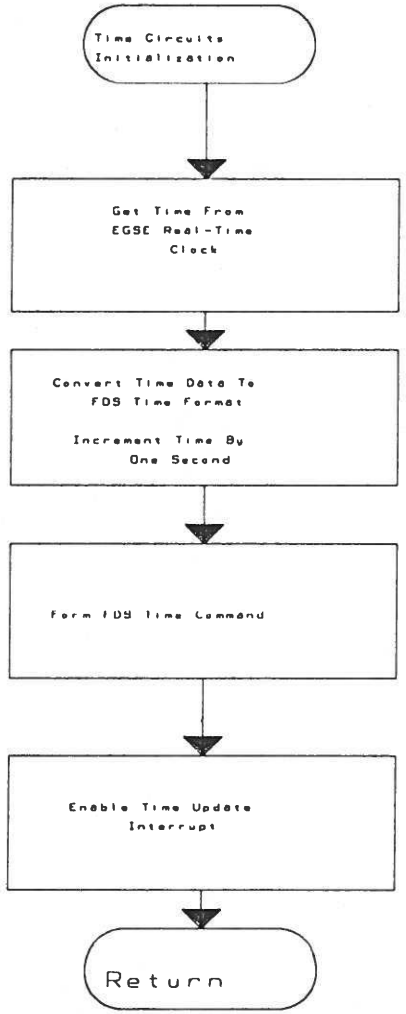
```
EN_FRAME_1            0x40    /* C7 Active High */
EN_LINE_FRAME_SYNC    0x80    /* C8 Active High */
RST_INT_CLK           0xFE    /* C1 Active High */
RST_EXT_CLK           0xFD    /* C2 Active High */
RST_COUNTER_PRESET    0xFB    /* C3 Active High */
RST_COUNTER_CLR       0xF7    /* C4 Active Low  */
RST_COUNTER_RCO       0xEF    /* C5 Active High */
RST_FRAME_2           0xDF    /* C6 Active High */
RST_FRAME_1           0xBF    /* C7 Active High */
RST_LINE_FRAME_SYNC   0x7F    /* C8 Active High */

FRAME_1                  0
FRAME_2                  1
MAX_PIXEL_VAL         0xFFF
MAX_NUM_PIXEL         (128 * 128)
```


Run Program
Perform EGSE Self Test
FAIL
PASS
User Input
Main Menu
MSFC TRMM-Interface Simulator Mode
GSFC Ground System Mode
Detailed Analysis Mode
Diagnostic Mode
Real-Time Display
Process Raw LIS Data
Display LIS Data
Print LIS Data
Archive LIS Data
Input-Driven Commands
Automatic Commands
Input-Driven Commands
Automatic Commands
EGSE Tests
LIS Tests

