SOFTWARE DESIGN DOCUMENT

for the

DETECTOR CONTROL ELECTRONICS (DCE)
FLIGHT SOFTWARE

for the

COSMIC ORIGINS SPECTROGRAPH (COS)

Contract No. NAS5-98043
CDRL No. DM-03

Prepared for:

Goddard Space Flight Center
Greenbelt, MD

Prepared by:

Experimental Astrophysics Group, Space Sciences Laboratory
University of California, Berkeley

COS-UCB-009

Prepared By:	**Reviewed By:**
Daniel Blackman FUV Detector Software Lead	Geoff Gaines FUV Detector Systems Engineer
Robert Lampereur COS Flight Software Engineer	
Allison Elliott COS Flight Software Engineer	Grant Blue BATC COS Software Manager

Approved By:

Dr. Kenneth Brownsberger HST-COS Software Scientist University of Colorado, Boulder	Date

Revision Record

Rev	Date	Reason for Revision	Project Released
-		Original Issue	

NOTE: The margins of this document are marked with a vertical black line to indicate where changes (additions, modifications, corrections, and deletions) were made from the previous issue.

Table of Contents

Table of Figures

1 SCOPE

1.1 Identification

This Software Design Document (SDD) establishes the software design for the Far UltraViolet (FUV) Detector Control Electronics (DCE) Flight Software (FSW) Computer Software Configuration Item (CSCI) for the Cosmic Origins Spectrograph (COS) instrument. The COS instrument is to be installed into the Hubble Space Telescope (HST) as part of a future HST Servicing Mission. The developer of the FUV Detector Subsystem, which includes the DCE FSW, is the Experimental Astrophysics Group (EAG) at the Space Science Laboratory, University of California, Berkeley.

This document has been prepared in accordance with the requirements of contract NAS5-98043, between Goddard Space Flight Center (GSFC) Code 442, Greenbelt, Maryland, and University of Colorado/Center for Astrophysics and Space Astronomy (CU/CASA), Boulder, Colorado. It is written in accordance with SCM-1050, HST COS Data Requirements Document, Part DM-03.

1.2 System Overview

COS is a fourth generation axial replacement instrument for the Hubble Space Telescope. The capabilities of COS allow astronomers to pursue programs of scientific interest that are not feasible with previous generation HST instruments.

Figure 1.2-1 is a top-level block diagram showing the flow of information to and from the COS instrument. As shown in the figure, scientific proposals are converted into the appropriate command blocks that are uplinked to the HST. Commands are received and distributed by the Science Instrument Command and Data Handling System (SI C&DH) which is the interface between the HST spacecraft and the HST science instruments. The SI C&DH sends the COS specific commands to the COS instrument via the Remote Interface Unit (RIU). The Control Section (CS) is responsible for accepting and processing the macro commands (or macros) forwarded by the RIU. Some macros require the CS to communicate with the DCE. Upon receipt of a macro command, the CS validates it and sends the appropriate sub-system level commands to the DCE.

Within the FUV Detector Subsystem, the DCE FSW provides the FUV detector with the computational, interface, and memory resources necessary for performing FUV detector control activities. Commands from the CS are accepted and processed by the DCE FSW. The DCE FSW outputs engineering telemetry data in the form of housekeeping packets. The DCE FSW controls the FUV detector hardware that generates science data, however, it sees only counts and not location information for each photon event. Science data does not pass through the DCE FSW, but rather is output from FUV detector hardware to the Detector Interface Board (DIB) for processing by the DIB flight software.

1.3 Document Overview

This Software Design Document describes how the requirements that have been allocated to the COS DCE FSW are satisfied by the design. The organization of this document is outlined below with a description of each major section.

- Section 1 - defines the scope of this document and introduces the FUV Detector Subsystem and DCE FSW.
- Section 2 - lists documents that have been directly or indirectly referenced by this document.
- Section 3 - presents the top-level design of the DCE FSW.
- Section 4 - presents the detailed design of the DCE FSW.
- Section 5 - presents DCE FSW development environment.
- Section 6 - describes the DCE hardware and software operating environments.

- Section 7 - contains a glossary of document acronyms and terms.

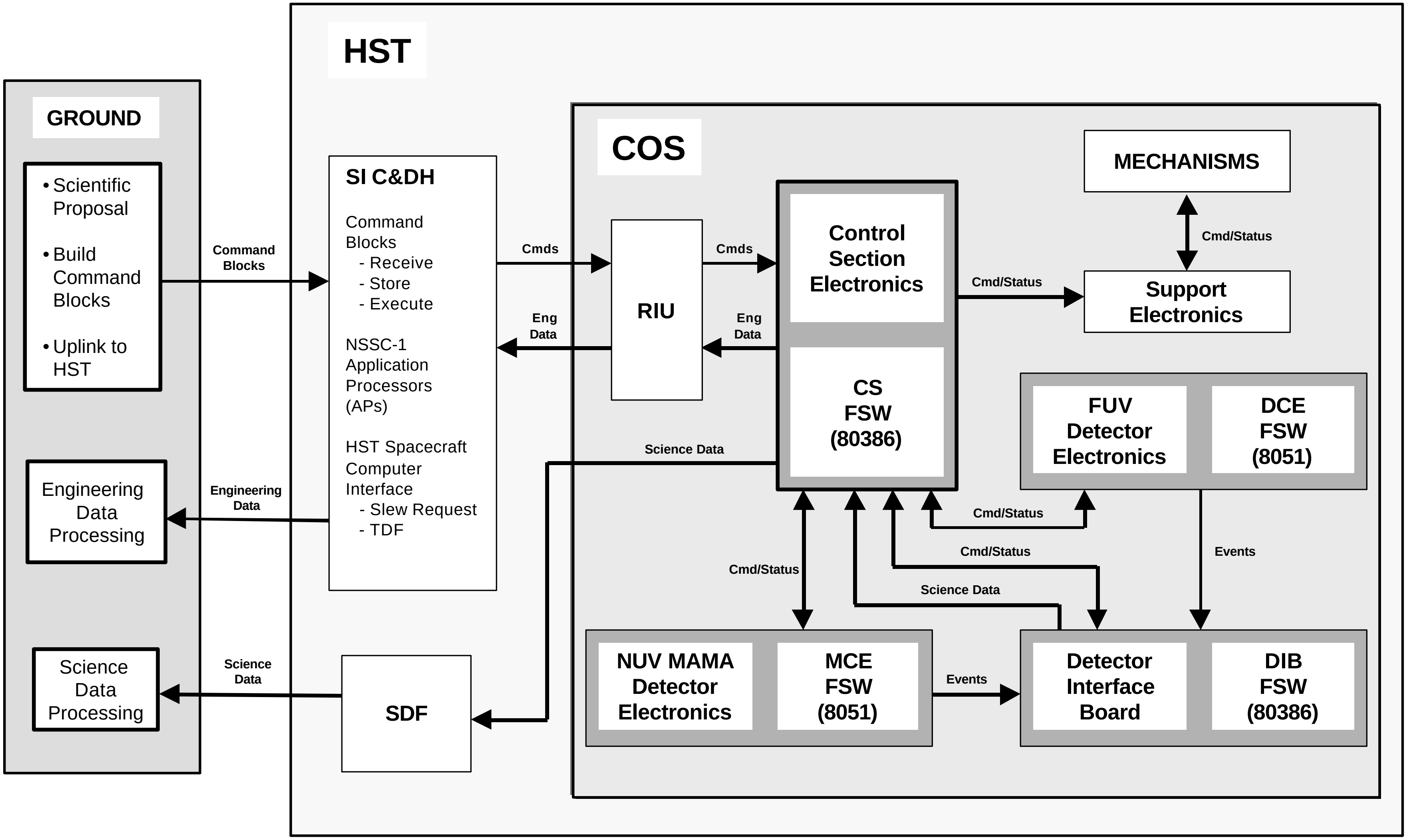


Figure 1.2-1. COS Block Diagram

2 APPLICABLE DOCUMENTS

2.1 Relationship To Other Documents

This Software Design Document describes both the top-level and detailed design of the COS DCE flight software. The design indicates how the system level requirements assigned to the DCE FSW are being fulfilled. Software requirements are detailed in the DCE FSW Requirements Document (COS-UCB-004). Requirements traceability is documented in matrix format in the FUV Software Test Plan (COS-UCB-008). Many details of FUV detector operations along with the command and housekeeping interface to the MEB are described in the FUV ICD (COS-UCB-001) and the COS Flight Software User's Guide (DM-03).

2.2 Referenced Documents

These documents provide additional details concerning the design and implementation of features required of the DCE FSW.

2.2.1 COS Documentation

- Interface Control Document, FUV Detector Subsystem, COS-UCB-001
- Configuration Management Document, FUV Detector Subsystem, COS-UCB-003
- DCE Flight Software Requirements Document, COS-UCB-004
- FUV Detector Configuration Management Plan, COS-UCB-005
- FUV Detector Software Test Plan, COS-UCB-008
- FUV Detector Software Test Procedures
- COS Control Section Flight Software Requirements Document, DM-03
- FUSE DPU Source Code (*source code is referenced since FUSE has no Software Design Document*)
- DCE Hardware/Software Interface Document
- COS Flight Software User's Manual, DM-03

2.2.2 Commercial Off-the-Shelf Documentation

- MCS 51 Micro-controller Family User’s Manual, Order Number 272383-002, Intel, Feb ’94
- Archimedes MCS-51 Macro Assembler, Version 5.02, Archimedes Software Inc.
- Archimedes L51 Linker/Locator, Version 3.52, Archimedes Software Inc.
- Archimedes MCS-51 Object-to-Hex File Converter, Version 2.1, Archimedes Software Inc.

3 TOP-LEVEL DESIGN

The top-level design portion of this document addresses the overall functionality of the DCE FSW. It shows how the system requirements are fulfilled by the overall flight software design.

The top-level design is presented from of an operational perspective and attempts to do the following:

- provide high-level descriptions of the internal processes of the DCE FSW and how those processes map to the system requirements
- identify flow of control and the events that can affect the processing of data
- examine the data flowing in and out of the system
- describe the hardware interfaces external to the DCE FSW
- address the flight software operating modes

The top-level design is broken down into several sections. The first section gives a functional overview of the DCE FSW. It describes each Computer Software Component (CSC) of the software and explains how that component fulfills certain system requirements. This section comes first since it provides an overview of what the DCE FSW does, and not how it does it.

The control flow section shows how the software components are combined to form the DCE FSW. This section shows the flow of control through the system, and how processing logic is affected by the various events that occur in the system. The System Level Control Flow Diagram (Figure 3.3-1) is included to graphically show how the DCE FSW works.

The data flow section shows how data flows in and out of the DCE FSW. This section is only concerned with data that enters and leaves the flight software. The Context Level Data Flow Diagram (Figure 3.4-1) is included to graphically show the system-level data flows.

The hardware interfaces section describes the interfaces that are external to the DCE flight software. The interfaces are dealt with at a high-level. The intent is to describe the type of hardware control and status that is provided by the DCE FSW. Each interface described in this section is represented on the Context Level Data Flow Diagram (Figure 3.4-1).

The last section looks at the DCE FSW operating modes. This section describes the various operating modes and how the user transitions from one mode to another.

3.1 Functional Overview

The FUV Detector Control Electronics (DCE) is the main control and coordination center for the FUV Detector Subsystem. It provides the FUV detector with the computational, interface, and memory resources necessary for performing FUV detector control operations. The DCE flight software resides in the DCE and provides the control and status capabilities of the FUV detector.

When power is applied to the FUV Detector Subsystem, the DCE FSW begins executing. The FSW is responsible for initializing its own operating environment and putting the DCE hardware in a known configuration. After the FSW has initialized, it is ready to accept commands from and provide status to the Control Section.

The DCE FSW accepts and validates incoming command packets sent from the CS. The command packets allow the CS to control and configure the FUV detector. The DCE FSW supports commands that configure the digitizers, set voltage limits, ramp the high-voltage, move the door mechanism, and generally support FUV detector operations.

The DCE FSW returns housekeeping data to the CS. Housekeeping provides insight into the state of the FUV detector. The FSW collects engineering data and formats it for output to the CS via the housekeeping interface.

Engineering data comes in many forms including temperatures, voltages, currents and other sensor data that are converted to a digital format.

The DCE FSW provides other necessary functions beyond command and status. It performs internal limit checking of critical detector parameters, reports the rate at which detector events are occurring to protect the FUV detector from an overlight condition, and performs various background checks to ensure the integrity of the operating environment.

Though the DCE FSW controls the FUV detector hardware, it does not process the science data generated by the FUV hardware. The FSW sees only counts and not coordinate information for each photon event. Science data does not pass through the DCE FSW, but rather is output from the FUV detector to the Detector Interface Board (DIB) for processing by the DIB flight software.

3.2 Computer Software Components (CSCs)

This section examines the functionality of the DCE flight software from a high level. It looks at the Computer Software Components (CSCs) that make up the DCE FSW. A CSC is a logical grouping of software that performs a common function. When viewed at the detailed level, a CSC consists of one or more software units. A software unit is a function, task or interrupt service routine. In this section only the high level functionality is addressed. The detailed design section of this document looks at the software units that comprise the various CSCs.

This section introduces the various CSCs that comprise the DCE FSW. It shows how the CSCI level requirements map to the CSCs. This section does not call out specific requirements that are being fulfilled, but rather addresses requirements in logical groupings. For a complete mapping of the DCE FSW requirements to the flight software design, see section 6 of this document, *Design Requirements Traceability*. For a mapping of DCE FSW requirements to software tests, see the FUV Detector Software Test Plan (COS-UCB-008).

The requirements allocated to the DCE FSW CSCI as stated in the DCE Flight Software Requirements Document (COS-UCB-004) are grouped as follows:

- DCE FSW Initialization
- Command Reception and Processing
- Detector Operations and Protection
- Door Control and Protection
- Memory Management

The DCE flight software must provide certain functionality based on the requirements allocated to the DCE FSW CSCI. The following is a list of the CSCs that comprise the DCE FSW CSCI. One or more of the CSCs may satisfy the requirement groups listed above.

- *Resets and Initialization CSC* - Initializes the DCE and FUV detector electronics along with the DCE FSW operating environment. This CSC satisfies *DCE FSW Initialization* requirements.

- *Command Reception and Processing CSC* - Accepts and processes command packets and memory upload data from the CS. This CSC satisfies *Command Reception and Processing* requirements.

- *Housekeeping Collection and Reporting CSC* - Provides status of the FUV Detector Subsystem to the CS via housekeeping data. This includes the logging of all diagnostic and error conditions detected by the DCE FSW. This CSC satisfies *Command Reception and Processing* and *Memory Management* requirements.

- *Current Limit Checking CSC* - Monitors critical detector hardware limits and maintains the detector in a safe configuration. This CSC satisfies *Detector Operations and Protection* requirements.

- *Global Rate Protection CSC* - Provides count rate protection of the detector to protect it from potential overlight conditions. This CSC satisfies *Detector Operations and Protection* requirements.
- *Detector Control CSC* - Configures and controls the FUV detector hardware. This includes the control and monitor of high-voltage ramping of the FUV detector. This CSC satisfies *Detector Operations and Protection* requirements.
- *Door Control CSC* - Provides control and status of the FUV door mechanism. This CSC satisfies *Door Control and Protection* requirements.
- *Background Monitoring CSC* - Performs internal tests to ensure the integrity of the DCE FSW operating environment. This CSC satisfies *Memory Management* requirements.
- *Support Functions CSC* - Provides a variety of commands that support FUV detector operations including the cyclic-redundancy checks, memory loads, and memory copies among other capabilities. This CSC satisfies *DCE FSW Initialization*, *Command Reception and Processing*, and *Memory Management* requirements.
- *Time Management CSC* - Allows the DCE FSW to perform certain functions on a periodic basis. This CSC satisfies *Command Reception and Processing*, *Detector Operations and Protection*, and *Door Operations and Protection* requirements.

The following sections address each CSC individually. They provide a functional description of the software component.

3.2.1 Resets and Initialization CSC

This CSC is responsible for initializing the DCE and FUV detector electronics along with the DCE FSW operating environment. It consists of the initialization software that is executed in response to any system reset. The DCE FSW implements a reset and initialization scheme that ensures the FSW is in a known configuration after any type of reset occurs.

Resets come in two flavors: unexpected resets and expected resets. An unexpected reset occurs due to a single event upset (SEU) or some other anomalous condition. An SEU occurs when a bit in memory gets flipped causing the software to execute out of an unexpected area of memory. Expected resets occur as part of normal detector operations. They include the power-on reset that occurs when power is applied to the DCE, and commanded resets that occur when a reset command is issued to the DCE or when the CS toggles the DCE hardware reset line.

All resets, whether unexpected and expected, are classified as one of three types:

- *Power-On Reset*: Only occurs as a result of power being applied to the DCE hardware.
- *Watchdog Timer Reset*: Occurs as a result of an SEU or the CS *not* sending a command to the DCE at least once every 10 seconds.
- *Commanded Reset*: Occurs when either the DCE is commanded by the CS to reset itself, or when the CS toggles the DCE hardware reset line. Commanded resets come in several forms and are described in detail in section 4.2.2 of the detailed design.

The reset and initialization software ensures that evidence of the state of the DCE FSW remains intact when an unexpected reset occurs. This allows ground personnel to analyze the contents of DCE memory to determine the nature of the problem. Information left over in DCE memory after an unexpected reset can be very helpful in reconstructing and understanding what happened.

The detailed description of how the DCE FSW handles each type of reset is covered in the detailed design section of this document.

3.2.2 Command Reception and Processing CSC

Activities performed by the FUV Detector Subsystem in support of science observations and detector maintenance are initiated and controlled through the use of command information sent to the DCE FSW from the CS. The DCE FSW provides the ability to accept the incoming command packets from the CS and perform the required actions indicated by the command op-code.

The DCE FSW implements a commanding structure that consists of command packets that are made up of 32-bit command words. A DCE command packet consists of multiple command words with each command word processed individually by the DCE FSW. After all of the command words associated with a command packet are received, the DCE FSW takes the appropriate action based on the command op-code.

The DCE FSW supports the following command performance constraints:

- The FSW is required to process command packet data from the CS at a rate no faster than 1 command packet per second. The DCE FSW may process command packets faster, but all command execution and housekeeping packet traffic must complete within one second of the command packet being received.
- The DCE FSW resets the DCE when it has not received a valid command from the CS in the last 10 seconds. This assumes the CS commanded the DCE FSW to enable watchdog timer resets.
- A command packet may be preceded by up to 512 32-bit words of upload data. A command packet consists of 32-bit words that may be sent in any order, provided that the command word containing the op-code is transmitted last.

All commands are sent via a command link to the DCE. On the CS side, a FIFO is in place to buffer up commands to the DCE. On the DCE side, each 32-bit word in the FIFO is transferred into a set of four 8-bit registers. The DCE reads these registers to form a 32-bit command word.

When 32-bits of command data arrives from the CS, the DCE hardware generates an interrupt. The FSW reacts to the interrupt and processes the 32-bit command word. The DCE FSW uses the most significant 16-bits of the command word to identify the type of data contained in the least significant 16-bits of the command word.

This CSC consists of an interrupt service routine to handle the receipt of 32-bit command words arriving at the command interface. It also contains a task for processing the commands entering the system. It does not, however, include all of the command functions. Command functions are allocated to the various CSCs based on functionality. A special Support Functions CSC exists that contains the command functions that do not map well to one of the other CSCs.

3.2.3 Housekeeping Collection and Reporting CSC

The main insight into the operations of the FUV Detector Subsystem comes through the housekeeping data. To gain the required visibility into the detector operations, all key detector parameters are identified and placed into the housekeeping data stream to the CS. Within the constraints of the engineering telemetry bandwidth and in accordance with the FUV ICD document, each key parameter is output in housekeeping at a rate sufficient to determine the health of the FUV detector and to perform a trend analysis on that parameter.

The DCE FSW provides the ability to collect, format, and output the engineering data necessary to achieve the scientific objectives of the FUV Detector Subsystem and to maintain the health and safety of the detector. The FSW responds with a housekeeping packet every time it receives a command from the CS.

In general, a housekeeping packet includes talk backs for all commanded settings; monitors for various temperature, voltage, current, and other sensors; command packet counters; photon event counters; pulse-height histogram data; self-test and diagnostic information; and status flags. Memory download data is sent out as a separate packet of data through the housekeeping channel and is not considered part of the Housekeeping CSC.

The DCE FSW supports the following housekeeping performance constraints:

- The FSW outputs engineering telemetry data to the CS at the rate of 512 32-bit housekeeping words per command response.
- Housekeeping may also include 512 32-bit memory download words transmitted in response to a memory download command from the CS.

The housekeeping interface between the DCE FSW and the CS is similar to the commanding interface. The DCE uses four 8-bit registers that serially shift data into the housekeeping FIFO in the CS. The DCE FSW writes each 32-bit housekeeping word to the housekeeping registers and the hardware takes care of shifting out the data. The FIFO on the CS side takes care of buffering up the housekeeping data for processing by the CS FSW.

This CSC includes a task that outputs the latest housekeeping data to the housekeeping interface. It also includes the commands required to set memory monitor addresses in the DCE FSW. The detailed description of the housekeeping data format is documented in Appendix D of the FUV ICD.

3.2.4 Current Limit Checking CSC

The DCE FSW is capable of collecting and monitoring key housekeeping data items for each of the identified critical FUV detector components. Key detector parameters are monitored at a rate that is sufficient to allow DCE FSW to be capable of performing predetermined corrective actions when a safety violation is discovered.

Most monitors are simply reported in housekeeping with no limit checking being performed by the DCE FSW. Some monitors are deemed critical, like high-voltage current, and are checked often by the DCE FSW. Other monitors, like door current, are also checked but less frequently (e.g. once a second in support of door operations). The DCE FSW is responsible for checking these critical monitors for out-of-limit conditions and taking corrective action.

If a current limit or the auxiliary power limit is violated, the DCE FSW turns off high-voltage power along with auxiliary power. It also puts all controls into a safe configuration. The limit values used by the DCE FSW can be modified either directly via DCE commands or by changing patchable constants.

This CSC includes a task to perform the periodic monitoring of critical detector limits. It also contains the command functions to change limit values and enable/disable limit checking.

3.2.5 Global Rate Protection CSC

The DCE FSW is capable of performing global event rate monitoring of the FUV detector. Global rate protection is implemented to protect the detector from an overlight condition. The FUV hardware has three types of counters: front end counters (FECA & FECB), digitized event counters (DECA & DECB), and science data counters (SDC1 & SDC2). For a complete description of these counters, reference the FUV ICD.

Global rate protection only looks at FEC counter value. The FEC counts are accessible to the DCE FSW via the hardware referred to as Counters A & B. Counters A & B are read on a periodic basis and compared against on-board values. If the either counter exceeds its maximum value, high-voltage is commanded to HV LOW. The threshold values and interval values for global rate protection can be changed on command.

This CSC includes a task that periodically performs global rate protection. The task is invoked once per second. It also includes the command functions to customize the count rate protection scheme, which include changing the threshold value and persistence value.

3.2.6 Detector Control CSC

The DCE FSW is capable of controlling the FUV detector electronics. The FUV detector consists of two detector segments named A and B. To support the goal of maintaining optimum detector performance over the life of the instrument, the DCE FSW provides the ability to configure each detector segment along with its associated detector interface electronics on command by the user.

The DCE FSW provides the capability to setup each of the detector segments and their associated interfaces. To aid in maintaining optimum FUV detector performance, the DCE FSW provides the ability to issue instructions to select detector gains, offsets, thresholds, and any other variable detector parameters.

Control of the FUV detector segments falls into one of two categories: detector setup and high-voltage ramping. Detector setup refers to activities that must occur prior to ramping up of the high-voltage. High-voltage ramping refers to the activities required to bring the FUV detector up to its operating voltage. The following three subsections address the topics of detector setup, high-voltage ramping and high-voltage status flags.

This CSC includes a task that controls and monitors the ramp up process. It includes the command functions to configure the digitizers, voltage settings and stim pulses. It also includes the command functions that initiate high-voltage ramp up and set the maximum allowable voltage level.

3.2.6.1 Detector Setup

The FSW provides a means of initializing the detectors for operational use. The flight software can configure the digitizers, high-voltage and stim pulses in preparation for exposures. The digitizer setup commands tune the way the digitizers process photon events. The voltage setup commands set values used by the high-voltage ramping process. The stim pulse commands control the stim pulse hardware. The following is a list of the types of detector setup commands processed by the DCE FSW.

Digitizer setup commands

- establish threshold and performance trim settings
- tune image position settings (offset and stretch, both segments, both axis)

Voltage setup commands

- set the low voltage value (voltage to go to when transitioning from high to low voltage, or when a count rate protection limit is exceeded)
- set the nominal high voltage value (voltage level at which to stop ramping HV)
- set the HV ramp rate (pause between HV ramp steps)
- set the maximum voltage level enforced by the hardware (this hardware protection does not let the FSW command high-voltage higher than the maximum limit)

Stim Pulse commands

- turn on and off the stim pulses
- set the rate at which stim pulse events occur

3.2.6.2 High-Voltage (HV) Ramping

High-voltage ramp up increases the MCP voltage on the detector in small increments starting at the current high-voltage level (LOW, NOM, NOMA, or NOMB) until it reaches the commanded nominal high-voltage value (NOM, NOMA or NOMB). The high-voltage ramp up process can be tailored using the detector voltage setup commands described above. The HV ramp rate is used to pause for a set amount of time between successive commands to increase the high-voltage. The DCE FSW supports a HV ramp command that starts the ramping process from the current high-voltage level and raises it to the specified level.

3.2.6.3 High-Voltage (HV) State Bits

The HV state bits are designed to reflect the expected state of the high voltage. This means that the DCE FSW updates the HV state bits whenever it configures the hardware to a new high-voltage state. Reconfiguration of the HV hardware occurs because a command was received from the CS to change the HV state, or because an event occurred within the DCE FSW (e.g. current limit violation) that causes the FSW to change the HV state autonomously.

The DCE FSW changes the HV state bits under the following conditions:

- a voltage set command (LFHVSET or LFHSTATE) was received from the CS
- a power command (LFHVPWR) was received from the CS
- the DCE FSW took autonomous action to change the HV state (a count rate violation puts HV in LOW, NOMA or NOMB depending on the violation; a limit violation turns off the HV; a DCE reset turns off the HV)

3.2.7 Door Control CSC

Reliable and safe configuration of the FUV detector door is a critical factor in supporting the capability of the detector to meet its scientific performance requirements. The door must be open for the FUV detector to perform useful science. The DCE FSW performs the functions necessary to configure and position the door using the support electronics.

Door control falls into two categories: motor control and actuator control. Motor control commands allow the door to be moved using the door motor. This is the primary method of moving the door mechanism. Actuator control commands allow the door to be moved in the event the door motor fails. The actuator is a limited life mechanism and can be reset using the door motor.

Motor control commands

- turn on aux power supply
- start a timer to guarantee that the door move completes within a certain time
- enable commanding of the door motor via a timer (another level of protection)
- set direction of the move
- turn on motor power
- enable door end switch override to allow the door to be reset after actuator use

Actuator control comands

- turn on aux power supply
- start a timer to guarantee that the actuator activation completes within a certain time
- select one of two actuator windings
- turn on actuator power

Since the FUV door may only be opened when the COS instrument is in a vacuum, it is critical that the door not inadvertently open. The door has interlocks to prevent inadvertent powering of the door motor or actuators. Some of these interlocks are in hardware while others are in software. The DCE FSW requires at least two separate command packets to move the door. Additionally, the DCE FSW requires that all door and actuator moves be completed within 3 minutes. If the command timer expires, a *door stop* is performed to put the door in a safe state. The state of the DCE that constitutes a *door stop* is covered in the detailed design portion of this document.

The FSW does not monitor door moves since the door hardware uses optical switches to determine when a move is done. The FSW monitors the aux power current limit when a door move is in progress. Nominally, the current limit protection software monitors this limit, however, the door control software also monitors it during a door move. If the door control software detects a limit break, a *door stop* is performed.

There are a number of events that can result in a *door stop* occurring. A *door stop* command, door timeout, or DCE reset causes any door move to stop. The FSW stops the door motion by reverting to its initialization state: door and actuator power off, and door commands disabled. The door latch opening or closing also causes the FSW to perform a *door stop*.

This CSC includes a task that monitors the current draw of the door and the door latch position. The CSC also includes the command functions that initiate moving the door, turn on and off the aux power supply, and initiate user of the actuator.

3.2.8 Background Monitoring CSC

The DCE FSW performs certain self-checks during initialization and during normal operations. The FSW reports any unusual or unexpected conditions as diagnostic messages. The DCE FSW performs two types of background monitoring: CRC checking of memory, and high-water mark monitoring of the stack. The CRC checking is designed to actively look for problems, such as SEUs, in memory. Stack monitoring is intended to track execution of code that causes the stack to grow unexpectedly large.

This CSC includes a task that performs a CRC check on defined memory ranges and a task that performs a background built-in test (BBIT) on the stack to determine its high-water mark. There are no command functions associated with this CSC since background monitoring cannot be disabled.

3.2.9 Support Functions CSC

The support functions include all DCE command functions that support DCE operations, but do not conveniently fall into one of the other CSCs. The types of command functions that are considered support functions include: no-op command, memory load and copy commands, jump commands, watchdog timer commands, and reset commands among others.

This CSC includes any command functions used in support of detector operations that do not fall into one of the other CSCs.

3.2.10 Time Management CSC

The DCE provides hardware timers that are used by the DCE FSW to track time. The FSW must be aware of time in order to perform certain time critical tasks.

There is a need for two hardware timers: a timer tick set to 49.152 ticks per second, and a watchdog timer set to 10 seconds. Both timers are programmed during initialization. The watchdog timer must be reset within 10 seconds otherwise a watchdog timer reset occurs.

