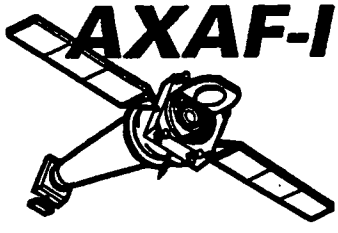
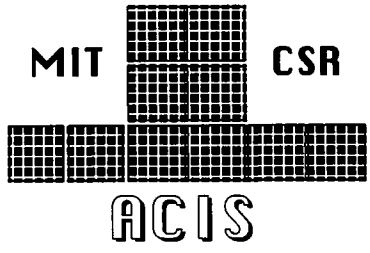
CSR





Advanced X-ray Astrophysics Facility

AXAF - I CCD Imaging Spectrometer

EGSE Software Development Plan

Submitted to:

George C. Marshall Space Flight Center
National Aeronautics and Space Administration
Marshall Space Flight Center, AL 35812

Submitted by:

Center for Space Research
Massachusetts Institute of Technology
Cambridge, MA 02139

AXAF-I CCD Imaging Spectrometer
(ACIS)

EGSE Software Development Plan

Document No. 36-02403-01

Contract # NASA-37716

February 16, 1995

Submitted to:

George C. Marshall Space Flight Center
National Aeronautics and Space Administration
Marshall Space Flight Center, AL 35812

Submitted By:

Massachusetts Institute of Technology
Center for Space Research
77 Massachusetts Avenue
Cambridge, MA 02139

Approvals:

Robert Goeke
Deputy Project Manager
Massachusetts Institute of Technology

Philip J. Gray
Project Manager
Massachusetts Institute of Technology

CSR

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
CENTER FOR SPACE RESEARCH
CAMBRIDGE, MASSACHUSETTS 02139

REVISION LOG	TITLE: Materials & Processes Selection & Verification Plan	DOC. NO. 36-02403

Revision	Date (mm/dd/yy)	ECO No.	Section(s) Affected	Reason	Approval
01	02/15/95			Preliminary Release	

TABLE OF CONTENTS

1.0 INTRODUCTION;

1.1 Purpose

The purpose of this Electronic Ground Support Equipment (EGSE) Software Development Plan is to define the responsibilities, techniques, procedures, and methodologies that will be used at the Center for Space Research (CSR) to assure that the use of the EGSE will not damage the ACIS instrument and in addition will not corrupt any data that is used in the calibration or t validation of the ACIS instrument.

1.2 Scope;

This document is written in response to Section 3.4.1.3 of the AXAF Level II Project requirements Document (PRD) and Section 2.0 paragraph 3. of the Software Quality Assurance Requirements for MSFC Projects (CQ 5330.1A) The requirements of the MSFC Software Management and Development Requirements Manual (MM 8075.1). The tailored requirements found in CQ-5530.1A are used as a guide in the format of this document. Appendix A contains the tailored matrix for CQ-5530.1A.

2.0 APPLICABILITY

The processes described in this plan are used in the development of the EGSE which is used in the support of component, calibration, system level, verification and acceptance testing of the Advanced X-ray Astrophysics Facility (AXAF) Charge Coupled Device (CCD) Imaging Spectrometer (ACIS) instrument. In this document, a component is defined as major subassembly such as the Digital Processing Assembly or the (DPA) or the Detector Electronics Assembly) (DEA) This plan is in effect until the ACIS instrument is accepted by MSFC.

2.1. Applicable Documents

The following documents can be used as requirements for the design and manufacture of EGSE and form a part of this document to the extent specified herein. The issue in effect at the time of contract award shall apply unless otherwise listed below.