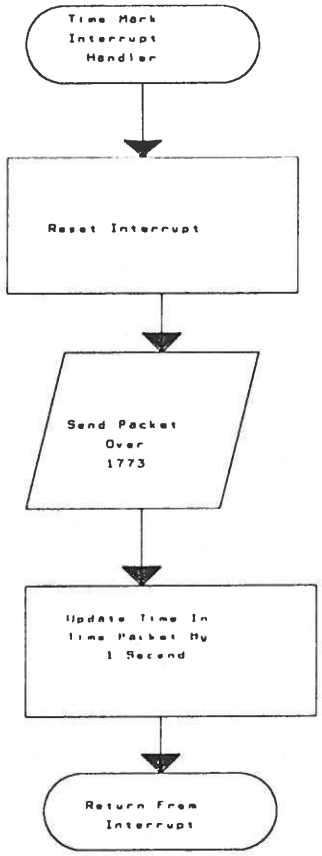


* When Using FPYIM In MSFC Mode Only

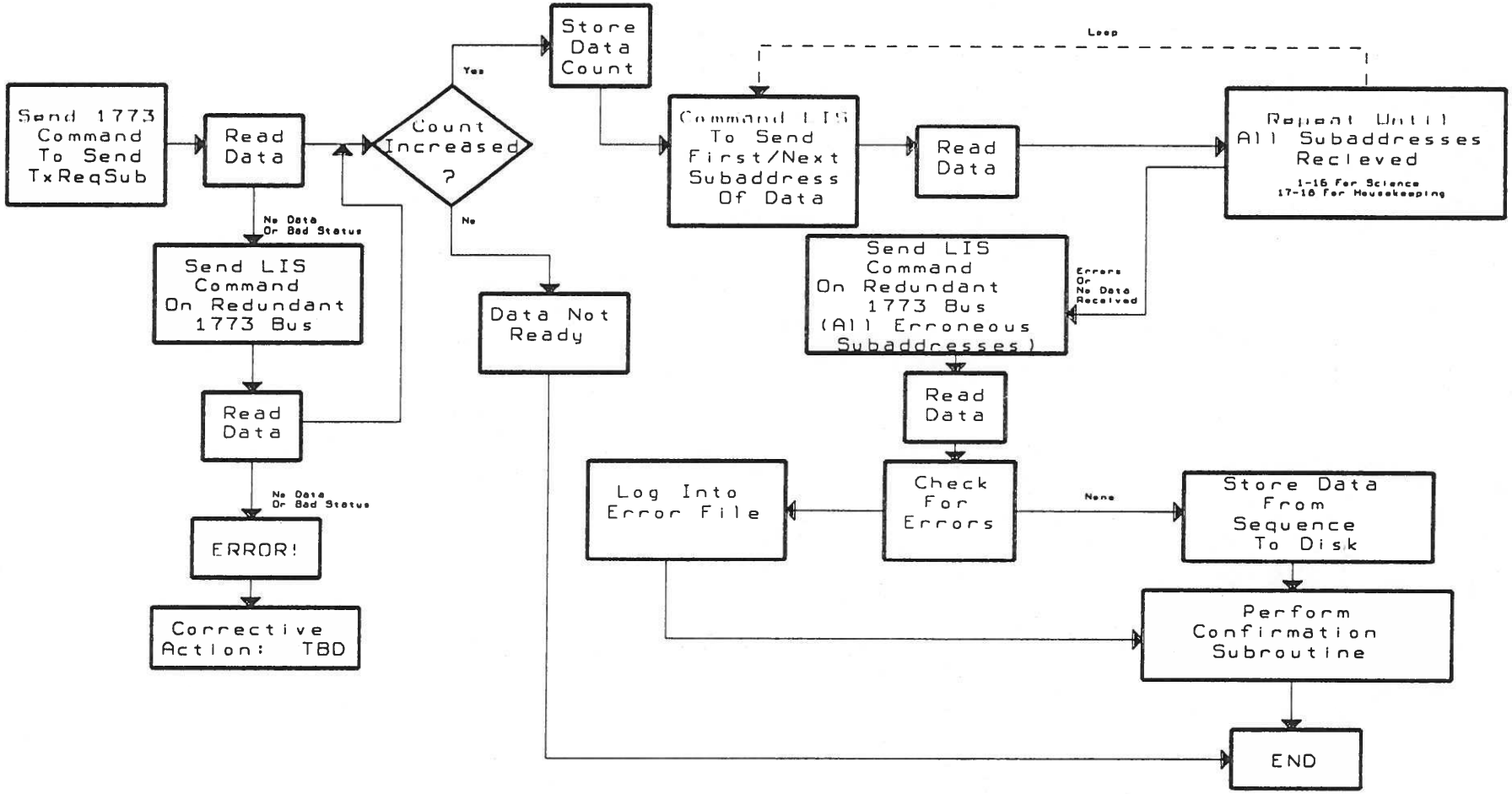
Time Circuit Initialization



Time Update Interrupt

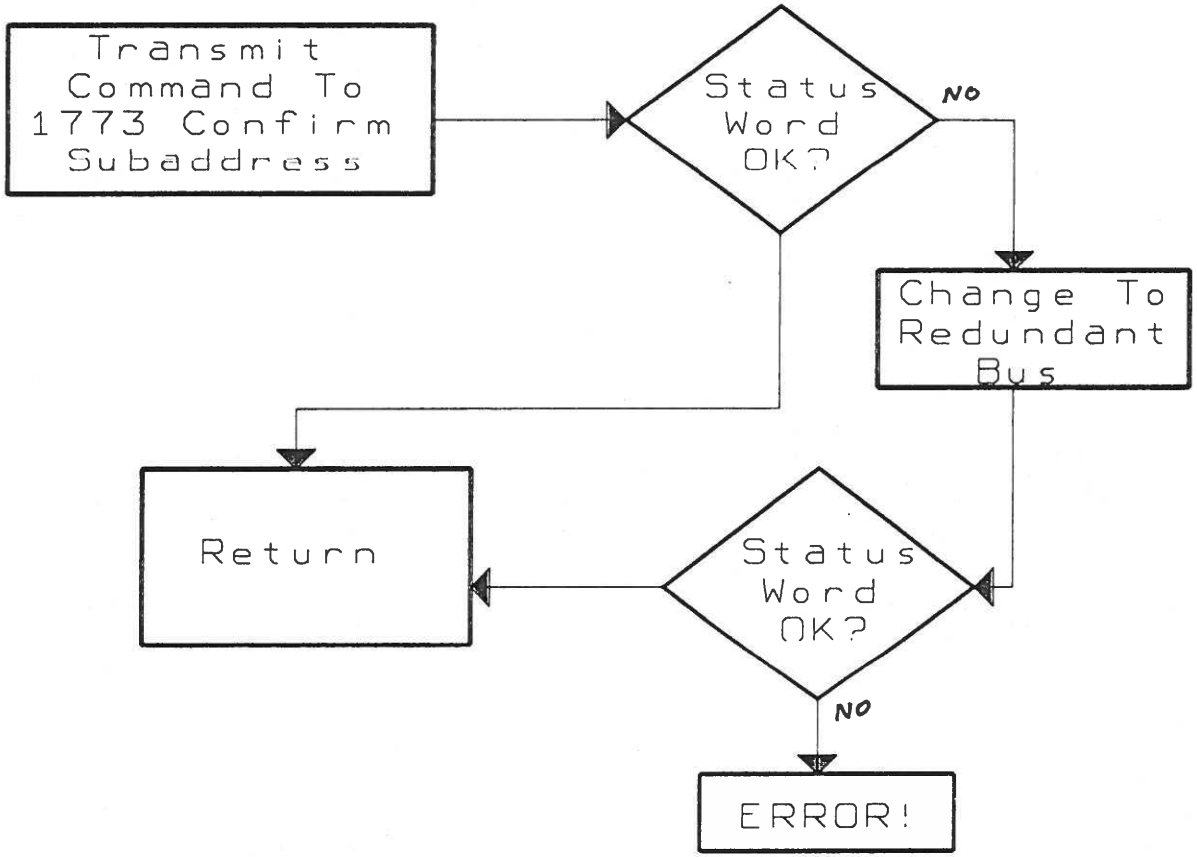


FDS Data Transfer From LIS



TxReqSub = 28 for Housekeeping
30 for Science Data

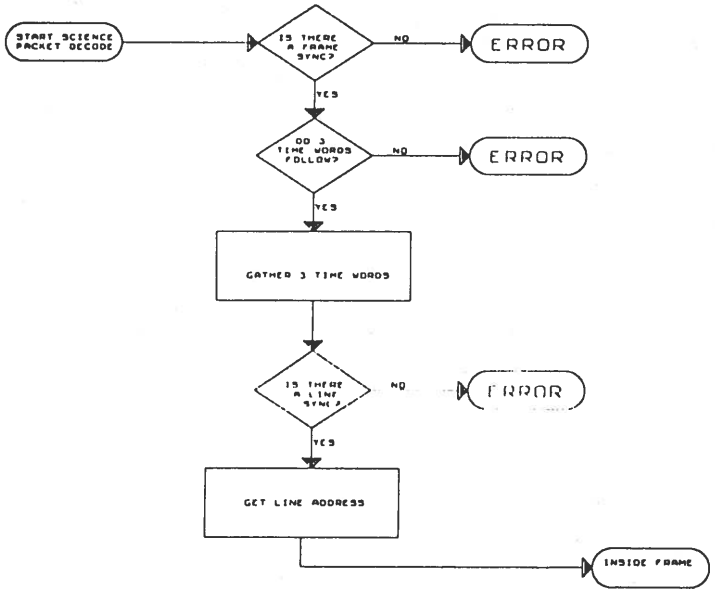