The timer tick interrupt service routine keeps track of the number of times it is invoked. By doing this, it can maintain software timers. The software timers are used to control the following activities:

- expiration of the door move timer
- expiration of the door latch timer
- collection of Counter A & B data
- invocation of the count rate protection software
- invocation of the high-voltage ramping software

The DCE hardware has a 16MHz clock that is fed into the 8051 microcontroller's timer hardware. The 8051 hardware automatically divides the clock down by 12, which effectively results in a 1.333 MHz clock. The 8051 timer is set to free running 16-bit mode which means that it will rollover (i.e. tick) once every 49.152 milliseconds when its counter overflows. The timer tick calculation is as follows:

*1 second / 1,333,333.333 counts * 65536 counts / 1 tick = 0.049152 seconds/tick = 49.152 msec/tick*

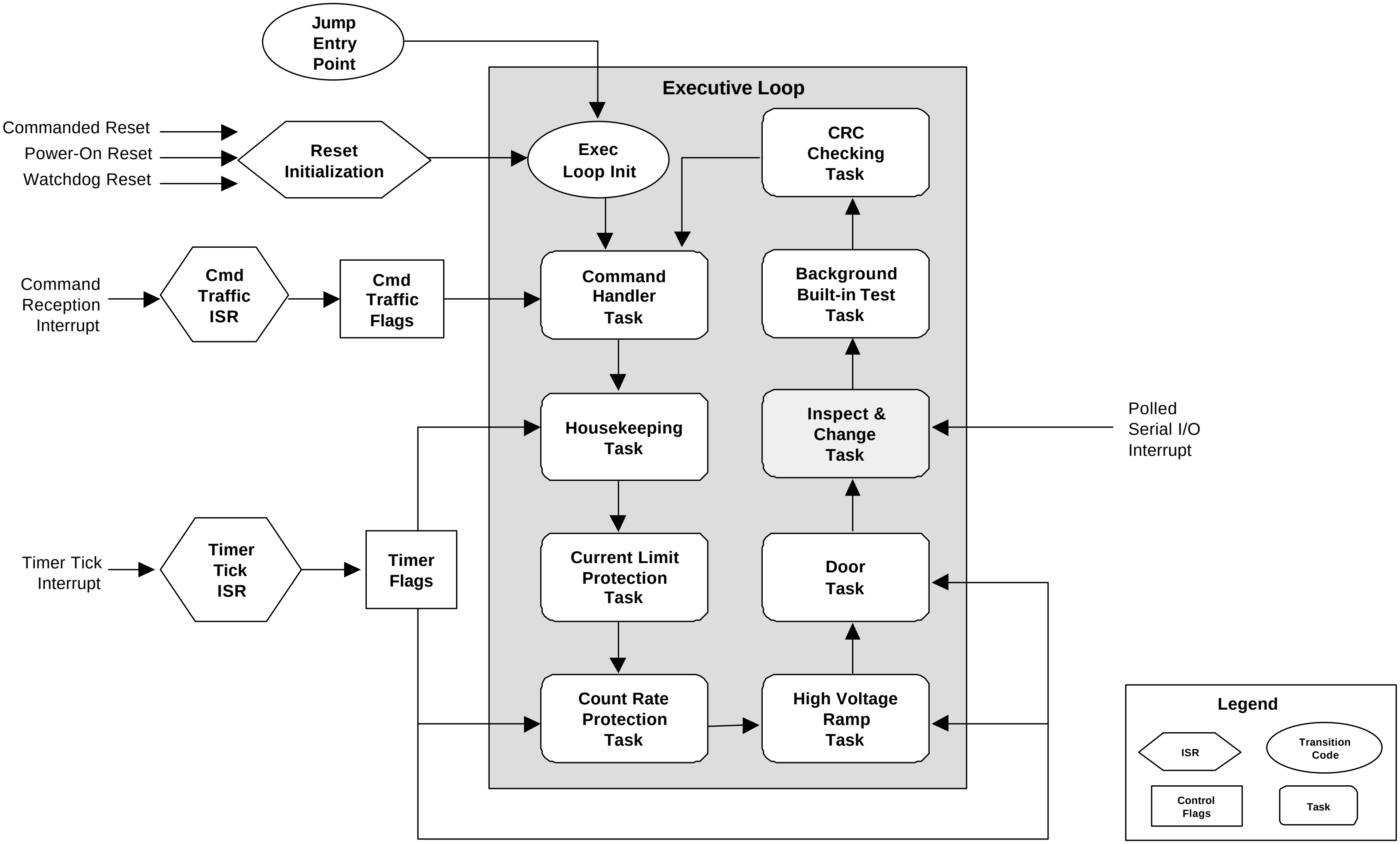
This CSC includes the interrupt service routine that manages time for the rest of the DCE FSW.

3.3 Control Flow

The control flow section shows how the software components are combined to form the DCE FSW. This section shows the flow of control through the system, and how processing logic is affected by the various events that occur in the system. The System Level Control Flow Diagram is included to graphically show how the DCE FSW works.

The basic design of the software control flow is that of an initialization sequence followed by an infinite processing loop. This control flow originates with a reset event, such as power-on to the DCE. The main software processing loop, called the executive loop, can be interrupted by a hardware signal that directs the 8051 to execute a special sequence of code called an interrupt service routine (ISR). ISRs are designed to execute quickly and terminate, restoring control to the executive loop at the point it was interrupted.

Figure 3.3-1 is the System Level Control Flow Diagram for the DCE flight software.



- This diagram is an idealized version of the Executive Loop with the order of task invocation being approximate. The actual order that tasks are invoked in the executive loop is documented in the detailed design section.
- The *Jump Entry Point* is the point of entry for the *jump* command when jumping from Boot to Operate, Operate to Operate, or Boot to Boot.
- The *Inspect & Change Task* is greyed because the current design does not have this task included in the software. The hooks are left in place just in case the task becomes necessary for test purposes.

Figure 3.3-1. System Level Control Flow Diagram

3.4 Data Flow

The data flow section shows how data flows in and out of the DCE FSW. This section is only concerned with data that enters and leaves the flight software. The Context Level Data Flow Diagram is included to graphically show the system level data flows.

The data flow diagram depicts the DCE FSW as a single entity with interfaces to the DCE and CS hardware. The diagram uses three types of symbols to represent the various entities in the system. A circle represents the DCE flight software. Round-cornered boxes represent the DCE interface hardware that is used by the DCE FSW to control and/or status some piece of DCE hardware, or to communicate with the CS. The square-cornered boxes represent the DCE or CS hardware with which the DCE FSW is communicating.

It should be noted that the diagram depicts the flow of data between MON51 test software and the DCE FSW. The flight version of the DCE FSW will not contain MON51 related software, however, the data flow is included here for completeness.

Figure 3.4-1 is the Context Level Data Flow Diagram for the DCE flight software.

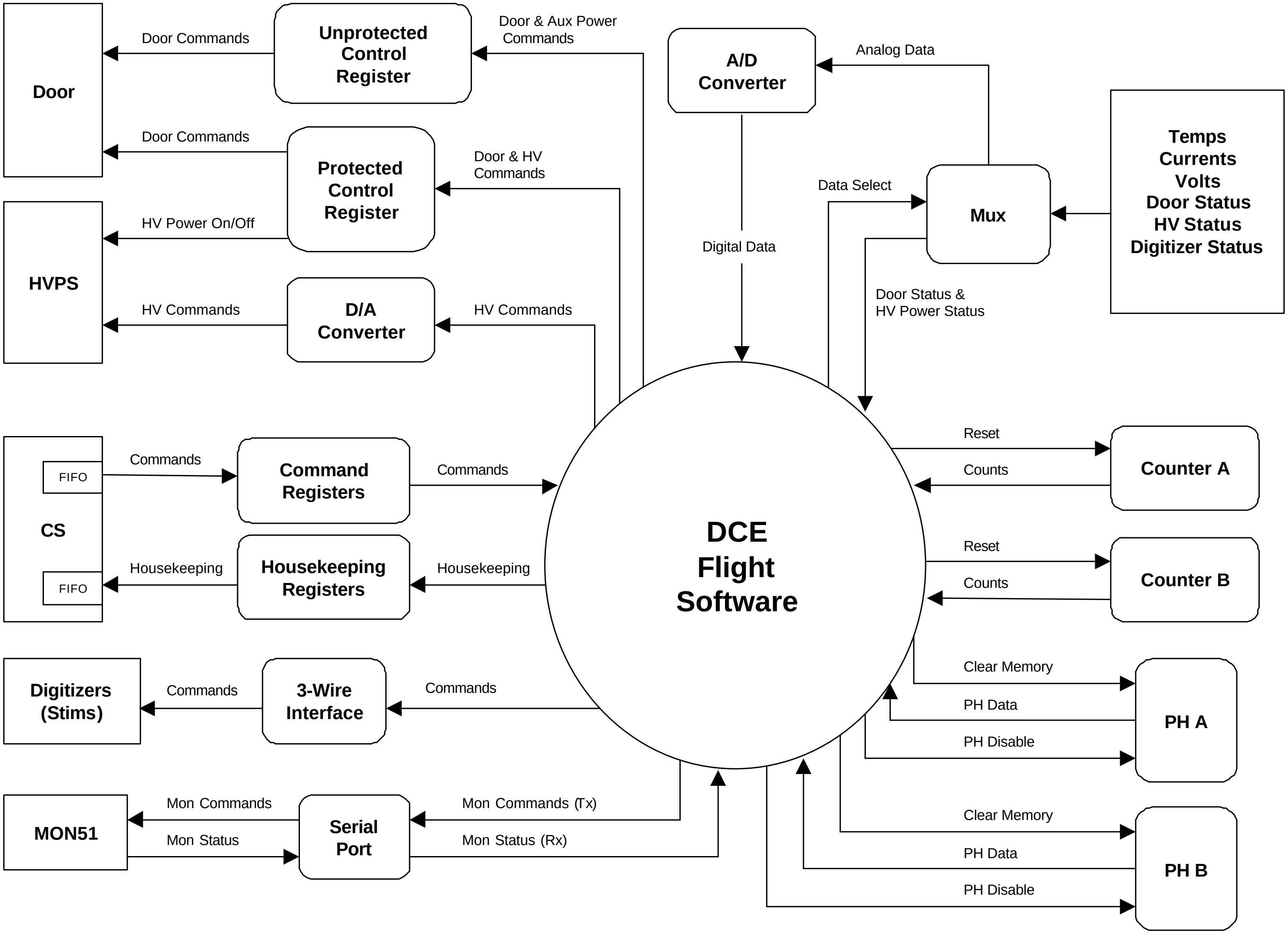


Figure 3.4-1. Context Level Data Flow Diagram

3.5 Hardware Interfaces

This section describes the hardware interfaces that are external to the DCE flight software. External interfaces refer to the following:

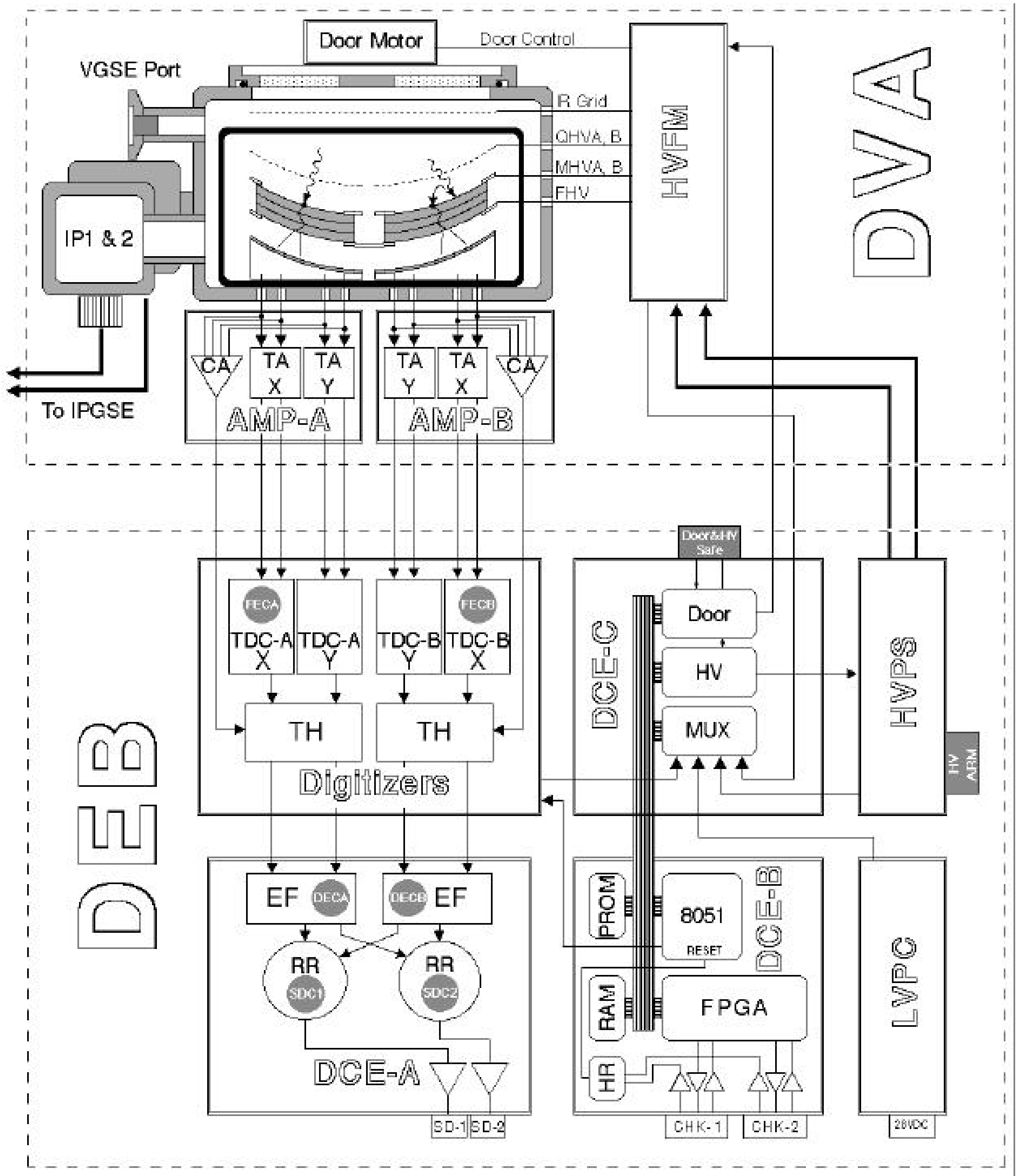
- DCE hardware that the DCE FSW controls and reads status from
- CS hardware that the DCE FSW receives commands from and sends housekeeping data to

External interfaces are represented by the round-cornered boxes depicted in the Context Level Data Flow Diagram shown in Figure 3.4-1. As a point of comparison, internal interfaces refer to the 8051 micro-controller ports and memory mapped I/O address used by the DCE FSW to access the external interfaces. The detailed design section of this document provides a complete description of 8051 ports and memory mapped I/O and how they are used by the FSW.

The external interfaces are dealt with at a high-level. The intent of this section is to describe the type of hardware control and status that the DCE FSW provides. The low-level commanding of the hardware (i.e. which bits in which register to set or clear) is defined in the DCE Hardware-to-Software Interface document.

This section is broken down into two subsections. The first subsection looks at the interfaces between the DCE FSW and the CS, and the second subsection examines the interfaces between the DCE FSW and the DCE hardware. Within those two subsections, all of the external interfaces are described. Each external interface is described in terms of the control provided by the DCE FSW and the status returned by the hardware.

Figure 3.5-1 (shown below) is the FUV Detector Functional Block Diagram. It provides an overview of the FUV detector hardware. It is included here to provide additional information about the hardware that is controlled by the DCE FSW.



TA - Timing Amplifier
CA - Charge Amplifier
TDC - Time-to-Digital Converter
TH - Threshold Ckt
EF - Event Formatter
RR - Round Robin Arbitrator

FEC - Front End Counter
PRC - Preamp Reset Counter
DEC - Digitized Event Counter
SDC - Science Data Counter
HR - Hardware Reset Ckt

GG 11/99

Figure 3.5-1. FUV Detector Functional Block Diagram

3.5.1 Interfaces Between the CS and DCE FSW

3.5.1.1 Command Registers

All DCE commands are sent via a command link from the CS to the DCE. On the CS side, a FIFO is in place to buffer up commands to the DCE. On the DCE side, each 32-bit word in the FIFO is transferred into a set of four 8-bit command registers. The act of reading the last of the 8-bit registers by the DCE FSW clears the "busy" bit and allows the next 32-bit CS FIFO value to be placed into the command registers.

The DCE has two redundant command links, A and B, which are both always active. Either link may be used to command the DCE at any time. The DCE FSW reads one set of command registers to process a command if one is present. If no command is present, it then reads the alternate set of command registers.

3.5.1.2 Housekeeping Registers

The housekeeping interface between the DCE and the MEB is similar to the commanding interface. The DCE uses four 8-bit registers that serially shift out data into the housekeeping FIFO in the CS. The DCE FSW writes each 32-bit housekeeping word to the housekeeping registers and the DCE hardware takes care of shifting out the data into the housekeeping FIFO. The FIFO on the CS side takes care of buffering up housekeeping data for processing by the CS FSW.

3.5.2 Interfaces Between the DCE FSW and the FUV Hardware

3.5.2.1 Control Registers

The DCE FSW controls the high-voltage power supply (HVPS) and the door mechanism by setting and clearing bits in one of two control registers. The control bits that can cause severe degradation to the FUV detector if used incorrectly are allocated to a protected control register that has been equipped with a *lock* bit. The use of the lock bit is described in detail in the DCE Hardware-to-Software Interface document. The remaining control bits that cannot cause damage to the FUV detector have been allocated to an unprotected control register.

3.5.2.1.1 Protected Control Register

The protected control register contains control bits that can cause damage to the FUV detector if used incorrectly. The control bits allocated to this register are as follows:

- *HV power* - Turns on/off power to both the grid & MCP.
- *Ion pump on/off* - For GSE use only since ion pumps are not used in orbit.
- *PROM/RAM* - When zero, 8051 uses PROM for executing code. When one, it uses 8K RAM.
- *Door end switch override* - When override is enabled, the door auto-stop capability is not in effect. When disabled, the door auto-stop capability turns off power to the door motor.
- *Actuator A and B power* - Turns on/off power to actuators A and B.
- *Door motor power* - Turns on/off power to the door motor.

3.5.2.1.2 Unprotected Control Register

The unprotected control register contains control bits that cannot damage the FUV detector. The control bits allocated to this register are as follows:

- *Aux power* - Enables/disables auxiliary power for moving the door motor or actuators.
- *Door direction* - Sets the direction of a door move.
- *Actuator enable* - Enables/disables actuator support circuitry which allows the actuators to be powered. The DCE FSW mirrors the state of actuator enable bit in software for use in processing actuator commands.

3.5.2.2 HVPS Digital-to-Analog Converter (DAC)

The DCE FSW uses the High-Voltage Power Supply (HVPS) Digital-to-Analog Converter (DAC) to control the HVPS. The DAC converts a digital value to a high-voltage value in the approximate range of -2000 to -6000 volts at the detector head.

The DCE FSW uses the DAC to command the HVPS to a new voltage level and to set the maximum voltage level. Status information concerning the state of the HVPS is returned to the FSW via the multiplexer (MUX) and Analog-to-Digital Converter (ADC).

3.5.2.3 Digitizer 3-Wire Interface

The digitizer is the electronics that interprets photon events. The criterion used to qualify an event can be controlled via commands. The digitizers commands are sent via a synchronous serial link implemented with 8051 port pins. The digitizer returns setting talk-backs as an input to the MUX/ADC. The digitizer is programmed with default values upon power-on reset, and after that, settings are modified only on command.

The DCE FSW uses a 3-wire interface to send commands to the digitizer hardware. The DCE FSW sends values that set event filtering values among other adjustment values for controlling digitizer performance. The FSW sends over digital values that are converted to analog values by the digitizer.

3.5.2.4 Serial I/O Port

The serial I/O port is a feature built-in to the 8051 microcontroller. It is used to provide a back door method of communicating with the DCE FSW. Test versions of the DCE FSW might include capabilities to use the serial I/O port for communicating with special COTS test software (MON51).

3.5.2.5 Analog-to-Digital Converter (ADC)

An analog-to-digital converter (ADC) converts sensor voltages to digital values that are stored in the housekeeping data area. The sensors can collect temperature, voltage or current information, and hardware talk-back information.

The FSW must use the multiplexer (MUX) to select the appropriate ADC input source and to tell the ADC to perform the conversion. After selecting the desired source and initiating the conversion, the FSW must wait a short time for the input to settle. After waiting the settle time, the FSW reads the ADC converted value and stores it in the appropriate housekeeping location.

Certain ADC inputs (e.g. current limits) must be gathered at a high rate, while all other ADC inputs are read just before the FSW sends out the housekeeping data. The current limit protection software collects current limit information from the ADC and uses it in its protection algorithm.

3.5.2.6 Multiplexer (MUX)

As described in the ADC section, the multiplexer is used to select the appropriate input to the analog-to-digital converter. There are a number of possible ADC inputs that can be selected using the MUX.

In addition to selecting ADC inputs by writing to the MUX address, the same address is used to gather status information about the door and high-voltage. The state of the door and high-voltage can be determined by reading the MUX address. It should be noted that the DCE FSW is not reading from the MUX but rather from another latch that has the same address as the MUX. Some of the status bits reflect whether HV and grid have been powered on, and door switches (latched, opened and closed).

3.5.2.7 Counters

Counters reflect the number of photon events that have passed through the various stages of the digitizing process. There are several types of counters that count specific types of events. The FUV hardware has three types of counters: front end counters (FECA & FECB), digitized event counters (DECA & DECB), and science data counters (SDC1 & SDC2). For a complete description of these counters, reference the FUV ICD.

Each counter counts only the photon events that meet some criterion specified by the digitizer and its threshold settings. The count rate protection software uses the FEC counters to determine when a global count rate violation has occurred. Each type of counter has an A & B counter that maps to each FUV detector segment.

3.5.2.8 Pulse Height Histograms

Pulse Height (PH) Histogram data consists of 128 bins of consecutive 16-bit words. This data represent the total number of photons that were detected and distributed into bins according to their pulse-height as determined by the digitizer hardware. There are two PH Histogram bins: one for MCP Segment A and one for MCP Segment B.

The FSW periodically reads the PH bins and outputs that information to the CS through housekeeping. The act of reading the PH data causes the bins to be reset to zero.

The memory used to store PH data is accessible to both the PH Counter hardware and the 8051. An attempt to simultaneously access the same memory by both the 8051 and the PH Counter hardware could result in corrupted data. The PH Counter hardware design requires the 8051 to assert a PH disable signal while accessing the PH Counter memory and de-assert the signal when done.

3.5.2.9 Miscellaneous

There are several external interfaces that are used for GSE purposes only. They are mentioned here for completeness.

- *LEDs* - used during testing to verify that something is happening
- *Reset pin* - resets the DCE hardware
- *External reset link* - external reset controlled by the CS

3.6 Operating Modes

This section examines the two DCE FSW operating modes: Boot and Operate. It describes the functionality the operating modes provide and how the user transitions from one mode to another. The concept of Boot and Operate modes is borrowed directly from the CS flight software design.

The DCE FSW is composed of three related software executables known as the *Boot code image*, *Upper Operate code image* and the *Lower Operate code image*. All three executables are built from the same source code files. This means that the same executable code is built into all images, however, only certain functionality is available depending on the operating mode. The software determines the operating mode based on the area of memory from which the code is executing. For example, if the code is executing from PROM address space, the mode is Boot. If the code is executing from 32K RAM, the mode is Operate.

The Boot code image is burned into PROM and cannot be changed once the COS instrument is on-orbit. The Operate code image is stored in the EEPROM within the CS and can be updated over the life of the COS instrument. The following sections describe each DCE FSW operating mode in detail.

3.6.1 Boot Mode

The DCE embedded processor begins execution of the Boot flight software contained in PROM whenever power is applied to the FUV Detector Subsystem or when a reset occurs. The primary purpose of the Boot software is to provide a robust, non-volatile code image from which to load the Operate code images into RAM. Once the DCE is in Boot mode, the ground can then send commands to put the FUV Detector Subsystem into an operational state.

The Boot code provides enough capabilities to upload an Operate code image and transfer microprocessor control to the Operate code. This allows the Operate code image to be actively managed on the ground and new versions to be uploaded to the COS instrument. The Boot code also allows memory tests to be performed on the Upper Operate and Lower Operate code areas in 32K RAM area since Boot runs out of PROM.

The Boot code provides a subset of the capabilities found in the Operate code. The following is a list of Boot code capabilities:

- Perform initialization functions, including reset tests and memory initialization.
- Command reception and processing
- Housekeeping data reporting including memory monitors
- Memory uploads from the CS into DCE RAM
- Memory dumps from DCE memory to the CS
- Begin execution of the Operate code image
- Processing of a subset of DCE commands (but *no* HV, door or digitizer commands)
- Background CRC checking and stack monitoring

Since Boot does not support any HV, door or digitizer commanding, there is no need for count rate protection or current limit protection capabilities.

For a list of all commands valid in Boot mode see the DCE Command section of this document.

3.6.2 Operate Mode

Operate mode is entered after a DCE Operate code image has been loaded into DCE memory and the ground has issued the jump-to-operate command. After receiving the jump-to-operate command, the Boot code transfers control to the specified Operate code image. Up to two Operate images can be stored in DCE memory at the same time: one in the lower code area (LCA) and one in the upper code area (UCA).

The Operate code provides a superset of FUV detector control capabilities. The following is a list of Operate code capabilities:

- Command reception and processing
- Housekeeping data reporting including memory monitors
- Memory uploads from the CS into DCE RAM
- Memory dumps from DCE memory to the CS
- Begin execution of the other Operate code image
- Processing of all DCE commands including HV, door and digitizer commands
- Ramp detector high-voltage
- Current limit protection
- Count rate protection
- Move the door mechanism
- Background CRC checking and stack monitoring

For a list of all commands valid in Operate mode see the DCE Command section of this document.

3.6.3 Mode Transitions

As described above, Boot code begins executing as the result of power being applied to the FUV detector electronics or due to a microprocessor reset. Once in Boot mode, the ground commands the CS to copy the DCE Operate image to the DCE. Once the image is successfully copied, the ground issues a command to transfer control to the Operate code. This is the standard process for transitioning between Boot and Operate modes.

Boot mode can be entered under the following conditions:

- a power-on reset occurs
- a watchdog timer reset occurs while in Boot mode (because the watchdog timer was not stroked)
- a reset command was received and processed by the Boot mode software
- a watchdog timer reset occurs while in Operate mode
- a reset command was received and processed by the Boot mode software

Operate mode can be entered under the following conditions:

- a jump-to-Operate command was received and processed while executing in Boot mode
- a jump-to-Operate command was received and processed while executing UCA Operate code
- a jump-to-Operate command was received and processed while executing LCA Operate code

The following figure shows all the possible means of transferring between Boot and Operate modes.

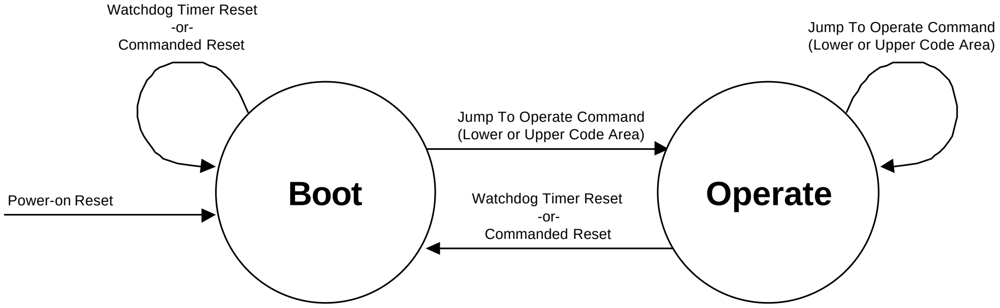


Figure 3.6-1. DCE Flight Software Operating Modes

4 DETAILED DESIGN

4.1 Design Approach

This section describes the overall software design approach to the DCE FSW. There are several factors that impact the way in which the DCE FSW is designed. This section examines these factors and attempts to describe how they influence the design.

4.1.1 Reuse from Previous Missions

The most influential factor is the reuse of software from previous missions. Previous missions that have used the same detector subsystem as the COS FUV detector include Far Ultraviolet Spectroscopic Explorer (FUSE) and Galaxy Explorer (GALEX). The reuse of software from one or both of these projects has a profound impact on how the COS DCE FSW is designed.

The lineage of the DCE flight software starts with FUSE and is followed by GALEX and COS. The bulk of the COS software design is based on the FUSE code.

Since the DCE FSW is relying on heavy design and code reuse from previous missions using the same detector subsystem, the detailed design reflects this reuse. Each subsection of the Software Architecture section includes a reuse description that details the level of reuse we expect to achieve. It also describes any changes or modifications we expect to make for COS.

4.1.2 Memory Utilization and Commanding

4.1.2.1 Memory and Op-Code Usage

Data structures are in fixed locations in memory and generally do not change from one version to the next. Command op-codes are dedicated to a specific function and are never reused. Although some commands may be retired and supporting code removed, the command op-code associated with that command function is never reused.

4.1.2.2 Commandable "Energy"

Commands for the HV and the door mechanism are considered *special* because they have the potential to severely degrade the performance of the detector. As special commands, the “energy” of the HV or door is not increased except when enabled by a separate software interlock. The energy level can be reduced, however, regardless of the state of the software interlock. This means that commands to turn *off* the door power do not require an “enable” flag to execute the command. A command to turn *on* the door power, however, does not turn on power unless a “Door Enable” command is issued first.

The general concept is that of a subsystem “energy level". A subsystem with power applied to it has a higher energy level than a subsystem that is powered off. For example, detector MCPs running at nominal HV have a higher energy level than detector MCPs running at low HV. Increasing HV from low to nominal requires software protections while reducing HV from nominal to low does not.

4.1.3 Boot and Operate Software Designs

4.1.3.1 Equivalent Executable Images

To maximize FUSE code reuse, the COS DCE Boot and Operate images are logically equivalent. The DCE FSW is designed such that the Boot and Operate images use the exact same source code files. Since the images run out of different areas in DCE memory, the linker produces executable images with addressing relative to the target memory area. This means the images are logically equivalent, however, not identical. Figure 4.1-1 below shows how the Boot and Operate code images are located in DCE memory. For a detailed view of the contents of DCE memory, see the DCE memory map described in section 4.4.2.3.

The code is constructed such that a single set of source code can be used to build both Boot and Operate images. At the beginning of any task or command functions that is *not* executed while in Boot mode, code exists to check the current operating state of the FSW. If the software is executing out of the PROM address range, the DCE FSW does not execute the function or task associated with Boot mode. Should the DCE FSW receive a command while in Boot mode that is not a valid Boot mode command, it issues a diagnostic stating that an invalid Boot mode command was received.

The Boot code image also contains a *data area* that consists of a GSE Operate Test Image. This test image is intended for use only during GSE testing and is located with Boot in PROM for easy copying to RAM.