2.1.1 Non MIT Documents

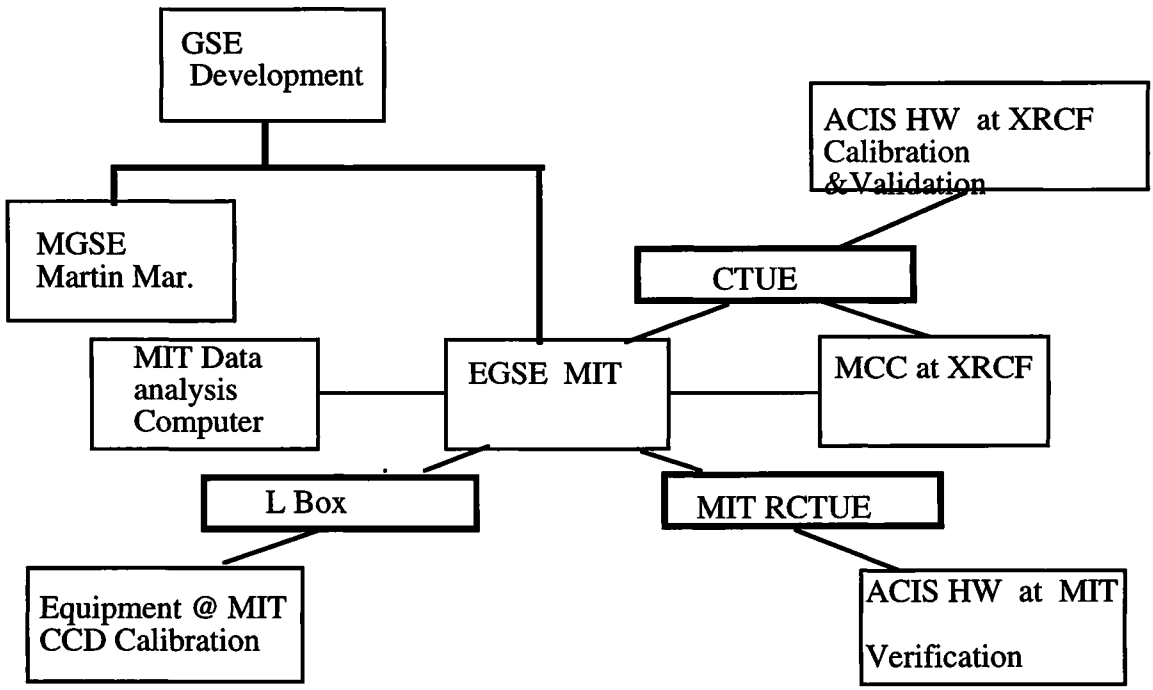
MM-8075.1	MSFC Software Management and Development requirements Manual (Jan. 22, 1991)
CQ-5330.1A	MSFC Software Quality Assurance Requirements for MSFC Projects (Rev A)
MSFC-SPEC-1836B	AXAF Level II project Requirements Document (PRD)
TRW 52100.200.92.0078	TRW AXAF HRMA & SI X-ray Calibration Plan (TXCP)

2.1.2 MIT Documents

ACIS 36-01102-02	ACIS GSE CEI Specification
ACIS 36-01103-03	ACIS SI Software Requirements Specification
ACIS 36-01203	ACIS Verification and Validation Plan
ACIS 36-01206	ACIS Configuration Management Plan (CMP)
ACIS-36-01215	ACIS Software Test Plan (STP)
ACIS 36-01410-03	ACIS Instrument Program & Command List
ACIS 36-02104-04	ACIS DPA Hardware Spec. & System Description

3.0 OVERVIEW

The EGSE is intended to be used in support of component, calibration, and system testing efforts. Support includes providing and interface at system level to provide commands to and accept telemetry from the CTUE. The EGSE also provides an interface to the test conductor that ensures safe control of the instrument. Additional EGSE requirements are specified in the EGSE CEI Specification.



EGSE Interface Definition

Figure - 1 shows what the EGSE is connected to during various stages of ACIS development. In addition it shows how the GSE development effort is split between MIT and Martin Marietta (MM).

1. While at MIT, the EGSE is used to calibrate individual CCDs via a dedicated interface called the lasagna box (L- box).
2. It could be connected to the ACIS instrument through a MIT developed RCTUE which look like a CTUE for some functions. Functions handled by MM are not implemented in the RCTUE. This allows for testing of ACIS via commands and telemetry.
3. The interface at XRCF includes two addition al computers. The Master Control Computer connected via ethernet and a MIT science branch developed computer connected via ?? called the Analysis Computer. ACIS is connected via ?? supplied CTUE.