FDS Confirmation Protocol



Confirm Subaddress = 27 for Housekeeping
29 for Science Data

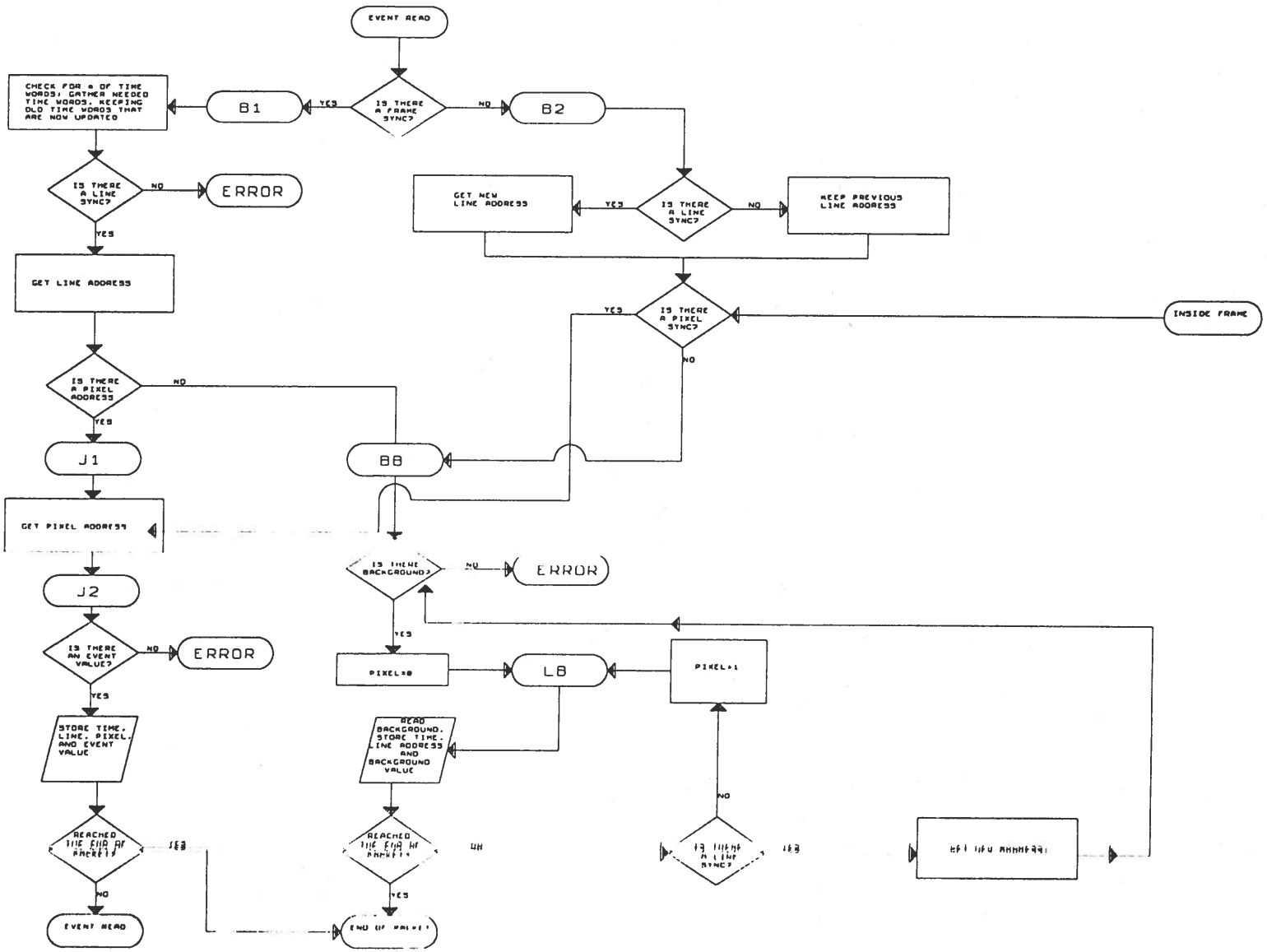
LIS-GSE Science Packet Decoding

(Chart 1 of 2)



LIS-GSE Science Packet Decoding

(Chart 2 of 2)



Appendix A:

Derivation Of Translation Equations

Menu for Focal Plane Simulator Board

1) Load Stair-Step Pattern.
2) Load Alternating Pattern.
3) Set Specific Voltage Level for CCD Output.
4) One Pixel Output.
5) Load Random Lightning Pattern.
6) Load Stored Random Lightning Pattern.
7) Load Double Stair-Step Patterns Into Two Memories.
8) Load Zero And Random Patterns Into Two Memories.
9) Load Specific Events Against A Random Background.