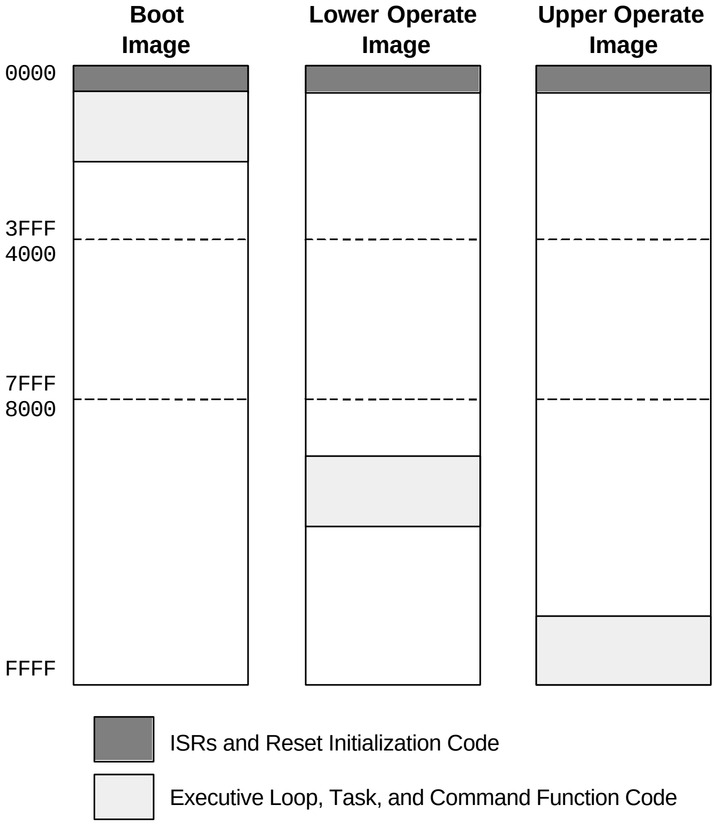

Boot Image
Lower Operate Image
Upper Operate Image
0000
3FFF
4000
7FFF
8000
FFFF
ISRs and Reset Initialization Code
Executive Loop, Task, and Command Function Code

Figure 4.1-1. Code Image Location in DCE Memory

4.1.3.2 Common Variable Space

Both Boot and Operate use the same space in RAM to store variable information. In addition, Operate images that exist in the upper and lower code area also share the same RAM space for variables. Any software variables that are common to Boot and Operate are overwritten whenever a system reset occurs, or when the ground commands the DCE to jump to Operate. This means that the state of software variables is not preserved in the event of a reset.

4.2 Software Architecture

Software architecture deals with the physical grouping of code versus the logical groupings called out in the top-level design section. The top-level design looks at the functionality of the DCE FSW from a logical perspective while the detailed design organizes processing logic into physical components.

This section addresses the physical software components that make up the DCE FSW. In the DCE FSW there are five major software component types: initialization code, an executive loop, interrupt service routines, tasks, and command functions. Each type of component is discussed in the following sections. Before looking at the major software components individually, a discussion of the operating system is necessary. The operating system overview describes how these software components are used together.

4.2.1 Operating System

The DCE flight software does not use a commercial off-the-shelf (COTS) operating system (OS). The operating system is built from the ground up using a simple design. The DCE FSW operating system consists of an executive loop that invokes a series of functions in a round-robin fashion along with interrupts that provide pre-emptive processing. The functions invoked by the executive loop are called *tasks*. Each task provides a unique set of functionality to the DCE FSW. Some of the tasks that make up the flight software include the command handler task, high-voltage ramping task, count rate protection task, among others. Each of these tasks is described in greater detail in following sections. The concept of tasks in the context of the DCE FSW is described in greater detail later in the document.

4.2.1.1 Overview

The easiest way to describe how the DCE FSW operating system works is to compare it to another operating system. Since the HST program is familiar with the operating system used by the Control Section Flight Software, it will be used as the point of comparison. The CS FSW uses the iRMK 1.3 commercial off-the-shelf operating system produced by the Intel Corporation. The iRMK operating system (or iRMK kernel as it is more commonly referred) handles all aspects of task switching and control.

Like the DCE OS, the iRMK kernel uses interrupts to provide pre-emptive processing. *Unlike* the DCE OS, the iRMK kernel allows the user to assign priorities to each task in the system. iRMK automatically determines when the highest priority task in the system is ready to run, switches it in, and starts executing it. It saves the state of the current task before switching in the next task. Since iRMK completely manages the switching of tasks, the developer does not need to deal with saving the task state. iRMK does that automatically.

In the DCE OS, there is no task priority concept. All tasks run at the same priority level. The round-robin nature of the executive loop means that all tasks are invoked in turn and the order cannot be dynamically altered. The following section details the concept of tasks in the DCE FSW and how they are constructed.

Due to the lack of task prioritization in the DCE OS, timing is less uniform than might be found with the iRMK OS used in the CS. This means the DCE does not provide uniform timing of certain events. An example of this is the execution of a command that takes a long time to complete (e.g. memory dump). A memory dump can impact the timing of other functions like high-voltage ramping. Performing a memory dump while ramping high-voltage might mean the DCE cannot maintain the 100 millisecond timing granularity of HV ramping.

It is worth mentioning, that performing a memory dump while ramping high-voltage will not be performed operationally. Based on the speed at which the DCE FSW executes as well as how the DCE is commanded in an operational environment, this timing non-uniformity is not considered a problem.

For a list of timing issues that result from the DCE OS design, see section 4.2.8.2.1, *Timing Considerations*, later in this document.

4.2.1.2 Tasks

Tasks in the DCE flight software are constructed differently than tasks in the CS flight software. As described in the operating system overview section above, tasks are invoked in round-robin fashion by the executive loop. The executive loop does not manage the switching in and out of tasks, which would take care of retaining a task's state information between invocations. The tasks, therefore, must be capable of retaining their own state information.

To ensure timely progress through the executive loop, tasks must execute quickly and return control to the executive loop. To accomplish this, each task runs to completion, however, it performs only a portion of its total work each time it is invoked. For example, the task that calculates a CRC only calculates one byte of CRC each time it is invoked. The task must be invoked many times in order to calculate the CRC for an entire range of memory. Each time a task is invoked it:

- checks its current state
- determines if its state should change based on the presence of certain conditions (i.e. events)
- performs the work associated with that state
- saves it state and returns control to the executive loop

This means that tasks in the DCE FSW must be constructed as state machines. Each task must manage its own state, keep track of which events have occurred, and determine whether its state should change as a result of a particular event occurring. Each task is designed to record its current state in a data structure and change its state in response to events. Events include things like the arrival of a command word, a one-second timer tick, or a timeout flag. Each task responds to a unique set of events. A complete description of each task, its states, and the events that drive it are covered in detail in the following sections.

4.2.2 Resets and Initialization

The DCE can be reset for a variety of reasons. A reset occurs whenever power is applied to the DCE, or whenever the watchdog timer has not been stroked quickly enough. Each of these resets causes the DCE FSW to start executing from its PROM. This section examines the resets that can occur in the system and how the DCE FSW initializes itself as a result of the various resets.

FUV detector initialization consists of the steps the DCE FSW must go through to properly setup its operating environment. Initialization steps include configuring the various pieces of DCE hardware, initializing global memory areas and starting execution of the various tasks in the system. This section covers the DCE FSW requirements that deal with resetting the system and causing the DCE FSW to re-initialize.

Figure 4.2-1 shows the logic flow associated with resets.

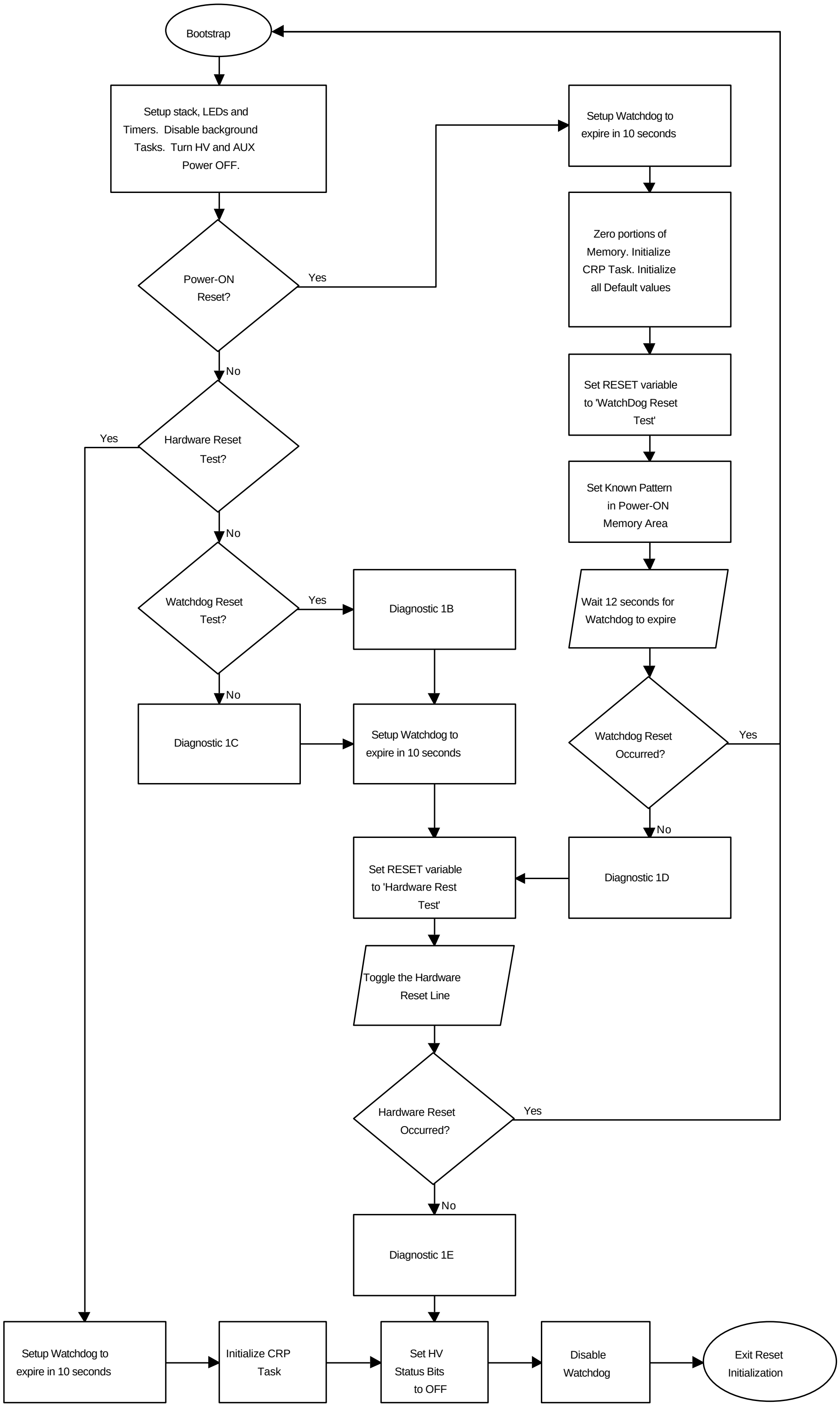

Bootstrap
Setup stack, LEDs and Timers. Disable background Tasks. Turn HV and AUX Power OFF.
Power-ON Reset?
Yes
No
Hardware Reset Test?
Yes
No
Watchdog Reset Test?
Yes
No
Diagnostic 1B
Diagnostic 1C
Setup Watchdog to expire in 10 seconds
Zero portions of Memory. Initialize CRP Task. Initialize all Default values
Set RESET variable to 'WatchDog Reset Test'
Set Known Pattern in Power-ON Memory Area
Wait 12 seconds for Watchdog to expire
Watchdog Reset Occurred?
Yes
No
Diagnostic 1D
Setup Watchdog to expire in 10 seconds
Set RESET variable to 'Hardware Rest Test'
Toggle the Hardware Reset Line
Hardware Reset Occurred?
Yes
No
Diagnostic 1E
Setup Watchdog to expire in 10 seconds
Initialize CRP Task
Set HV Status Bits to OFF
Disable Watchdog
Exit Reset Initialization

Figure 4.2-1. DCE Resets

4.2.2.1 Power-On Reset

A power-on reset (POR) is treated as if the hardware has just been powered on. While this kind of reset is intended to be used only during an actual power-on event, it is also a commandable event that forces the DCE into a default, known state. A power-on reset causes a complete initialization of the software to occur. This is described in detail in the *Complete Initialization Sequence* found in section 4.2.2.4.1.

Interrupts are enabled during a POR, allowing the Command Traffic ISR to process incoming commands even before the Command Handler Task is ready to check the command buffer. The POR purposefully does not write to memory addresses above 0x8800 in order to preserve the state of that memory. During initialization, the watchdog reset and hardware resets are tested.

4.2.2.2 Watchdog Reset

A watchdog reset occurs when the watchdog timer has not been "stroked" (i.e. reset) often enough. If the watchdog count down timer reaches zero before being reset, a watchdog timer reset occurs. This type of reset can be due to a single event upset (SEU), a coding error, or some spurious event. Just like the POR, a watchdog timer reset can be commanded from the ground.

This is a catchall-type reset that is named for its most likely cause: the 8051 built-in watchdog reset timer, which can cause the 8051 to jump to its reset vector. This type of reset assumes that power had been previously applied to the DCE. A watchdog reset causes a minimal initialization of the software to occur. This is described in detail in the *Minimal Initialization Sequence* found in section 4.2.2.4.2.

4.2.2.3 Commanded Resets

A commanded reset occurs as a result of a reset command being sent to the DCE FSW from the CS. There are four possible ways to command the DCE FSW to reset the DCE. Two of the commands result in complete reset of the DCE FSW, and the other two result in a minimal initialization. The four ways to command DCE reset include:

- The CS can reset the DCE by toggling the DCE hardware reset line. This does not result in a command being sent to the DCE FSW. This reset command results in the DCE FSW executing its minimal initialization sequence.

- The CS sends an Initiate Watchdog Reset command to the DCE. This reset command results in the DCE FSW executing a minimal initialization sequence.

- The CS sends a single command word that contains the value 0x80000000. This reset command results in the DCE FSW executing a complete initialization sequence.

- The CS sends an Initiate Power-On Reset command to the DCE. This reset command results in the DCE FSW executing a complete initialization sequence. (NOTE: The DCE forces the complete initialization sequence to run by scrambling the *known pattern* in memory to make it appear as if a powered up of the DCE occurred.)

4.2.2.3.1 Reset Summary Table

Source of Reset	**Initialization Sequence Executed by FSW**
Power-on Reset: power is applied to the DCE hardware	Complete Initialization Sequence
Watchdog Reset: Watchdog timer expires	Minimal Initialization Sequence
Commanded Reset #1: CS sends a command to perform a complete initialization (aka Initiate Power-On Reset command)	Complete Initialization Sequence
Commanded Reset #2: CS sends a command word containing 0x80000000	Complete Initialization Sequence
Commanded Reset #3: CS toggles DCE hardware reset line	Minimal Initialization Sequence
Commanded Reset #4: CS sends a command to perform a minimal initialization (aka Initiate Watchdog Reset command)	Minimal Initialization Sequence

4.2.2.4 Initialization

The DCE FSW initializes itself after any reset. The initialization code can distinguish between a POR and any other reset in order to determine how much internal memory to initialize. After a power-on reset, the DCE FSW initializes its memory to a known state by zeroing certain areas of RAM. After any other type of reset, the memory is preserved so that evidence of activity prior to the reset can be analyzed to determine the cause of the reset.

Some initialization is required after any reset to ensure FSW control flow is fully deterministic. This means that all variables needed to properly operate the FSW are initialized to avoid a continuous reset situation. To ensure a continuous reset situation can never hang the DCE FSW, a watchdog reset counter forces a POR after 255 reset events have occurred.

The DCE FSW uses one of two initialization sequences (complete or minimal) depending on the type of reset that occurred. The two initialization sequences are described in more detail in the following subsections.

4.2.2.4.1 Complete Initialization Sequence

A complete initialization sequence is performed in response to a power-on reset occurring, or when the DCE FSW is commanded to perform a complete reset. The following initialization steps are performed:

- Setup the stack.
- Setup the LEDs.
- Disable background tasks.
- Setup the timers.
- Execute the *DCE Safe Controls* procedure (turn off HV power, enable PROMs, disable door end switch override, turn off actuator A & B power, turn off door motor power, set door direction to STOP, turn off auxiliary power, and disable actuator circuitry).
- Configure the watchdog timer to expire in 10 seconds.
- Clear internal registers.
- Initialize 8K RAM.
- Set known pattern in memory to show that we have been powered up and that we performed the complete initialization sequence.
- Zero variables in 32K RAM.
- Initialize all default values including HV Ramping variables, Current Limit Protection variables, and Count Rate Protection variables.

- Initialize the housekeeping buffer to all zeroes.
- Configure digitizers with default values (this turns off the STIM generator).
- Set reset variable to *watchdog reset test*.
- Setup for background CRC calculations.
- Wait for watchdog reset to occur by performing background CRC calculations.
- Report diagnostic showing a power-on reset occurred.
- Perform the minimal initialization sequence.

4.2.2.4.2 Minimal Initialization Sequence

A minimal initialization sequence is performed in response to a watchdog timer reset occurring, or when the DCE FSW is commanded to perform a minimal reset. The following initialization steps are performed:

- Setup the stack.
- Setup the LEDs.
- Disable background tasks.
- Setup the timers.
- Execute the *DCE Safe Controls* procedure (see Complete Initialization Sequence above).
- Setup watchdog timer hardware.
- Initialize Count Rate Protecton prediction table.
- Disable watchdog timer.
- Start background CRC calculations.
- Set HV State Bits to OFF.

4.2.2.4.3 Executive Loop Initialization Sequence

After reset initialization completes (i.e. after executing either the Complete or Minimal Initialization Sequence mentioned above) but before the executive loop is entered, the executive loop initialization sequence is executed.

Executive loop initialization is performed as part of a reset or whenever a *jump* command is processed. Whenever the DCE FSW is commanded to jump to Boot or Operate, it enters the code image at this point.

The executive loop initialization sequence is as follows:

- Setup the stack.
- Clear the common use (F0) flag.
- Clear the LEDs.
- Clear the op-code in the command buffer.
- Clear the command received registers to put the BUSY bit in a known state.
- Start the background CRC checking task.
- Initialize the HV ramping task variables.
- Update housekeeping with the Operate code version number.
- Enable interrupts.

4.2.2.5 Reuse

Source: FUSE

Design Reuse: 90%

Code Reuse: 90%

4.2.3 Executive Loop

As described previously, the DCE FSW does not use a commercial off-the-shelf operating system. The operating system is built from the ground up using a simple design consisting of an executive loop that invokes a series of functions (called tasks) in a round-robin fashion. Interrupts provide pre-emptive processing capabilities.

Following reset initialization, the DCE FSW passes control to the main routine. The main routine consists of an entry point (named REENTRY) followed by an executive loop that invokes each of the system's tasks in round-robin fashion. There are no timing delays built into the executive loop. It runs as quickly as possible.

The DCE executive loop can be considered a single-threaded process that contains several tasks. Tasks are designed as functions that are called repeatedly, executing a segment of code that may change the task's state. Before the task returns control to the executive loop, it stores all information needed for the next time the routine is called. The tasks perform their processing quickly in order to return control to the executive loop. The passing of data and control among the tasks is controlled by the executive loop. For a description of how a task is structured, see section 4.2.1.2 above.

The executive loop is mentioned here because it is the piece of code that ties all of the tasks together. It provides the basic control flow of the DCE FSW. See figure 3.3-1, the System Level Control Flow Diagram, for a graphic representation of how the executive loop functions within the context of the DCE flight software.

Any timing considerations associated with the executive loop are described below in section 4.2.8.2.1, Timing Considerations.

4.2.3.1 Reuse

Source: FUSE

Design Reuse: 90%

Code Reuse: 50%

The COS project has higher documentation and coding standards which require the executive loop to be restructured and the documentation enhanced.

4.2.4 Interrupt Service Routines

Interrupt service routines provide pre-emptive processing in the DCE FSW. Since the operating system consists of a round-robin loop that executes tasks in a pre-determined order, only interrupts can provide pre-emptive processing. The DCE FSW supports two interrupts: the timer interrupt and the command traffic interrupt. This section describes the interrupt service routine (ISR) associated with each interrupt.

4.2.4.1 Timer Tick ISR

The timer tick interrupt is physically linked to the 8051 timer hardware. The timer hardware is programmed to "tick" approximately 49 times per second. See section 3.2.10, Time Management CSC, for a full description of how time is managed by the DCE.

The timer interrupt triggers the Timer Tick ISR to execute. The ISR sets flags used by some tasks to control timing behavior. The Timer Tick ISR keeps track of the number of times it is invoked, and maintains a one second software timer as a result. The one second timer is used to control the following activities:

- expiration of the door timer

- collection of Counter A & B data
- invocation of the count rate protection software
- invocation of the high-voltage ramping software

The DCE FSW could run without this ISR, however, some timed activities would not function properly. It is also possible to program the 8051 timer for a variety of time intervals which would change the timing characteristics of the flight software. The 8051 timer is programmed to run as described in section 3.2.10 above.

See figure 3.3-1, the System Level Control Flow Diagram, for a graphic representation of how this ISR affects the control flow of the DCE flight software.

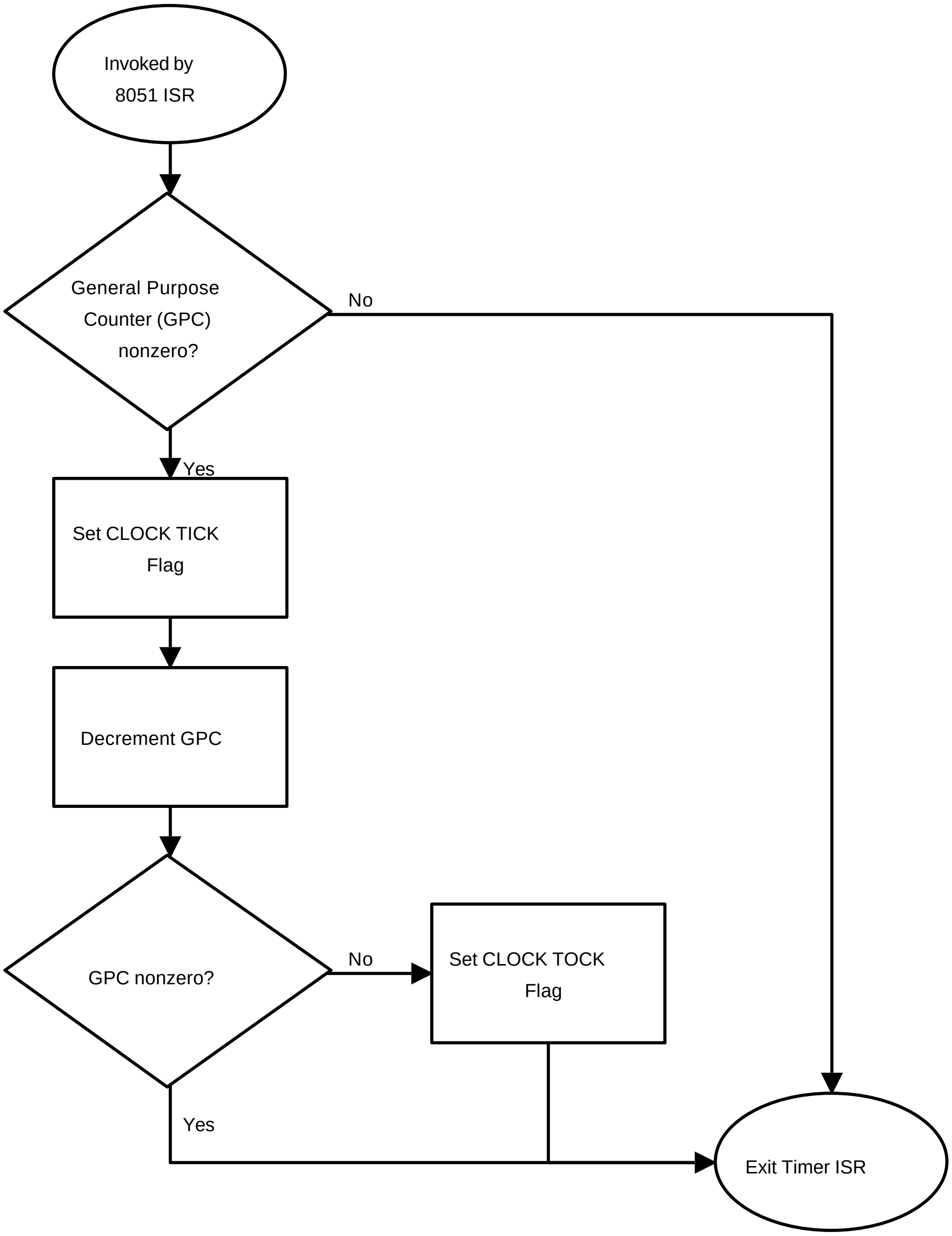
4.2.4.1.1 Processing Logic

This section presents the processing logic of this ISR in a pseudo-programming language format.

```
ISR is invoked when the 8051 timer hardware counts down to zero

General Purpose Counter (GPC) is set to one second when the timer hardware is initialized
Check if GPC is zero when ISR starts (if so, it means we are ignoring the interrupt)
IF (GPC != 0) THEN
   Set CLOCK TICK flag
   DECREMENT GPC
   Check GPC again (if zero, it means one second has elapsed)
   IF (GPC == 0) THEN
     Set CLOCK TOCK flag
     <additional timer flags can be inserted at this point>
   ENDIF
ENDIF
```

4.2.4.1.2 Logic Flow: Timer Tick ISR



4.2.4.1.3 Reuse

Source: New

Design Reuse: 0%

Code Reuse: 0%

FUSE used their commanding rate of 1 command per second as the basis for timing. COS requires finer granularity in its timing to handle count rates that are based on a 1Hz timer, and high-voltage ramping based on a 100 millisecond basis among other potential timing issues.

4.2.4.2 Command Traffic ISR

A command traffic interrupt goes off when a 32-bit command word has been placed into the set of four 8-bit command registers. The DCE has two sets of command registers with an interrupt for each set. This interrupt causes the Command Traffic ISR to execute. At initialization the command traffic interrupt is enabled so that command packets can be received by the DCE FSW before the Command Handler Task has initialized itself.

The main function of the ISR is to move data into memory for processing by the Command Handler Task. As command packets are received, they are processed by this ISR. The command packets are written into RAM in the Command Buffer region based on the address information contained in the command packet. After all of the command packets for a particular command are received, the Command Handler Task processes the command.

This ISR also provides a means of resetting the software if the executive loop should hang for any reason. Any command packet that consists of a one in the most significant bit with zeros in all other bits causes the ISR to command a reset of the DCE.

See figure 3.3-1, the System Level Control Flow Diagram, for a graphic representation of how this ISR affects the control flow of the DCE flight software.

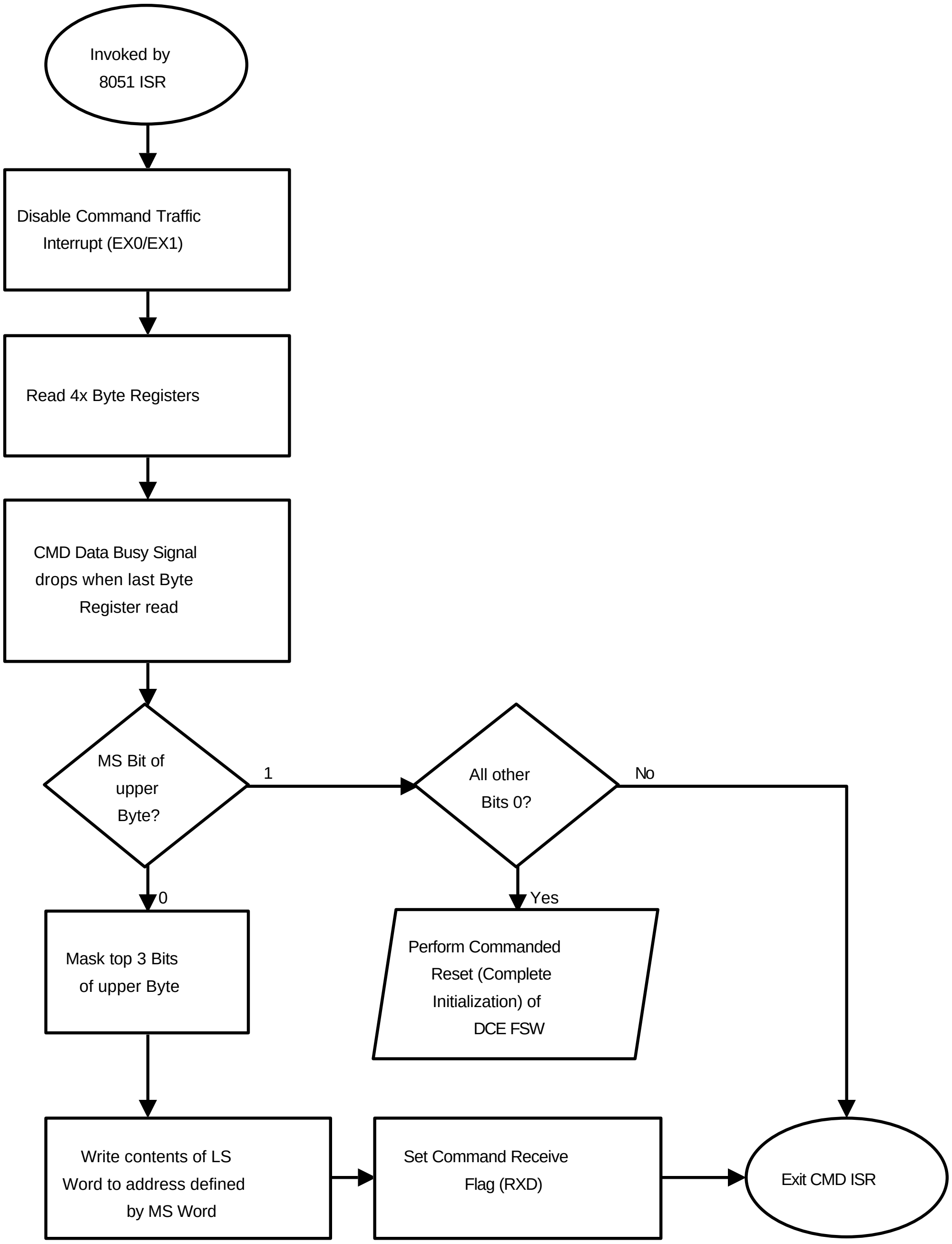
4.2.4.2.1 Processing Logic

This section presents the processing logic of this ISR in a pseudo-programming language format.

```
ISR is invoked by 8051 interrupt when a 32-bit command value has been serial-shifted into
   the four 8-bit DCE command registers

Disable Command Traffic interrupt (EX0 or EX1)
Get 4 bytes of command data
Check to see if the MSB has the most significant bit set
IF (MS Bit == 1) THEN
   IF (remaining 31-bits are all zeros) THEN
      Perform a Commanded Reset (performs complete initialization)
   ENDIF
ELSE
   Mask off MS 3-bits of command word
   Get the destination address from the MS 16-bits of the command word
   Set the command received (RXD) flag
   Write LS 16-bits of command word to the destination address
ENDIF
Enable interrupts
```

4.2.4.2.2 Logic Flow: Command Traffic ISR



4.2.4.2.3 Reuse

Source: New

Design Reuse: 0%

Code Reuse: 0%

FUSE did not have a Command Traffic ISR. This is new to COS due to hardware changes in how command information is transferred to the DCE.