3.1 Organization

The organization of the ACIS project at CSR consists of three branches. This organization is shown in Figure 2. The science branch at CSR is headed by the ACIS Deputy Instrument Principal Investigator. The engineering branch is headed by the ACIS Project Manager (PM). EGSE development is the responsibility of the Project Engineering function (PE) with interface issues involving other institutions coordinated by the ACIS System Engineer (SE). The Performance Assurance branch is headed by the Performance Assurance Manager (PAM) with assistance from a Software Quality Assurance (SQA) engineer. The EGSE development effort is being managed by the PE.

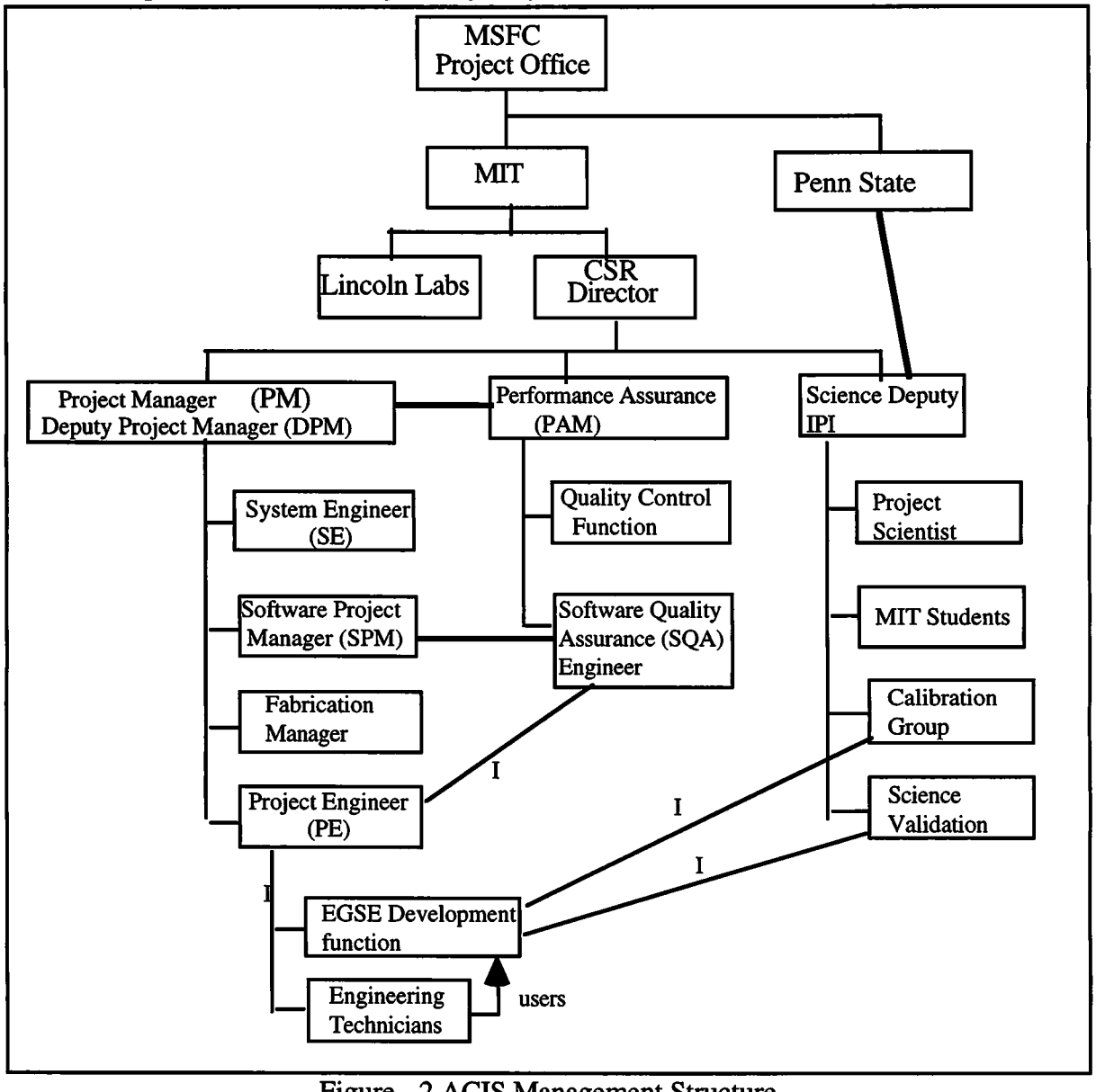


Figure - 2 ACIS Management Structure

3.2 Requirements;

The requirements for the EGSE come from the TRW Calibration plan and the MIT documents identified in section 2.1.2 above. The EGSE in intended to be used only by MIT scientists, engineers or technicians.

2.1 Crucial Requirements

Some requirements have more impact on the ACIS mission if they are not implemented correctly. In effect they are mission crucial. Such requirements involve the controlling of the focal plane door. The implementation of these requirements are noted throughout this document as "crucial".

4.0 SOFTWARE DEVELOPMENT CYCLE

The EGSE is developed using an integrative development process. That is, support functions are derived from requirements and are implement in an incremental fashion as ACIS test, validation and calibration requirements are defined. Software will be developed using a structured approach. This development process uses the following development phases.

41 Requirements Phase

During this development phase, the developer interviews the requester of support and documents the interview . This phase may be performed a number of times until the particular support function has been demonstrated or validated.

4.2 Construction Phase

The developer uses the requirements documentation to design and implement the support function. All designs involving crucial requirements will be documented. From the design, the function is coded and tested. Code involving crucial functions will be unit tested. The results of each functions unit test will be documented.

4.3 Verification and test

Tests that are performed to validate the EGSE will be documented along with the values of parameters used in the test. Crucial functions will be verified at limits specified in the requirements.

4.5 Operations and Maintenance