Discrete Telemetry Board

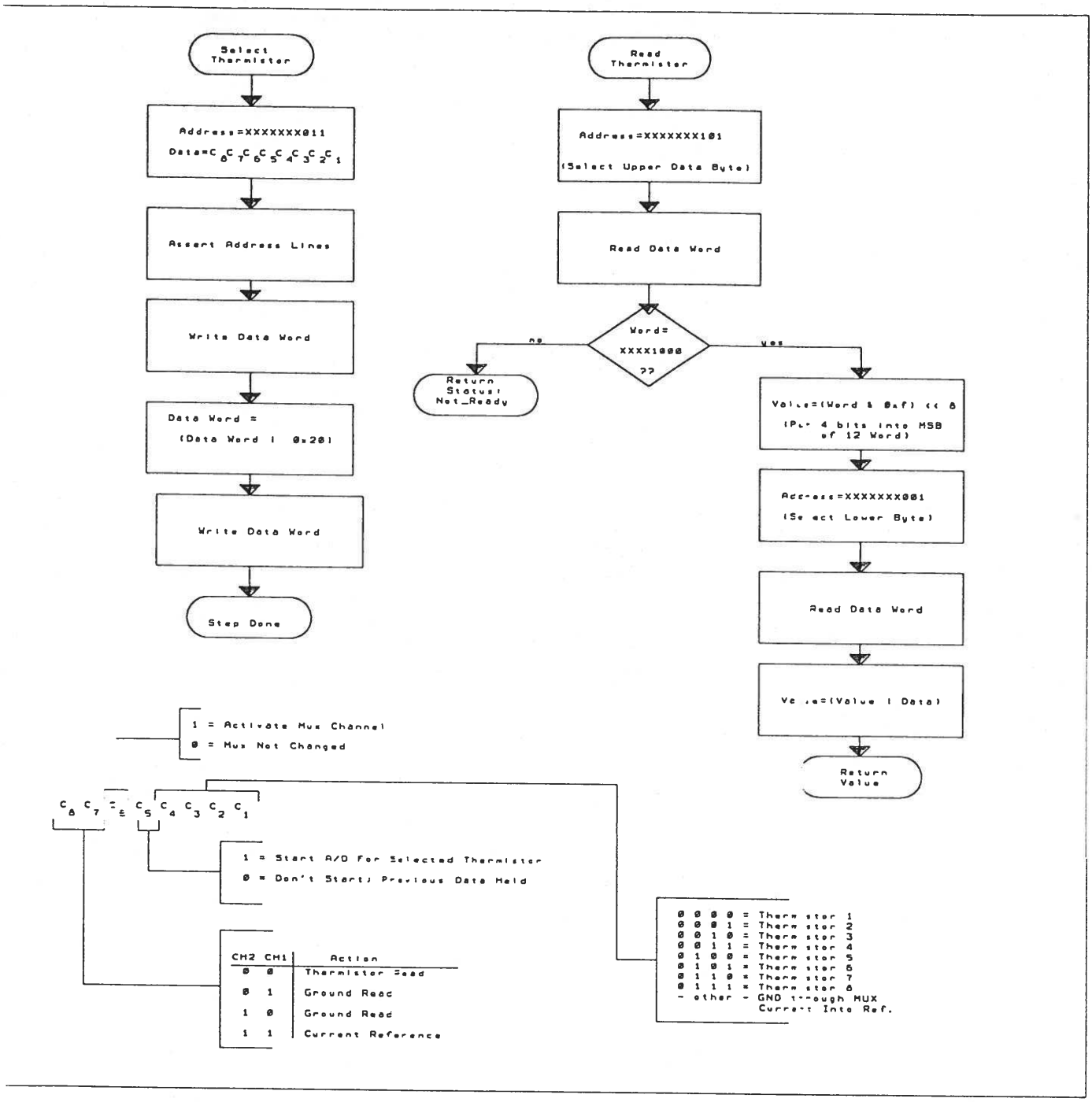
Board Address = $XXXXXXXA_2A_1A_0$ (binary)