4.2.5 Tasks

This section describes the various tasks that comprise the DCE flight software. The concept of a task is fully described in section 4.2.1.2. For details on how tasks are structured and how they function, the reader is referred to that section. This section describes each task in terms of its functionality.

See figure 3.3-1, the System Level Control Flow Diagram, for a graphic representation of how tasks function within the context of the DCE flight software.

4.2.5.1 Command Handler Task

The DCE FSW provides a method for the successful and timely reception and interpretation of command packets from the MEB. The Command Handler Task is responsible for processing all commands entering the DCE subsystem. Commands are received from the Control Section on a once-per-second basis.

The Command Traffic ISR accepts command packets and upload data from the MEB and stores the information in the Command Buffer region of RAM. Once there, the Command Handler Task can process the command. The task polls the memory location that stores the command op-code. Once an op-code is detected, the command is processed.

The task first checks for a valid command format. If the command is valid, it is executed. The task copies the contents of the Command Buffer to an area in the housekeeping region, thus preserving the parameters for use by the command. It also provides a command echo in housekeeping.

Executing the command means the Command Handler Task calls the appropriate command function. When the DCE is in Boot mode, only a subset of all DCE commands can be executed. More specifically, all commands are available in Boot *except* for the commands that enable hardware functions to the HV and Door.

The Command Handler Task also sets a flag used by the Housekeeping Task to indicate that a housekeeping response is required. Every command op-code, valid or not, elicits a response from the DCE. If the command is invalid and not executed, the DCE reports the attempt in housekeeping. The command response may be interpreted as readiness for a new command packet at the DCE. A housekeeping packet contains enough information to determine whether the command packet was executed properly. The task may also produce a housekeeping packet that contains download data in response to a download command.

When the task returns control to the executive loop, it is always in the same state. It is always searching for the presence of a command op-code.

The Command Handler Task has no DCE commands associated with it that cause it to perform different types of processing. The Command Handler Task is responsible for processing all commands and invoking the appropriate command function, however, none of those command functions alter its behavior.

For a complete list of all DCE commands, see section 4.2.8.2.1. For a detailed description of the DCE commands and their parameters, see Appendix B of the COS DM-05 document.

4.2.5.1.1 Processing Logic

This section presents the processing logic of this task in a pseudo-programming language format.

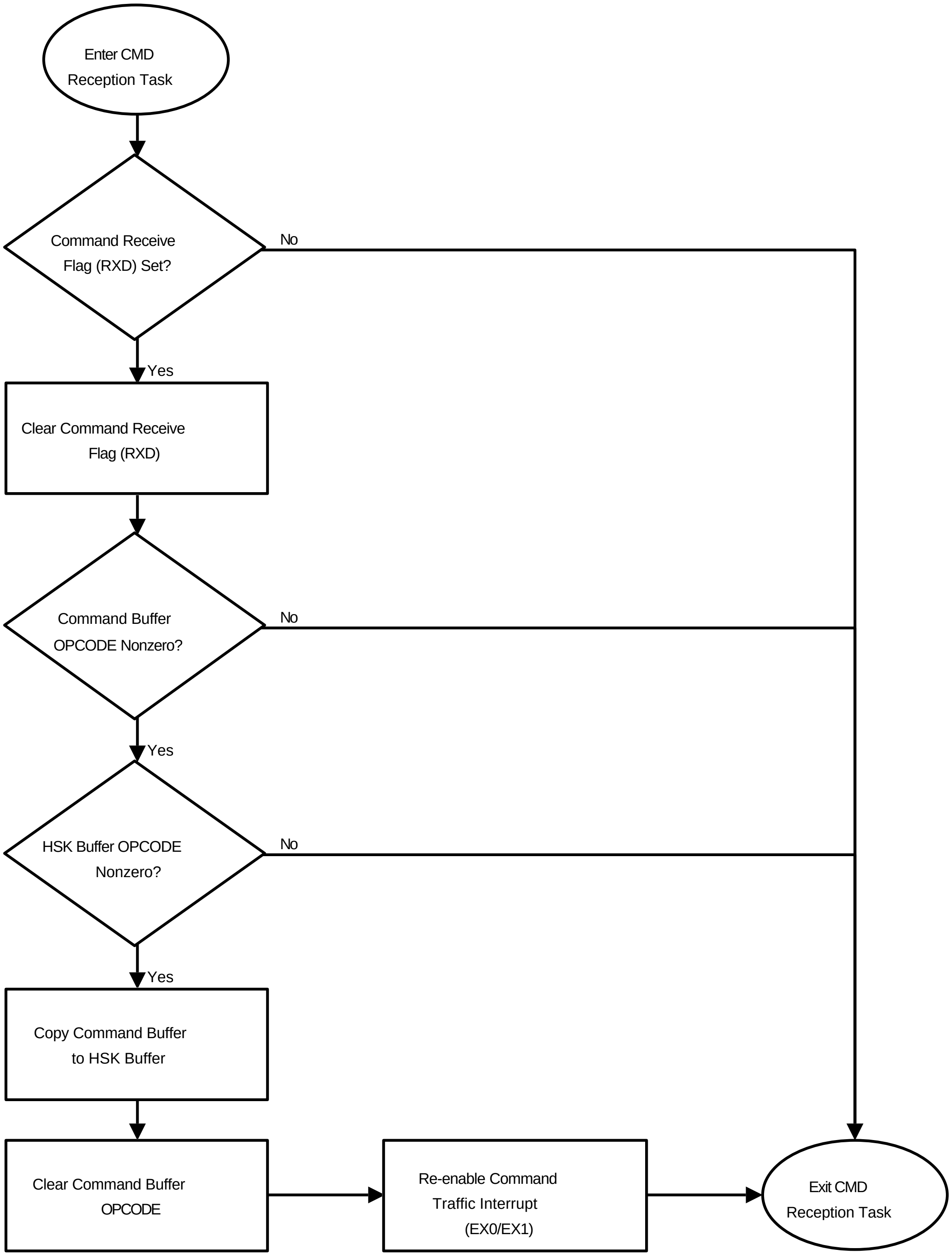
```
Command Reception Sub-Task
Check to see if a command word has been received (the RXD flag is set in the ISR)
IF (RXD flag == SET) THEN
   Clear the command received (RXD) flag
   Check to see if a valid op-code was entered into the command buffer
   IF (command buffer op-code != 0) THEN
      Check to see if the command can be copied to the housekeeping buffer
      IF (housekeeping buffer op-code != 0) THEN
         Copy the contents of the command buffer to the housekeeping buffer
         Reset the command buffer op-code to 0.
         Enable the Command Traffic interrupt (EX0 or EX1)
      ENDIF
   ENDIF
ENDIF
```

Command Processing Sub-Task

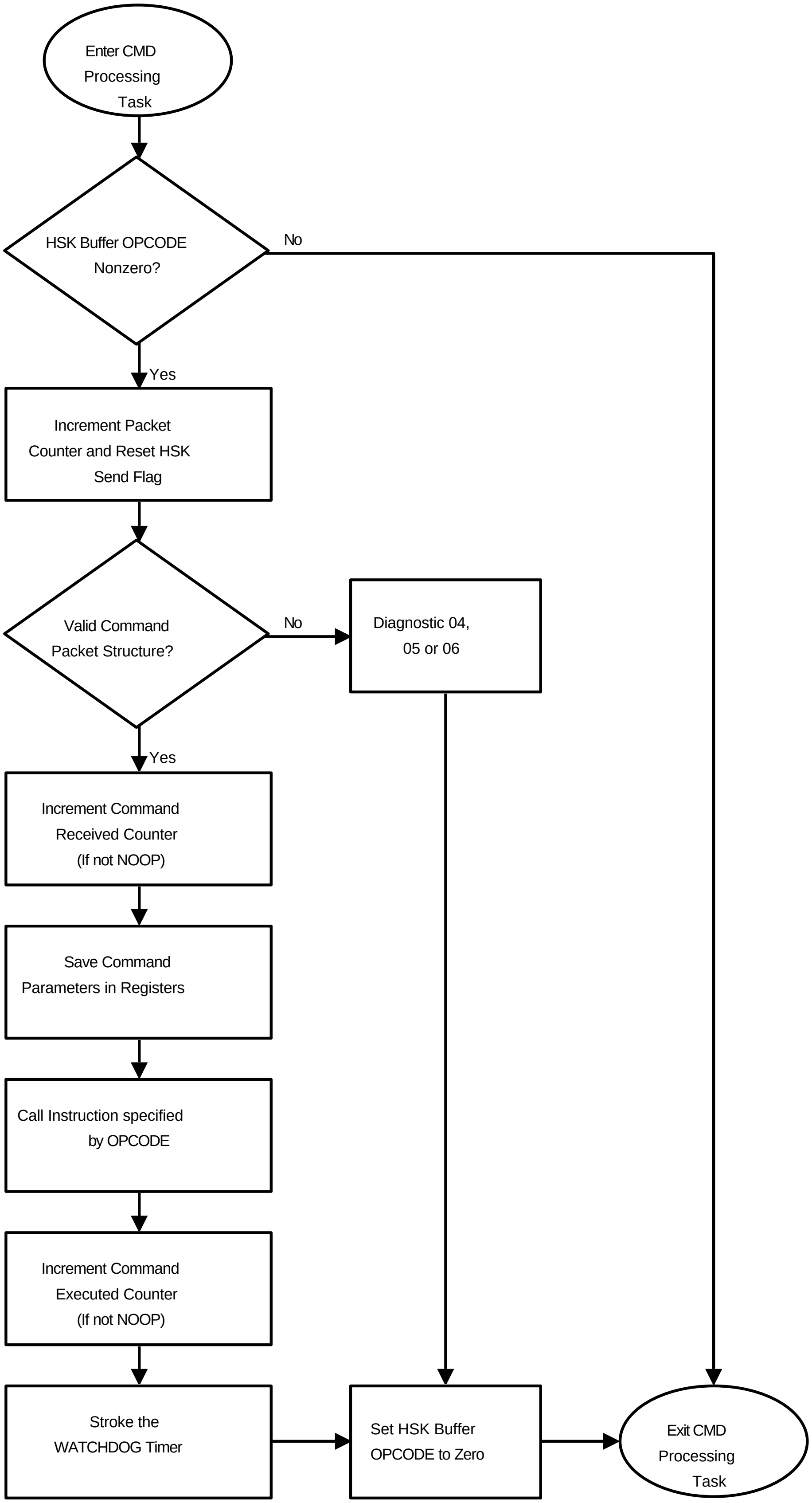
```
Check to see if a valid op-code was entered into housekeeping buffer
IF (housekeeping buffer op-code != 0) THEN
   INCREMENT packet counter
   Send housekeeping flag = SET
   Check command packet for a valid command structure (command must have
      a duplicate op-code, most significant bit of the command must be 1, all
      parameters must inverse value)
   IF (command has a valid structure) THEN
      IF (op-code != NOOP) THEN
         INCREMENT commands received count
      ENDIF
      Copy command parameters from housekeeping buffer into registers
      CALL the appropriate command function based on the op-code using
         registers containing command parameters
      IF (op-code != NOOP) THEN
         INCREMENT commands executed count
      ENDIF
      Stroke the watchdog timer
   ELSE
      Issue diagnostic
   ENDIF
   Set op-code in housekeeping buffer to 0
ENDIF
```

4.2.5.1.2 Logic Flow: Command Handler Task

Command Reception Sub-Task


Enter CMD Reception Task
Command Receive Flag (RXD) Set?
No
Yes
Clear Command Receive Flag (RXD)
Command Buffer OPCODE Nonzero?
No
Yes
HSK Buffer OPCODE Nonzero?
No
Yes
Copy Command Buffer to HSK Buffer
Clear Command Buffer OPCODE
Re-enable Command Traffic Interrupt (EX0/EX1)
Exit CMD Reception Task

Command Processing Sub-Task


Enter CMD Processing Task
HSK Buffer OPCODE Nonzero?
No
Yes
Increment Packet Counter and Reset HSK Send Flag
Valid Command Packet Structure?
No
Diagnostic 04, 05 or 06
Yes
Increment Command Received Counter (If not NOOP)
Save Command Parameters in Registers
Call Instruction specified by OPCODE
Increment Command Executed Counter (If not NOOP)
Stroke the WATCHDOG Timer
Set HSK Buffer OPCODE to Zero
Exit CMD Processing Task

4.2.5.1.3 Reuse

Source: FUSE

Design Reuse: 80%

Code Reuse: 80%

Double buffering of commands is added to the COS design to avoid overwriting an executing command.

4.2.5.2 Count Rate Protection Task

The Count Rate Protection Task checks the digitizer counters on a periodic basis for excessive counts. If it detects excessive counts over a specified interval, it forces the HV to the HV_LOW state. This task asserts direct control over the HV state when a Count Rate limit is exceeded. It causes the HV Ramping Task to stopped if it is running.

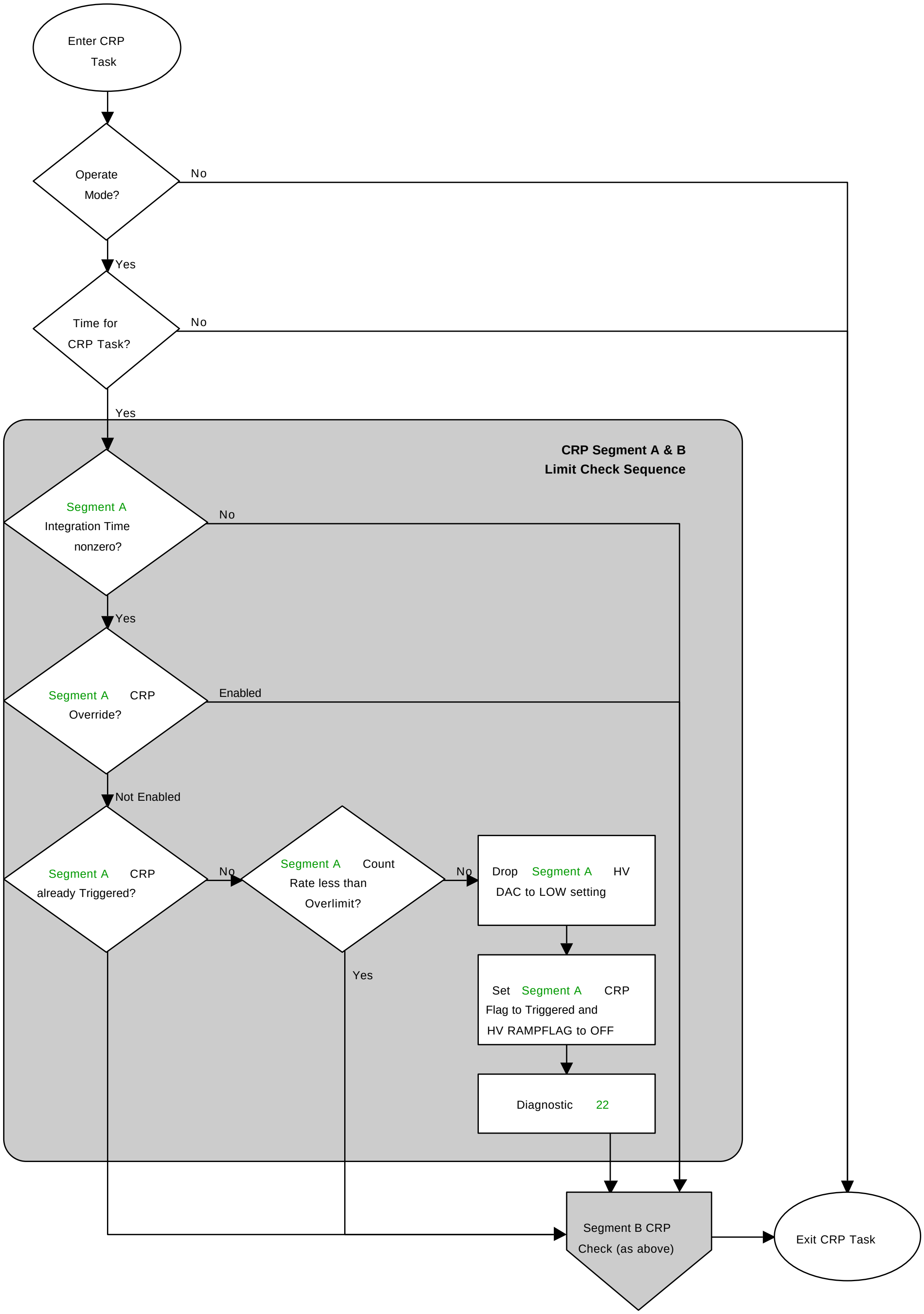
Default parameters for the Count Rate Protection Task are loaded during power-on reset initialization. The parameters can then be modified via DCE commands. Every second, this task reads the digitizer counters and compares the accumulated counts against a table in memory. If the accumulated counts exceed any value in the table, the protection function is triggered.

4.2.5.2.1 Processing Logic

This section presents the processing logic of this task in a pseudo-programming language format.

```
Check to see if periodic timer has elapsed (0.1 second timer is used to determine when
   to invoke the Count Rate Protection (CRP) Task)
IF (periodic timer has elapsed) THEN
   Check to see if integration time of global rate check
   FOR EACH (segment A & B)
      IF (integration time[segment] != 0) THEN
         Check if override status is set
         IF (override == CLEAR) THEN (NOTE: no command exists to set or clear the override)
            Check if the CRP has already triggered due to excessive global counts
            IF (CRP triggered[segment]) == FALSE THEN
               Check to see if global counts for segment are over specified limit
               IF (segment is in an overlight condition) THEN
                  Set the HV of the segment to low
                  CRP triggered[segment] = TRUE
               ENDIF
            ENDIF
         ENDIF
      ENDIF
   ENDFOR
ENDIF
```

4.2.5.2.2 Logic Flow: Count Rate Protection Task


Enter CRP Task
Operate Mode?
No
Yes
Time for CRP Task?
No
Yes
CRP Segment A & B Limit Check Sequence
Segment A Integration Time nonzero?
No
Yes
Segment A CRP Override?
Enabled
Not Enabled
Segment A CRP already Triggered?
No
Segment A Count Rate less than Overlimit?
No
Yes
Drop Segment A HV DAC to LOW setting
Set Segment A CRP Flag to Triggered and HV RAMPFLAG to OFF
Diagnostic 22
Segment B CRP Check (as above)
Exit CRP Task

4.2.5.2.3 Reuse

Source: FUSE

Design Reuse: 90%

Code Reuse: 80%

COS uses count rates rather than accumulated counts (like in FUSE) in the count rate protection algorithm. The ability to manage the HV State Bits needs to be added. Logic needs to be added that determines whether the code is running from PROM (Boot code) or from 32K RAM (Operate code).

4.2.5.3 Current Limit Protection Task

The Current Limit Protection Task checks the HV and Aux current readings for over-limit conditions. It reads the current values, records the information as data samples, and determines if an over-limit condition occurred. In the event of an over-limit, the task reports a diagnostic and turns off HV, if warranted.

This task runs as fast as possible to monitor the power current levels. It reads the analog-to-digital converter (ADC) for HV current and Aux current. It saves the samples in RAM so that they can be downloaded for analysis if necessary. This task is capable of reporting HV glitches without turning off the HV. It is also capable of recognizing a sustained over-limit condition. If the current level remains out of limits, the task powers off the HV and reports the over-limit event in housekeeping.

If a single over-limit event is detected, this task reports a diagnostic. Subsequent over-limit samples are counted but no diagnostic is posted until the consecutive out-of-limits count is reached. If a power current value within limits is read, the consecutive out-of-limits counter is reset to zero. When the consecutive out-of-limits count is exceeded, this task turns off power to the HV supply. After a limit violation has occurred, there is no autonomous method for turning HV back on.

Default parameters for the Current Limit Protection Task are loaded during power-on reset initialization. The parameters can then be modified via DCE commands.

IMPORTANT: This task runs as fast as possible, however, there is no requirement on this task to run fast enough to handle any sort of "thermal runaway" problem. Based on the design of the OS, this task does not run uniformly. There could be a 200ms gap over a 1000ms time range due to the way the Housekeeping Task works.

4.2.5.3.1 Processing Logic

This section presents the processing logic of this task in a pseudo-programming language format.

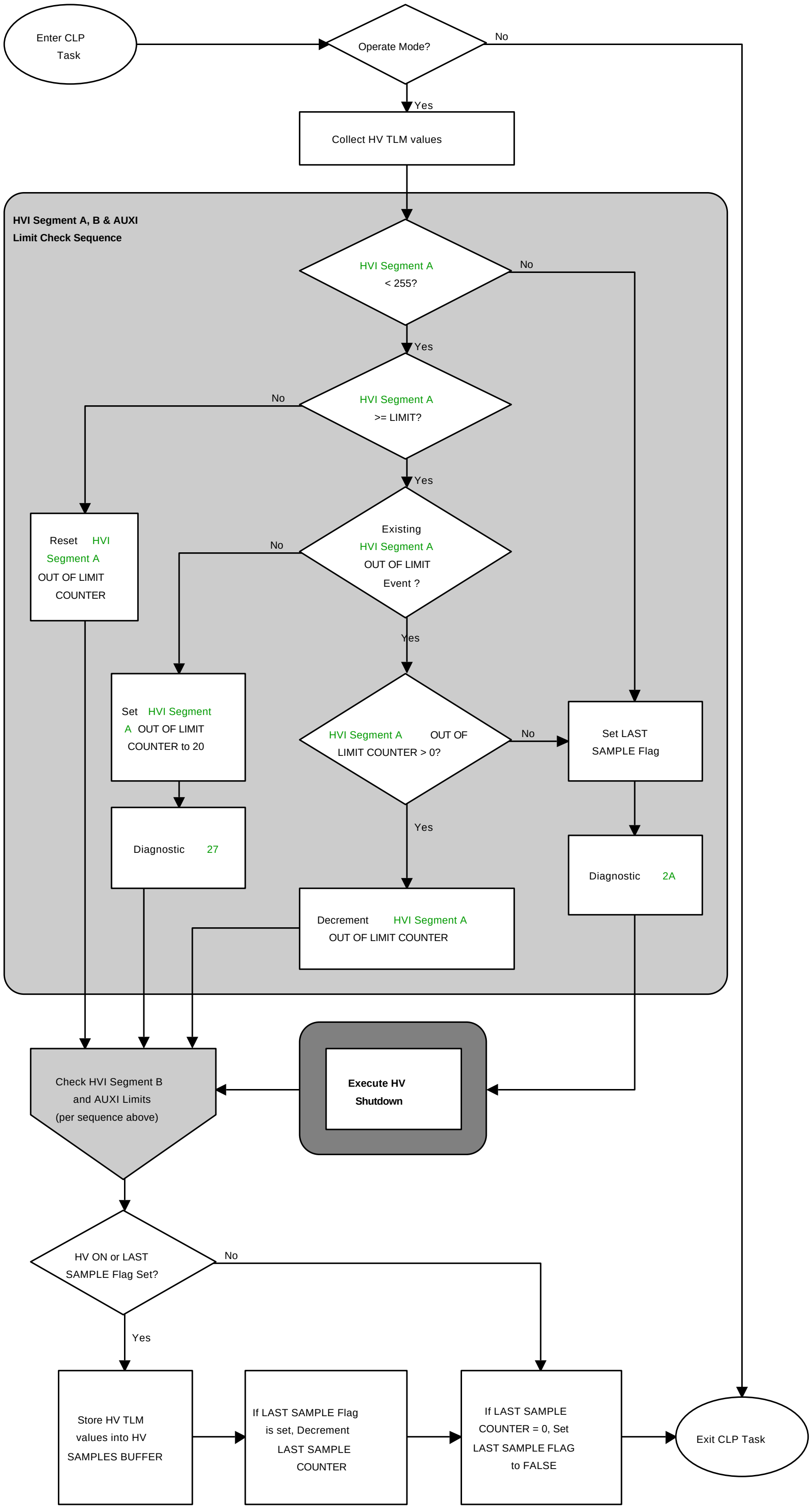
```
Increment offset into Sample Buffers
Get value of current level of segment A
IF (current value is "pegged" (maxed out)) THEN
   Execute high voltage shutdown sequence
      (stop ramping, turn off HV, issue diagnostic, last sample flag = SET)
ELSE
   IF (current value >= limit value) THEN
     IF (first OOL condition) THEN
        Set consecutive out-of-limits (COOL) count = 20
        Issue a diagnostic that an initial limit break occurred
     ELSE
        IF (COOL count == 0) THEN
           Execute high voltage shutdown sequence (see above)
        ELSE
           DECREMENT COOL count
        ENDIF
     ENDIF
   ELSE
      COOL count = 0
   ENDIF
ENDIF

IF (high voltage == ON) OR (last sample count > 0) THEN
   Record value in sample buffer and histogram
ENDIF

Check HV current on segment B (same as segment A)
Check current on Aux Power (same as segment A)

IF (last sample flag == SET) THEN
   IF (last sample count == 0) THEN
      last sample count = 20
   ENDIF
   DECREMENT last sample count
   IF (last sample count == 0) THEN
      last sample flag = CLEAR
   ENDIF
ENDIF
```

4.2.5.3.2 Logic Flow: Current Limit Protection Task


Enter CLP Task
Operate Mode?
No
Yes
Collect HV TLM values
HVI Segment A, B & AUXI Limit Check Sequence
HVI Segment A < 255?
No
Yes
HVI Segment A >= LIMIT?
No
Yes
Reset HVI Segment A OUT OF LIMIT COUNTER
Existing HVI Segment A OUT OF LIMIT Event ?
No
Yes
Set HVI Segment A OUT OF LIMIT COUNTER to 20
HVI Segment A OUT OF LIMIT COUNTER > 0?
No
Set LAST SAMPLE Flag
Diagnostic 27
Yes
Diagnostic 2A
Decrement HVI Segment A OUT OF LIMIT COUNTER
Check HVI Segment B and AUXI Limits (per sequence above)
Execute HV Shutdown
HV ON or LAST SAMPLE Flag Set?
No
Yes
Store HV TLM values into HV SAMPLES BUFFER
If LAST SAMPLE Flag is set, Decrement LAST SAMPLE COUNTER
If LAST SAMPLE COUNTER = 0, Set LAST SAMPLE FLAG to FALSE
Exit CLP Task

4.2.5.3.3 Reuse

Source: FUSE

Design Reuse: 100%

Code Reuse: 90%

The ability to manage the HV State Bits needs to be added. Logic needs to be added that determines whether the code is running from PROM (Boot code) or from 32K RAM (Operate code).

4.2.5.4 CRC Checking Task

The CRC Checking Task computes a 16-bit cyclic redundancy check (CRC) value that characterizes a defined region of DCE memory. The task reports changes in a memory region when the newly computed CRC for that region differs from the previously computed value. This task reports changes as diagnostic messages and does not attempt to correct any memory errors.

This task is a background task and does not attempt to calculate the CRC for an entire region each time it is invoked. It processes one byte of data through the CRC algorithm every time the task is called. After the CRC for a memory region has been computed, the task checks to see that the value matches the CRC value stored in memory. If the values differ, the task reports a diagnostic and stores the newly computed CRC in housekeeping. Computed CRC values have designated locations in housekeeping memory.

The task is initialized to point to the first memory region in a list of memory regions to be checked. In general, the task completes its check of a region and automatically starts computation of the next region in the list. After all regions in the list have been checked, it starts back at the beginning of the list.

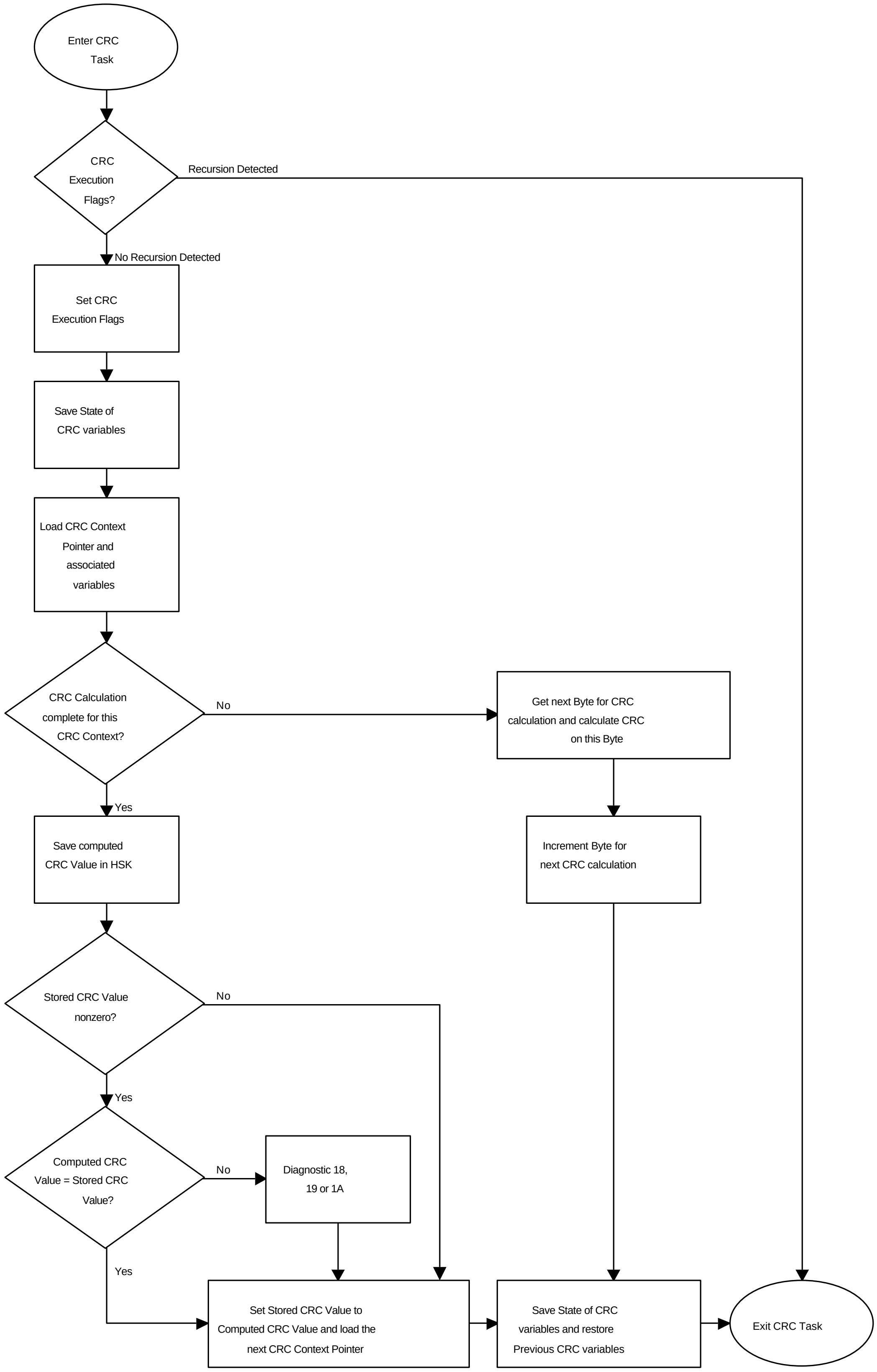
The Compute CRC command causes the CRC Task to stop its current CRC calculation and start a new one. This command causes the CRC Task to calculate a CRC for a user-specified range of DCE memory. Once that CRC value is calculated, the task re-initializes itself and starts checking from the first memory region in the list.