Support functions for component test as specified by the development engineers will be demonstrated prior to TBD with sufficient time to incorporate the most recent component design. Support functions for ACIS system test will be available for use as ACIS hardware and software is available to accept commands and produce telemetry. Support functions for calibration requirements as specified by the SCIENCE branch and will be available TBD. Validation testing consists of long and short form tests. The EGSE will be ready to support validation when sufficient hardware and software is available. Although the EGSE will be available after calibration at XRCF, its role has yet to be defined.

5.0 CONFIGURATION MANAGEMENT

The purpose of configuration management is to enable the reliable rebuilding and identification of current and the previous versions of EGSE executable software. In addition, configuration management of test scripts and ACIS configurations will help ensure consistent test results.

5.1 EGSE Software

The software for EGSE consists of source code, build files, libraries, development tools and executables. The primary means of controlling the EGSE software is by use of a 'string' constant

consisting of the particular EGSE software part number and version or date/time stamp. This 'string' constant will appear in the executable code so that it is observable from a core dump of memory. In addition, these shall be a user friendly mechanism to query the EGSE for this 'string' constant. The 'string' constant will be unique for each release of the software. In order to ensure consistency, versions of all software and software tools used in constructing the EGSE will be recorded for each release of EGSE software.

5.2 ACIS Test scripts and CCD Calibration test procedures

The EGSE shall provide a repository for test scripts and configuration parameters that is controlled. Only after these scripts and ACIS configurations have been tested and approved by the PE will they be stored. Test scripts and configurations shall also be identified with part number and version.

5.3 ACIS and CCD Calibration and Validation Data

The test results of engineering validation scripts will be stored. A mechanism for identifying which script generated the data and when the test was run shall be developed by the EGSE designer.

6.0 APPROVAL AUTHORITY

The Deputy Project Manager (DPM) has approval authority for each release of software used in the EGSE support function for ACIS. In addition the PE approves the initiation of EGSE requirement implementation or change.

7.0 GROUP CHARTERS AND RESPONSIBILITIES

Requirements for the EGSE can come from the Science, Quality Assurance, various ICDs or Engineering branches of . ACIS development.

- The component design group consists of the engineers responsible for designing the DPA, DEA, and CCD components. They specify the requirements and recommend to the DPE weather the delivered EGSE support function should be approved.