Where the X represents a DIP-Switch selectable, TBD value

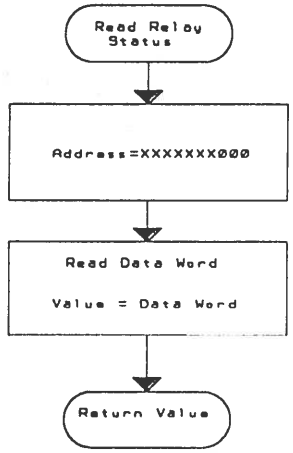
Interrupt = < IRQ3,... , IRQ7>; DIP-switch selectable (TBD)

$A_2A_1A_0$ = Address of a particular board function

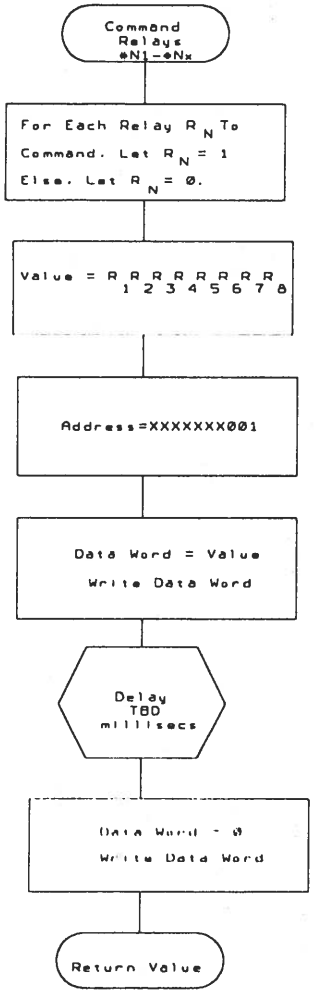
Address	Action	Associated Data Word
$A_2A_1A_0$		
0 0 0	Relay Status	Read 8-Bits
0 0 1	Relay Control	Write 8-Bits
0 1 0	Time-Mark Off/On	Write 1=ON; 0=OFF
0 1 1	Set Multiplexer	Write 8-Bits
1 0 0	Lower A/D Value	Read 8-Bits
1 0 1	Upper A/D Value	Read 8-Bits
1 1 0	Interrupt Enable	don't care
1 1 1	Interrupt Disable	don't care



Passive Analog (Thermistor) Data Collection



MSB=Relay8; LSB=Relay1
if (bit = 0) contact closed
if (bit = 1) contact open



Relay Command And Status
(Power Distribution Control)

Appendix A:

Derivation Of Translation Equations

LIS HOUSEKEEPING DAS MEASUREMENT SYSTEM

The LIS Housekeeping Data Acquisition System multiplexes 16 input channels and converts the input voltage, ranging from -5.0 to +5.0 volts, to a 12-bit digital word (0 to 4095 counts). The 16 input channels are identified as CH0 to CH15.

To compensate for gain and offset drift CH14 and CH15 are used to measure the Analog Ground Reference (AGR = 0 volts) and a Precision Reference Voltage (PRV = 2.5 volts). The two measurements are then used during the conversion to engineering units to correct for any DAS calibration change.

To convert any channel (CHx) from counts back to measured volts the following linear equation is used.

Measured Volts (MV) = A * CHx (counts) + B

The A and B values are made adaptive by using the following equations to compute A and B.

A (volts/count) = PRV / (CH15 - CH14)

B (volts) = - A * CH14

substituting for A;
B = - PRV * CH14 / (CH15 - CH14)

Now the Measured Volts equation becomes by substituting for A and B;

MV = PRV * CHx / (CH15 - CH14) - PRV * CH14 / (CH15 - CH14)

collecting terms;
MV = PRV * (CHx - CH14) / (CH15 - CH14)

The PRV is designed to be <u>2.5 volts</u>. A precision measurement of PRV will be made during the final testing of the electronics assembly.