4.2.5.4.1 Processing Logic

This section presents the processing logic of this task in a pseudo-programming language format.

```
Check anti-recursion flag to verify this function was not called by another function
IF (not recursively called) THEN
   SET anti-recursion flag
   Save off current CRC variables
   Load CRC context pointer and variables for the region we are calculated
   Check if the CRC is done for this region
   IF (CRC done) THEN
      Save computed CRC in housekeeping
      Check computed CRC against the stored value
      IF (saved image CRC != 0) THEN
         Check computed CRC against the stored CRC
         IF (computed CRC != saved image CRC) THEN
            Report diagnostic
         ENDIF
      ENDIF
      saved image CRC = computed CRC
      Load next CRC context pointer and variables
   ELSE
      Get next byte of data
      Compute 1 byte of CRC
      Increment pointer to next byte of data
   ENDIF
   Save current CRC variables
   Restore previous CRC variables
ENDIF
```

4.2.5.4.2 Logic Flow: CRC Checking Task


Enter CRC Task
CRC Execution Flags?
Recursion Detected
No Recursion Detected
Set CRC Execution Flags
Save State of CRC variables
Load CRC Context Pointer and associated variables
CRC Calculation complete for this CRC Context?
No
Get next Byte for CRC calculation and calculate CRC on this Byte
Yes
Save computed CRC Value in HSK
Increment Byte for next CRC calculation
Stored CRC Value nonzero?
No
Yes
Computed CRC Value = Stored CRC Value?
No
Diagnostic 18, 19 or 1A
Yes
Set Stored CRC Value to Computed CRC Value and load the next CRC Context Pointer
Save State of CRC variables and restore Previous CRC variables
Exit CRC Task

4.2.5.4.3 Reuse

Source: FUSE

Design Reuse: 100%

Code Reuse: 90%

The code needs accommodate the calculation of a CRC over a user-specified range in addition to its preset ranges.

4.2.5.5 *Background Built-in Test Task*

The Background Built-in Test Task performs a number of sanity checks on the DCE and reports anomalies as diagnostic messages. One test is a check on the 8051 stack. The task reports when an unexpected amount of stack space is used. This is necessary because the stack is the only dynamically allocated memory in the DCE system.

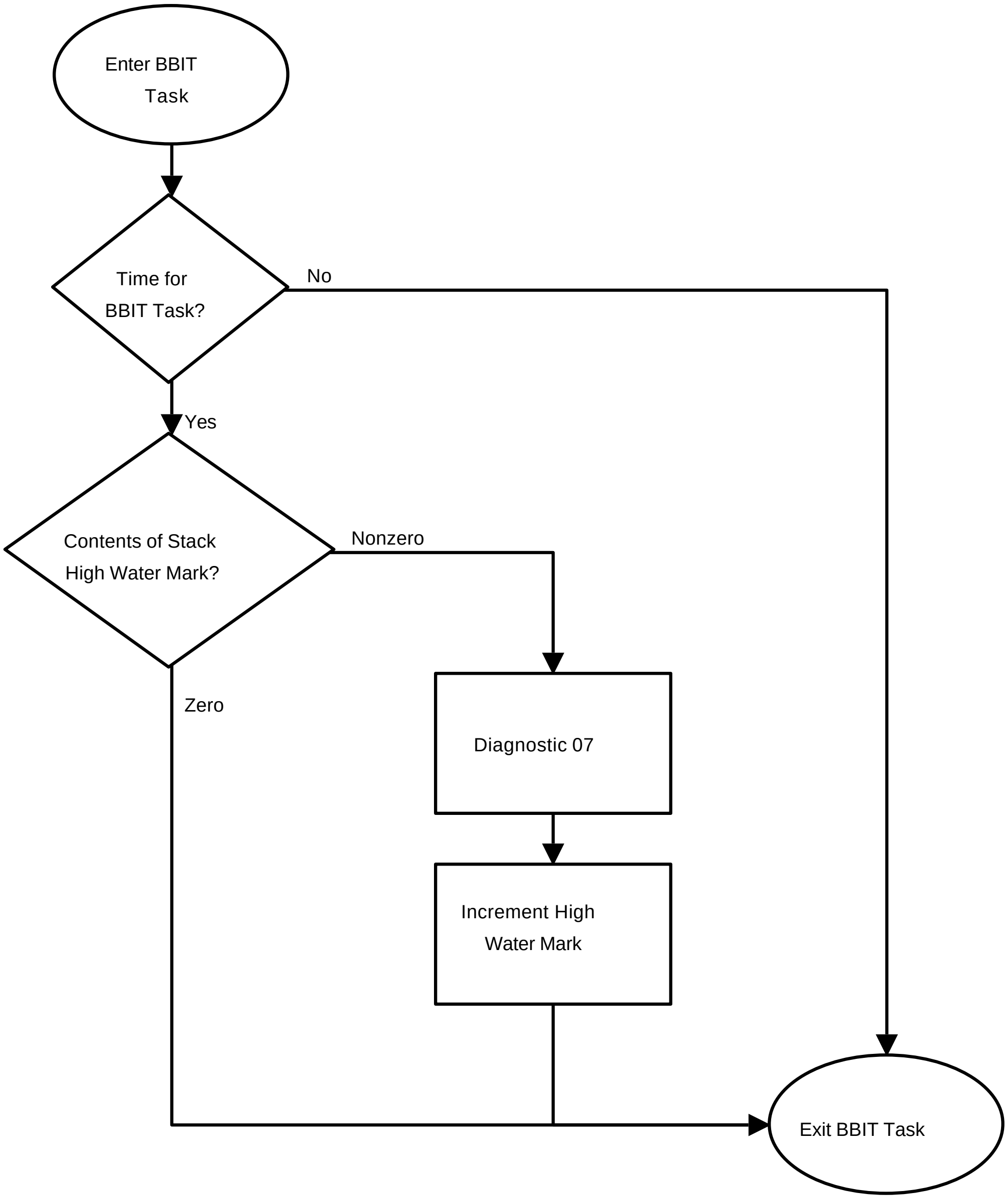
The task uses a high-water mark in memory to determine if the stack has overgrown its allocated area. The high-water mark is a pointer into memory. The stack is initialized with a known pattern. As this task runs, it checks the contents of the high-water mark against the known pattern. If the known pattern is present, the stack is within its allowable area. If the known pattern is not present, it means the stack has grown beyond its allowable limits. In this case, the high-water mark is incremented and a diagnostic is reported in housekeeping.

4.2.5.5.1 Processing Logic

This section presents the processing logic of this task in a pseudo-programming language format.

```
ASSUMPTION: stack area was initialized by reset code and high-water mark set by
            task initialization code
Check to see if periodic timer has elapsed (1.0 second timer is used to determine when
   to invoke the BBIT Task)
IF (periodic timer has elapsed) THEN
   IF (contents of stack at high-water mark != 0) THEN
      Report diagnostic
      INCREMENT high-water mark
   ENDIF
ENDIF
```

4.2.5.5.2 Logic Flow: Background Built-in Task



4.2.5.5.3 Reuse

Source:	FUSE
Design Reuse:	100%
Code Reuse:	90%

Modification of the executive loop design forced a change in how the Background Built-in Test Task accesses the stack.

4.2.5.6 High Voltage Ramping Task

The High Voltage Ramping Task is responsible for increasing the MCP voltage on the detector in small increments. When ramping the detector, the task starts at the current HV level and increments the voltage until it reaches the commanded nominal HV value. The HV ramp rate is used to pause for a set amount of time between successive HV commands.

Before the HV ramping can begin, the ground must setup internal parameters used in the ramping process: the low HV value, the nominal HV value, and the HV ramp rate. The ground must also set the maximum voltage level (HV MAX) enforced by the hardware. This ensures that the flight software cannot command the high voltage to a level higher than the maximum level allowed by the hardware.

With the configuration complete, high voltage ramping can being. High voltage ramping is initiated via a command from the ground. The task uses the previously set parameters to determine how to perform the ramping. The task writes to the HV DAC registers to command small increments in high voltage. After each DAC command, the task waits for the specified settle time before sending the next HV command. This process continues until the HV has reached the nominal HV value. The task does not change the HV MAX register.

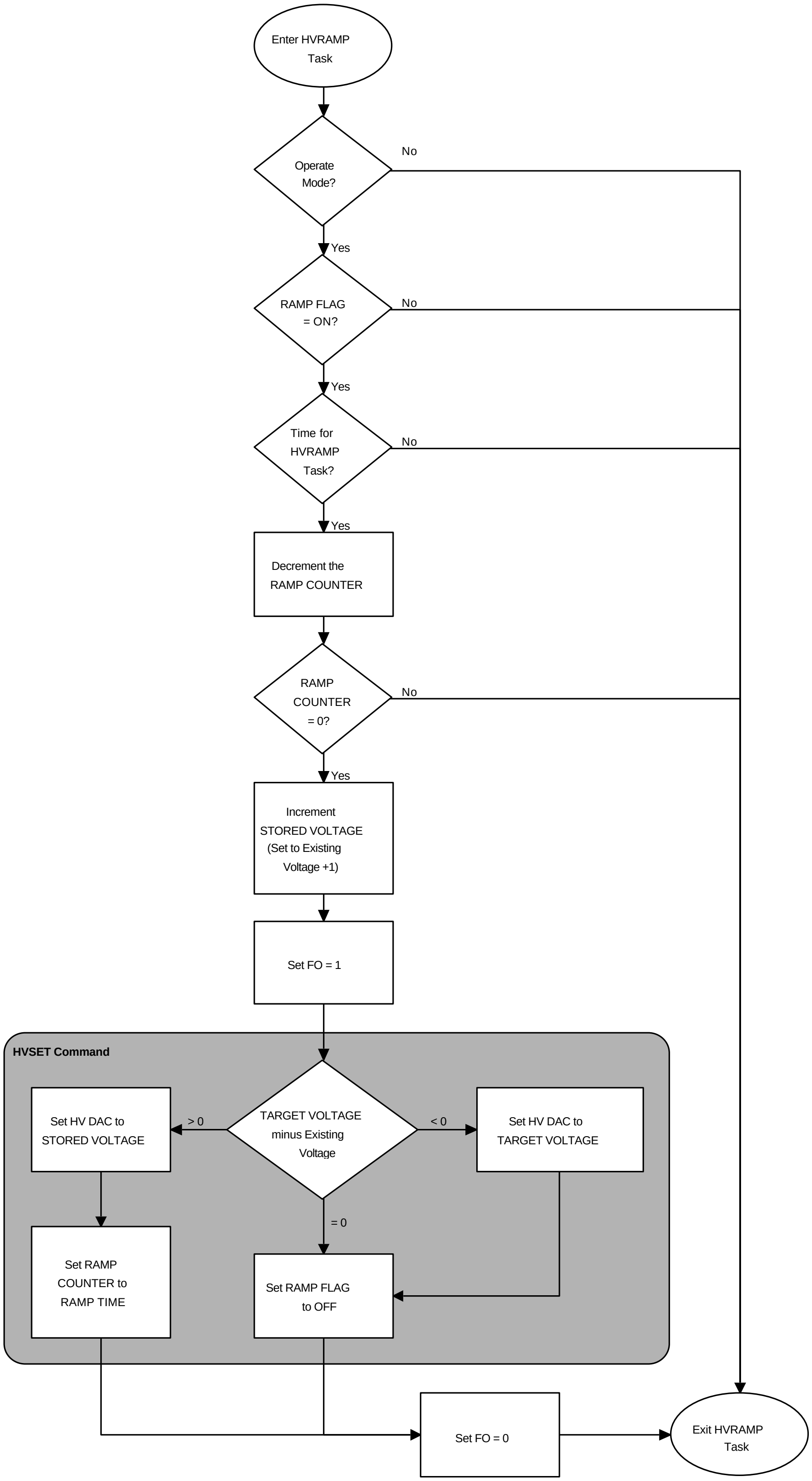
The High Voltage Ramping Task maintains a task state variable that tracks whether ramping is in progress or not. The HVSET and HVSTATE commands set the ramp flag. The Count Rate Protection Task and Current Limit Protection Task can stop ramping by clearing the ramp flag.

4.2.5.6.1 Processing Logic

This section presents the processing logic of this task in a pseudo-programming language format.

```
Initial task state of Ramping Task is set in the HVSET command (the task state can be
   changed by the Count Rate Protection Task)
Check to see if the periodic timer has elapsed (0.1 second timer is used to determine when
   to invoke the HV Ramp Task)
IF (periodic timer has elapsed) THEN
   Check to see if there is a ramp up is in progress -or-
      if we are setting HV to a lower setting
   IF (ramp flag == RAMPING OR setting HV to lower setting) THEN
      Check the ramp counter to see if it is time to change the HV setting
         (the ramp counter is a count down counter that is initialized to RAMP TIME)
      DECREMENT ramp counter
      IF (ramp counter == 0) THEN
         Compare the target voltage against the current voltage
         IF (target voltage > current voltage) THEN
            Store current voltage + 1
            CALL HVSET to command DAC to stored voltage
            (HVSET command sets the DAC and resets the ramp counter to RAMP TIME)
         ELSE IF (target voltage == current voltage) THEN
            /* we have achieved the new HV state set in the STATE or SET command */
            ramp flag  = NOT_RAMPING
         ELSE /* target voltage < current voltage */
            CALL HVSET to command DAC to target voltage
            /* we have achieved the new HV state set in the STATE or SET command */
            ramp flag = NOT_RAMPING
         ENDIF
      ENDIF
   ENDIF
ENDIF
```

4.2.5.6.2 Logic Flow: High-Voltage Ramping Task


Enter HVRAMP Task
Operate Mode?
No
Yes
RAMP FLAG = ON?
No
Yes
Time for HVRAMP Task?
No
Yes
Decrement the RAMP COUNTER
RAMP COUNTER = 0?
No
Yes
Increment STORED VOLTAGE (Set to Existing Voltage +1)
Set FO = 1
HVSET Command
Set HV DAC to STORED VOLTAGE
> 0
TARGET VOLTAGE minus Existing Voltage
< 0
Set HV DAC to TARGET VOLTAGE
= 0
Set RAMP COUNTER to RAMP TIME
Set RAMP FLAG to OFF
Set FO = 0
Exit HVRAMP Task

4.2.5.6.3 Reuse

Source: FUSE

Design Reuse: 90%

Code Reuse: 75% (never tested at the system level, and not currently in use by FUSE)

The HV Ramping Task dynamic range has changed by a factor of 8 or 16. Logic needs to be added that determines whether the code is running from PROM (Boot code) or from 32K RAM (Operate code).

4.2.5.7 Door Task

The Door Task performs periodic checks of the door timer and the actuator latch. If the condition warrants, this task performs a Door Stop. The task performs two basic functions. It monitors the 3-minute door timer to see if the timer has elapsed, and it monitors the actuator latch to see if it has changed states. The door is autonomously commanded to stop by this task either three minutes after the last door command is sent to the DCE, or two seconds after the actuator latch has changed states.

The task periodically checks the motor timer to determine if the 3-minute timer has elapsed. If the timer has not elapsed, the task checks the Aux current against its allowable limits. If the Aux current is out of limits (for even one sample) a diagnostic message is reported and a *Door Stop* is performed. A Door Stop consists of turning off door motor power, turning off actuator power, disabling the software enable for door commanding, turning off switches used to control direction, and resetting the motor timer. If the Aux current is within its limits, the task decrements the motor timer and checks to see if the motor timer has expired. If it has, a Door Stop is performed.

The task also periodically monitors the actuator latch to see if it changed states. The first time the task runs, it initializes the state of the actuator latch. The task reads the MUX 8-bit register to get the state of the actuator latch. Every subsequent time the task runs, it checks to see if the actuator latch has changed states. If the task detects a state change, it reports a diagnostic and performs a Door Stop.

This task can maintains a state variable that indicates the state of the door mechanism. The states it tracks include: Aux power off, DoorStoppedSomewhere DoorStoppedOpen, DoorStoppedClosed, DoorEnabledSomewhere, DoorEnabledOpen, DoorEnabledClosed, DoorOpening, DoorOpen, DoorClosing, DoorClosed. All these states are detected and reported in housekeeping.

4.2.5.7.1 Processing Logic

This section presents the processing logic of this task in a pseudo-programming language format. The Door Task actually consists of two separate sub-tasks. Since the two sub-tasks are logically linked, they are included here together. The two sub-tasks consist of the Door Timer Sub-Task and Actuator Latch Checking Sub-Task.

Door Timer Sub-Task

```
/* value of Motor Timer is set in the Door commands */
Check to see if periodic timer has elapsed (1.0 second timer is used to determine when
   to invoke the Door Timer Sub-Task)
IF (periodic timer has elapsed) THEN
   Check to see if motor timer is zero. (If timer is zero, no valid door command
      was issued. If timer is non-zero, a valid door command was received.)
   IF (motor timer != 0) THEN
      Toggle LEDs (for GSE test purposes)
      IF (aux current limit != 0) AND (aux current >= aux current limit) THEN
         Report diagnostic
         Perform a door stop (see below)
      ELSE
```

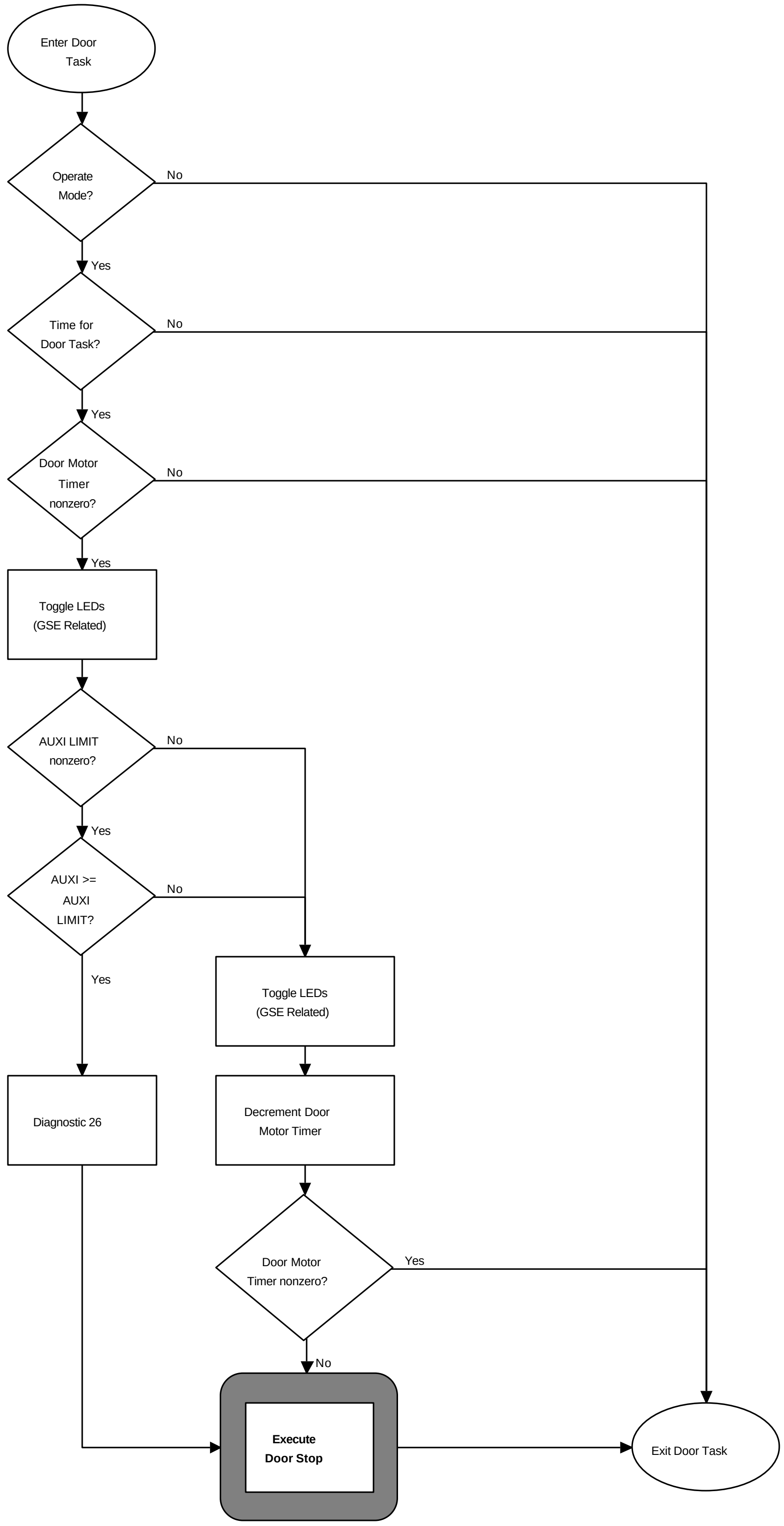
```
         Toggle LEDs (for GSE test purposes)
         DECREMENT motor timer
         IF (motor timer == 0) THEN
            Perform a door stop (turn off bridge switches (direction),
                                 motor power, arm flag (software enable),
                                 disable motor safety override,
                                 turns off actuator power, reset motor timer)
         ENDIF
      ENDIF
   ENDIF
ENDIF
```

Actuator Latch Checking Sub-Task

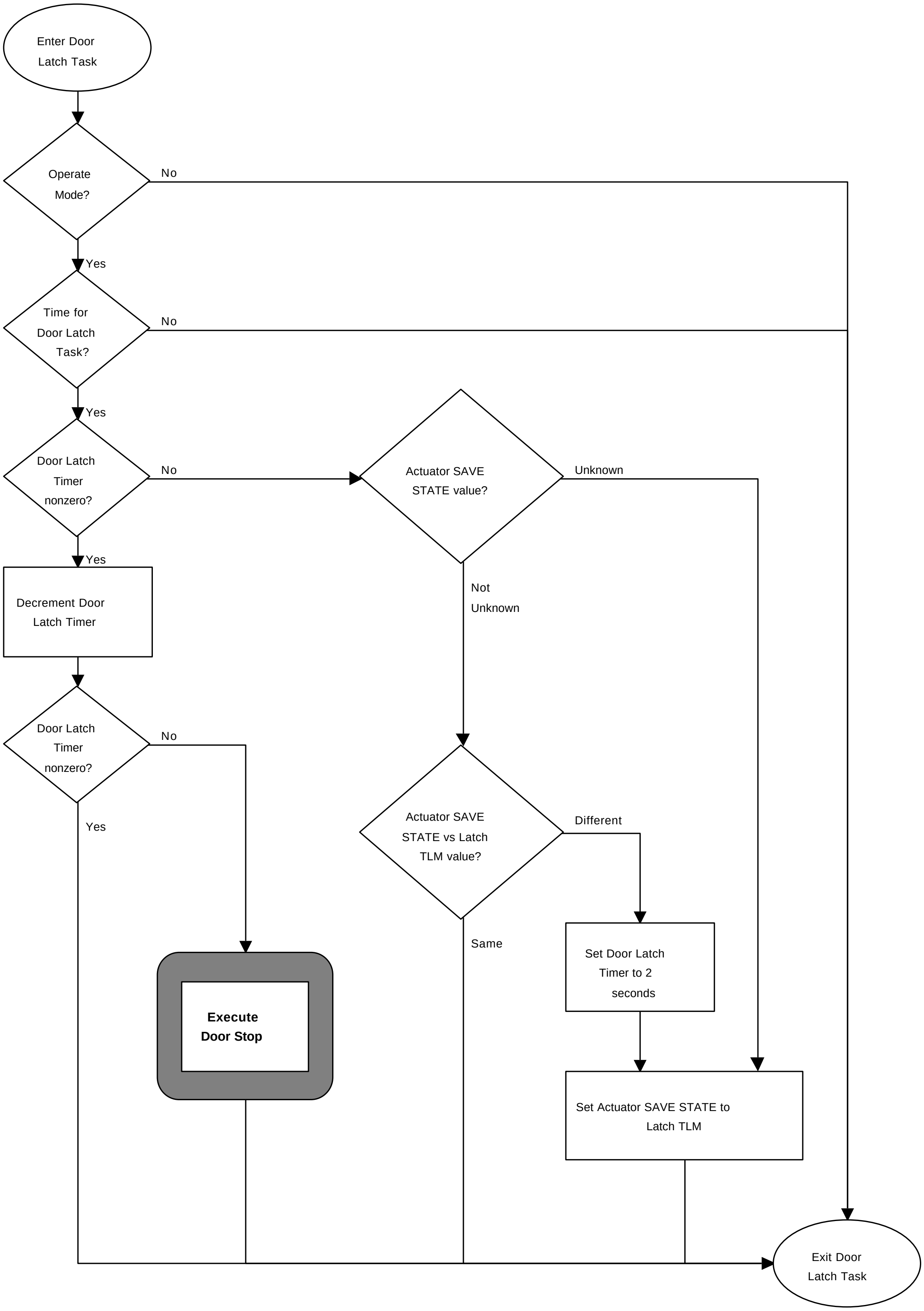
```
Saved state of actuator latch is set to UNKNOWN during initialization
Check to see if periodic timer has elapsed (1.0 second timer is used to determine when
   to invoke the Door Task)
IF (periodic timer has elapsed) THEN
   IF (latch timer == 0) THEN
      Get current value of the actuator latch
      Check the current state of the actuator latch against the saved value
         (UNKNOW, OPEN or CLOSED)
      IF (saved state == UNKNOWN) THEN
         saved state = current state
      ELSE
         IF (current state != saved state) THEN
            Set latch timer to 2 seconds to ensure door is driven into door latch
            saved state = current state
         ENDIF
      ENDIF
   ELSE
      DECREMENT latch timer
      IF (latch timer == 0) THEN
          Perform door stop (see above)
      ENDIF
   ENDIF
ENDIF
```

4.2.5.7.2 Logic Flow: Door Task

<u>Door Timer Sub-Task</u>



Actuator Latch Sub-Task


Enter Door Latch Task
Operate Mode?
No
Yes
Time for Door Latch Task?
No
Yes
Door Latch Timer nonzero?
No
Yes
Actuator SAVE STATE value?
Unknown
Not Unknown
Decrement Door Latch Timer
Door Latch Timer nonzero?
No
Yes
Actuator SAVE STATE vs Latch TLM value?
Different
Same
Set Door Latch Timer to 2 seconds
Execute Door Stop
Set Actuator SAVE STATE to Latch TLM
Exit Door Latch Task

4.2.5.7.3 Reuse

Source: FUSE

Design Reuse: 100%

Code Reuse: 90%

Logic needs to be added that determines whether the code is running from PROM (Boot code) or from 32K RAM (Operate code).

4.2.5.8 Housekeeping Task

The Housekeeping Task is responsible for collecting counter information, collecting hardware and software status, and sending out the housekeeping packet to the CS. The Housekeeping Task is comprised of two sub-tasks. The Collect Counter Data Sub-Task collects counter information, while the Send Housekeeping Data Sub-Task collects other status and sends out the housekeeping packet.

The Collect Counter Data Sub-Task periodically reads Counters A and B, and places the counter information in the housekeeping area. It constructs a time stamp that indicates when the counter information was collected.

When signalled by the Command Handler Task, the Send Housekeeping Data Sub-Task collects hardware and software status, and populates the housekeeping area with it. After collecting status information, it sends out the housekeeping packet to the CS.

When the housekeeping flag is set by the Command Handler Task, the Send Housekeeping Data Sub-Task constructs a time stamp to reflect when the housekeeping packet was sent. It reads all of the essential monitors for the latest values. The monitors include A/D converted values and the MUX register value that contains HV and door status. After reading the monitors, the sub-task checks to see if a current limit violation occurred. If so, it saves the values that caused the limit violation in housekeeping. The sub-task also reads the PH A and PH B data and stores them in the housekeeping area. The memory monitor information is then collected and stored in the housekeeping area. The housekeeping area is the central repository for all status information. Each housekeeping item has a dedicated location in memory, and all status information is stored in the appropriate location in the housekeeping area.

With all of the housekeeping information collected, the sub-task then sends the housekeeping packet to the CS. The packet consists of 512 32-bit housekeeping words. The housekeeping words are written sequentially to a set of four 8-bit registers. After all of the housekeeping words have been written to the registers, the task resets the HST error messages contained in housekeeping so that they are not duplicated in the next housekeeping packet.

4.2.5.8.1 Processing Logic

This section presents the processing logic of this task in a pseudo-programming language format. The Housekeeping Task actually consists of two separate sub-tasks. Since the two sub-tasks are logically linked, they are included here together. The two sub-tasks consist of the Collect Counter Data Sub-Task and Send Housekeeping Data Sub-Task.