- The Quality Assurance Branch specifies the EGSE system support requirements for ACIS testing. The Software Quality Assurance (SQA) engineer recommends to the DPE weather the delivered support function should be approved. SQA will assure that EGSE software is developed in accordance with this plan. In addition SQA will generate EGSE support function requirements necessary to support the system test of ACIS.

- The Project Engineering Function (PE) receives a copy of all ICDs. The PE ensures that the EGSE developer reviews appropriate ICD and incorporates applicable requirements..

- The Science Branch, ACIS Calibration section, specifies the EGSE support function necessary for CCD and ACIS instrument calibration. They recommend to the DPE weather the delivered EGSE support function should be approved.

- The Science Branch ACIS Science Users Section, specify the EGSE support function necessary for validation of science application requirements. It is anticipated that an additional workstation will be connected to the ethernet They recommend to the DPE weather the delivered EGSE support function should be approved.

8.0 FACILITIES AND RESOURCES

The EGSE consists mainly of a Sun Sparc workstation and some included device driver hardware and software. It is anticipated that this is all that is needed to develop the support functions for EGSE. This implies that all software tools will reside within the workstation.

9.0 DOCUMENTATION

9.1 The requirements , design, and EGSE test results will be documented in an EGSE Project Notebook. User documentation will consist mainly of appropriate help screens. Provision will be made by the EGSE design to print these screens.

9.2 Computer generated ACIS Validation test results will be archived in a directory . These test results will be retrievable by some test identification or date/time scheme TBD.

10.0 PROGRAM MANAGEMENT AND PLANNING

This section establishes and presents EGSE management policies with regard to EGSE requirement definition, construction ,verification & test, operations & management. Technical coordination and review is included in each of the following sections.

10.1 Requirement Definition

During this development phase, the developer interviews the requester of support and documents the interview in a EGSE project notebook. All EGSE requirements will be reviewed by the PE who will ensure that all crucial requirements are identified.

10.2 Construction Phase

The developer uses the notes taken during the requirements phase to design and implement the support function. All designs involving crucial requirements will be documented. The PE will ensure that the design is reviewed. From the design, the function is coded and tested. Code involving crucial functions will be unit tested. The results of each functions unit test will be documented. The PE will ensure that the units test documentation has been audited by Software Quality Assurance.

10.3 Verification and Test

Tests that are performed to validate the EGSE will be described in the project notebook along with the values of parameters used in the test. Crucial functions will be verified at limits specified in the requirements.

10.4 Operations and Maintenance

The major EGSE support functions are component test, ACIS systems test, calibration and validation. Support functions for component test will be specified by the development engineers. The EGSE will be available for component test when the master ACIS schedule indicates the begin of testing for the Digital Processing Assembly and the Detector Electronics Assembly. ACIS system test begins whenever there is sufficient hardware and software to accept commands and produce telemetry. The requirements for system test will be specified by Software Quality Assurance and the Science branch. Calibration requirements are specified by the SCIENCE branch and will be available for calibration at XRCF.

11.0 SOFTWARE STANDARDS AND PROCEDURES

The following standards should be followed for all EGSE software and will be followed for all software involving crucial support functions.

12.0 EGSE VERIFICATION AND VALIDATION STANDARDS

A test procedure for crucial support functions will be developed by and run by SQA. This test procedure will be developed to validate documented requirements. The results of the test will be recorded in the EGSE project development notebook.

13.0 QUALITY REQUIREMENTS

13.1. SQA Program Requirements

This section defines the SQA review, reporting, and auditing procedures used at MIT to ensure that the EGSE does not corrupt data and handles crucial commands properly.

13.2 Software Documentation

One attribute of quality of software is that requirement and design documentation is complete. Software source code commenting requirements are described in Appendix - II.

13.3 Requirements Traceability

SQA is included in the release process for all EGSE software. It is during this process that SQA ensures that requirements have been satisfied.

13.4 Project reviews

13.4.1 Formal Reviews

At least one week prior to delivery of documents to MSFC for a formal review, SQA will review the Document List that is generated by the SPE. This list identifies all the documents and revision that will be submitted for the formal review. SQA will review software related documents identified on this list to ensure that effected EGSE documentation with indicated revision is or will be available for shipment to MSFC. Discrepancies will be brought to the attention of the PE.