To convert from MV to Engineering Units (EU) requires the following linear equation:

EU = Ax * MV + Bx; where Ax and Bx vary as a function of the type measurement. The following chart defines the EU equations for each measurement.

LIS HOUSEKEEPING DAS MEASUREMENT SYSTEM

The LIS Housekeeping DAS measurement list:

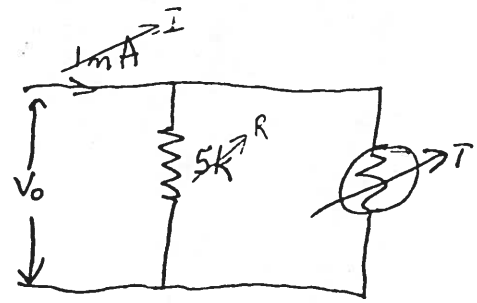
CHANNEL NUMBER	MEASUREMENT NAME	Engineering Unit (EU) Conversion Equations
0	Spare	
1	Spare	
2	Spare	
3	Spare	
4	Optics Filter Temperature	EU (degrees-C) = MV * 71.43 - 273
5	Focal Plane Array Temperature	EU (degrees-C) = MV * 71.43 - 273
6	Controller Board Temperature	EU (degrees-C) = MV * 71.43 - 273
7	Power Converter Temperature	EU (degrees-C) = MV * 71.43 - 273
8	Power Converter Input Current	EU (amps) = MV * 1.0
9	- 15.0 Volt Analog Circuit Power	EU (volts) = MV * 3.318
10	+ 5.0 Volt Analog Circuit Power	EU (volts) = MV * 3.318
11	- 5.2 Volt Analog Circuit Power	EU (volts) = MV * 1.153
12	+5.2 Volt Analog Circuit Power	EU (volts) = MV * 1.153
13	+5.0 Volt Digital Circuit Power	EU (volts) = MV * 1.110
14	Analog Ground Reference (CH14)	Used to maintain DAS calibration
15	Precision Reference Voltage (CH15)	Used to maintain DAS calibration

DEFINITIONS:

Precision Reference Voltage (PRV) = 2.5 Volts
Measured Volts (MV) = (PRV) * (CHx - CH14) / (CH15 - CH14)
CHx = The 12 bit digital value of the measured channel x (where x = 0 to 13)
CH14 = The 12 bit digital value of the measured Analog Ground Reference
CH15 = The 12 bit digital value of the measured Precision Reference Voltage

LIS Thermistor Transfer Function

↳ S-311-P-18 ~~-07R6~~ -02-7R6



$$R_{eq} = \frac{RT}{R+T}$$

$$V_o = I R_{eq} = I\left(\frac{RT}{R+T}\right)$$

extreams:

@ −80°C : $T = 1660$ K ohms
@ 0°C : $T = 7355$ ohms
@ +100°C : $T = 152.8$ ohms
@ +150°C : $T = 41.9$ ohms

@ −80°C: $V_o \doteq (1\times10^{-3})\left[\frac{5\times10^3 * 1660\times10^3}{5\times10^3 + 1660\times10^3}\right]$
≈ 4.985 Volts

@ 0°C : $V_o \doteq (1\times10^{-3})\left[\frac{5\times10^3 * 7355}{5\times10^3 + 7355}\right]$
≈ 2.98 Volts

@ +150°C : $V_o \doteq (1\times10^{-3})\left[\frac{5\times10^3 * 41.9}{5\times10^3 + 41.9}\right]$
≈ 0.041 Volts

$V_o = I\left(\frac{RT}{R+T}\right) \rightarrow \frac{V}{I} = \frac{RT}{R+T}$

$$\left(\frac{V}{I}\right)R + \left(\frac{V}{I}\right)T = RT$$

$$\left(\frac{V}{I}\right)R = T\left(R - \frac{V}{I}\right)$$

$$T = \frac{\left(\frac{V}{I}\right)R}{R - V/I}$$

Let $V_o = 2.98$ Volts

↳ $T = \frac{[(2.98)/1\times10^{-3}]\,5\times10^3}{5\times10^3 - (2.98/1\times10^{-3})} \approx 0.03°C$