Collect Counter Data Sub-Task

```
Check to see if periodic timer has elapsed (1.0 second timer is used to determine when
   to invoke the Collect Counter Data Sub-Task)
IF (periodic timer has elapsed) THEN
   Construct timestamp for Counters and store in housekeeping
   Read Counters A & B and store in housekeeping
ENDIF
```

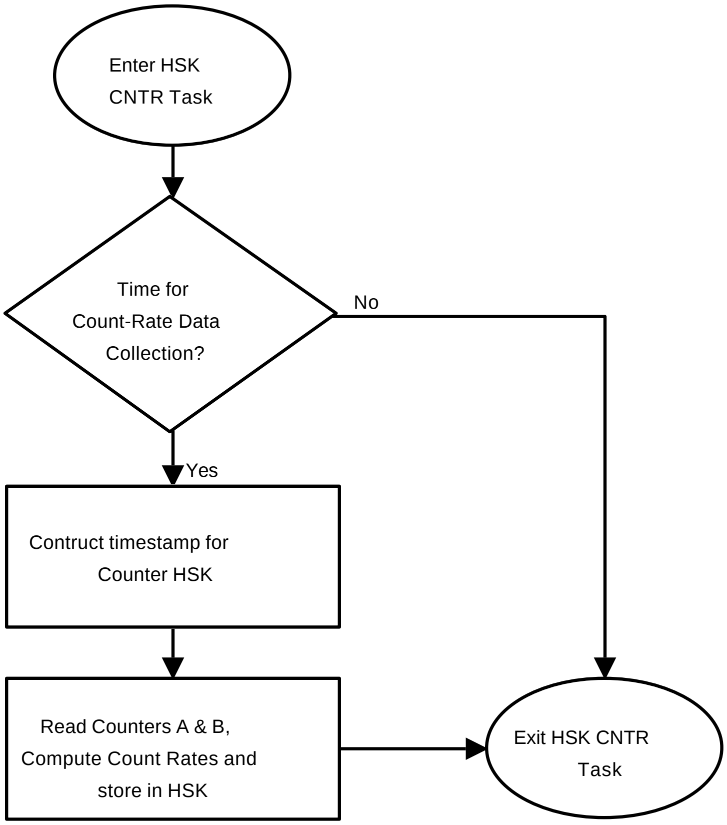
Send Housekeeping Data Sub-Task

```
Check to see if the send housekeeping flag was set
```

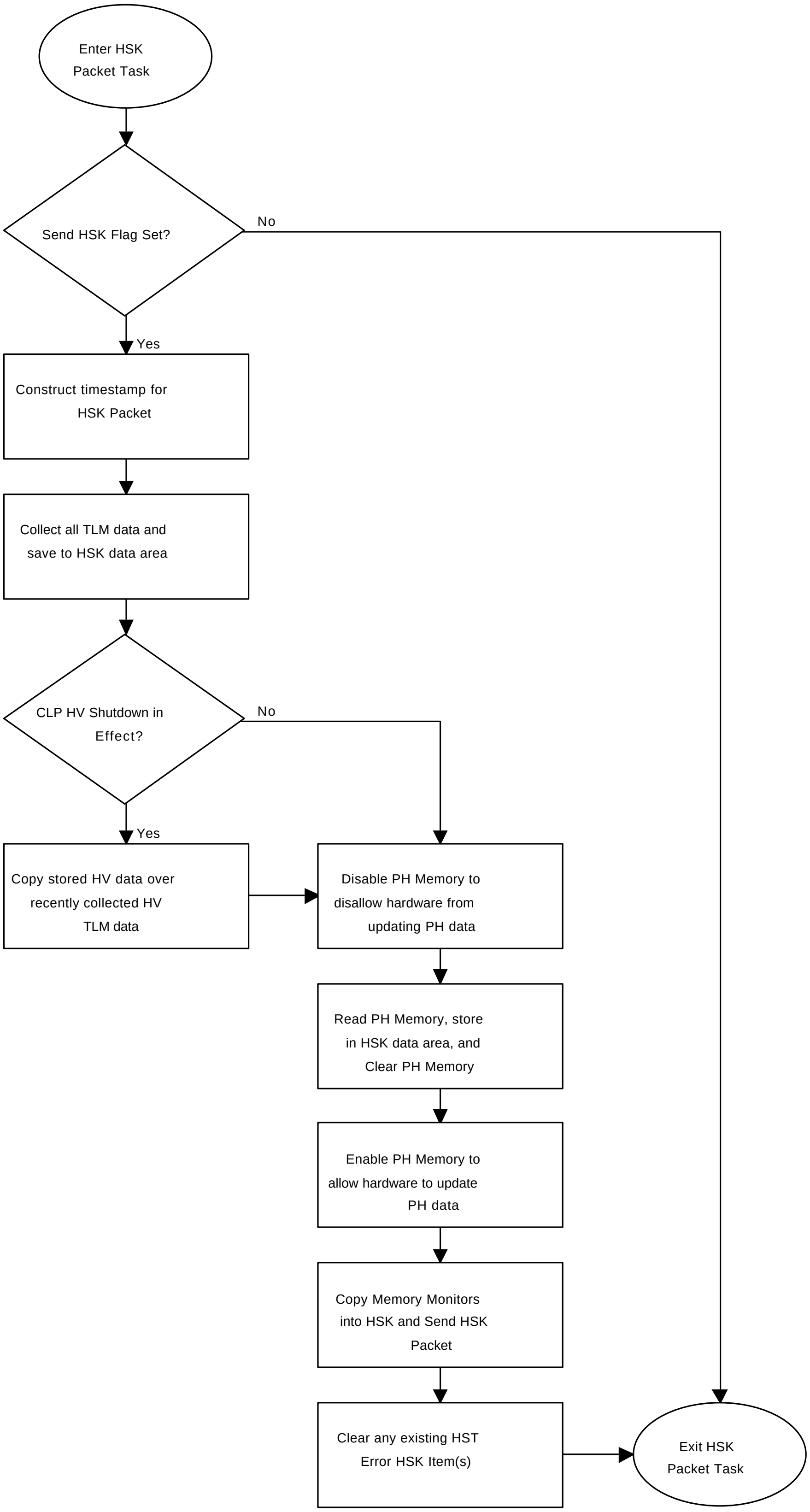
```
IF (send housekeeping flag was set) THEN
   Construct a time stamp to reflect when command was received/housekeeping sent
   Read all essential monitors (A/D converted values, read mux for HV/Door status)
   Check to see if current limit protection violation occurred
   IF (a violation occurred) THEN
      Copy saved limit violation values over collected sample and put them in
        housekeeping so we can see the conditions of the limit violation
   ENDIF
   Disable PH memory to disallow hardware from updating it
   Read entire PH memory and put in housekeeping (unlike FUSE which only reads
      1/8th of the PH data at a time)
   Clear PH memory
   Enable PH memory for access by hardware
   Collect memory monitor information and store in housekeeping
   Send out housekeeping packet to CS
   Reset HST error housekeeping items
ENDIF
```

4.2.5.8.2 Logic Flow: Housekeeping Task

Collect Counter Data Sub-Task


Enter HSK CNTR Task
Time for Count-Rate Data Collection?
No
Yes
Contruct timestamp for Counter HSK
Read Counters A & B, Compute Count Rates and store in HSK
Exit HSK CNTR Task

Send Housekeeping Data Sub-Task


Enter HSK Packet Task
Send HSK Flag Set?
No
Yes
Construct timestamp for HSK Packet
Collect all TLM data and save to HSK data area
CLP HV Shutdown in Effect?
No
Yes
Copy stored HV data over recently collected HV TLM data
Disable PH Memory to disallow hardware from updating PH data
Read PH Memory, store in HSK data area, and Clear PH Memory
Enable PH Memory to allow hardware to update PH data
Copy Memory Monitors into HSK and Send HSK Packet
Clear any existing HST Error HSK Item(s)
Exit HSK Packet Task

4.2.5.8.3 Reuse

Source: FUSE

Design Reuse: 50%

Code Reuse: 25%

The Housekeeping Task must output the housekeeping words to the CS. Hardware differences between FUSE and COS require housekeeping data to be collected differently. New timing constraints are being placed on the collection of housekeeping data to ensure they are collected on a once per second basis. The addition of memory monitors requires a change to the Housekeeping Task.

4.2.6 Inspect and Change Task

The Inspect and Change Task is a modified Archimedes MON51 utility. This task uses the 8051 serial port to communicate with a PC running the MON51 host software. The serial I/O port hardware is a not-for-flight part, therefore, this task does nothing when running on flight hardware except consume a minimal number of processor cycles. This task is especially useful when running on breadboard or prototype hardware. It allows the programmer to view the contents of DCE memory via the MON51 host software.

The current design does not include this task, however, it is kept here as part of the FUSE design. In the event this task is needed to support software testing, the hooks are already in place in the software design.

4.2.7 Miscellaneous Functions

This section describes important DCE FSW functions that do not fall into the ISR or Task categories.

4.2.7.1 Error/Diagnostic Reporting Function

The Error/Diagnostic Reporting Function is used throughout the DCE FSW by any function that needs to report a HST error or a diagnostic. The function takes care of formatting the housekeeping areas dedicated to HST errors and diagnostics with the new error/diagnostic information.

There is a difference between how HST errors and diagnostics are handled by the DCE FSW. HST errors are queued in memory from the oldest to the newest error. If the area in housekeeping dedicated to HST errors fills, the newest errors are dropped. HST errors are also cleared from memory after they have been sent out in a housekeeping packet. They only ever appear once in a housekeeping packet.

Diagnostics, in contrast, are queued in newest to oldest order and persist in housekeeping. Each diagnostic has a serial number to uniquely identify it. Old diagnostics drop off the end when the queue is full and there is no more room to store them.

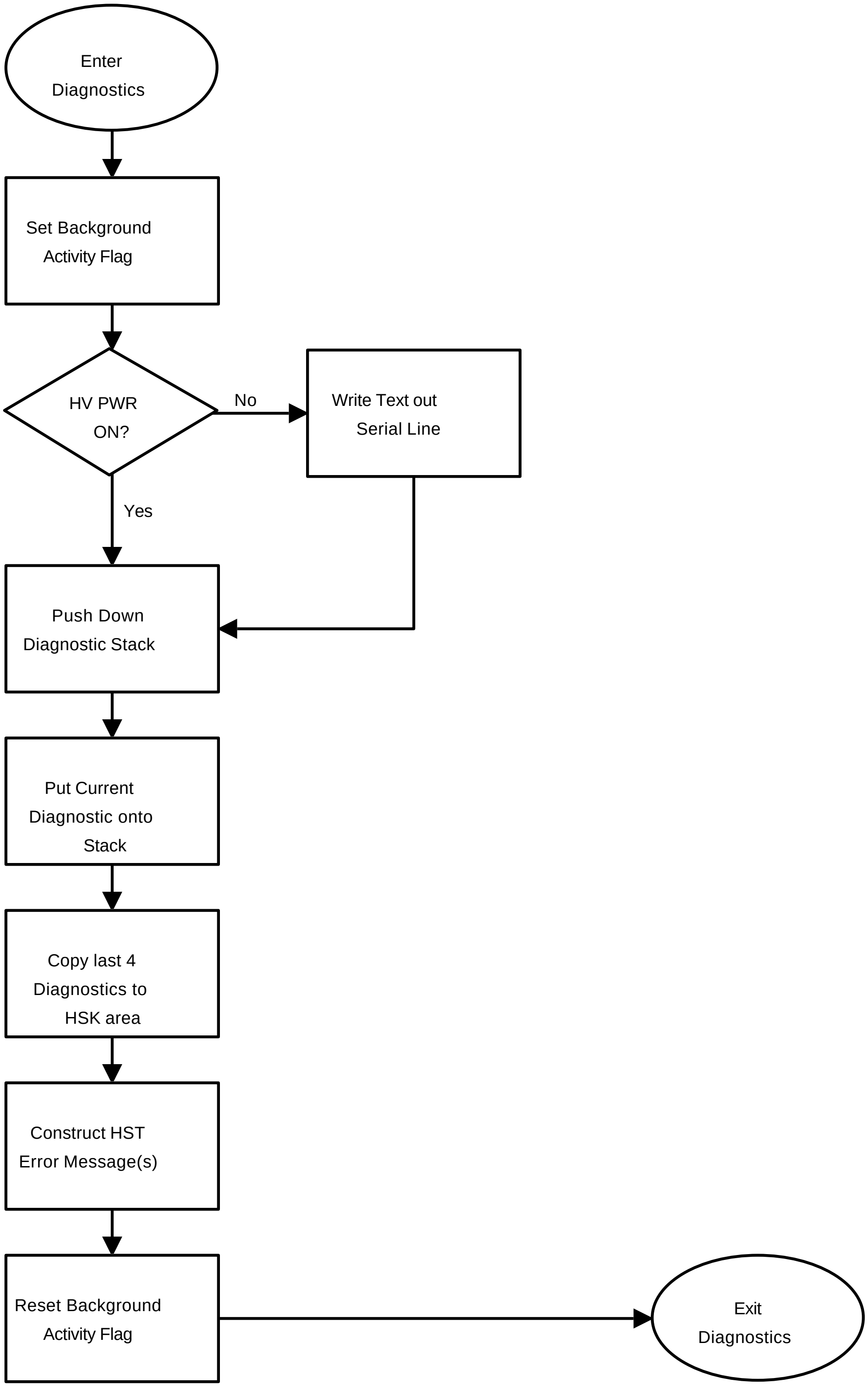
4.2.7.1.1 Processing Logic

This section presents the processing logic of this function in a pseudo-programming language format.

```
Invoked by any function or task that needs to report an error or diagnostic
Set background activity flag to prevent recursive calls
Record state of hardware high-voltage power on bit
IF (high voltage bit is OFF) THEN
   Report diagnostic text out serial port (this is part of the MON51 activites)
ENDIF
Push diagnostic stack down by one
Insert new diagnostic at top of stack
Copy top four diagnostics to another part of housekeeping (for GSE purposes)
```

```
Construct HST error message area
Restore background activity flag
```

4.2.7.1.2 Logic Flow: Error/Diagnostic Reporting Function


Enter Diagnostics
Set Background Activity Flag
HV PWR ON?
No
Write Text out Serial Line
Yes
Push Down Diagnostic Stack
Put Current Diagnostic onto Stack
Copy last 4 Diagnostics to HSK area
Construct HST Error Message(s)
Reset Background Activity Flag
Exit Diagnostics

4.2.7.1.3 Reuse

Source: FUSE

Design Reuse: 50%

Code Reuse: 25%

This function must now handle the formatting and management of HST errors.

4.2.7.2 High-Voltage State Bit Management

The DCE Flight Software reports the status of the detector's high voltage power supply through several hardware and software talkbacks. The behavior of these talkbacks is described below. The DCE HV talkbacks change as a result of HV commands sent to the DCE by the CS FSW (or the FUV detector EGSE), or through autonomous action taken by the DCE FSW. As the DCE HV State Bits (LFHSTATE) are not simply a reported value from hardware registers or latches, the behavior of LFHSTATE must be clearly defined for all of the recognized events which can change the state of DCE High Voltage. The following section contains a matrix showing the eight defined states for LFHSTATE, and the eleven recognized "events" that can change the value of LFHSTATE.

4.2.7.2.1 Behavior of LFHSTATE

The following table describes how the HV State Bits (LFHSTATE) behave with respect to the eleven recognizable "events" which can drive HV State Transitions. For example, if LFHSTATE is 0 (HV Power OFF), and the command LFHVPWRON (Event B) is sent to the DCE FSW, then the new value of LFHSTATE will be 7 (HV Power ON).

	New LFHSTATE, as a result of one of the following "Events"											
		A	B	C	D	E	F	G	H	I	J	K
Current LFHSTATE	0	0	7	0	0	0	0	0	0	0	0	0
	1	0	1	0	1	2	3	4	6	4	0	0
	2	0	2	0	1	2	3	4	6	4	0	0
	3	0	3	0	1	2	3	4	6	4	0	0
	4	0	4	0	1	2	3	4	6	4	0	0
	5	0	5	0	1	2	3	4	6	4	0	0
	6	0	6	0	1	2	3	4	6	4	0	0
	7	0	7	0	1	2	3	4	6	4	0	0

<u>HV states:</u>

0 - HV Power OFF
1 - HV NOMA
2 - HV NOMB
3 - HV NOM
4 - HV LOW (or CRP)
5 - invalid
6 - HV SET
7 - HV Power ON

<u>Recognized "Events" which can drive HV State Transitions:</u>

A - POR - Power ON Reset
B - cmdHVPWRON - LFHVPWRON command to power-on HV

C - cmdHVPWROFF - LFHVPWROFF command to power-off HV

D - cmdHVSTATEhvnoma - LFHSTATE command to HV NOMA state

E - cmdHVSTATEhvnomb - LFHSTATE command to HV NOMB state

F - cmdHVSTATEhvnom - LFHSTATE command to HV NOM state

G - cmdHVSTATElow - LFHSTATE command to HV LOW state

H - cmdHVSET - LFHVSET command to any HV level

I - CRP Trigger, Segment A and/or B.

J - WatchDog Reset or Commanded Reset

K - HVI/AUXI Shutdown, Segment A and/or B.

4.2.8 DCE Commands

This section describes the DCE Commands that may be used by the CS to command and configure the DCE. The command information captured in this section is a snapshot of the state of command definition at the time this design document was created. For the official information concerning DCE commands, see Appendix B of the DM-05.

4.2.8.1 DCE Command List

The list consists of the command name, the DCE software mode(s) in which it is valid, the mnemonic assigned by the DM-05, any parameters it takes, the source of any reuse, and a description of the software modifications made if the command function is being reused from a previous mission.

Command Name	Mode	Mnemonic	Parameters	Source	Scope of Change
DCE Internal Command					
No-Op	boot/ oper	LFDNOOP	Param 1: GSE Exposure Flag	FUSE	None
Send Housekeeping Packet	boot/ oper	LFDHKREQ	None	New	New
Clear Diagnostic Code Stack	boot/ oper	LFDDIAGC	None	New	New COS command created from a FUSE test command
Process Memory Upload	boot/ oper	LFDUPLOD	Param 1: Destination address in RAM dataspace Param 2: Number of bytes to load Param 3: CRC value of load	FUSE	Added option to bypass CRC check
Download Memory Block	boot/ oper	LFDDNLOD	Param 1: Memory address in DCE RAM dataspace Param 2: Number of Bytes to Transfer	FUSE	Added function to transmit the Download Packet to the CS
DCE Memory Copy	boot/ oper	LFDCOPY	Param 1: Source address in DCE Memory Param 2: Destination address in DCE dataspace Param 3: Number of bytes to copy Param 4: Memory Space of source	New	New

			(DATA/CODE)		
Jump to Upper/Lower Code Area	boot/ oper	LFDJMPCS	Param 1: Destination code area	FUSE	Added destination parameter
Jump to Specified Address and Execute Code There	boot/ oper	LFDGOTO	Param 1: DCE destination memory address	FUSE	None
Set DCE Memory Monitor	boot/ oper	LFDMADDR	Param 1: Memory monitor to set Param 2: DCE Memory Address to Monitor Param 3: Mem space containing monitored addr	New	New
Calculate CRC	boot/ oper	LFDCRC	Param 1: Starting address in DCE memory Param 2: Number of bytes to compute Param 3: Memory space to check (DATA/CODE)	New	New
Initiate Power-on Reset to DCE	boot/ oper	LFDRSETP	None	FUSE	None
Initiate Watchdog Reset	boot/ oper	LFDRSETW	None	FUSE	None
Watchdog Timer Disable/Enable	boot/ oper	LFDWDOG	Param 1: 0/1 (Disable/Enable)	New	New COS command created from a FUSE test command
Digitizer Commands					
Stim Pulse Control	oper	LFGSTIM	Param 1: Stim Pulse Rate (OFF/LOW/MID/MAX) Param 2: Segment selection (A/B)	FUSE	Added modifiable pule-rate parameter and segment selection parameter. Ensures FSW is in Operate mode.
Digitizer Begin Walk Adjust	oper	LFGBWK	Param 1: Begin walk setting adjustment Param 2: Segment selection (A/B) Param 3: Direction (DISP/XDISP)	FUSE	Added Segment Selection and Direction parameters. Ensures FSW is in Operate mode.
Digitizer End Walk Adjust	oper	LFGEWK	Param 1: End walk setting adjustment Param 2: Segment selection (A/B) Param 3: Direction (DISP/XDISP)	FUSE	Added Segment Selection and Direction parameters. Ensures FSW is in Operate mode.

Dig. Lower Charge Threshold	oper	LFGLQT	Param 1: Lower charge threshold setting Param 2: Segment selection (A/B)	FUSE	Added Segment Selection parameter. Ensures FSW is in Operate mode.
Digitizer Upper Charge Threshold	oper	LFGUQT	Param 1: Upper charge threshold setting Param 2: Segment selection (A/B)	FUSE	Added Segment Selection parameter. Ensures FSW is in Operate mode.
Digitizer Timing Threshold	oper	LFGTT	Param 1: Dispersion timing threshold setting Param 2: Segment selection (A/B) Param 3: Direction for threshold (DISP/XDISP)	FUSE	Added Segment Selection and Direction parameters. Ensures FSW is in Operate mode.
Digitizer Image Shift	oper	LFGSHF	Param 1: Image shift setting Param 2: Segment selection (A/B) Param 3: Direction of shift (DISP/XDISP)	FUSE	Added Segment Selection and Direction parameters. Ensures FSW is in Operate mode.
Digitizer Image Stretch	oper	LFGSTR	Param 1: Image Stretch Setting Param 2: Segment selection (A/B) Param 3: Direction of stretch (DISP/XDISP)	FUSE	Added Segment Selection and Direction parameters. Ensures FSW is in Operate mode.
High Voltage Commands					
Set High Voltage Current Limit	oper	LFHVILIM	Param 1: High voltage current limit	FUSE	Ensures FSW is in Operate mode.
Set Voltage Level for HVLOW	oper	LFHVLOW	Param 1: High Voltage Setting Param 2: Segment selection (A/B)	New	New
Set Voltage Level for HVNOM	oper	LFHVNOM	Param 1: High voltage setting Param 2: Segment selection (A/B)	New	New
Set High Voltage Ramp Rate	oper	LFHRAMPT	Param 1: Time between ramp steps	New	New
Max. Setting for High Voltage	oper	LFHVMAX	Param 1: High Voltage Setting Param 2: Segment selection (A/B)	FUSE	Added Segment Selection parameter. Ensures FSW is in Operate mode.
Enable/Disable HV Commanding	oper	LFHVENA	Param 1: 0/1 (Disable/Enable)	FUSE	Ensures FSW is in Operate mode.
QDE Grid High Voltage On/Off	oper	LFHQPWR	Param 1: 0/1 (Off/On)	FUSE	Ensures FSW is in Operate mode.

HV Power Supply On/ Off	oper	LFHVPWR	Param 1: 0/1 (Off/On)	FUSE	Ensures FSW is in Operate mode.
Transition to High Voltage State	oper	LFHSTATE	Param 1: HV state (NOMA/NOMB/NOMAB/LOW)	New	New
High Voltage Setting	oper	LFHVSET	Param 1: Target voltage value Param 2: Segment selection (A/B)	New	Modified from FUSE DAC commands.
FUV Count Rate Protection	oper	LFPCRP	Param 1: Smoothing interval Param 2: Segment selection (A/B) Param 3: Maximum event counts per second	FUSE	Parameter is a rate rather than a count. Ensures FSW is in Operate mode.
Door Commands					
Auxiliary Power Supply On/Off	oper	LFRAXPWR	Param 1: 0/1 (Off/On)	FUSE	Ensures FSW is in Operate mode.
Set Aux Pwr Supply Current Limit	oper	LFRILIM	Param 1: Door current limit	FUSE	Ensures FSW is in Operate mode.
Door Motor Enable	oper	LFRMENA	Param 1: 0/1 (Disable/Enable)	FUSE	Ensures FSW is in Operate mode.
Select Door Direction	oper	LFRMDIR	Param 1: Door direction (SAFE/CLOSE/OPEN)	FUSE	Ensures FSW is in Operate mode.
Move the FUV Detector Door	oper	LFRMPWR	Param 1: 0/1 (Stop door move/Start door move)	FUSE	Ensures FSW is in Operate mode.
HOP Actuator Enable/Disable	oper	LFRACTEN	Param 1: 0/1 (Disable/Enable)	FUSE	Ensures FSW is in Operate mode.
Actuator 1 On/Off	oper	LFRACT1	Param 1: 0/1 (Off/On)	FUSE	Ensures FSW is in Operate mode.
Actuator 2 On/Off	oper	LFRACT2	Param 1: 0/1 (Off/On)	FUSE	Ensures FSW is in Operate mode.
HOP Actuator Reset	oper	LFRACTRS	Param 1: Actuator reset function	FUSE	Ensures FSW is in Operate mode.
Override Door Endswitch Protection	oper	LFRLSOVD	Param 1: 0/1 (Disable override/Enable override)	FUSE	Ensures FSW is in Operate mode.

4.2.8.2 DCE Command Descriptions

4.2.8.2.1 Maintenance Commands

No-Op (LFDNOOP)

This command increments the packet counter and returns a housekeeping packet, but doesn't increment the commands received or commands executed counters. This command is used by the control section to stroke the DCE watchdog timer once each second. For more information on commanding the DCE, see the FUV ICD.

Send a Housekeeping Packet (LFDHKREQ)

This command increments counters and sends the regular housekeeping packet to the control section.

Clear Diagnostic Code Stack (LFDDIAGC)

This command clears the 32 item diagnostic code stack reported in housekeeping to zeros.

- Clear the internal list of diagnostics kept by the DCE flight software and reported in housekeeping. (The incrementing diagnostics serial number reported with the diagnostics is not reset by this command.)
- The Housekeeping Packet returned from this command includes the cleared diagnostic code stack.

Process Memory Upload (LFDUPLOD)

This command copies memory contents from the DCE transfer buffer to the specified area of RAM with a CRC check. The DCE transfer buffer (addresses 040-43F hex) should be loaded with the data to be copied before this command is sent to the DCE.

- If the CRC parameter is set:
  - Compute a CRC for the number of bytes in the data transfer buffer
  - Set LFMXFER to this CRC value
  - If the calculated CRC matches the predicted CRC (sent as a parameter):
    - Transfer the number of bytes from the transfer buffer to the destination address
- If the CRC parameter is not set:
  - Transfer the number of bytes from the transfer buffer to the destination address
- The Housekeeping Packet returned from this command includes the new value for LFMXFER

Download Memory Block (LFDDNLOD)

This command sends a download packet of 512 longwords to the control section, followed by a housekeeping packet containing the value of a CRC of the requested address range. If less than 1024 bytes are requested, the download packet still contains 512 long words, but only the requested range is valid and the CRC is performed only on the selected range.

- Compute a CRC on the selected address range
- Set LFMXFER to the value of the computed CRC in the Housekeeping Data Structure
- Copy a block of memory from the specified address in RAM dataspace to the transfer area (dump buffer)
- Output the dump buffer to the Control Section in a Download Packet.
  - The Dump Packet always consists of 512 32-bit long words.
  - The most significant two bytes of each long word is the 16-bit transfer buffer address
  - The least significant two bytes of each long word is a 16-bit memory data word of the selected download memory block.
  - Only the requested number of bytes are valid. The remaining words in the dump packet are invalid

  - See the FUV ICD for further details on Download Packets
- The Housekeeping Packet returned from this command includes:
  - The CRC computed over specified range of memory
  - The starting address for the dump (LFDCBUF)
  - The number of valid bytes in the dump

DCE Memory Copy (LFDCOPY)

This command copies a DCE memory range in codespace or dataspace to another address in DCE dataspace.

- Copy NBYTES from SADDRESS in MEMSPACE to DADDRESS in dataspace
- The Housekeeping Packet returned from this command includes all normal command talkbacks

Jump to Upper/Lower Code Area (LFDJMPCS)

Used by the control section to transition from boot to operate mode, this commands causes the microprocessor to begin executing at a predefined entry point in either the lower or upper code area.

- If DCE Mode is BOOT:
  - If CODEAREA parameter is LOWER:
    - Jump to the predefined entry point in lower code area
  - If CODEAREA parameter is UPPER:
    - Jump to the predefined entry point in the upper code area
  - If CODEAREA parameter is TOGGLE:
    - Jump to the predefined entry point in boot
- If DCE Mode is OPERATE:
  - If CODEAREA parameter is LOWER:
    - Jump to the predefined entry point in lower code area
  - If CODEAREA parameter is UPPER:
    - Jump to the predefined entry point in the upper code area
  - If CODEAREA parameter is TOGGLE:
    - If currently executing from lower code, jump to the predefined entry point in upper code
    - If currently executing from upper code, jump to the predefined entry point in lower code
- The Housekeeping Packet returned from this command will include updated status bits indicating which code segment is currently executing

Jump to a Specified Address and Execute Code There (LFDGOTO)

This command causes the 8051 microprocessor to begin executing at the requested address.

- Transfer 8051 microprocessor control to the address specified
- Start executing at the new address
- This command may or may not return a Housekeeping Packet, depending on the jump address

Set DCE Memory Monitor (LFDMADDR)

This command sets a memory monitor in dataspace, codespace, or 8051 internal address space. There are eight separate memory monitors which are updated and reported in each housekeeping packet.

- Check MEMSPACE parameter:
  - If value is INTERNAL:

        - Set bit MONITOR (value of the MONITOR parameter) of LFDMDATA to DATA in the Housekeeping data structure
        - Set bit MONITOR of LFDMEXTR to INTERNAL in the Housekeeping data structure
    - If value is DATA:
        - Set bit MONITOR of LFDMDATA to DATA in the Housekeeping data structure
        - Set bit MONITOR of LFDMEXTR to EXTERNAL in the Housekeeping data structure
    - If value is CODE:
        - Set bit MONITOR of LFDMDATA to CODE in the Housekeeping data structure
        - Set bit MONITOR of LFDMEXTR to EXTERNAL in the Housekeeping data structure
- Set item MONITOR of LFDMADD to the requested address in the Housekeeping data structure
- The Housekeeping Packet returned from this command includes new values for LFDMDATA (code/data select), LFDMEXTR (external/internal memory), LFDMADD (monitor addresses), and LFDMONS (array that reports the monitored values – computed using LFDMDATA, LFDMEXTR, and LFDMADD)

Calculate CRC (LFDCRC)

In response to the command the DCE will compute a CRC on the selected DCE memory region.