13.4.2 Informal Reviews (Baseline Quality Reviews)

These reviews will be conducted by SQA prior to any baseline release of executable code that is identified with an alphabetic revision ID and involves crucial requirements. . This review ensures that: (1) the code has been tested and meets specifications, except as noted; (2) the delivered software (SW) executable is produced from the sources indicated with the tools indicated. (3) that all software design documentation complies with this plan and other applicable CSR/ACIS plans and procedures. (7) that tool and techniques used to produce and validate the Software Sub-System are identified and controlled.

13.5 Tools and Techniques

SQA will assure that all purchased and developed tools that directly affect the contents of the software that effects crucial support functions are uniquely identified, For example, compilers, linkers, loaders, boot transfer programs, checksum generation programs fall under this category.

These tools and programs are identified in TBD. Text editors, Case tools, and unit test programs need not be controlled.

13.6 Software Testing

13.6.1 Unit Test;

All code involving crucial support functions will be unit tested to ensure that the individual function performs the required functions and outputs the proper results and data. Proper results are determined by using the design limits of the calling (client) function as specified in the design specification defining the called (server) function. Unit testing is typically white box testing and may require the use of software stubs and symbolic debuggers. This testing helps ensure proper operation of a module because tests are generated with knowledge of the internal workings of the module. Unit testing requirements and documentation guidelines are shown in Appendix - III.

13.6.2 Systems/Integration Test

There are two levels of systems/integration testing. One level is the process of testing software capability; e.g. being able to send a message via a Direct Memory Access (DMA) port, or the ability to acquire a row of Charged Coupled Device (CCD) data. During this level, each module is treated as a black box, while conflicts between functions or classes and between software and appropriate hardware are resolved. Integration testing requirements are shown in Appendix - III. Test cases should provide unexpected parameter values when design documentation does not explicitly specify calling requirements for client functions. A second level of systems/integration testing occurs when sufficient modules have been integrated to demonstrate a ACIS scenario e.g. type of science mode or the ability to send commands and receive telemetry . During this phase, composite builds, or baselines, of the software are married to the engineering versions of the hardware to evaluate the combined hardware/software performance for each operational function. The instrumentation and test equipment specified by engineering may be required to identify and resolve hardware/software deficiencies. Both hardware and software documentation is reworked as necessary.

13.6.3 Validation Testing

Validation testing is performed for support functions involving crucial commands. It begins when sufficient hardware and software have been integrated that enables validation of the requirements identified in the EGSE Development Workbook .SQA has the responsibility of ensuring that these test have been run. The PE will ensure that there will be sufficient physical and human engineering resources to support this V & V effort.

13.7 Release, Problem Reporting and Corrective Action

13.7.1 Release Procedures.

The need for control increases when crucial commands are involved. It is anticipated that most releases will be numeric. However, different control procedures will be used depending on function

a. Preliminary software releases and preliminary releases involving crucial functions are identified with numeric revision identification and will be used early in the development process. These release identifiers will be used on software executables when sufficient functionality has been developed and can be used by

others outside of the EGSE development team. When used in the above instance, the declared revision or release code i.e. 10, 11, 12, will be identified on the ECO and, appears in the executable code.

b. Baseline or EGSE releases involving crucial functions are identified with alphabetic revision codes and will be used in the development process at XRCF and later.

13.7.2 Change Control

Change control for software will begin during the EGSE test phase and must be in place when software identified with a numeric release is given to someone outside of EGSE development for use in their work. Change procedures as described in this document and the CMP are used throughout the life of the ACIS project.

a. While a Preliminary Software Sub-System is used by individuals outside of software development, the following procedures will be used. The Software Sub-System release will be accompanied by an ECO that identifies the version of the release and briefly describes functionality.

b. When a baseline release of a Software Sub-System is identified with alphabetic revision, all changes will be controlled as specified in the CMP. The installation of baseline software can occur only after the appropriate ECO has been approved by the CCB.