- Set LFMXFER to 0 in the Housekeeping Data Structure to indicate a CRC check is in progress
- Fill the CRC data structure with ADDRESS and NBYTES parameters and a pointer to code that specifies the memory space type specified by the MEMSPACE parameter
- Pass pointer to the data structure to the CRC checking task
- The next time the CRC Checking Task runs, it will begin calculating the CRC
- Housekeeping will report LFMXFER as 0 until the calculation is complete. Then it will report the calculated CRC in LFMXFER

Initiate Power-On Reset to DCE (LFDRSETP)

This command activates the internal hardware reset circuitry of the DCE and reboots the DCE into its power-on reset state with the DCE in Boot mode executing from PROM. This command includes a 10 second watchdog timer test. Any commands sent to the DCE during this time will be lost. No housekeeping packet is returned from this command.

- Reset the flag that indicates a Power On Reset has been previously performed
- Initiate a Watchdog Reset
- This results in the DCE performing its complete initialization sequence (See section 4.2.2.4 of this document)
- No Housekeeping packet will be returned from this command

Initiate Watchdog Reset (LFDRSETW)

This command causes a watchdog reset and initializes the DCE to it's watchdog reset state with the detector in Boot mode and executing from PROM.

- Wiggle the hardware reset line
    - If this fails:
        - Issue a diagnostic
        - Jump to the Reset Vector
- This results in the DCE performing its minimal initialization sequence (See section 4.2.2.4 of this document)
- No Housekeeping packet will be returned from this command

DCE Watchdog Timer Disable/Enable (LFDWDOG)

This command enables or disables the DCE watchdog timer. When enabled, the DCE will initiate a watchdog rest if it does not receive a command from the Control Section within 10 seconds. Any command (op-code), including invalid commands, will reset the timer. When the DCE is powered-up, or after any reset, the watchdog timer is disabled. The watchdog timer state is not affected when the DCE transitions from Boot to Operate.

- Set Register to the value of the parameter to enable or disable the watchdog timer
- The Housekeeping Packet returned from this command includes the new state of the watchdog timer in Housekeeping item LFDWDOG (hardware talkback)

4.2.8.2.2 Digitizer Commands

Stim Pulse Control (LFGSTIM)

This command configures the internal stim pulses. Stim pulses are used for image calibration and typically are enabled before each exposure. The rate will depend on the target.

- If the SEGMENT parameter is B:
  - Set Digitizer B stim pulse rate to the value of the RATE parameter
  - Set LFGSTIMB to the value of the RATE parameter in the Housekeeping Data Structure
- If the SEGMENT parameter is not B:
  - Set Digitizer A stim pulse rate to the value of the RATE parameter
  - Set LFGSTIMA to the value of the RATE parameter in the Housekeeping Data Structure
- The Housekeeping Packet returned from this command includes the new value updated above

Digitizer Begin Walk Adjustment (LFGBWK)

This command is used to tune FUV detector performance for Segment A or B by optimizing the image resolution performance. The Begin and End Walk are adjusted together to ensure that the event position encoding is independent of the event amplitude. The Begin Walk adjustment is primarily seen on the “begin” side of the anode.

- If the SEGMENT parameter is B:
  - If the DIR parameter is Y:
    - Set Digitizer B, Y-direction Begin Walk adjustment to the value of the SETTING parameter
  - If the DIR parameter is not Y:
    - Set Digitizer B, X-direction Begin Walk adjustment to the value of the SETTING parameter
- If SEGMENT parameter is not B:
  - If the DIR parameter is Y:
    - Set Digitizer A, Y-direction Begin Walk adjustment to the value of the SETTING parameter
  - If the DIR parameter is not Y:
    - Set Digitizer A, X-direction Begin Walk adjustment to the value of the SETTING parameter
- The Housekeeping Packet returned from this command includes the updated Begin Walk setting in hardware-talkback LFGBWKAX, LFGBWKAY, LFGBWKBX, or LFGBWKBY

Digitizer End Walk Adjustment (LFGEWK)

This command is used to tune FUV detector performance for Segment A or B by optimizing the image resolution performance. The Begin and End Walk are adjusted together to ensure that the event position encoding is independent of the event amplitude. The End Walk adjustment is primarily seen on the “end” side of the anode.

- If the SEGMENT parameter is B:
  - If the DIR parameter is Y:

        - Set Digitizer B, Y-direction End Walk adjustment to the value of the SETTING parameter
    - If the DIR parameter is not Y:
        - Set Digitizer B, X-direction End Walk adjustment to the value of the SETTING parameter
- If SEGMENT parameter is not B:
    - If the DIR parameter is Y:
        - Set Digitizer A, Y-direction End Walk adjustment to the value of the SETTING parameter
    - If the DIR parameter is not Y:
        - Set Digitizer A, X-direction End Walk adjustment to the value of the SETTING parameter
- The Housekeeping Packet returned from this command includes the updated End Walk setting in hardware-talkback LFGEWKAX, LFGEWKAY, LFGEWKBX, or LFGEWKBY

<u>Digitizer Lower Charge Threshold Adjust (LFGLQT)</u>

This command is used to tune FUV detector performance for segment A or B. The lower charge threshold setting establishes a minimum charge criteria for valid events. Any events not meeting the minimum charge criteria are not reported in the science data stream.

- If the SEGMENT parameter is B:
    - Set Digitizer B lower charge threshold to the value of the SETTING parameter
- If the SEGMENT parameter is not B:
    - Set Digitizer A lower charge threshold to the value of the SETTING parameter
- The Housekeeping Packet returned from this command includes the updated lower charge threshold hardware talkback, LFGLQTA or LFGLQTB

<u>Digitizer Upper Charge Threshold Adjust (LFGUQT)</u>

This command is used to tune FUV detector performance for segment A or B. The upper charge threshold setting establishes the maximum charge criteria for valid events. Any events exceeding the maximum charge criteria are not reported in the science data stream.

- If the SEGMENT parameter is B:
    - Set Digitizer B upper charge threshold to the value of the SETTING parameter
- If the SEGMENT parameter is not B:
    - Set Digitizer A upper charge threshold to the value of the SETTING parameter
- The Housekeeping Packet returned from this command includes the updated upper charge threshold hardware talkback, LFGUQTA or LFGUQTB

<u>Digitizer Timing Threshold (LFGTT)</u>

This command is used to tune FUV detector performance for Segment A or B. The command sets a timing threshold in the dispersion or cross-dispersion direction that defines the criteria for valid events. Any events not meeting the timing criteria are not reported in the science data stream.

- If the SEGMENT parameter is B:
    - If the DIR parameter is Y:
        - Set Digitizer B, Y-direction timing threshold to the value of the SETTING parameter
    - If the DIR parameter is not Y:
        - Set Digitizer B, X-direction timing threshold to the value of the SETTING parameter
- If SEGMENT parameter is not B:
    - If the DIR parameter is Y:
        - Set Digitizer A, Y-direction timing threshold to the value of the SETTING parameter

  - If the DIR parameter is not Y:
    - Set Digitizer A, X-direction timing threshold to the value of the SETTING parameter
- The Housekeeping Packet returned from this command includes the updated timing threshold setting in hardware talkback LFGTTAX, LFGTTAY, LFGTTBX, or LFGTTBY

Digitizer Image Shift (LFGSHF)

This command is used to tune FUV detector performance for segment A or B. The command shifts the segment image space in the dispersion or cross-dispersion direction. The image space is offset to include both the segment space and the stim pulse.

- If the SEGMENT parameter is B:
  - If the DIR parameter is Y:
    - Set Digitizer B, Y-direction image shift to the value of the SETTING parameter
  - If the DIR parameter is not Y:
    - Set Digitizer B, X-direction image shift to the value of the SETTING parameter
- If SEGMENT parameter is not B:
  - If the DIR parameter is Y:
    - Set Digitizer A, Y-direction image shift to the value of the SETTING parameter
  - If the DIR parameter is not Y:
    - Set Digitizer A, X-direction image shift to the value of the SETTING parameter
- The Housekeeping Packet returned from this command includes the updated image shift setting in hardware talkback LFGSHFAX, LFGSHFAY, LFGSHFBX, or LFGSHFBY

Digitizer Image Stretch (LFGSTR)

This command is used to tune FUV detector performance for Segment A or B. The command stretches the Segment image space in the dispersion or cross-dispersion direction. The image space is enlarged (or reduced) to include both the segment space and the stim pulse.

- If the SEGMENT parameter is B:
  - If the DIR parameter is Y:
    - Set Digitizer B, Y-direction image stretch to the value of the SETTING parameter
  - If the DIR parameter is not Y:
    - Set Digitizer B, X-direction image stretch to the value of the SETTING parameter
- If SEGMENT parameter is not B:
  - If the DIR parameter is Y:
    - Set Digitizer A, Y-direction image stretch to the value of the SETTING parameter
  - If the DIR parameter is not Y:
    - Set Digitizer A, X-direction image stretch to the value of the SETTING parameter
- The Housekeeping Packet returned from this command includes the updated image stretch setting in hardware talkback LFGSTRAX, LFGSTRAY, LFGSTRBX, or LFGSTRBY

4.2.8.2.3 High Voltage Commands

Set High Voltage Current Limit (LFHVILIM)

This command establishes the high voltage current limit for each detector segment. Setting a current limit of 0 will disable limit checking. A DCE background task samples the high voltage current periodically, based on available CPU time. With minimal CPU loading, the task samples the current every millisecond. During the worst case loading, the task monitors the current every 200 milliseconds. If a current sample exceeds the limit, the DCE will report a diagnostic in the Housekeeping Packet. If 20 consecutive out-of-limit samples occur, the DCE will shut off high voltage and report another diagnostic in Housekeeping. (See 4.2.5 Task Description for the Current Limit Protection Task)

- Set LFHVILIM to the value of the LIMIT parameter in the Housekeeping Data Structure
  - This value will be used by the Current Limit Protection Task
- The Housekeeping Packet returned from this command includes the new value of LFHVILIM

Set Voltage Level for HVLOW State (LFHVLOW)

This command establishes the high voltage setting that will be used when commanding the segment to the HV_LOW state using the LFHSTATE command. A different setting can be used for each segment.

- If the SEGMENT parameter is B:
  - Set LFHVLOB to the value of the VOLTAGE parameter in the Housekeeping Data Structure
    - This value will be used by the HV Ramp Task
- If the SEGMENT parameter is not B:
  - Set LFHVLOA to the value of the VOLTAGE parameter in the Housekeeping Data Structure
    - This value will be used by the HV Ramp Task
- The Housekeeping Packet returned from this command includes the new value of the Housekeeping item updated above

Set Voltage Level for HVNOM State (LFHVNOM)

This command establishes the high voltage setting that will be used when commanding the segment to the HV_NOM state using the LFHSTATE command. A different setting can be used for each segment.

- If the SEGMENT parameter is B:
  - Set LFHVNOMB to the value of the VOLTAGE parameter in the Housekeeping Data Structure
    - This value will be used by the HV Ramp Task
- If the SEGMENT parameter is not B:
  - Set LFHVNOMA to the value of the VOLTAGE parameter in the Housekeeping Data Structure
    - This value will be used by the HV Ramp Task
- The Housekeeping Packet returned from this command includes the new value of the Housekeeping item updated above

Set High Voltage Ramp Rate (LFHRAMPT)

This command specifies the time between steps when ramping up the high voltage. The ramp rate applies to both segments.

- Set LFHRAMPT to the value of the RAMPRATE parameter in the Housekeeping Data Structure
  - This value will be used by the HV Ramp Task
- The Housekeeping Packet returned from this command includes the new value of LFHRAMPT

Maximum Setting for High Voltage (LFHVMAX)

This command sets the maximum high voltage that is allowed by the hardware for Segment A or B. The maximum high voltage setting is reset to zero counts each time the high voltage is turned on. Therefore this command should be sent to the DCE after LFHVPWR and before ramping the high voltage.

- If the SEGMENT parameter is B:
  - Set the hardware high voltage clamp for Segment B to the value of the VOLTAGE parameter
- If the SEGMENT parameter is not B:

    - Set the hardware high voltage clamp for Segment A to the value of the VOLTAGE parameter
- The Housekeeping Packet returned from this command includes the new value of Housekeeping item LFHVMAXA/B (hardware talkback)

Enable/Disable High Voltage Commanding (LFHVENA)

This command controls the software interlock on commands affecting high voltage. Only commands that increase the high voltage state are required to have the high voltage commanding enabled.

- If the HIVOLT parameter is DISABLE:
    - Check if LFHVENA is already disabled
        - If already disabled (LFHVENA = 0)
            - Issue diagnostic 0A
            - Exit
        - If not disabled (LFHVENA=1)
            - Clear the high voltage enable bit (this value of this bit is used by Command Handler Task)
- If the HIVOLT parameter is ENABLE:
    - Check if LFHVENA is already enabled
        - If already enabled (LFHVENA = 1)
            - Issue diagnostic 09
            - Exit
        - If not enabled (LFHVENA=0)
            - Set the high voltage enable bit (this value of this bit is used by Command Handler Task)
    - The Housekeeping Packet returned from this command includes the new value of LFHVENA (hardware talkback)

QDE Grid High Voltage On/Off (LFHQPWR)

This command controls power to the two grids positioned above the Micro Channel Plate (MCP) detector segments. The grid voltage forces photoelectrons generated at the top of the microchannel plate back down to the MCP. This process improves the detector efficiency by approximately 30%. The grid may be turned on and off as long as HV is at HVLow or lower.

- If the POWER parameter is ON:
    - Verify high voltage commanding is enabled
        - If commanding is not enabled (LFHVENA = 0):
            - Issue diagnostic 08
            - Exit
        - If commanding is enabled (LFHVENA = 1):
            - Turn on grid power
- If the POWER parameter is OFF:
    - Turn off grid power
- The Housekeeping Packet returned from this command includes the new value for Housekeeping item LFHVQPWR, the grid power hardware talkback

High Voltage Power Supply On/Off (LFHVPWR)

This command turns on or off the high voltage power supply.

- If the POWER parameter is ON:

  - Verify high voltage commanding is enabled
    - If commanding is not enabled (LFHVENA = 0):
      - Issue diagnostic 08
      - Exit
    - If commanding is enabled (LFHVENA=1):
      - Set the Segment A high voltage DAC to 0 by calling the LFHSET Command Function:
        - VOLTAGE parameter = 0
        - SEGMENT parameter = A
      - Set the Segment B high voltage DAC to 0 by calling the LFHSET Command Function:
        - VOLTAGE parameter = 0
        - SEGMENT parameter = B
      - Set the Segment A HVMAX to 0 by calling the LFHVMAX Command Function:
        - VOLTAGE parameter = 0
        - SEGMENT parameter = A
      - Set the Segment B HVMAX to 0 by calling the LFHVMAX Command Function:
        - VOLTAGE parameter = 0
        - SEGMENT parameter = B
      - Set LFHVSTATE to 7 in the Housekeeping Data Structure to indicate the high voltage has been commanded to the Powered On state
      - Clear the high voltage current limit checking flags
      - Turn on the high voltage power
- If the POWER parameter is OFF:
  - Set LFHVSTATE to 0 in the Housekeeping Data Structure to indicate the high voltage has been commanded to the Powered Off state
  - Clear the high voltage current limit checking flags
  - Turn off the high voltage power
- The Housekeeping Packet returned from this command includes the new values for LFHVSTATE, LFHVPWR, the high voltage power hardware talkback, LFHVSETA/B, LFHVMONA/B, and LFHVMAXA/B

Transition to High Voltage State (LFHSTATE)

This command is used to transition between the four predefined high voltage states: LOW, NOMA, NOMB, NOMAB. When ramping up, the ramp rate parameter LFHRAMPT is used to ramp the high voltage. When going to a lower high voltage level, there is no ramp and the voltage is set directly to the new level.

- If the STATE parameter is LOW:
  - If Segment A voltage (LFHVDACA) is lower than LFHVLOA or Segment B voltage (LFHVDACB) is lower than LFHVLOB:
    - Verify high voltage commanding is enabled
      - If commanding is not enabled (LFHVENA = 0):
        - Send diagnostic 08
        - Exit
  - Send the LFHVSET command
    - VOLTAGE parameter = LFHVLOA
    - SEGMENT parameter = A
  - Send the LFHVSET command

        - VOLTAGE parameter = LFHVLOB
        - SEGMENT parameter = B
    - Set LFHVSTATE to 4 in the Housekeeping Data Structure to indicate the high voltage has been commanded to the LOW setting
- If the STATE parameter is NOMA:
    - If Segment A voltage (LFHVDACA) is lower than LFHVNOMA or Segment B voltage (LFHVDACB) is lower than LFHVLOB:
        - Verify high voltage commanding is enabled
            - If commanding is not enabled (LFHVENA = 0):
                - Send diagnostic 08
                - Exit
    - Send the LFHVSET command
        - VOLTAGE parameter = LFHVNOMA
        - SEGMENT parameter = A
    - Send the LFHVSET command
        - VOLTAGE parameter = LFHVLOB
        - SEGMENT parameter = B
    - Set LFHVSTATE to 1 in the Housekeeping Data Structure to indicate the high voltage has been commanded to the NOMA setting
- If the STATE parameter is NOMB:
    - If Segment A voltage (LFHVDACA) is lower than LFHVLOA or Segment B voltage (LFHVDACB) is lower than LFHVNOMB:
        - Verify high voltage commanding is enabled
            - If commanding is not enabled (LFHVENA = 0):
                - Send diagnostic 08
                - Exit
    - Send the LFHVSET command
        - VOLTAGE parameter = LFHVLOA
        - SEGMENT parameter = A
    - Send the LFHVSET command
        - VOLTAGE parameter = LFHVNOMB
        - SEGMENT parameter = B
    - Set LFHVSTATE to 2 in the Housekeeping Data Structure to indicate the high voltage has been commanded to the NOMB setting
- If the STATE parameter is NOMAB:
    - If Segment A voltage (LFHVDACA) is lower than LFHVNOMA or Segment B voltage (LFHVDACB) is lower than LFHVNOMB:
        - Verify high voltage commanding is enabled
            - If commanding is not enabled (LFHVENA = 0):
                - Send diagnostic 08
                - Exit
    - Send the LFHVSET command
        - VOLTAGE parameter = LFHVNOMA
        - SEGMENT parameter = A

  - Send the LFHVSET command
    - VOLTAGE parameter = LFHVNOMB
    - SEGMENT parameter = B
  - Set LFHVSTATE to 3 in the Housekeeping Data Structure to indicate the high voltage has been commanded to the NOMAB setting
- The Housekeeping Packet returned from this command includes the new value of LFHVSTATE

<u>High Voltage Setting (LFHVSET)</u>

This command sets the selected detector segment to the voltage specified. Typically, this command will not be used for operations, the LFHSTATE command will be used instead. When using the LFHVSET command, voltages should not be commanded above the LFHVMAX values.

- If the SEGMENT parameter is B:
  - If the value of the VOLTAGE parameter is > LFHVSETB
    - Verify high voltage commanding is enabled
      - If commanding is not enabled (LFHVENA = 0):
        - Send diagnostic 08
        - Exit
  - Clear Ramp Flag LFHRMPB
  - Check if CRP has been triggered
    - If it has, call CRP_B_INIT, the routine to reset the CRP machine for Segment B
  - Set LFHVSTATE to 6 in the Housekeeping Data Structure to indicate the high voltage has been commanded using the LFHVSET command

  ** *Internal calls to this routine from the HV Ramp Task and CRP Task break in here*
  - If the F0 flag is off:
    - Store the value of the VOLTAGE parameter in the saved register
  - Clear F0 flag
  - Check if the high voltage ramping flag is set
    - If the flag is not set (LFHRMPB = 0):
      - Check the value of the ramp time constant
        - If LFHVRMPT is not zero:
          - Load LFHVRMPT to the timeout counter
          - If the value of the voltage parameter is zero:
            - Write the VOLTAGE value to the DAC Register, setting voltage
          - If the value of the voltage parameter is non-zero:
            - Turn on the Ramp Flag (LFHRMPB = 1)
        - If LFHVRMPT = 0:
          - Write the VOLTAGE value to the DAC Register, setting voltage
    - If the flag is set (LFHRMPB = 1):
      - Write incremented voltage value to the DAC Register, setting voltage
      - Load LFHVRMPT to timeout counter
  - Set LFHVTGTB to the value of the VOLTAGE parameter in the Housekeeping Data Structure
- If the SEGMENT parameter is not B:

  - If the value of the VOLTAGE parameter is > LFHVSETA
    - Verify high voltage commanding is enabled
      - If commanding is not enabled (LFHVENA = 0):
        - Send diagnostic 08
        - Exit
  - Clear Ramp Flag LFHRMPA
  - Check if CRP has been triggered
    - If it has, call CRP_A_INIT, the routine to reset the CRP machine for Segment A
  - Set LFHVSTATE to 6 in the Housekeeping Data Structure to indicate the high voltage has been commanded using the LFHVSET command

  ** *Internal calls to this routine from the HV Ramp Task and CRP Task break in here*

  - If the F0 flag is off:
    - Store the value of the VOLTAGE parameter in the saved register
  - Clear F0 flag
  - Check if the high voltage ramping flag is set
    - If the flag is not set (LFHRMPA = 0):
      - Check the value of the ramp time constant
        - If LFHVRMPT is not zero:
          - Load LFHVRMPT to the timeout counter
          - If the value of the voltage parameter is zero:
            - Write the VOLTAGE value to the DAC Register, setting voltage
          - If the value of the voltage parameter is non-zero:
            - Turn on the Ramp Flag (LFHRMPA = 1)
        - If LFHVRMPT = 0:
          - Write the VOLTAGE value to the DAC Register, setting voltage
    - If the flag is set (LFHRMPA = 1):
      - Write incremented voltage value to the DAC Register, setting voltage
      - Load LFHVRMPT to timeout counter
  - Set LFHVTGTA to the value of the VOLTAGE parameter in the Housekeeping Data Structure
- The Housekeeping packet returned from this command includes new values for LFHRMPA(B), LFHVTGTA(B), LFHVSETA(B) (hardware talkbacks of commanded voltage), LFHVMONA(B) (hardware talkbacks of actual voltage)

FUV Detector Count Rate Protection (LFPCRP)

This command initializes global Count Rate Protection (CRP) processing for Segment A or B. CRP is a bright object protection algorithm that ensures the rate of valid detector events do not exceed a specified threshold. If the threshold is exceeded, the DCE flight software will command High Voltage to the HVLOW state.

- If the SEGMENT parameter is B:
  - Set LFPINTB to the value of the INTERVAL parameter in the Housekeeping Data Structure
    - This value is used by the Count Rate Protection Task
  - Set LFPCNTB to the value of the COUNT parameter in the Housekeeping Data Structure
    - This value is used by the Count Rate Protection Task
  - Call CRP_B_INIT, the routine to reset the CRP machine for Segment B
- If the SEGMENT parameter is not B:

    - Set LFPINTA to the value of the INTERVAL parameter in the Housekeeping Data Structure
        - This value is used by the Count Rate Protection Task
    - Set LFPCNTA to the value of the COUNT parameter in the Housekeeping Data Structure
        - This value is used by the Count Rate Protection Task
    - Call CRP_A_INIT, the routine to reset the CRP machine for Segment A
- The Housekeeping Packet returned from this command includes the new value of the Housekeeping items updated above

4.2.8.2.4 Door Commands

Auxiliary Power Supply On/Off (LFRAXPWR)

This command controls power to the auxiliary power supply. The door and door sensors are powered by this supply.

- If the POWER parameter is non-zero (POWER = ON):
    - Turn on the auxiliary power supply
- If the POWER parameter is zero (POWER = OFF):
    - Turn off the auxiliary power supply
- The Housekeeping Packet returned from this command includes the new value for LFRAXPWR, the Auxiliary Power hardware talkback

Set Auxiliary Power Supply Current Limit (LFRILIM)

This command sets the auxiliary power supply current limit. If the auxiliary power supply current exceeds the limit, the auxiliary power supply will be turned off.

- Set LFRILIM to the value of the LIMIT parameter in the Housekeeping Data Structure
    - This value will be used by the Door Task and the Current Limit Protection Task
- The Housekeeping Packet returned from this command includes the new value for LFRILIM

Door Motor Enable (LFRMENA)

This command enables or disables the primary method of opening the detector door, the door motor.

- Check if door commanding is enabled (LFRMENA = 1)
    - If commanding is enabled:
        - If the DOOR parameter is non-zero:
            - Issue diagnostic 0C
            - Exit
        - If the DOOR parameter is zero:
            - Set LFRMENA to 0 in the Housekeeping Data Structure
                - This value will be used by the Door Task
    - If commanding is disabled:
        - If the DOOR parameter is zero:
            - Issue diagnostic 0D
            - Exit
        - If the DOOR parameter is non-zero:
            - Set LFRMENA to 1 in the Housekeeping Data Structure
                - This value will be used by the Door Task
- Start the three-minute door activity timeout

- The Housekeeping Packet returned from this command includes the new value for LFRMENA

Select the Door Direction (LFRMDIR)

This command selects which direction the door will move when commanded with LFRMPWR.

- Verify door commanding is enabled (LFRMENA=1)
    - If commanding is disabled:
        - Issue diagnostic 0B
        - Exit
    - If commanding is enabled:
        - Start the three-minute door activity timeout
        - If the DIR parameter = 2 (OPEN):
            - Configure the hardware to open the door
                - The door will not move until power is applied with the LFRMPWR command
        - If the DIR parameter = 1 (CLOSE):
            - Configure the hardware to close the door
                - The door will not move until power is applied with the LFRMPWR command
        - If the DIR parameter = 0 (STOP):
            - Configure the hardware to stop the door
        - If the DIR parameter = 3:
            - Issue diagnostic 14 (illegal command)
            - Exit
        - Set LFRMDIR to the value of the DIR parameter in the Housekeeping Data Structure
- The Housekeeping Packet returned from this command will include the new value for LFRMDIR

Move the FUV Detector Door (LFRMPWR)

This command initiates the FUV detector door motion. It must be preceded by a door direction command (LFRIDIR) and a door enable command (LFRMENA). The door motor will move until it activates an endswitch that stops door motion using hardware logic.

- Verify door commanding is enabled (LFRMENA = 1)
    - If commanding is disabled:
        - Issue diagnostic 0B
        - Exit
    - If commanding is enabled:
        - Start the three-minute door activity timeout
        - If the MOVE parameter is START:
            - Power the motor to open the door
        - If the MOVE parameter is STOP:
            - Power off the motor to stop door movement
- The Housekeeping Packet returned from this command includes new values for:
    - LFRPOS (hardware position readback – changes w/ door movement)
    - LFROP (hardware status indicating door open or not-open)
    - LFRCL (hardware status indicating door closed or not-closed)

HOP Actuator Enable/Disable (LFRACTEN)

This command enables or disables the secondary method of opening the detector door, the High Output Paraffin (HOP) actuators.

- Start the three-minute door activity timeout
- If the ACTUATOR parameter is ENABLE:
  - Set the actuator enable bit
    - This value will be used by the Door Task
- If the ACTUATOR parameter is DISABLE:
  - Clear the actuator enable bit
    - This value will be used by the Door Task
- The Housekeeping Packet returned from this command includes the new value for LFRACTEN, the actuator enable hardware talkback

Actuator 1 On/Off (LFRACT1)

This command is used to open the FUV detector door when the primary door method fails. This command powers a heater inside a High output Paraffin (HOP) actuator that releases a spring-loaded mechanism that opens the door. This command must be preceded by LFRACTEN.

- If the POWER parameter is ON:
  - Verify actuator commanding is enabled (LFRACTEN = 1)
    - If commanding is disabled:
      - Issue diagnostic 10
      - Exit
    - If commanding is enabled:
      - Start the three-minute door activity timeout
      - Power the heater for actuator winding 1
- If the POWER parameter is OFF:
  - Start the three-minute door activity timeout
  - Turn off power to the heater for actuator winding 1
- The Housekeeping Packet returned from this command includes the new value for LFRACT1 (hardware talkback)

Actuator 2 On/Off (LFRACT2)

This command is used to open the FUV detector door when the primary door method fails. This command powers a heater inside a High output Paraffin (HOP) actuator that releases a spring-loaded mechanism that opens the door. This command must be preceded by LFRACTEN.

- If the POWER parameter is ON:
  - Verify actuator commanding is enabled (LFRACTEN = 1)
    - If commanding is disabled:
      - Issue diagnostic 10
      - Exit
    - If commanding is enabled:
      - Start the three-minute door activity timeout
      - Power the heater for actuator winding 2
- If the POWER parameter is OFF:

    - Start the three-minute door activity timeout
    - Turn off power to the heater for actuator winding 2
- The Housekeeping Packet returned from this command includes the new value for LFRACT2 (hardware talkback)

<u>HOP Actuator Reset (LFRACTRS)</u>

This command is used to relatch the FUV detector door to the door motor after opening the door using the HOP actuators.

- If the FUNCTION parameter is zero:
    - Perform the door stop function, including:
        - Turn off door motor power
        - Set the door direction to STOP
        - Disable door commanding
        - Turn off actuator power
        - Disable actuator commanding
- If the FUNCTION parameter is non-zero:
    - Verify door commanding is enabled (LFRMENA = 1)
        - If commanding is disabled:
            - Issue diagnostic 0B
            - Exit
        - If commanding is enabled:
            - Check the hardware latch bit (Housekeeping Item LFRLA)
            - If the latch bit indicates the door is latched (LFRLA = 1):
                - This means the actuators never fired:
                    - Send diagnostic 15
                    - Exit
            - If the latch bit indicates the door is unlatched (LFRLA = 0):
                - Override endswitches by calling the Override Door Endswitch Protection Command Function (LFRLSOVD):
                    - OVERRIDE parameter = ENABLE
                - Set the door direction to open by calling the Door Direction Command Function (LFRMPWR):
                    - DIR parameter = OPEN
                - Power the door motor by calling the Motor Power Command Function (LFRMPWR):
                    - POWER parameter = ON
- The Housekeeping Packet returned from this command includes the new values for all Housekeeping items updated above

<u>Override Door Endswitch Protection (LFRLSOVD)</u>

This command overrides the protection against driving the door against endstops. This command is required for relatching the door mechanism with the door motor after opening the door using the HOP actuators. This function is part of the LFRACTRS command, which relatches the door. Therefore LFRLSOVD should not be used under normal operations.

- Start the three-minute door activity timeout
- Set LFRLSOVD to the value of the OVERRIDE parameter in the Housekeeping Data Structure

- If the OVERRIDE parameter is ENABLE (non-zero):
  - Set the bit to enable the override and allow the door to be driven against the endstops
- If OVERRIDE parameter is DISABLE (zero):
  - Set the bit to disable the override so that the door will not be driven against the endstops
- The Housekeeping Packet returned from this command includes the new value for LFRLSOVD

4.2.9 Timing Considerations

This section addresses some of the timing issues concerning the DCE flight software design. These timing issues are not considered to be problems. They are listed here so the user is aware of them. The following is a list of the timing issues that have been identified as a result of the design.

- A housekeeping packet does not get sent to the CS within the same amount of time for every DCE command. For commands that take a short amount of time to process (e.g. no-op), the housekeeping packet is sent to the CS sooner than it would be for DCE commands that take longer to process (e.g. memory dump). Depending on the length of time it takes to process the command, the housekeeping packet is sent back to the CS at different times. See the DCE ICD section 8.5 Packet Protocol for more information.

- The CRC Checking Task is impacted by everything else in the system since it functions like the lowest priority task. If commands are sent to the DCE that take long to process, the CRC calculation is held off. This can result in a CRC calculation taking longer than expected. It also means that a CRC command can take a varied amount of time to complete depending on system loading.

- Power-on Reset requires 10 seconds to complete. Watchdog Reset requires approximately 100 milliseconds to complete. In either case, no housekeeping packet is returned as a result of a reset.

- A maximum 200 millisecond "hit" to Current Limit Protection Task can occur due to the Housekeeping Task taking so long to A/D convert data and output data to the housekeeping interface.

4.3 Internal Hardware Interfaces

This section describes the internal DCE FSW hardware interfaces. Internal interfaces refer to the hardware that is accessed directly by the DCE FSW that allows the external hardware to be controlled. More specifically, internal interfaces refer to the 8051 micro-controller ports and memory mapped I/O address used by the DCE FSW to access the external interfaces.

An understanding of the DCE hardware layout is essential to the design of the FSW. The 8051 communicates with DCE hardware through both ports and memory mapped I/O. The DCE Hardware/Software Interface document fully describes how the DCE FSW is required to control the DCE hardware. It provides a list of all memory mapped I/O locations and ports along with the hardware to which they interface. The following sections describe 8051 ports and memory mapped I/O in more detail.

4.3.1 8051 Ports

Ports are 8-bit latches that provide hardware control similar to memory mapped I/O (e.g. flipping a bit in a hardware latch). They are not, however, addressed using external memory addresses. They are accessed internal to the 8051 using a series of 256 registers. Each register has a unique address (0-255).

The ports are mapped to internal 8051 address space. To access a port (or port register), special 8051 instructions are used to access a port's memory location (read/write). These instructions are different than those used to access MMIO.

The software can access individual bits by using a bit address, rather than a port address. The bit address can be used to toggle individual bits in the various port registers whereas port registers access 8-bit values. The bit addresses essentially overlay the port registers. Only a certain range of port registers can be accessed using bit addresses. The bit addresses are also set/cleared using a special 8051 machine language instruction that is different from the port register access instructions.

Figure 4.3-1 provides an overview of how ports are mapped to internal 8051 memory. It shows how accessing different internal memory ranges provides different types of access to the ports. Figure 4.3-2 explodes the 0x80 through 0xFF port address range and shows the specific registers that exist within that memory range. The MCS 51 Micro-controller Family User's Manual should be consulted for specific information about 8051 ports.

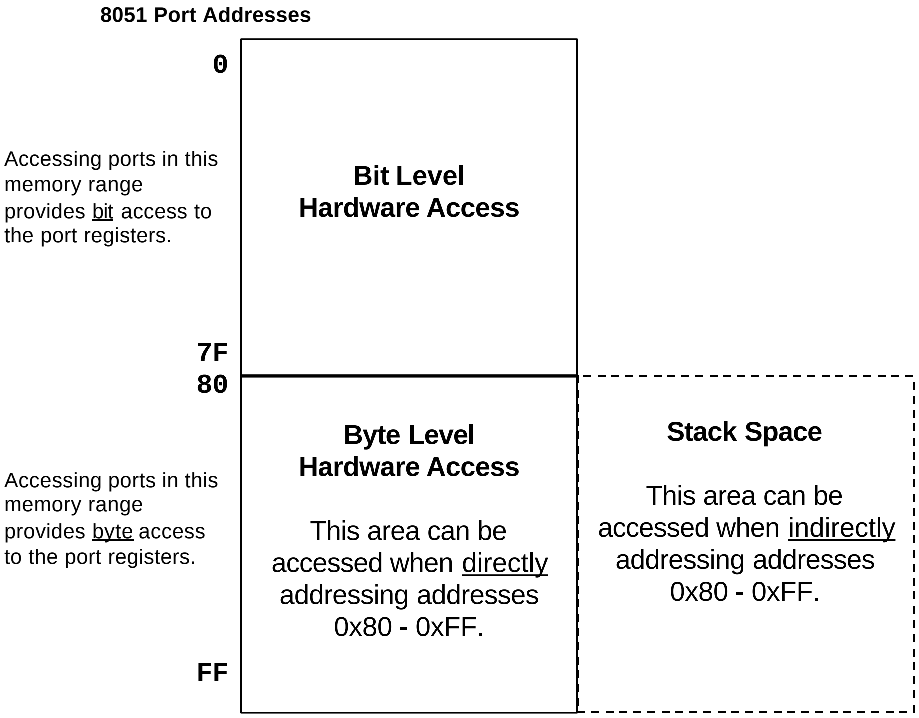

8051 Port Addresses
0
Accessing ports in this memory range provides bit access to the port registers.
Bit Level Hardware Access
7F
80
Byte Level Hardware Access
This area can be accessed when directly addressing addresses 0x80 - 0xFF.
Stack Space
This area can be accessed when indirectly addressing addresses 0x80 - 0xFF.
Accessing ports in this memory range provides byte access to the port registers.
FF

Figure 4.3-1. Port Mapping of Internal 8051 Memory

F8		CH	CCAP0H	CCAP1H	CCAP2H	CCAP3H	CCAP4H		FF
F0	B								F7
E8		CL	CCAP0L	CCAP1L	CCAP2L	CCAP3L	CCAP4L		EF
E0	ACC								E7
D8	CCON	CMOD	CCAP0M	CCAP1M	CCAP2M	CCAP3M	CCAP4M		DF
D0	PSW								D7
C8	T2CON	T2MOD	RCAP2L	RCAP2H	TL2	TH2			CF
C0									C7
B8	IP	SADEN							BF
B0	P3							IPH	B7
A8	IE	SADDR							AF
A0	P2								A7

98	SCON	SBUF							9F
90	P1								97
88	TCON	TMOD	TL0	TL1	TH0	TH1			8F
80	P0	SP	DPL	DPH				PCON	87

Figure 4.3-2. 8051 Internal Registers Providing Byte Access

The following is a list of DCE hardware that is controlled using ports:

- *Digitizer 3-Wire Interface* - Provides digitizer control.
- *Lock Bit* - Locks and unlocks the protected control register.
- *Format Bit* - Currently unused in COS, however, it is used during test for timing the software. The name “format bit” is legacy from FUSE.
- *B0/B1 Memory Select Bits* - External memory bank select. Effectively allows memory ranges to be re-mapped. This means that by configuring the bits a certain way, you can change “code” address space from PROM to RAM without changing the executable image.
- *Pulse Height Memory Disable Bit* - Disables access to the PH memory by the science data (PH) processing hardware. This is done while the PH memory is being accessed by the 8051 to avoid both the 8051 and PH hardware from accessing the PH memory at the same time.
- *Serial I/O Port Communication* - Used by the Inspect & Change Task if installed. COS does not expect to use this port.
- *Self-Reset Bit* - Allows the DCE flight software to reset itself.
- *Timer* - The 8051 has 3 internal timers (0, 1 and 2). Timers 0 and 1 are used in series to provide the timer tick.
- *Interrupt Controller* - The 8051 has an internal interrupt controller with 8 hard-wired interrupts.

4.3.2 Memory Mapped I/O

Most control of the DCE is supported by memory mapped I/O (MMIO). Hardware resources are allocated specific addresses in external memory. When a MMIO location is written, a register in the DCE hardware is updated. When an MMIO location is read, a DCE hardware register supplies data from the DCE hardware. Memory mapped I/O locations do not access memory in the traditional sense. They access hardware registers used to control and status the DCE hardware.

4.3.2.1 Memory Mapped I/O Controlled Hardware

The following is a list of the DCE hardware controlled using memory mapped I/O. The DCE Hardware/Software Interface document contains all of the details associated with the control and status of the DCE hardware using memory mapped I/O locations.

- Counters A & B (read)
- PH Data A & B (read)
- Door Status and HV Power Status (read)
- ADC Digitized Data (temperatures/volts/current/power) (read)
- MUX Select input to ADC (write)
- Unprotected control register (write)
- Protected control register (write) - control bits for door and HV
- DAC HV Settings, Max (write)
- Rx Registers for Command traffic (read)

- Tx Registers for Housekeeping traffic (write)
- LEDs (write) - primarily for GSE use

4.4 Processing Hardware

4.4.1 8051 Micro-controller

The 8051 micro-controller is used to control the DCE because of its reliability and thermal qualities. Its simplicity is well suited to the functions allocated to the software.

The DCE uses the UT69RH051 (8051), which is a close variant of the 8051FC micro-controller. Software for the 8051 is installed in the DCE as a PROM device, which the 8051 reads from address 0 in the event of any reset. The 8051 executes code as long as it is powered and the clock is running. The 8051 has no idle state. The MCS 51 Micro-controller Family User's Manual should be consulted for specific information about the 8051. Additional information concerning the 8051FC variant should be obtained from a production specification.

4.4.1.1 8051 External Memory Addressing

4.4.1.1.1 Harvard and Von Neuman Architectures

The DCE hardware is designed to use a combination of Harvard and Von Neuman computer architectures. It is important to understand the difference between these two architectures to fully understand the DCE FSW design.

The organization of software code and data in the 8051 is known as a Harvard Architecture, whereby code and data each occupy a distinct 64K area of memory with both areas having the same address space (0 to 0xFFFF). For example, a piece of executable code can start at address 0x100 and a variable can also exist at address 0x100. The 8051 knows to access code memory when it fetches an instruction op-code, and to access data memory when it reads or writes a variable.

The DCE memory layout above addresses 0x8000 are designed as Von Neuman Architecture in which code and data occupy the same physical region of memory. In this region of memory, the code and data have different address ranges since they occupy the same physical memory space.

See figures 4.4-1 and 4.4-2 for more information concerning the DCE memory map.

4.4.1.1.2 Interrupt Vectors

ISRs for both Boot and Operate are located in PROM due to the 8051 architecture. The 8051 does not allow ISR vectors to be dynamically relocated. ISR vectors must exist at physical address 0x0. This needs to be understood since it has a profound impact on the design of the Boot and Operate software. Without special bank memory switching (see below), this limitation means that the ISRs running in Operate are actually the ISRs contained in PROM.

4.4.1.1.3 Special Bank Memory Switching

It is expected that the Actel will allow re-mapping of the unused portions of the 8K RAM space. The 8K RAM memory area is actually a 32 KB device with only 8 KB currently being used. By modifying the Actel to allow real-time re-mapping of memory, the PROM space can be re-mapped to this unused RAM space. This would allow the ground to upload new ISR code for use by the Operate code. The existing hardware design has the limitation of not allowing ISR code to be changed in flight for the Operate code.

4.4.1.2 8051 Internal Registers

The 8051 micro-controller contains 256 (0x100) addressable registers. The terms "internal register" and "internal memory" and "ports" are used interchangeably when referring to the 8051.

Not all of the registers above address 0x80 are implemented, and most of those directly control the 8051 itself. Such registers include timers, serial port data, interrupt control, and port bits as well as standard general purpose registers R0 - R7 and accumulator register, etc. For a complete discussion of 8051 internal registers, see the MCS 51 Micro-controller Family User’s Manual.

This section is included here since internal registers are integral to the 8051 architecture. Form more detailed information on 8051 internal registers, see section 4.3.1 of this document.

4.4.2 External Memory

4.4.2.1 PROM

The programmable read-only memory (PROM) device is programmed and installed in hardware. It contains the Boot code that executes when power is applied to the DCE hardware. Since the Boot code is burned into PROM, it means that the code cannot be changed in flight. Due to the design of the Boot and Operate code images, the PROM also contains a version of Operate code as well. The Boot/Operate image can be copied to RAM and executed on command.

ISRs for both Boot and Operate are located in PROM due to the 8051 architecture. The 8051 does not allow ISR vectors to be dynamically relocated. ISR vectors must exist at physical address 0x0. This means that Operate ISRs are actually running from PROM.

4.4.2.2 RAM

There are two random access memory (RAM) devices named 8K RAM and 32K RAM. As a design choice, memory is allocated to specific purposes somewhat arbitrarily: data passing to and from the CS (i.e. command and housekeeping traffic) has been located in 8K RAM; code space, global variables, private data and A/D samples are all stored in certain locations in 32K RAM. The 8K RAM is completely initialized with zero values upon power-on reset only. The 32K RAM is never initialized as the result of a reset.

Nearly all memory in the DCE is statically allocated. Every memory object has a dedicated location in DCE memory. Previously allocated memory locations are generally not used again.

4.4.2.2.1 Contents of 8K RAM

Upload Buffer Stores upload data passed as command traffic to this 1K buffer.

Command Buffer Stores incoming command traffic in a small region of this 1K buffer.

Download Buffer Data is copied to this buffer in the event of a download command, which then copies the entire download buffer out the housekeeping channel as a special download housekeeping packet.

Housekeeping Buffer Housekeeping allocates a memory location for up to 1024 bytes of data. Housekeeping memory is the source of housekeeping traffic data and address tags. Many housekeeping memory locations remain undefined in the FSW, but the Housekeeping Task always transmits the complete 512-word contents of memory.

4.4.2.2.2 Contents of 32K RAM

Lower Code Area LCA is defined as the address space used to store the Lower Code Operate Image.

Upper Code Area UCA is defined as the address space used to store the Upper Code Operate Image.

Current Samples HV A, HV B and Auxiliary current. All values monitored by the Current Limit Protection Task. Circular queue of the last 1024 current samples along with a histogram containing data since it was last cleared.

Variables Memory locations are reserved for variables used by certain tasks. The variable space is used for data structures that are too big or not appropriate for storage in the Housekeeping data region. For example, the count rate protection table is included in this section, rather than in housekeeping.

4.4.2.3 Memory Map

There are three general types of memory available for flight software use in the DCE: PROM, RAM and Memory Mapped I/O. Figure 4.4-1 graphically represents the DCE memory map.

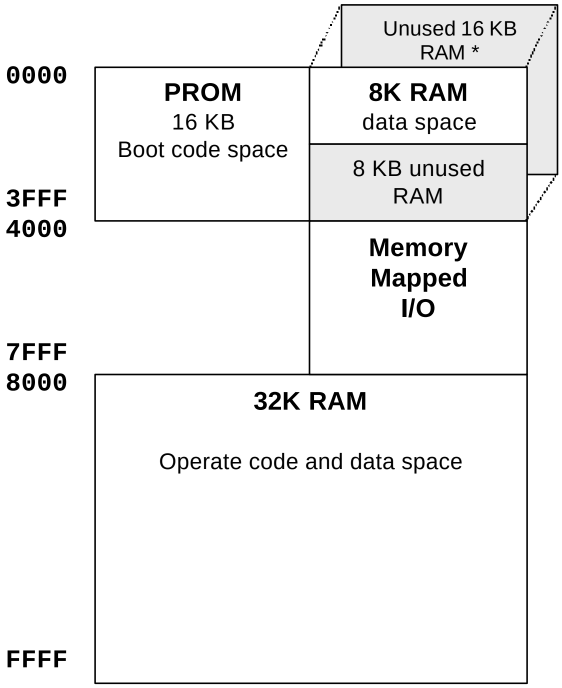
- PROM is 16 KB of non-volatile and non-modifiable memory that is used to hold and execute the DCE Boot code image.

- RAM is physically partitioned into two non-contiguous segments of volatile memory. One segment exists at addresses 0x0000 to 0x1FFF (8K RAM) and the other at addresses 0x8000 to 0xFFFF (32K RAM). The 8K RAM segment is used as data memory. The 32K RAM segment is used to store the DCE Operate code and variables. It should be noted that the 8K RAM memory area is actually a 32 KB device with only 16 KB currently mapped in the hardware, and only 8 KB of the 16 being used.

- Memory mapped I/O resides in an area of reserved addresses. Selected locations in this address range are used to write control data to and read status from the DCE hardware. These contents of these memory locations are described in the DCE Hardware/Software Interface document.

The DCE hardware distinguishes between memory that stores *code* and memory that stores *data*. In the address space 0x0000 to 0x1FFF, both PROM and RAM are located at the same physical memory addresses. The DCE hardware determines which physical memory to use based on the type of memory access. It accesses PROM when it fetches an instruction for executing code, and it accesses RAM when it reads or writes a program variable. The DCE hardware design can support program execution from this RAM space with the implementation of special memory bank switching (see section 4.4.1.1.3).

The 32 RAM segment is set up in the DCE hardware to be used as either code memory or data memory. This allows the Operate code image to be written to this area of RAM and then executed. The DCE FSW divides this 32 RAM into four logical areas (see Figure 4.4-2). Two of these areas are reserved for loading Operate code images and are designated *Lower Code Area* and *Upper Code Area*. The other two areas are reserved for FSW data and are designated *Variables* and *Current Samples*.



* This 16 KB RAM space is only accessable by using special memory bank switching.

Figure 4.4-1. DCE Memory Map

Memory Usage	Address
Variables	8000 8FFF
Lower Code Area	9000 BFFF
Current Samples	C000 CFFF
Upper Code Area	D000 FFFF

Figure 4.4-2. DCE 32K RAM Memory Allocation

4.4.3 Hardware Environment

The following tables summarizes the major hardware components that form the hardware environment in which the DCE FSW resides:

ITEM	DESCRIPTION	COMMENTS
Microprocessor	UT69RH051 (radiation hardened 8051FX)	16 MHz
PROM	16 KB	Boot code and GSE Test Operate code image
RAM	32 KB	Operate code image and data
Low RAM	8 KB of data space (actually a 32 KB device with 16 KB physically mapped)	Command and Housekeeping data including data transfer areas

5 DEVELOPMENT ENVIRONMENT

Since there is so much reuse associated with the DCE FSW, a section describing the development environment is appropriate. The DCE FSW is being developed from heritage FUSE and GALEX software. While there are some important differences between FUSE and GALEX and COS, efforts have been made to keep the software similar if not the same.

5.1 Programming Language

All of the DCE flight software is written in 8051 Assembly Language. All of the DCE flight software is custom built with no COTS libraries being used.

5.2 Software Tools

The following table summarizes the software packages that form the software environment in which the DCE FSW is developed:

ITEM	DESCRIPTION	COMMENTS
Configuration Manager	Pkzip	Every tested version saved
8051 Assembler	Archimedes A51	Archimedes
Linker	Archimedes L51	Archimedes
Object-to-Hex	Archimedes OH51	Archimedes
8051 In-Circuit Emulator	High-Tex ICE	
Hex2cmd	Creates upload script	EAG
Hexmv	Moves load addresses to PROM space	Required for creating PROM burnable code images.

5.2.1 DCE Flight Software Build Process

The DCE flight software build process consists of a number of steps that start with source code files and end up with ABS files containing executable code. Figure 5.2-1 below depicts the process of building executable code from source code using the set of DCE flight software development tools.

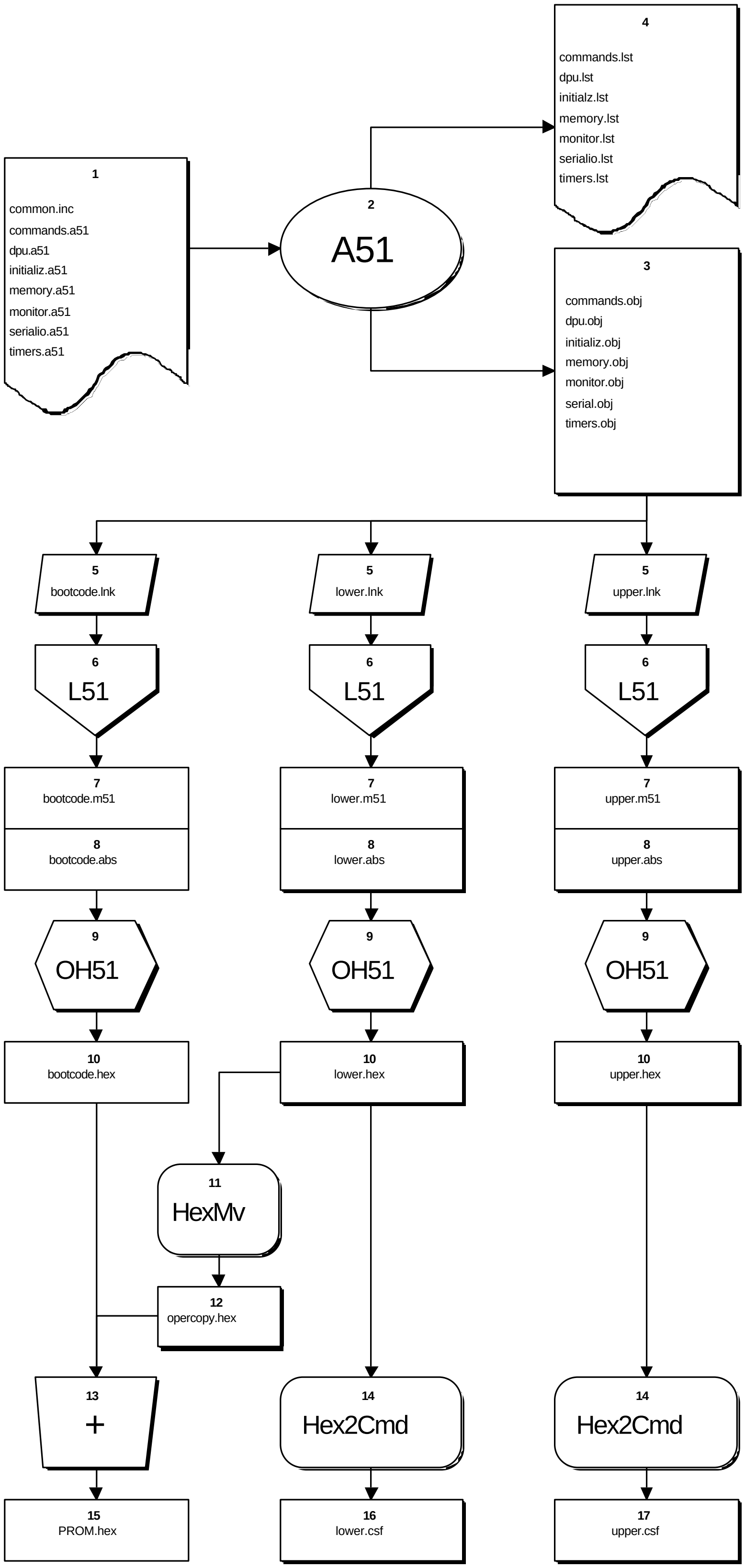


Figure 5.2-1. DCE FSW Build Process

The following is a list of *build notes* that apply to the flight software build process described in Figure 5.2-1. Each numbered item below corresponds to a numbered box in Figure 5.2-1.

1. Source files contain assembly code organized into modules. The file “common.inc” is referenced by all source files and contains module interface and hardware descriptions.
2. The 8051 Assembler reads the source files and produces object code and listings.
3. Object code files contain code text and symbol information.
4. Source listings are useful in understanding what code was produced by the assembler and where in each module that code and data are located.
5. Linker parameters (program arguments) specify where code modules and data are to be located in 8051 memory spaces. There is a set of linker parameters specified for each executable image.
6. The linker/locator program (L51) reads the linker parameter file and any specified object code files.
7. The map file produced by the linker describes where each code module and data areas are located in memory. To determine where any particular code or data item is located in 8051 memory, one must consult with this map in conjunction with the corresponding source listing file.
8. The linker also produces an “ABS” file that embodies an executable image and a map file.
9. The Object-to-Hex utility converts the abs file to hex format.
10. Hex format is a text file, more readable than abs, and used by various software tools.
11. UCB-created tool relocates the text of code to load at an address 16k-bytes lower in memory, which allows us to load a lower-code image executable into PROM space. A FSW software function is required to move the code back to its designed location on command.
12. HexMv creates an intermediate file of "lower code", which can be loaded into PROM space.
13. Bootcode and the intermediate file are combined into a single Hex file.
14. A UCB-created tool converts Hex files into UCB EGSE format for upload to RAM.
15. PROM Hex image that can then be burned into the PROM within the DCE.
16. Uploadable lower code image format suitable for UCB EGSE.
17. Uploadable upper code image format suitable for UCB EGSE.

The following table lists the output products of the software build process:

.LST Assembled listing. Useful for locating any software entity (like addresses of variables or the text of 8051 code). It also contains a copy of the source code.

.M51 Locator map. Useful in identifying the starting address of a software module and in locating the address of a software entity.

.CSF Uploadable script in UCB eGSE format.

.HEX Text code image used in the creation of PROMs.

.ABS OMF formatted file containing all executable code and symbol information.

5.2.2 Interface to SI Memory Manager

The SI Memory Manager (SIMM) is the tool used to update software versions in the science instrument. The SIMM tool is capable of creating load files that can update the CS EEPROM with a new DCE Operate FSW version. The primary input to the SIMM tool is an ABS file that contains the executable code and symbol information about the FSW. The DCE FSW development system provides an ABS file containing both executable code and symbol information to the SIMM tool.

6 DESIGN REQUIREMENTS TRACEABILITY

The following table contains a list of all DCE FSW requirements and the Computer Software Component (CSC) that fulfills the requirement.

Req't	Title	CSC
	Detector Initialization	
5.1.1.1	Initialize to Boot State After Reset	Reset and Initialization CSC
5.1.1.2	Indicate the Cause of the Reset	Reset and Initialization CSC
5.1.1.3	Commands to Reboot DCE FSW	Command Processing CSC; Reset & Init. CSC
5.1.1.4	Memory Initialization on Power-Up Only	Reset and Initialization CSC
5.1.1.5	Code In PROM (Non-Volatile Memory)	
	Error Detection	
5.1.2.1	Watchdog	
5.1.2.2	Command to Disable/Enable Watchdog	Command Reception and Processing CSC
5.1.2.3	Capability to Log Diagnostics	Housekeeping Collection and Reporting CSC
5.1.2.4	HST Error Logging	Housekeeping Collection and Reporting CSC
5.1.2.5	HST Error Format	Housekeeping Collection and Reporting CSC
	Command Reception and Processing	
5.2.1.1	Error-Detecting Command Format	Command Reception and Processing CSC
5.2.2.1	Command Rate: One Per Second	Command Reception and Processing CSC
5.2.3.1	Housekeeping Response Within 1 Second	Housekeeping Collection and Reporting CSC
5.2.3.2	Echoes for Command Op-code and Parameters	Housekeeping Collection and Reporting CSC
5.2.3.3	Counters for All Commands	Command Reception and Processing CSC
5.2.3.4	PHD Handling	Housekeeping Collection and Reporting CSC
5.2.4.1	Command to Jump Anywhere in Code	Support Functions CSC
	Detector Operations	
5.3.1.1	HV Ramping Function	Detector Control CSC
5.3.1.2	Commands to Set Digitizer Gains and Offsets	Detector Control CSC
5.3.1.3	Commands to Control High Voltage	Detector Control CSC
	Detector Protection	
5.3.2.1	Commands to Enable/Disable Global Rate Monitoring	Global Rate Protection CSC
5.3.2.2	Commands to Set Global Rate Monitoring Parameters	Global Rate Protection CSC
5.3.2.3	HV Current Over-Limit Checking	Current Limit Checking CSC Global Rate Protection CSC
5.3.2.4	HV Interlocks	Detector Control CSC
5.3.2.5	Take Protection Measures if Limit Exceeded	Current Limit Checking CSC
5.3.2.6	Command to Enable/Disable Limit Checking	Current Limit Checking CSC
	Door Control	
5.4.1.1	Door Direction	Door Control CSC

5.4.1.2	Door Power	Door Control CSC
5.4.1.3	Actuator Power	Door Control CSC
5.4.1.4	Door Override	Door Control CSC
5.4.1.5	Settable Door Current Limit	Door Control CSC
	Door Protection	
5.4.2.1	Door Timeout	Door Control CSC
5.4.2.2	Door Enable	Door Control CSC
5.4.2.3	Aux. Current Checking During Door Motion	Door Control CSC Current Limit Checking CSC
5.4.2.4	Latch Checking	Door Control CSC
	Memory Management	
5.5.1.1	Process Memory Uploading Commands	Support Functions CSC
5.5.1.2	Capability to Upload Data	Support Functions CSC
5.5.1.3	Check Upload Integrity	Support Functions CSC
5.5.1.4	Command to Disable/Enable CRC Check on Upload	Support Functions CSC
5.5.2.1	Process Memory Download Commands	Support Functions CSC
5.5.2.2	Download Data Timing	Support Functions CSC
5.5.3.1	CRC Background Check On Memory Regions	Support Functions CSC Background Monitoring CSC
5.5.4.1	HST Memory Monitors in Housekeeping	Housekeeping Collection and Reporting CSC
5.5.4.2	HST Memory Monitor Commanding	Support Functions CSC

7 ACRONYMS AND TERMS

Aux	Auxiliary Power
COS	Cosmic Origins Spectrograph
COTS	Commercial off the shelf
BATC	Ball Aerospace & Technologies Corp.
CRC	Cyclic Redundancy Check
CSC	Computer Software Component
CSCI	Computer Software Configuration Item
DCE	FUV Detector Control Electronics
EAG	Experimental Astrophysics Group, Space Sciences Laboratory, UC Berkeley
FSW	Flight software
FUSE	Far Ultraviolet Spectrographic Explorer
FUV	Far Ultraviolet
GSE	Ground Support Equipment
GSFC	Goddard Space Flight Center
HST	Hubble Space Telescope
HV	High Voltage
HVPS	High Voltage Power Supply
Hz	Hertz, cycles per second
ICD	Interface Control Document
ICE	In-Circuit Emulator
I/O	Input/Output
I&T	Integration & Test
LVPS	Low Voltage Power Supply
MEB	Main Electronics Box
NASA	National Aeronautics and Space Administration
PROM	Programmable Read-Only Memory
PHD	Pulse Height Distribution
RAM	Random Access Memory
ROM	Read-Only Memory
SEU	Single Event Upset

SRD	Software Requirements Document
TDC	Time-to-Digital Converter
UCB	University of California, Berkeley