13.7.3 Problem Reporting

Problems involving EGSE will be communicated to the EGSE development by any appropriate means. Problems involving crucial functions will be validated by EGSE development. The validation process will be recorded in the project notebook.

13.8 Commercially Off the Shelf Available software. (COTS)

If commercially available software is used in support of crucial functions, only the features used will be validated.

If COTS software is modified, then all affected COTS software must be designed developed, tested and controlled as if it were EGSE developed software.

13.9 Non Deliverable Software

Not applicable since the EGSE is not delivered.

13.10 Acceptance and Delivery

There is no software acceptance data package since the EGSE is not delivered. There are internal release procedures as described in Release Procedures Section 12 of this document .

14.0 ACRONYMS & ABBREVIATIONS

ACIS	AXAF CCD Imaging Spectrometer
AXAF	Advanced X-ray Astrophysics Facility
CCB	Configuration Control Board
CCD	Charge Coupled Device
CEI	Contract End Item
CMP	Configuration Management Plan
CTUE	Command and Telemetry Unit (Engineering)
CSR	Center for Space Research
DEA	Detector Electronics Assembly
DMA	Direct Memory Access
DPA	Digital Processing Assembly
DPM	Deputy Project Manager
DR	Document Requirements
ECO	Engineering Change Order
EGSE	Electronic Ground Support Equipment
ICD	Interface Controlled Description
IPI	Instrument Principal Investigator
MIT	Massachusetts Institute of Technology
MSFC	Marshall Space Flight Center
PAM	Product Assurance Manager
PE	ACIS Project Engineer
PECO	Proposed ECO
PM	ACIS Project Manager
RCTUE	Remote Command and Telemetry Unit (Engineering)
ROM	Read Only Memory
SE	System Engineer
SIS	Science Instrument Software
SPM	Software Project Manager
SQA	Software Quality Assurance
SQAP	ACIS Software Quality Assurance Plan
SW	Software
V&V	Verification and Validation
TBD	To Be Determined

15.0 APPENDICES

Appendix I - Software Tailoring Matrix

EGSE MIT Software Tailoring Matrix			Feb 15, 1995
Section	**Description**	**Yes/No**	**Remarks**
4.0	SQA PROGRAM MANAGEMENT AND PLANNING	Yes	
4.1	SQA Plan	Yes	
4.2	Organizational Structure	Yes	
4.3	Training	NO	Not required
5.0	SQA PROGRAM REQUIREMENTS	Yes	
5.1	Program Resource Allocation Monitoring	NO	Not applicable
5.2	SQA Program Audits	NO	Use Reviews
5.2.1	Unscheduled Audits	NO	Not needed
5.2.2	Audits of SQA Organization	NO	Covered by SI SQAP
5.2.3	Audit Reports	NO	
5.3	SQA Records	NO	Covered by SI SQAP
5.4	SQA Status Reports	NO	
5.5	Software Documentation	Yes	
5.6	Requirements Traceability	Yes	
5.7	Software Development Process	Yes	
5.8	Project Reviews	Yes	
5.8.1	Formal Reviews	Yes	
5.8.2	Informal Reviews	Yes	
5.8.2.1	Design Walkthroughs	Yes	
5.8.2.2	Code Walkthroughs	NO	
5.9	Tools and Techniques	Yes	
5.10	Software Configuration Management	Yes	
5.11	Software Testing	Yes	
5.12	Problem Reporting and Corrective Action	Yes	
5.13	Trend Analysis	NO	Develop Cycle to short
5.14	Subcontractor Controls	NO	Not used
5.15	Commercially Available, Reusable and Government Furnished Software (GFS)	Yes	
5.16	Non-deliverable software	Yes	
5.17	Acceptance and delivery	NO	SW is not delivered

Appendix II Coding Documentation Requirements

a. A high level language shall be used except when approved by the SPM.

b. Each method, function and class will be identified with it's own comment header. The contents of the header should identify the purpose and any assumptions the user or caller must be aware of.

c. Coding documentation will, at a minimum, describe reasons for code branching and a description of each variable name. Where possible, each name will be described in the header in which the item being named is defined.

d. Naming conventions shall be used that clearly distinguish literal constants, variables, methods and class/object names. Class/object names should be nouns; methods should be verbs. Variables shall not be re-used for different purposes, except in trivial cases such as loop counts and indices. In addition, all names will contain at least 2 (two) characters to facilitate global pattern searches.

e. Coding complexity conventions for a class shall be established. The design will not exceed a predetermined complexity value without the approval of the SPM. If Cyclomatic is used, a value (Vg) of 10, is recommended. A description of how to calculate Cyclomatic complexity index can be found in Chap 13 of "Software Engineering, a Practitioners Approach" by Roger S. Pressman.; McGraw-Hill.

f. Dispatcher logic shall include a default clause, and loops shall include an escape clause. ' Fall through' logic should not be used without approval from the SPM.

Appendix III Testing Requirements

a. Unit Testing:

1. Environment; Specify testing environment. i.e. if and when stubs and drivers and/or other application routines, special hardware and/or conditions are to be used.

2. Logic Complexity: The number of test cases must exceed the number of paths through the code + 1. Refer to Cyclomatic complexity in Appendix - II above.

3. Boundary Analysis: Decision based in specified values such as "IF name < 10, then do .." Specify tests that will execute the "IF" statement using values for name at 9, 10, and 11. This includes looping instructions, while, for, and tests that use LT, GT, LE, GE operators.

4. Error handling: Design tests that verify the recording of all detected and reportable errors that a program is designed to find and report.

5. Global parameter modification: When a program modifies global variables, design tests that verify the modification. That is; initialize the variable independent of the program, verify memory contents, run the program, and check that memory contents have been modified.

6. Mathematical limit checking: Design tests that use out of range values that could cause the mathematical function to calculate erroneous results.

7. Cessation of test: Specify the conditions under which a testing session stops and a

new build is made. Regression testing is required, according to steps 2 through 6 above, of all lines of code that have been modified.

8. Documentation: The documentation must show that the tests have shown that the topics in items 2 through 6 above have been addressed.

b. Integration Testing;

This type of testing addresses the issues associated with the dual problems of verification and program construction. Integration is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing. The objective is to take unit tested modules and build a program structure that has been dictated by design. The following topics are addressed in the STP.

1. crucial module definition: Decide which classes/modules containcrucial control operations. These classes/modules should be unit tested as soon as possible and not wait for subordinate class/object completion. Use of program stubs may be necessary.

2. Object grouping: Decide what modules comprise an integration group by use of scenarios and appropriate architecture diagrams. It is desirable to integrate at low levels to make bug definition easier. Choose objects that are related to a specific function like command uplink.

3. Depth vs. breadth testing: Decide how to test a group of objects/classes It is suggested that breadth testing be used when interfacing with the hardware. Use stubs, if required, to test dispatcher control modules. Use depth testing when a function is well defined and can be demonstrated, e.g. and application mode like timed exposure.

4. Regression testing: Integration regression testing is required whenever an interface attribute has been changed, e.g. the value of a passed parameter.

5. Top down vs. bottom up: Use top down testing to verify major control or decision points. Use bottom up to test hardware driver type functions.

6. The preferred method of designing of each test case is to make the server place a value in the parameter being tested and then the client can invoke to server to obtain the value. When conditions can't be produced to cause the server to place the desired value in a parameter, then a debugger could be used to place the value.

c. System testing:

System testing is actually a series of different tests whose primary purpose is to fully exercise the computer-based system. Each test may have a different purpose, but all tests are performed to expose system limitations. System testing will follow formal test procedures based on hardware, software, and science requirements as specified in the STP.

d. Validation Testing

The purpose of validation is to prove that the ACIS instrument performs as specified in the requirements documents listed above in the Applicable Documents Section of this document.

1. Validation tests/procedures will identify a testing method and pass/fail criteria.

2. When ranges are specified in the requirements, tests cases will include boundary values

at, n-1, n, n+1, where possible.

3. When LT or GT limits are specified, the measured value should be recorded.

Testing Documentation:

Testing documentation must be sufficient to provide evidence that the testing objectives, as stated in the preceding sections or the STP, have been met.